CHAPTER 5

# Regression Methods

In this chapter, one of the most commonly used data science techniques—fitting data with functions or *function fitting* will be explored. The basic idea behind function fitting is to predict the value (or class) of a dependent attribute y, by combining the predictor attributes **X** into a function, $y = f(\mathbf{X})$. Function fitting involves many different techniques and the most common ones are *linear regression* for numeric prediction and *logistic regression* for classification. These two, form the majority of the material in this chapter. Regression models continue to be one of the most common analytics tools used by practitioners today.[1]

Regression is a relatively old technique dating back to the Victorian era (1830s to the early 1900s). Much of the pioneering work was done by Sir Francis Galton, a distant relative of Charles Darwin, who came up with the concept of *regressing toward the mean* while systematically comparing children's heights against their parents' heights. He observed there was a strong tendency for tall parents to have children slightly shorter than themselves, and for short parents to have children slightly taller than themselves. Even if the parents' heights were at the tail ends of a bell curve or normal distribution, their children's heights tended toward the mean of the distribution. Thus, in the end, all the samples regressed toward a population mean. Therefore, this trend was called *regression* by Galton (Galton, 1888) and, thus, the foundations for linear regression were laid.

In the first section of this chapter, the theoretical framework for the simplest of function-fitting methods: the *linear regression model*, will be provided. The main focus will be on a case study that demonstrates how to build regression models. Due to the nature of the function-fitting approach, one limitation

[1] Rexer Analytics survey available from http://www.rexeranalytics.com.



Data Science. DOI: https://doi.org/10.1016/B978-0-12-814761-0.00005-8

that modelers have to deal with is what is called the *curse of dimensionality*. As the number of predictors **X**, increases, not only will our ability to obtain a good model reduce, it also adds computational and interpretational complexity. *Feature selection* methods will be introduced that can reduce the number of predictors or factors required to a minimum and still obtain a good model. The mechanics of implementation, will be explored, to do the data preparation, model building, and validation. Finally, in closing some checkpoints to ensure that linear regression is used correctly will be described.

In the second section of this chapter *logistic regression* will be discussed. Strictly speaking, it is a classification technique, closer in its application to decision trees or Bayesian methods. But it shares an important characteristic with linear regression in its function-fitting methodology and, thus, merits inclusion in this chapter, rather than the previous one on classification.

## 5.1 LINEAR REGRESSION

Linear regression is not only one of the oldest data science methodologies, but it also the most easily explained method for demonstrating function

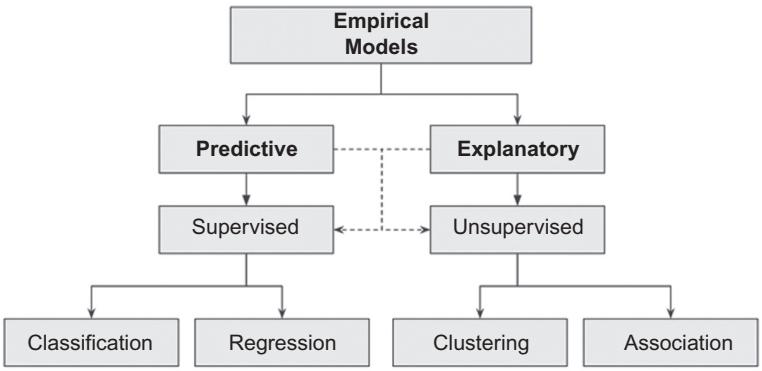
### PREDICTING HOME PRICES

What features would play a role in deciding the value of a home? For example, locality, the number of rooms, its age, the quality of schools in the vicinity, its location with respect to major sources of employment, and its accessibility to major highways, are some of the important considerations most potential home buyers would like to factor in. But which of these are the most significant influencers of the price? Is there a way to determine these? Once these factors are known, can they be incorporated into a model that can be used for predictions? The case study that will be discussed later in this chapter addresses this problem, using multiple linear regressions to predict the median home prices in an urban region given the characteristics of a home.

A common goal that all businesses have to address in order to be successful is growth, in revenues and profits. Customers are what will enable this to happen. Understanding and increasing the likelihood that someone will buy again from the company is, therefore, critical. Another question that would help strategically, for example in customer segmentation, is being able to predict how much money a customer is likely to spend, based on data about their previous purchase habits. Two very important distinctions need to be made here: understanding why someone purchased from the company will fall into the realm of *explanatory modeling*, whereas, predicting how much someone is likely to spend will fall into the realm of predictive modeling. Both these types of models fall under a broader category of *surrogate* or empirical models which relies on historical data to develop rules of behavior as opposed to *system* models which use fundamental principles (such as laws of physics or chemistry) to develop rules. See Fig. 1.2 for a taxonomy of data science. In this chapter, the predictive capability of models will be focused on as opposed to the explanatory capabilities. Historically much of applied linear regression in statistics has been used for explanatory needs. Later on in this chapter with the case of logistic regression, it will be demonstrated, how *both* needs can be met with good analytical interpretation of models.

(*Continued*)

**(Continued)**




fitting. The basic idea is to come up with a function that explains and predicts the value of the target variable when given the values of the predictor variables.

### 5.1.1 How it Works

A simple example is shown in Fig. 5.1: if one would like to know the effect of the number of rooms in a house (predictor) on its median sale price (target). Each data point on the chart corresponds to a house (Harrison, 1978). It is evident that on average, increasing the number of rooms tends to also increase median price. This general statement can be captured by drawing a straight line through the data. The problem in linear regression is, therefore, finding a line (or a curve) that best explains this tendency. If there are two predictors, then the problem is to find a surface (in a three-dimensional space). With more than two predictors, visualization becomes difficult and one has to revert to a general statement where the dependent variables are expressed as a linear combination of independent variables:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \tag{5.1}$$

Consider the problem with one predictor. Clearly, one can fit an infinite number of straight lines through a given set of points such as the ones shown in Fig. 5.1. How does one know which one is the best? A metric is

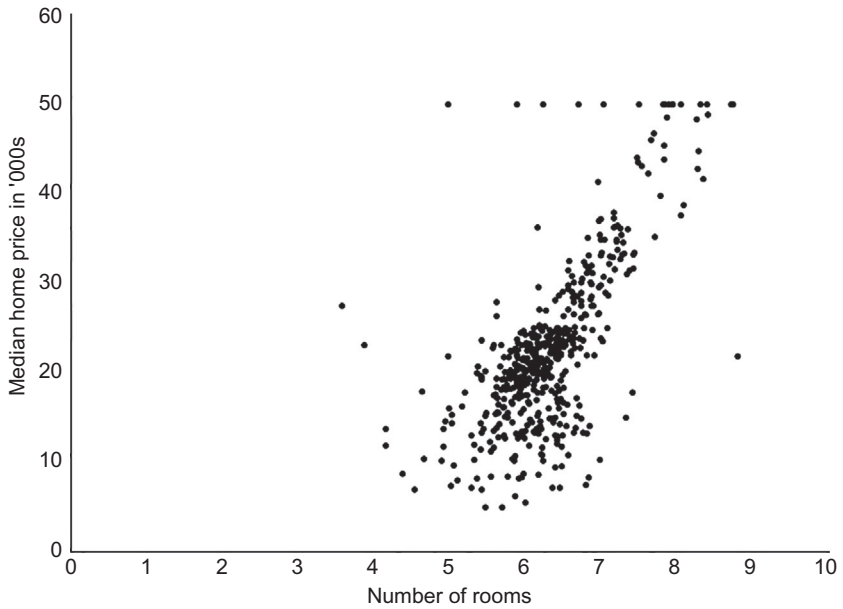



**FIGURE 5.1**
A simple regression model.

needed, one that helps quantify the different straight line fits through the data. Once this metric is found, then selecting the best line becomes a matter of finding the optimum value for this quantity.

A commonly used metric is the concept of an error function. Suppose one fits a straight line through the data. In a single predictor case, the *predicted* value, $\hat{y}$, for a value of $x$ that exists in the dataset is then given by:

$$\hat{y} = b_0 + b_1 x \tag{5.2}$$

Then, error is simply the difference between the actual target value and predicted target value:

$$e = y - \hat{y} = y - (b_0 + b_1 x) \tag{5.3}$$

This equation defines the error at a single location $(x, y)$ in the dataset. One could easily compute the error for all existing points to come up with an aggregate error. Some errors will be positive, and others will be negative. The difference can be squared to eliminate the sign bias and an average error for a given fit can be calculated as:

$$\frac{J}{n} = \frac{\sum e^2}{n} = \frac{\sum (y_i - \dot{y}_i)^2}{n} = \frac{\sum (y_i - b_0 - b_i x_i)^2}{n} \tag{5.4}$$

where $n$ represents the number of points in the dataset. $J$ is the total squared error. For a given dataset, the best combination of $(b_0, b_1)$ can then be found,

which minimizes the total error, $e$. This is a classical minimization problem, which is handled with methods of calculus. Stigler provides some interesting historical details on the origins of the method of least squares, as it is known (Stigler, 1999). Using the methods of calculus, the values of $b$ can be found, which minimize the total error $J$. Specifically, one can take partial derivatives of $J$ with respect to $b_1$ and $b_0$ and set them equal to zero. Chain rule of differential calculus gives us:

$$\begin{aligned}
&\partial J/\partial b_1 = \partial J/\partial \hat{y} \quad \partial \hat{y}/\partial b_1 \\
&\Rightarrow \partial J/\partial b_1 = 2(\Sigma(y_i - b_0 - b_1 x_i))\partial \hat{y}/\partial b_1 = 0 \\
&\Rightarrow \Sigma(y_i - b_0 - b_1 x_i)(-x_i) = 0 \\
&\Rightarrow -\Sigma(y_i x_i) + \Sigma(b_0 x_i) + \Sigma(b_1 x^2{}_i) = 0 \\
&\Rightarrow \Sigma(y_i x_i) = b_0 \Sigma(x_i) + b_1 \Sigma(x^2{}_i)
\end{aligned} \tag{5.5}$$

Similarly, one can use:

$$\begin{aligned}
&\partial J/\partial b_0 = 2(\Sigma(y_i - b_0 - b_1 x))\partial \hat{y}/\partial b_0 = 0 \\
&\Rightarrow \Sigma(y_i - b_0 - b_1 x_i)(-1) = 0 \\
&\Rightarrow -\Sigma(y_i) + \Sigma(b_0.1) + \Sigma(b_1 x_i)1 = 0 \\
&\Rightarrow -\Sigma(y_i) + b_0 \Sigma(1) + b_1 \Sigma(x_i) = 0 \\
&\Rightarrow \Sigma(y_i) = b_0 N + b_1 \Sigma(x_i)
\end{aligned} \tag{5.6}$$

Eqs. (5.5) and (5.6) are two equations in two unknowns, $b_0$ and $b_1$, which can be further simplified and solved to yield the expressions:

$$b_1 = (\Sigma x_i y_i - \bar{y}\Sigma x_i)/(\Sigma x^2{}_i - \bar{x} - \Sigma x_i) \tag{5.7}$$

$$b_0 = (\bar{y}\Sigma x^2{}_i - \bar{x}\Sigma x_i y_i)/(\Sigma x^2{}_i - \bar{x}\Sigma x_i) \tag{5.8}$$

$b_1$ can also be written as (5.9a):

$$b_1 = \text{Correlation } (y, x) \times \frac{s_y}{s_x} \tag{5.9a}$$

$$b_0 = y_{\text{mean}} - b_1 \times x_{\text{mean}} \tag{5.9b}$$

where Correlation($x$, $y$) is the correlation between $x$ and $y$ and $s_y$, $s_x$ are the standard deviations of $y$ and $x$. Finally, $x_{\text{mean}}$ and $y_{\text{mean}}$ are the respective mean values.

Practical linear regression algorithms use an optimization technique known as *gradient descent* (Fletcher, 1963; Marquardt, 1963) to identify the combination of $b_0$ and $b_1$ which will minimize the error function given in Eq. (5.4). The advantage of using such methods is that even with several predictors, the optimization works fairly robustly. When such a process is applied to the simple example shown, one gets an equation of the form:

$$\text{Median price} = 9.1 \times (\textit{number of rooms}) - 34.7 \tag{5.10}$$

where $b_1$ is 9.1 and $b_0$ is $-34.7$. From this equation, it can be calculated that for a house with six rooms, the value of the median price is about 20 (the prices are expressed in thousands of US dollars, c. 1970). In Fig. 5.1 it's

evident that for a house with six rooms, the actual price can range between 10.5 and 25. An infinite number of lines could have been fit in this band, which would have all predicted a median price within this range—but the algorithm chooses the line that minimizes the average error over the full range of the independent variable and is, therefore, the best fit for the given dataset.

For some of the points (houses) shown in Fig. 5.1 (at the top of the chart, where median price = 50) the median price appears to be independent of the number of rooms. This could be because there may be other factors that also influence the price. Thus, more than one predictor will need to be modeled and *multiple linear regression (MLR)*, which is an extension of simple linear regression, will need to be used. The algorithm to find the coefficients of the regression Eq. (5.1) can be easily extended to more than one dimension.

The single variable expression for error function in (5.4) can be generalized to multiple variables quite easily, $\hat{y}_i = b_0 + b_1x_1 + ... + b_Dx_D$. If we let $x = [x_0,x_1,...,x_D]$ and recognize that the intercept term can be written as $b_0x_0$ where $x_0 = 1$, then we can write (5.4) as $E = \sum_{i=1}^{N}(y_i - B^Tx_i)^2$ where $B$ is a vector of weights $[b_0, b_1,...,b_D]$ for a dataset with $D$ columns or features and $N$ samples. Similar to how we computed (5.7) and (5.8), we can take a derivative of $E$ with respect to each weight $B$ and will end up with $D$ equations to be solved for $D$ weights (one corresponding to each feature). The partial derivative for each weight is

$$\partial E/\partial b_j = \partial E/\partial \hat{y} * \partial \hat{y}_i/\partial b_j$$
$$\Rightarrow \partial E/\partial b_j = 2\Sigma\left(y_i - B^Tx_i\right)\partial \hat{y}_i/\partial b_j$$
$$\Rightarrow \partial E/\partial b_j = 2\Sigma\left(y_i - B^Tx_i\right)(-x_i)$$
$$\Rightarrow \Sigma y_i(-x_i) - B^T\Sigma(x_i)(-x_i)$$

When we consider all $D$ weights at the same time, this can be very simply written in matrix form as follows:

$$\partial E/\partial B = -\left(Y^TX\right) + B\left(X^TX\right) \tag{5.11}$$

Here $B$ is a $1 \times D$ matrix or vector. As in the case of simple linear regression, we can set this derivative to zero, to solve for the weights, $B$ and get the following expression: $-(Y^TX) + B(X^TX) = 0$. In this case, solving for $B$ now becomes a matrix inversion problem and results in $B = (X^TX)^{-1}Y^TX$. The reader can verify as an exercise that this matrix is dimensionally consistent (Hint: Use the fact that the matrix shape of $X$ is $N \times D$, $Y$ is $N \times 1$ and $B$ is $1 \times D$). MLR can be applied in any situation where a numeric prediction, for example "how much will something sell for," is required. This is in contrast to making categorical predictions such as "will someone buy/not buy" or "will/will not fail," where classification tools such as decision trees or logistic regression models are used. In order to ensure regression models are not arbitrarily deployed, several checks must be performed on the model to ensure that the regression is accurate which will be the focus of a later section in this chapter.

**Table 5.1** Sample View of the Classic Boston Housing Dataset

| CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | Target = MEDV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.00632 | 18 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.09 | 1 | 296 | 15.3 | 396.9 | 4.98 | 24 |
| 0.02731 | 0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.9 | 9.14 | 21.6 |
| 0.02729 | 0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 | 4.03 | 34.7 |
| 0.03237 | 0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 | 2.94 | 33.4 |
| 0.06905 | 0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.9 | 5.33 | 36.2 |
| 0.02985 | 0 | 2.18 | 0 | 0.458 | 6.43 | 58.7 | 6.0622 | 3 | 222 | 18.7 | 394.12 | 5.21 | 28.7 |
| 0.08829 | 12.5 | 7.87 | 0 | 0.524 | 6.012 | 66.6 | 5.5605 | 5 | 311 | 15.2 | 395.6 | 12.43 | 22.9 |
| 0.14455 | 12.5 | 7.87 | 0 | 0.524 | 6.172 | 96.1 | 5.9505 | 5 | 311 | 15.2 | 396.9 | 19.15 | 27.1 |

**Table 5.2** Attributes of Boston Housing Dataset

1. CRIM per capita crime rate by town
2. ZN proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS proportion of non-retail business acres per town
4. CHAS Charles River dummy variable (=1 if tract bounds river; 0 otherwise)
5. NOX nitric oxide concentrations (parts per 10 million)
6. RM average number of rooms per dwelling
7. AGE proportion of owner-occupied units built prior to 1940
8. DIS weighted distances to five Boston employment centers
9. RAD index of accessibility to radial highways
10. TAX full-value property-tax rate per $10,000
11. PTRATIO pupil-teacher ratio by town
12. B 1000(Bk − 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT percent lower status of the population
14. MEDV Median value of owner-occupied homes in $1000's

The housing example can be extended in order to include additional variables. This comes from a study of urban environments conducted in the late 1970s[2] (Harrison, 1978). The objectives of this are:

1. Identify which of the several attributes are required to accurately predict the median price of a house.
2. Build a multiple linear regression model to predict the median price using the most important attributes.

The original data consist of thirteen predictors and one response variable, which is the variable that needs to be predicted. The predictors include physical characteristics of the house (such as number of rooms, age, tax, and location) and neighborhood features (schools, industries, zoning) among others. The response variable is of course the median value (MEDV) of the house in thousands of dollars. Table 5.1 shows a snapshot of the dataset, which has altogether 506 examples. Table 5.2 describes the features or attributes of the dataset.

[2] http://archive.ics.uci.edu/ml/datasets/Housing.

### 5.1.2 How to Implement

In this section, how to set up a RapidMiner process to build a multiple linear regression model for the Boston Housing dataset will be demonstrated. The following will be described:

1. Building a linear regression model
2. Measuring the performance of the model
3. Understanding the commonly used options for the *Linear Regression* operator
4. Applying the model to predict MEDV prices for unseen data

#### *Step 1: Data Preparation*

As a first step, the data is separated into a training set and an unseen test set. The idea is to build the model with the training data and test its performance on the unseen data. With the help of the *Retrieve* operator, import the raw data (available in the companion website www.IntroDataScience.com) into the RapidMiner process. Apply the *Shuffle* operator to randomize the order of the data so that when the two partitions are seperated, they are statistically similar. Next, using the *Filter Examples Range* operator, divide the data into two sets as shown in Fig. 5.2. The raw data has 506 examples, which will be linearly split into a training set (from row 1 to 450) and a test set (row 451−506) using the two operators.

Insert the *Set Role* operator, change the role of MEDV to label and connect the output to a *Split Validation* operator's input training port as shown in Fig. 5.3. The training data is now going to be further split into a training set and a validation set (keep the default Split Validation options as

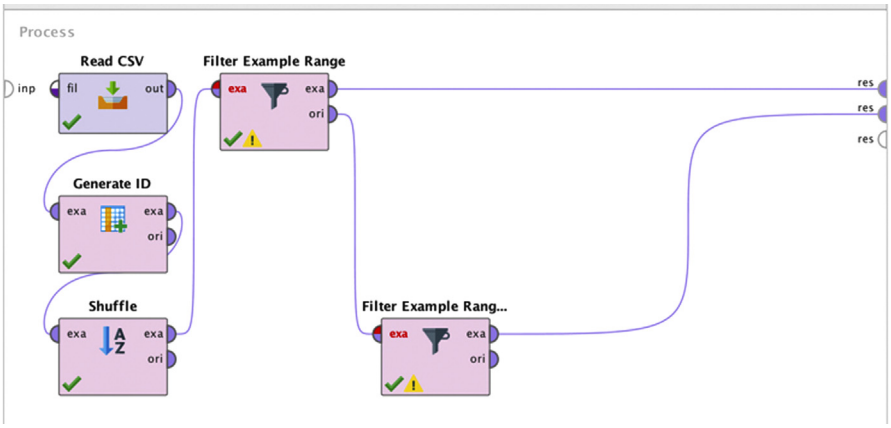



**FIGURE 5.2**
Separating the data into training and testing samples.

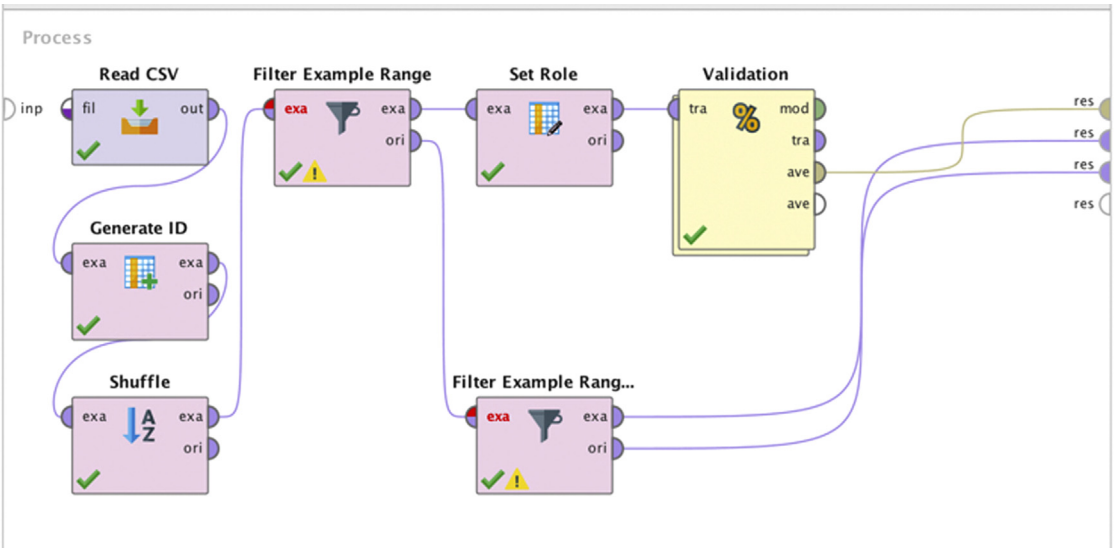



**FIGURE 5.3**
Using the split validation operator.

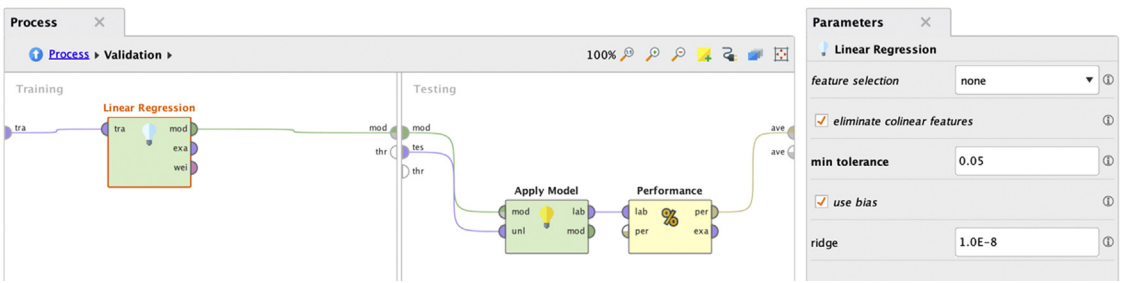



**FIGURE 5.4**
Applying the linear regression operator and measuring performance.

is, i.e., *relative*, *0.7*, and *shuffled*). This will be needed in order to measure the performance of the linear regression model. It is also a good idea to set the local random seed (to default the value of 1992), which ensures that RapidMiner selects the same samples if this process is run at a later time.

After this step, double-click the *Validation* operator to enter the nested process. Inside this process insert the *Linear Regression* operator on the left window and *Apply Model* and *Performance (Regression)* in the right window as shown in Fig. 5.4. Click on the *Performance* operator and check squared error, correlation, and squared correlation inside the Parameters options selector on the right (Fig. 5.5).

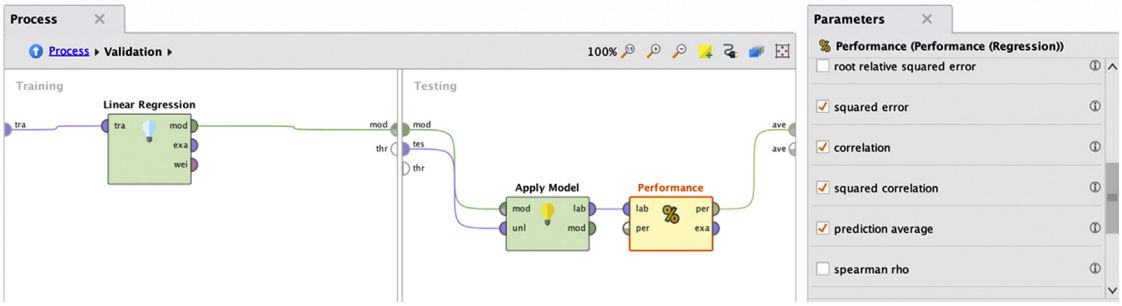



**FIGURE 5.5**
Selecting performance criteria for the MLR.

### *Step 2: Model Building*

Select the *Linear Regression* operator and change the *feature selection* option to none. Keep the default eliminate collinear features checked, which will remove factors that are linearly correlated from the modeling process. When two or more attributes are correlated to one another, the resulting model will tend to have coefficients that cannot be intuitively interpreted and, furthermore, the statistical significance of the coefficients also tends to be quite low. Also keep the use bias checked to build a model with an intercept [the $b_0$ in Eq. (5.2)]. Keep the other default options intact (Fig. 5.4).

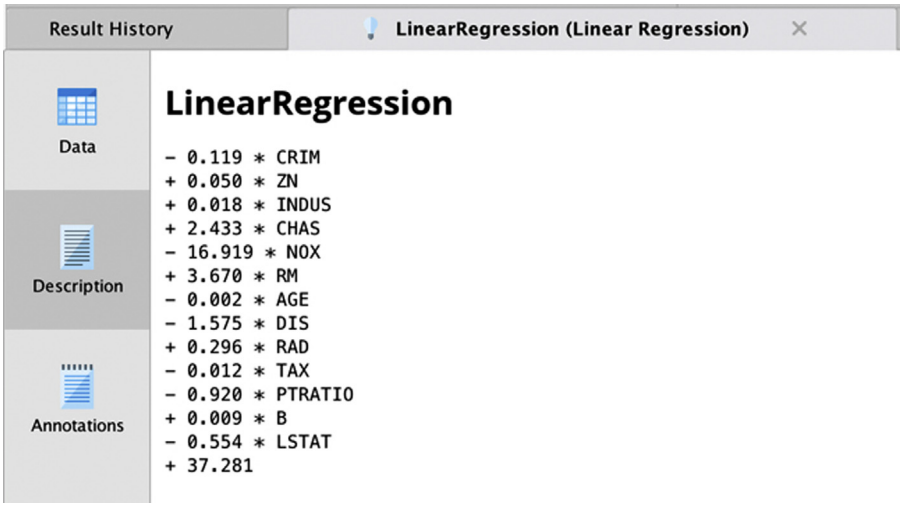
When this process is run the results shown in Fig. 5.6 will be generated.

### *Step 3: Execution and Interpretation*

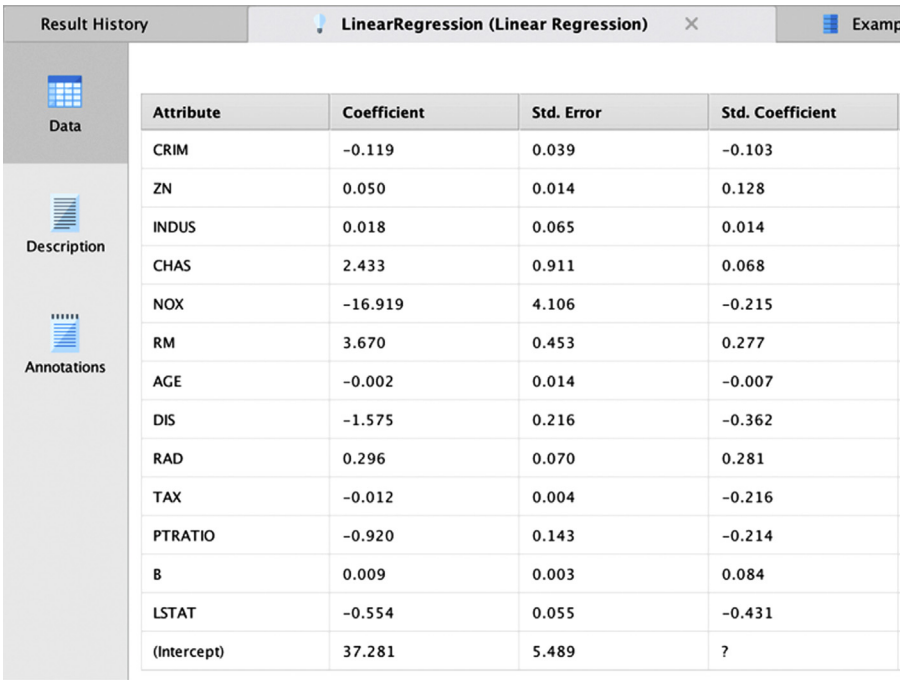
There are two views that one can examine in the *Linear Regression* output tab: the Description view, which actually shows the function that is fitted (Fig. 5.6A) and the more useful Data view, which not only shows the coefficients of the linear regression function, but also gives information about the significance of these coefficients (Fig. 5.6B). The best way to read this table is to sort it by double-clicking on the column named Code, which will sort the different factors according to their decreasing level of significance. RapidMiner assigns four stars (****) to any factor that is highly significant.

In this model, no *feature selection* method was used and as a result all 13 factors are in the model, including AGE and INDUS, which have very low significance. However, if the same model were to be run by selecting any of the options that are available in the drop-down menu of the *feature selection* parameter, RapidMiner would have removed the least significant factors from the model. In the next iteration, the *greedy* feature selection is used, and this will have removed the least significant factors, INDUS and AGE, from the function (Fig. 5.7A & B).

(A)




(B)




| Attribute | Coefficient | Std. Error | Std. Coefficient |
|---|---|---|---|
| CRIM | -0.119 | 0.039 | -0.103 |
| ZN | 0.050 | 0.014 | 0.128 |
| INDUS | 0.018 | 0.065 | 0.014 |
| CHAS | 2.433 | 0.911 | 0.068 |
| NOX | -16.919 | 4.106 | -0.215 |
| RM | 3.670 | 0.453 | 0.277 |
| AGE | -0.002 | 0.014 | -0.007 |
| DIS | -1.575 | 0.216 | -0.362 |
| RAD | 0.296 | 0.070 | 0.281 |
| TAX | -0.012 | 0.004 | -0.216 |
| PTRATIO | -0.920 | 0.143 | -0.214 |
| B | 0.009 | 0.003 | 0.084 |
| LSTAT | -0.554 | 0.055 | -0.431 |
| (Intercept) | 37.281 | 5.489 | ? |

**FIGURE 5.6**

(A) Description of the linear regression model. (B) Tabular view of the model. Sort the table according to significance by double-clicking on the Code column.

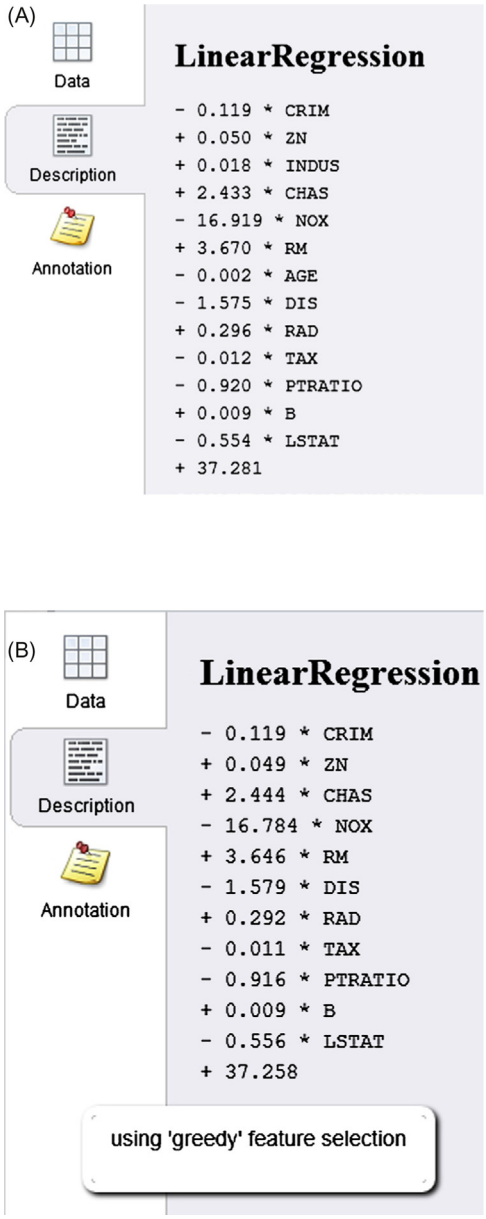



**FIGURE 5.7**
(A) Model without any feature selection. (B) Model with greedy feature selection.

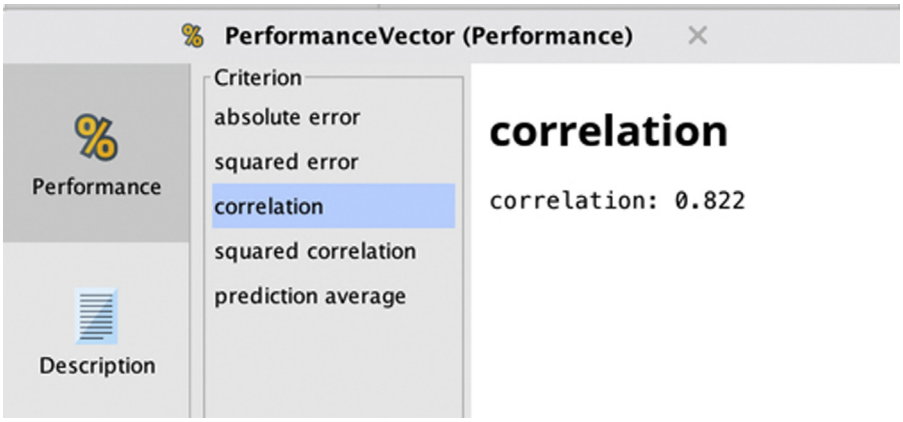



**FIGURE 5.8**
Generating the $r^2$ for the model.

Feature selection in RapidMiner can be done automatically within the *Linear Regression* operator as described or by using external wrapper functions such as forward selection and backward elimination. These will be discussed separately in Chapter 14, Feature Selection.

The second output to pay attention to is the Performance: a handy check to test the goodness of fit in a regression model is the *squared correlation*. Conventionally this is the same as the adjusted $r^2$ for a model, which can take values between 0.0 and 1.0, with values closer to 1 indicating a better model. For either of the models shown above, a value around 0.822 can be obtained (Fig. 5.8). The squared error output was also requested: the raw value in itself may not reveal much, but this is useful in comparing two different models. In this case it was around 25.

One additional insight that can be extracted from the modeling process is ranking of the factors. The easiest way to check this is to rank by *p*-value. As seen in Fig. 5.9, RM, LSTAT, and DIS seem to be the most significant factors. This is also reflected in their absolute *t*-stat values. The *t*-stat and *P*-values are the result of the hypothesis tests conducted on the regression coefficients. For the purposes of predictive analysis, the key takeaway is that a higher *t*-stat signals that the null hypothesis—which assumes that the coefficient is zero—can be safely rejected. The corresponding *p*-value indicates the probability of wrongly rejecting the null hypothesis. Its already been noted how the number of rooms (RM) was a good predictor of the home prices, but it was unable to explain all of the variations in median price. The $r^2$ and squared

| Attribute | Coefficient | Std. Error | Std. Coeff... | Tolerance | t-Stat | p-Value ↑ | Code |
|---|---|---|---|---|---|---|---|
| LSTAT | -0.556 | 0.052 | -0.432 | 0.490 | -10.661 | 0 | **** |
| RM | 3.646 | 0.443 | 0.275 | 0.581 | 8.236 | 0.000 | **** |
| DIS | -1.579 | 0.199 | -0.363 | 0.823 | -7.928 | 0.000 | **** |
| (Intercept) | 37.258 | 5.440 | ? | ? | 6.849 | 0.000 | **** |
| PTRATIO | -0.916 | 0.140 | -0.213 | 0.793 | -6.547 | 0.000 | **** |
| NOX | -16.784 | 3.794 | -0.213 | 0.812 | -4.424 | 0.000 | **** |
| RAD | 0.292 | 0.068 | 0.276 | 0.769 | 4.318 | 0.000 | **** |
| ZN | 0.049 | 0.014 | 0.128 | 0.877 | 3.465 | 0.001 | **** |
| TAX | -0.011 | 0.004 | -0.208 | 0.749 | -3.219 | 0.001 | *** |
| CRIM | -0.119 | 0.039 | -0.103 | 0.843 | -3.088 | 0.002 | *** |
| B | 0.009 | 0.003 | 0.083 | 0.905 | 2.953 | 0.003 | *** |
| CHAS | 2.444 | 0.905 | 0.068 | 0.991 | 2.701 | 0.007 | *** |

**FIGURE 5.9**
Ranking variables by their *P*-values.

error for that one-variable model were 0.405 and 45, respectively. This can be verified by rerunning the model built so far using only one independent variable, the number of rooms, RM. This is done by using the *Select Attributes* operator, which has to be inserted in the process before the *Set Role* operator. When this model is run, the equation shown earlier, Eq. (5.10), will be obtained in the model *Description*. By comparing the corresponding values from the MLR model (0.676 and 25) to the simple linear regression model, it's evident that both of these quantities have improved, thus, affirming the decision to use multiple factors.

One now has a more comprehensive model that can account for much of the variability in the response variable, MEDV. Finally, a word about the sign of the coefficients: LSTAT refers to the percentage of low-income households in the neighborhood. A lower LSTAT is correlated with higher median home price, and this is the reason for the negative coefficient on LSTAT.

### *Step 4: Application to Unseen Test Data*

This model is now ready to be deployed against the unseen data that was created at the beginning of this section using the second *Filter Examples* operator (Fig. 5.2). A new *Set Role* operator will need to be added, select MEDV under parameters and set it to target role prediction from the pull-down menu. Add another *Apply Model* operator and connect the output of *Set Role* to its unlabeled port; additionally, connect the output model from the Validation

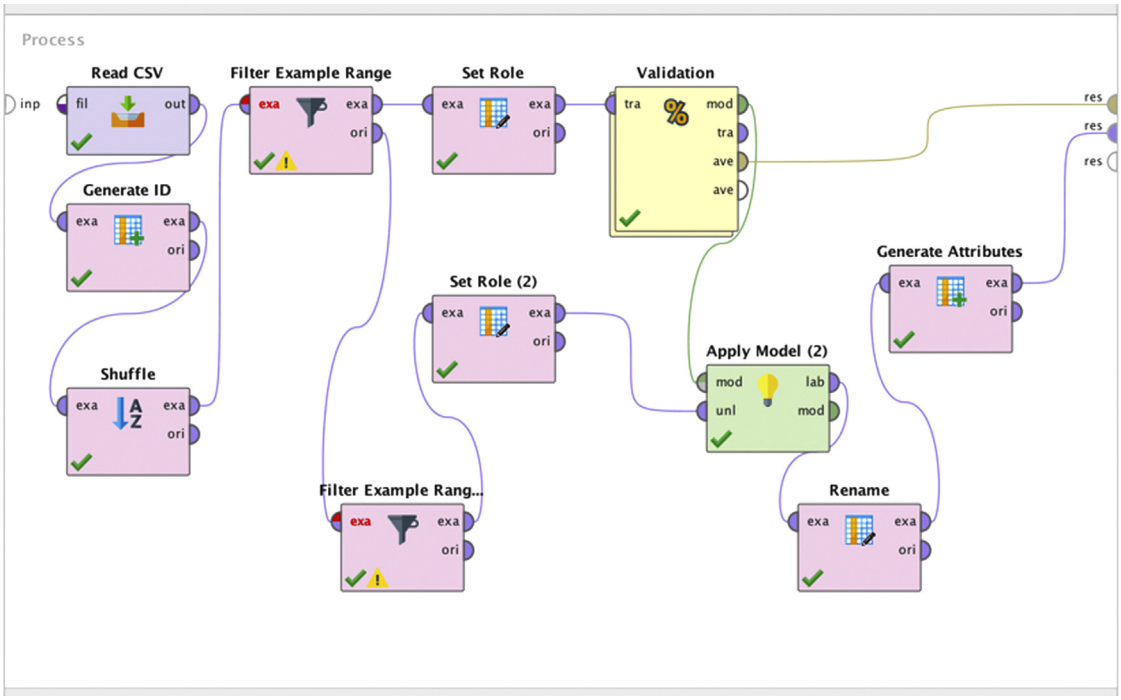



**FIGURE 5.10**
Setting up a process to do the comparison between the unseen data and the model predicted values.

process to the input model port of the new *Apply Model*. What has been done is, the attribute MEDV has been changed from the unseen set of 56 examples to a prediction. When the model is applied to this example set, one will be able to compare the prediction (MEDV) values to the original MEDV values (which exist in the set) to test how well the model would behave on new data. The difference between prediction (MEDV) and MEDV is termed residual. Fig. 5.10 shows one way to quickly check the residuals for the models application. A *Rename* operator would be needed to change the name of "prediction (MEDV)" to predictedMEDV to avoid confusing RapidMiner when the next operator, *Generate Attributes*, is used to calculate residuals (try without using the *Rename* operator to understand this issue as it can pop up in other instances where *Generate Attributes* is used). Fig. 5.11 shows the statistics for the new attribute residuals which indicate that the mean is close to 0 (−0.27) but the standard deviation (and hence, variance) at 4.350 is not quite small. The histogram also seems to indicate that the residuals are not quite normally distributed, which would be another motivation to continue to improve the model.

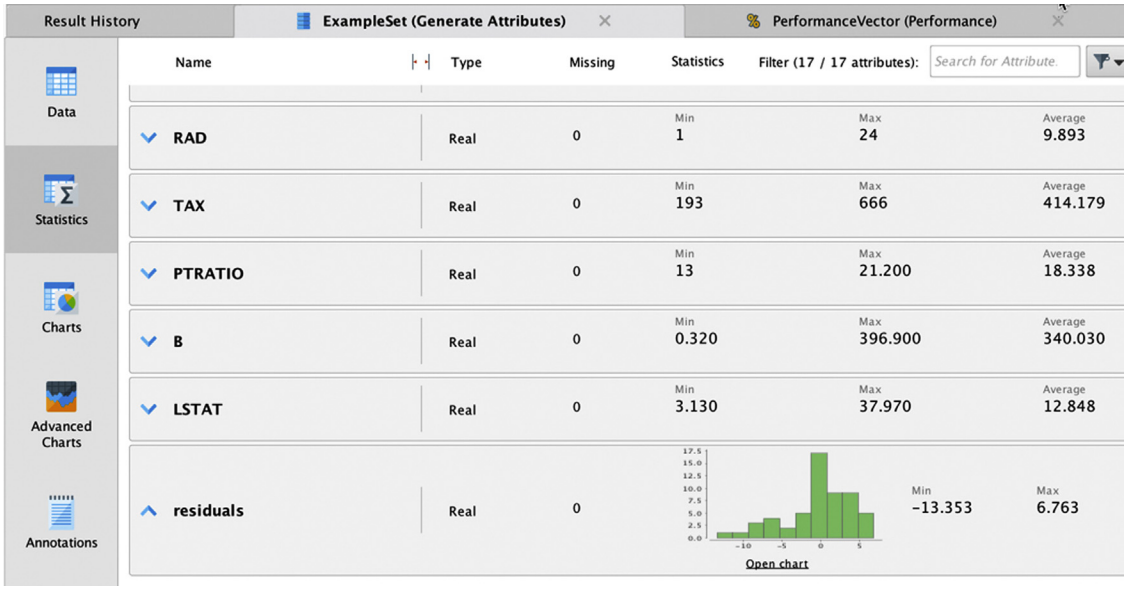



**FIGURE 5.11**
Statistics of the residuals for the unseen data show that some model optimization may be necessary.

### 5.1.3 Checkpoints

This section on linear regression can be wrapped up with a brief discussion of several checkpoints to ensure that any models are valid. This is a critical step in the analytics process because all modeling follows the garbage in, garbage out (GIGO) dictum. It is incumbent upon the data scientist to ensure these checks are completed.

***Checkpoint 1:*** One of the first checkpoints to consider before accepting any regression model is to quantify the $r^2$, which is also known as the coefficient of determination which effectively explains how much variability in the dependent variable is explained by the independent variables (Black, 2008). In most cases of linear regression, the $r^2$ value lies between 0 and 1. The ideal range for $r^2$ varies across applications; for example, in social and behavioral science models typically low values are acceptable. Generally, extremely low values ($\sim$ $<0.2$) indicate that the variables in the model do not explain the outcome satisfactorily. A word of caution about overemphasizing the value of $r^2$: when the intercept is set to zero (in RapidMiner, when one unchecks *use bias*, Fig. 5.5), $r^2$ values tend to be inflated because of the manner in which they are calculated. In such situations where a zero intercept are required, it makes sense to use other checks such as the mean and variance of the residuals.

The most common metric used for measuring how well a regression model fits the data is by using the coefficient of determination, $r^2$. This is defined as:

$$r^2 = 1 - \text{SSE}/\text{SSYY} \tag{5.12}$$

SSE is simply the sum of squared errors which is given by $\sum e^2$ in Eq. (5.4). SSYY is the aggregate mean deviation defined by:

$$\text{SSYY} = \sum (y - \bar{y})^2 \tag{5.13}$$

Intuitively, it is easy to see that if SSE is close to zero, then $r^2$ is close to 1—a perfect fit. If SSE is close to SSYY then $r^2$ is close to zero—the model is simply predicting the average value of $y$ for all values of $x$. The nice thing about $r^2$ is that because it only depends on $y$ and not the weights or independent variables, it can be used for any form of regression model: simple or multiple.

***Checkpoint 2:*** This leads to the next check, which is to ensure that all error terms in the model are normally distributed. To do this check in RapidMiner, a new attribute called error can be generated, which is the difference between the predicted MEDV and the actual MEDV in the test dataset. This can be done using the *Generate Attributes* operator. This is what was done in step 5 in the last section. Passing checks 1 and 2 will ensure that the independent and dependent variables are related. However, this does not imply that the independent variable is the cause and the dependent is the effect. Remember that *correlation is not causation*!

***Checkpoint 3:*** Highly nonlinear relationships, result in simple regression models failing these checks. However, this does not mean that the two variables are not related. In such cases it may become necessary to resort to somewhat more advanced analytical methods to test the relationship. This is best described and motivated by Anscombe's quartet, presented in Chapter 3, Data Exploration.

***Checkpoint 4:*** In addition to testing the goodness of a model fit using $r^2$, it is also important to ensure that there is no *overfitting* of data. Overfitting refers to the process of developing a model that is so well tuned to represent the training data that its least squares error is as low as possible, but this error becomes high when the model is used on unseen or new data. That is, the model fails to generalize. The following example illustrates this and also develops intuition for avoiding such behavior by using what is called Regularization.

Consider some sample data which represent an underlying simple function $y = 3x + 1$. If the data were to be plotted it would look like Fig. 5.12.

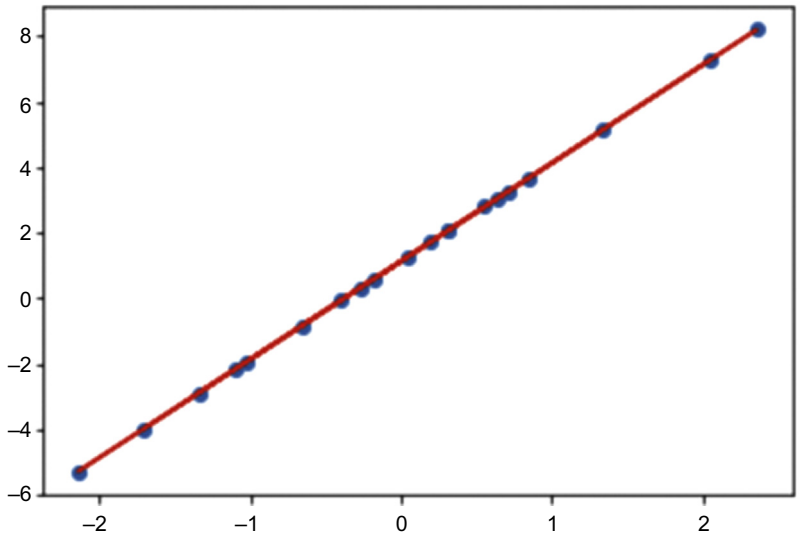



**FIGURE 5.12**
Linear regression line.

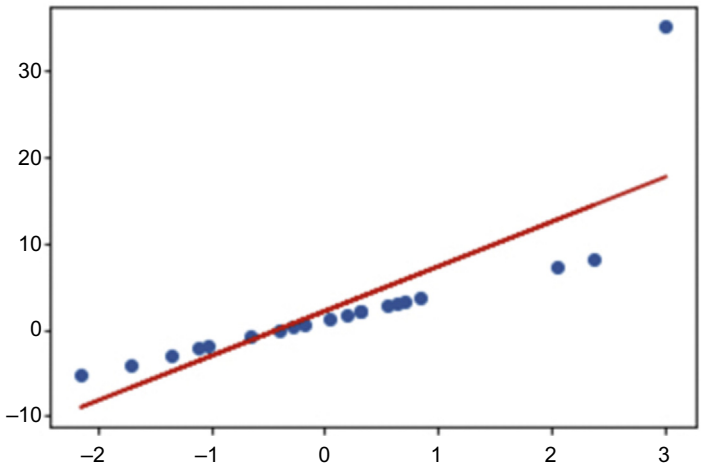



**FIGURE 5.13**
Linear regression line with an outlier.

A regression model could be fit to this data and a nice linear fit obtained, as shown by the line, as well as obtaining the following coefficients: $b_0 = 1.13$ and $b_1 = 3.01$, which is close to the underlying function. Now suppose that there is a new data point [3, 30] which is somewhat of an outlier and now the model has to be refit, the results would be similar to those shown in Fig. 5.13.

Compared to the previous fit, the outlier tends to pull the fitted line up toward the outlying point. In other words, as the model tries to minimize the squared distance between it and every point, a few outlying points tend to exert a disproportionate influence on the model's characteristics.

Another way to look at this is that the model tries to fit all the training data points to the best of its ability or causes *overfitting*. A consequence of overfitting is that the overall error on the training data will be minimized, but if the same model is tried on new data points (which were not used for training), the error tends to increase till it's out of proportion. One symptom of overfitting is the generation of large coefficients. In the example above, these coefficients are now $b_0 = 2.14$ and $b_1 = 6.93$, that is, they have increased by a factor of nearly 2 or more in each case

In order to avoid overfitting by ensuring that none of the weights or coefficients become large, one can add a penalty factor to the cost function that penalizes large weights. This process is known as *Ridge regression or L2-norm regularization*. The penalty includes the sum of the squared magnitude of all weights, $||b||^2 = b^2{}_1 + b^2{}_2 + \ldots$, that is, *L2-Norm* of $b_m$, where m is the number of attributes.

The cost function is modified as shown:

$$J_{RIDGE} = \Sigma N_i = (y_i - B^T x_i)^2 + \lambda |b|^2 \tag{5.14}$$

By following the usual steps and switching to a matrix form shown earlier, one arrives at the new solution for the weights as:

$$B = (\lambda I + X^T X)^{-1} X^T Y \tag{5.15}$$

where $I$ is the identity matrix and $\lambda$ is a penalty factor $> 0$.

Compare this to the standard solution that can be derived from Eq. (5.11):

$$B = (X^T X)^{-1} Y^T X. \tag{5.16}$$

When L2-norm is implemented, the fit looks much improved and the coefficients are $b_0 = 1.5$ and $b_1 = 4.02$ which are closer to the underlying function (shown in Fig. 5.14).

*Ridge regression* tends to push all the weights toward zero in order to minimize the cost function.

There is another scenario of overfitting, that involves selecting more than the optimal number of independent variables (and, thus, weights). Not all features exert the same influence on the predicted outcome, some features have more influence than others. However, as more features are included in a model, the training error continues to reduce. But the test error may spiral out of control and result in another form of overfitting.

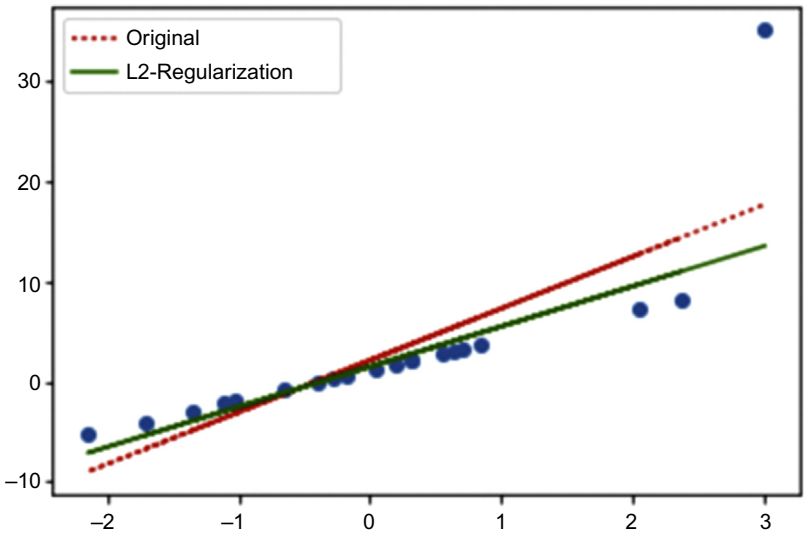



**FIGURE 5.14**
L2-regularization.

*Lasso regression* or *L1-norm regularization* addresses this concern where the goal is to select the optimal number of features. The formulation is similar to Ridge, but using the L1-norm: $||b|| = |b_1| + |b_2| + \cdots$

$$J_{LASSO} = \Sigma N_i = (y_i - B^T x_i)^2 + \lambda |b| \tag{5.17}$$

Sometimes the features are correlated with each other—when this happens the $X^TX$ matrix becomes singular and the closed form solution cannot be used. However, this should not be any concern if gradient descent is used to approximate the solution. Gradient descent is a technique that allows us to incrementally evaluate the coefficients, **b** for which the error **J** is minimized, when obtaining a closed form derivative for **dJ/db** is not possible. The idea is to take small computational steps toward the direction of minimum **J** by choosing the fastest path. In this case we can however get the derivative of the error function in closed form, which turns out to be:

$$\partial J / \partial b = -2X^T Y + 2X^T Xb + \lambda \text{sign}(b) = 0 \tag{5.18}$$

here $\text{sign}(b) = 1$ if $b > 0$, $-1$ if $b < 0$, 0 if $b = 0$.

Typically for practical situations, such closed form derivatives are seldom available and the gradient descent formulation is the alternative. The final gradient descent setup for both types of regularization are:

LASSO or L1:

$$b_{i+1} = b_i - \eta X^T(Y - \hat{Y}) + \lambda \times \text{sign}(b_i) \tag{5.19}$$

RIDGE or L2:

$$b_{i+1} = b_i - \eta X^T(Y - \hat{Y}) + \lambda \times b_i \tag{5.20}$$

where $\eta$ is the same *learning rate* parameter that crops up in neural networks (Chapter 4: Classification) and deep learning (Chapter 10). The number of steps, i is determined by the rate of convergence of the solution or by other stopping criteria. In RapidMiner, Rigde regression is implemented by providing a non-zero penalty factor in the box for 'ridge' under parameters—see Fig. 5.4.

## 5.2 LOGISTIC REGRESSION

From a historical perspective, there are two main classes of data science techniques: those that evolved (Cramer, 2002) from statistics (such as regression) and those that emerged from a blend of statistics, computer science, and mathematics (such as classification trees). Logistic regression arose in the mid-twentieth century as a result of the simultaneous development of the concept of the *logit* in the field of biometrics and the advent of the digital computer, which made computations of such terms easy. So, to understand logistic regression, the logit concept first needs to be explored. The chart in Fig. 5.15, adapted from data shown in Cramer (2002) shows the evolving trend from initial acceptance of the logit concept in the mid-1950s to the surge in references to this concept toward the latter half of the twentieth century. The chart is an indicator of how important logistic regression has become over the last few decades in a variety of scientific and business applications.

To introduce the logit, a simple example will be used. Recall that linear regression is the process of finding a function to fit the x's that vary linearly with y with the objective of being able to use the function as a model for prediction. The key assumption here are that both the predictor and target variables are continuous, as seen in the chart in Fig. 5.16. Intuitively, one can state that when $x$ increases, $y$ increases along the slope of the line. For example, advertisement spend and sales.

What happens if the target variable is not continuous? Suppose the target variable is the response to advertisement campaigns—if more than a threshold number of customers buy for example, then the response is considered

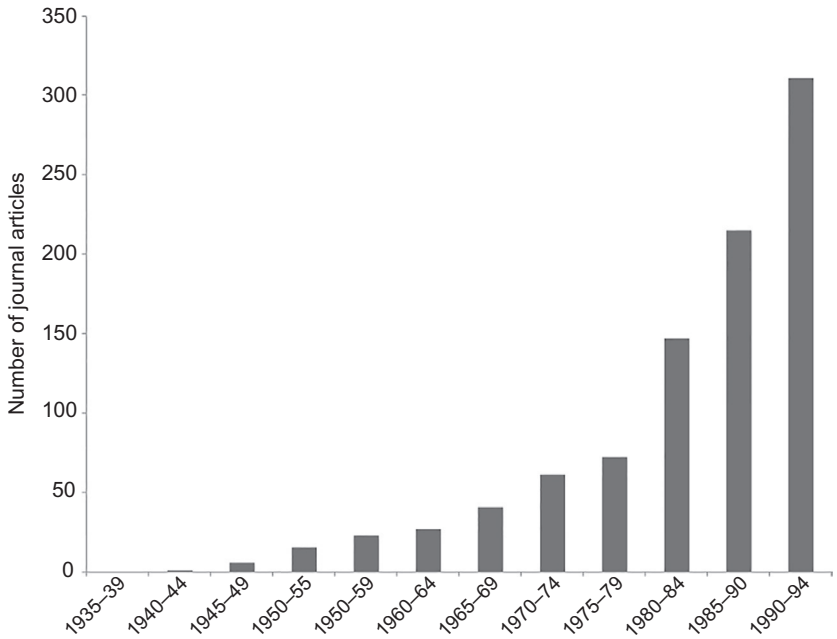
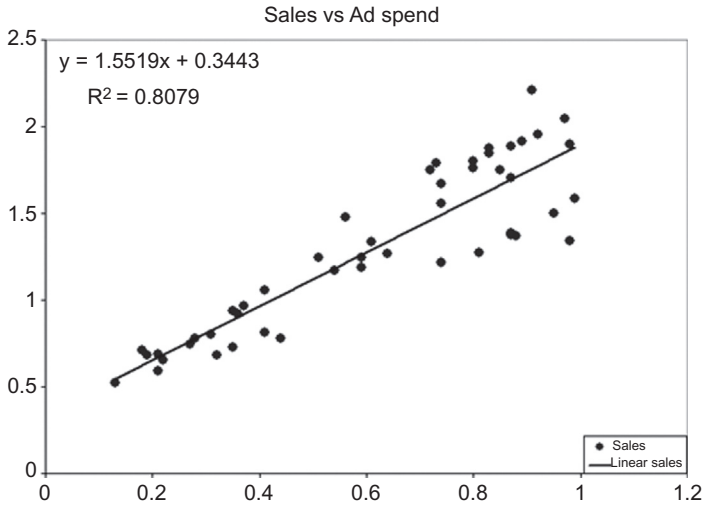



**FIGURE 5.15**
Growth of logistic regression applications in statistical research.




Linear Regression Model. We can make an intuitive assessment that increase in *Ad spend* also increases *Sales*. Using the straight line, we may also be able to predict.

**FIGURE 5.16**
Goal of linear regression.

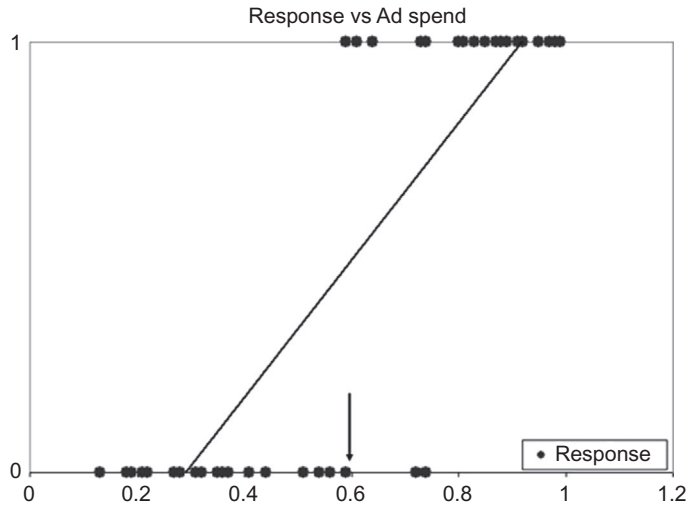



**FIGURE 5.17**
Fitting a linear model to discrete data.

to be 1; if not the response is 0. In this case, the target ($y$) variable is discrete (as in Fig. 5.17); the straight line is no longer a fit as seen in the chart. Although one can still estimate—approximately—that when $x$ (advertising spend) increases, $y$ (response or no response to a mailing campaign) also increases, there is no gradual transition; the $y$ value abruptly jumps from one binary outcome to the other. Thus, the straight line is a poor fit for this data.

On the other hand, take a look at the S-shaped curve in Fig. 5.18. This is certainly a better fit for the data shown. If the equation to this "sigmoid" curve is known, then it can be used as effectively as the straight line was used in the case of linear regression.

Logistic regression is, thus, the process of obtaining an appropriate nonlinear curve to fit the data when the target variable is discrete. How is the sigmoid curve obtained? How does it relate to the predictors?

### 5.2.1 How It Works

The dependent variable, $y$, will be re-examined. If it is binomial, that is, it can take on only two values (yes/no, pass/fail, respond/does not respond, and so on), then y can be coded to assume only two values: 1 or 0.

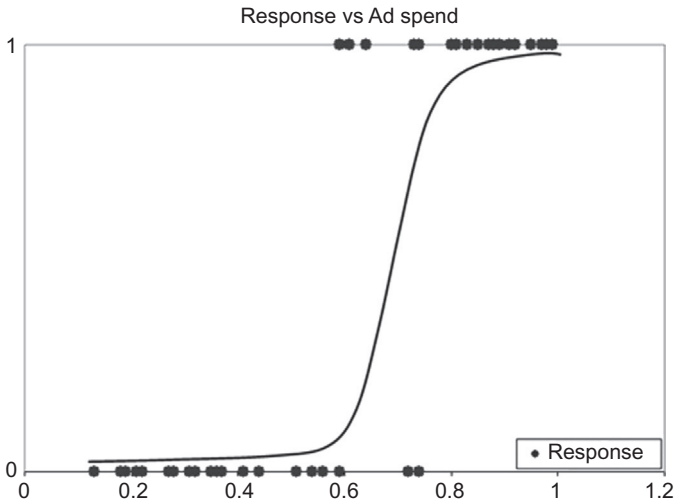



**FIGURE 5.18**
Fitting a nonlinear curve to discrete data.

The challenge is to find an equation that functionally connects the predictors, $x$, to the outcome y where y can only take on two values: 0 or 1. However, the predictors themselves may have no restrictions: they could be continuous or categorical. Therefore, the functional range of these unrestricted predictors is likely to also be unrestricted (between $-\infty$ to $+\infty$). To overcome this problem, one must map the continuous function to a discrete function. This is what the logit helps to achieve.

### *How Does Logistic Regression Find the Sigmoid Curve?*

As observed in Eq. (5.1), a straight line can be depicted by only two parameters: the slope ($b_1$) and the intercept ($b_0$). The way in which $x$'s and $y$ are related to each other can be easily specified by $b_0$ and $b_1$. However, an S-shaped curve is a much more complex shape and representing it parametrically is not as straightforward. So how does one find the mathematical parameters to relate the $x$'s to the $y$?

It turns out that if the target variable $y$ is transformed to the *logarithm of the odds of* $y$, then the *transformed* target variable is *linearly* related to the predictors, $x$. In most cases where the use of logistic regression is needed; the y is usually a yes/no type of response. This is usually interpreted as the

probability of an event happening ($y = 1$) or not happening ($y = 0$). This can be deconstructed as:

- If y is an event (response, pass/fail, etc.),
- and $p$ is the probability of the event happening ($y = 1$),
- then $(1 - p)$ is the probability of the event *not* happening ($y = 0$),
- and $p/(1 - p)$ are the *odds* of the event happening.

The logarithm of the odds, $\log(p/(1 - p))$ is linear in the predictors, $X$, and $\log(p/(1 - p))$ or the log of the odds is called the *logit function*.

The logit can be expressed as a linear function of the predictors $X$, similar to the linear regression model shown in Eq. (5.1) as:

$$\text{logit} = \log p/(1 - p) = b_0 x + b_1 \tag{5.21}$$

For a more general case, involving multiple independent variables, $x$, there is:

$$\text{logit} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \tag{5.22}$$

The logit can take any value from $-\infty$ to $+\infty$. For each row of predictors in a dataset, the logit an now be computed. From the logit, it is easy to then compute the probability of the response y (occurring or not occurring) as seen below:

$$p = e^{\text{logit}}/(1 + e^{\text{logit}}) \tag{5.23}$$

The logistic regression model from Eq. (5.22) ultimately delivers the probability of y occurring (i.e., $y = 1$), given specific value(s) of $x$ via Eq. (5.23). In that context, a good definition of logistic regression is that it is a mathematical modeling approach in which a best-fitting, yet least-restrictive model is selected to describe the relationship between several independent explanatory variables and a dependent binomial response variable. It is least-restrictive because the right side of Eq. (5.22) can assume any value from $-\infty$ to $+\infty$. Cramer (2002) provides more background on the history of the logit function.

From the data given the $x$'s are known and using Eqs. (5.22) and (5.23) one can compute the $p$ for any given $x$. But to do that, the coefficients first need to be determined, $b$, in Eq. (5.22). How is this done? Assume that one starts out with a trial of values for $b$. Given a training data sample, one can compute the quantity:

$$p^y \times (1-p)^{(1-y)}$$

where $y$ is the original outcome variable (which can take on 0 or 1) and $p$ is the probability estimated by the logit equation [Eq. (5.23)]. For a specific training sample, if the actual outcome was $y = 0$ and the model estimate of $p$ was high (say 0.9), that is, the model was wrong, then this quantity reduces

to 0.1. If the model estimate of probability was low (say 0.1), that is, the model was good, then this quantity increases to 0.9. Therefore, this quantity, which is a simplified form of a *likelihood* function, is *maximized for good estimates* and *minimized for poor estimates*. If one computes a summation of the simplified likelihood function across *all* the training data samples, then a high value indicates a good model (or good fit) and vice versa.

In reality, gradient descent or other nonlinear optimization techniques are used to search for the coefficients, $b$, with the objective of maximizing the likelihood of correct estimation (or $p^y \times (1 - p)^{(1 - y)}$, summed over all training samples). More sophisticated formulations of likelihood estimators are used in practice (Eliason, 1993).

### A Simple but Tragic Example

In the 1912 shipwreck of the HMS Titanic, hundreds of people perished as the ship struck an iceberg in the North Atlantic (Hinde, 1998). When the data is dispassionately analyzed, a couple of basic patterns emerge. 75% of the women and 63% of the first-class passengers survived. If a passenger was a woman and if she traveled first class, her probability of survival was 97%! The scatterplot in Fig. 5.19 depicts this in an easy to understand way (see the bottom left cluster).

A data science competition used the information from this event and challenged analysts to develop an algorithm that could classify the passenger list into survivors and non-survivors.[3] The training dataset provided will be used as an example to demonstrate how logistic regression could be employed to make this prediction and also to interpret the coefficients from the model.

Table 5.3 shows part of a reduced dataset consisting only of three variables: travel class of the passenger (pclass = 1st, 2nd, or 3rd), sex of the passenger (0 for male and 1 for female), and the label variable "survived" (true or false). When a logistic regression model is fit to this data consisting of 891 samples, the following equation is obtained for predicting the class *survived = false* (the details of a generic setup process will be described in the next section):

$$\text{logit} = -0.6503 - 2.6417 \times \text{sex} + 0.9595 \times \text{pclass} \tag{5.24}$$

Comparing this to Eq. (5.22), $b_0 = -0.6503$, $b_1 = -2.6417$, and $b_2 = 0.9595$. How are these coefficients interpreted? In order to do this, Eq. (5.23) will need to be recalled,

[3] http://www.kaggle.com/c/titanic-gettingStarted.

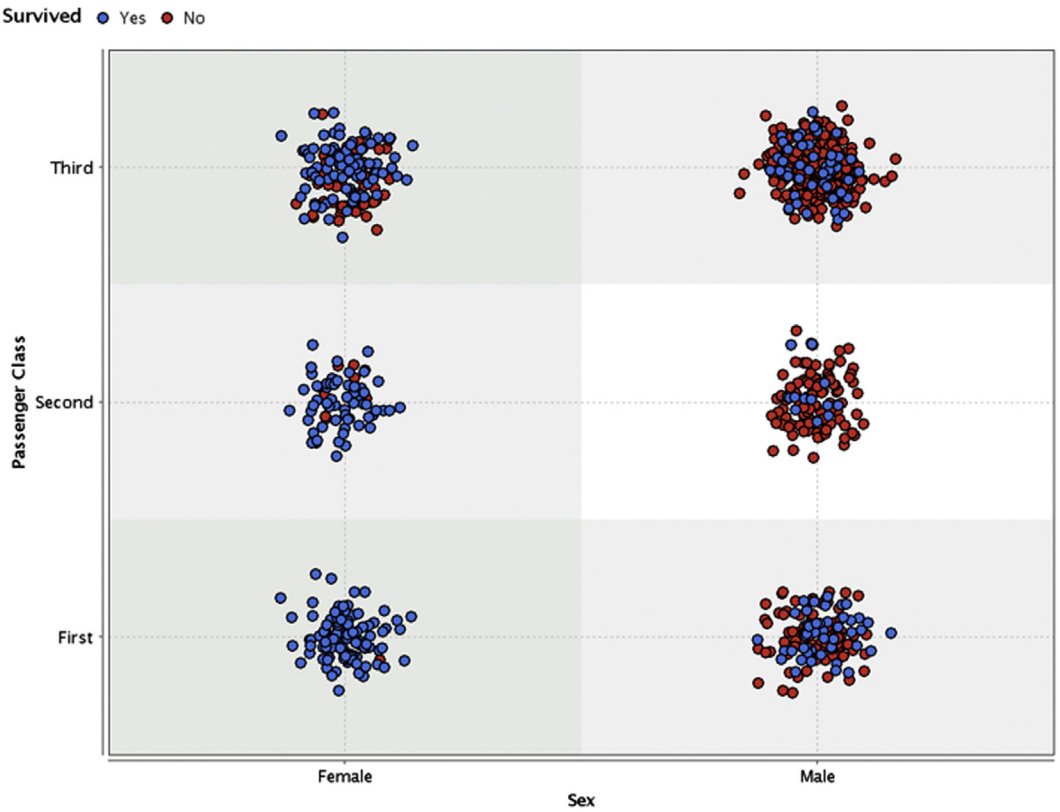



**FIGURE 5.19**
Probability of survival in the Titanic wreck based on gender and travel class.

$$p = e^{\text{logit}}/[1 + e^{\text{logit}}]$$

which indicates that as logit increases to a large positive quantity, the probability that the passenger did not survive (survived = false) approaches 1. More specifically, when logit approaches $-\infty$, $p$ approaches 0 and when logit approaches $+\infty$, $p$ approaches 1. The negative coefficient on variable *sex* indicates that this probability reduces for females (*sex* = 1) and the positive coefficient on variable $p$ indicates that the probability of not surviving (*survived* = false) increases the higher the *number* of the travel class. This verifies the intuitive understanding that was provided by the scatterplot shown in Fig. 5.19.

The odds form of the logistic regression model can also be examined, which is given as:

$$\text{odds (survived = false)} = e^{-0.6503} \times 2.6103^{\text{pclass}} \times 0.0712^{\text{sex}} \tag{5.25}$$

**Table 5.3** Portion of the Dataset From the Titanic Example

| pclass | Sex | Survived? |
|---|---|---|
| 3.0 | Male | 0.0 |
| 1.0 | Female | 1.0 |
| 3.0 | Female | 1.0 |
| 1.0 | Female | 1.0 |
| 3.0 | Male | 0.0 |
| 3.0 | Male | 0.0 |
| 1.0 | Male | 0.0 |
| 3.0 | Male | 0.0 |
| 3.0 | Female | 1.0 |
| 2.0 | Female | 1.0 |
| 3.0 | Female | 1.0 |
| 1.0 | Female | 1.0 |
| 3.0 | Male | 0.0 |
| 3.0 | Male | 0.0 |
| 3.0 | Female | 0.0 |
| 2.0 | Female | 1.0 |
| 3.0 | Male | 0.0 |

Recall that logit is simply given by log(odds) and essentially the same equation as Eq. (5.24) is used. A key fact to observe is that a positive coefficient in the logit model translates into a coefficient higher than 1 in the odds model (the number 2.6103 in the above equation is $e^{0.9595}$ and 0.0712 is $e^{-2.6417}$) and a negative coefficient in the logit model translates into coefficients smaller than 1 in the odds model. Again, it is clear that the odds of not surviving increases with travel class and reduces with gender being female.

An *odds ratio analysis* will reveal the value of computing the results in this format. Consider a female passenger ($sex = 1$). The survivability for this passenger could be calculated if she was in 1st class ($pclass = 1$) versus if she was in 2nd class as an odds ratio:

$$\begin{aligned}&\text{odds (survived = false 2nd class)}/\text{odds(survived = false 1st class)}\\&\quad = 2.6103^2/2.6103^1 = 2.6103\end{aligned} \tag{5.26}$$

Based on the Titanic dataset, the odds that a female passenger would not survive if she was in 2nd class increases by a factor of 2.6 compared to her odds if she was in 1st class. Similarly, the odds that a female passenger would not survive increases by nearly seven times if she was in 3rd class! In the next section, the mechanics of logistic regression are discussed as well as the process of implementing a simple analysis using RapidMiner.

**Table 5.4** A Sample From the Loan Default Dataset

| [Busage] | [Daysdelq] | [Default] |
|---|---|---|
| 87.0 | 2.0 | N |
| 89.0 | 2.0 | N |
| 90.0 | 2.0 | N |
| 90.0 | 2.0 | N |
| 101.0 | 2.0 | N |
| 110.0 | 2.0 | N |
| 115.0 | 2.0 | N |
| 115.0 | 2.0 | N |
| 115.0 | 2.0 | N |
| 117.0 | 2.0 | N |

### 5.2.2 How to Implement

The data used come from an example for a credit scoring exercise. The objective is to predict DEFAULT (Y or N) based on two predictors: loan age (business age) and number of days of delinquency. There are 100 samples Table 5.4.

#### *Step 1: Data Preparation*

Load the spreadsheet into RapidMiner. Remember to set the DEFAULT column as Label. Split the data into training and test samples using the *Split Validation* operator.

#### *Step 2: Modeling Operator and Parameters*

Add the *Logistic Regression* operator in the training subprocess of the *Split Validation* operator. Add the *Apply Model* operator in the testing subprocess of the *Split Validation* operator. Just use default parameter values. Add the *Performance (Binominal)* evaluation operator in the testing subprocess of *Split Validation* operator. Check the Accuracy, AUC, Precision, and Recall boxes in the parameter settings. Connect all ports as shown in Fig. 5.20.

#### *Step 3: Execution and Interpretation*

Run the model and view the results. In particular, check for the kernel model, which shows the coefficients for the two predictors and the intercept. The bias (offset) is $-1.820$ and the coefficients are given by: w[BUSAGE] = 0.592 and w[DAYSDELQ] = 2.045. Also check the confusion matrix for Accuracy, Precision, and Recall and finally view the ROC curves and check the area under the curve or AUC. Chapter 8, Model Evaluation, provides further details on these important performance measures.

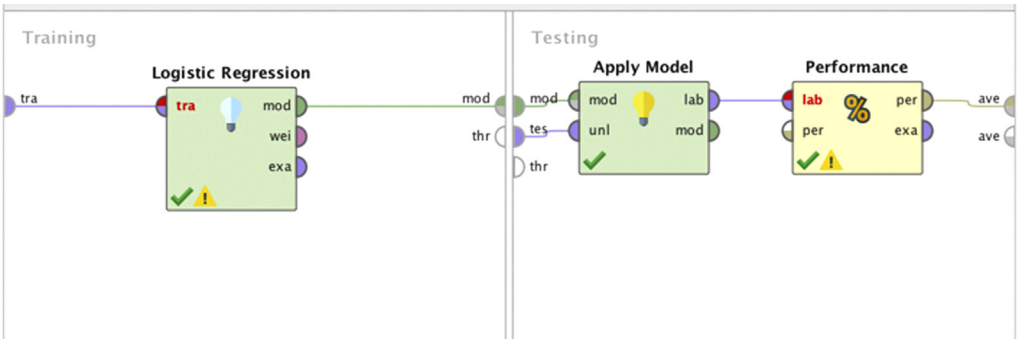



**FIGURE 5.20**
Setting up the RapidMiner process for a logistic regression model.

**accuracy: 83.33%**

| | true N | true Y | class precision |
|---|---|---|---|
| pred. N | 21 | 3 | 87.50% |
| pred. Y | 2 | 4 | 66.67% |
| class recall | 91.30% | 57.14% | |

**FIGURE 5.21**
Confusion matrix for the testing sample.

The accuracy of the model based on the 30% testing sample is 83%. (The ROC curves have an AUC of 0.863). The next step would be to review the kernel model and prepare for deploying this model. Are these numbers acceptable? In particular, pay attention to the class recall (bottom row of the confusion matrix in Fig. 5.21). The model is quite accurate in predicting if someone is NOT a defaulter (91.3%), however, its performance when it comes to identifying if someone IS a defaulter is questionable. For most predictive applications, the cost of wrong class predictions is not uniform. That is, a false positive (in the case of identifying someone as a defaulter, when they are not) may be less expensive than a false negative (in the case of identifying someone as a non-defaulter, when they actually are). There are ways to weight the cost of misclassification, and RapidMiner allows this through the use of the *MetaCost* operator.

### *Step 4: Using MetaCost*

Nest the *Logistic Regression* operator inside a *MetaCost* operator to improve class recall. The *MetaCost* operator is now placed inside *Split Validation* operator. Configure the *MetaCost* operator as shown in Fig. 5.22. Notice that false

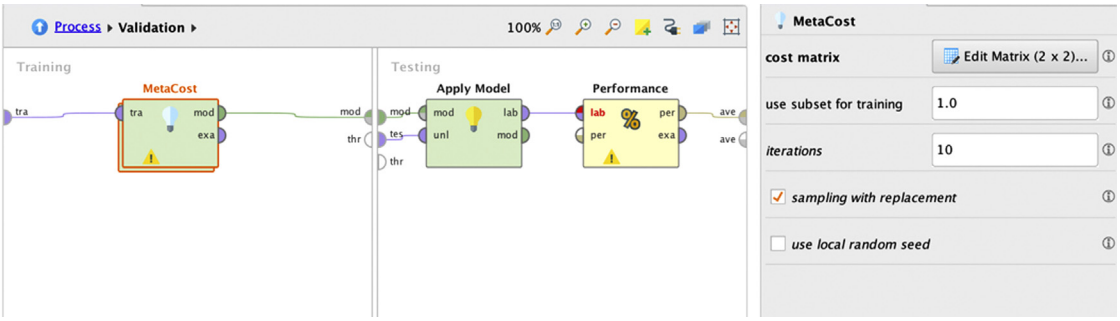



**FIGURE 5.22**
Configuring the MetaCost operator to improve class recall performance.

accuracy: 83.33%

| | true N | true Y | class precision |
|---|---|---|---|
| pred. N | 20 | 2 | 90.91% |
| pred. Y | 3 | 5 | 62.50% |
| class recall | 86.96% | 71.43% | |

**FIGURE 5.23**
Improved classification performance with the usage of MetaCost operator.

negatives have twice the cost of false positives. The actual values of these costs can be further optimized using an optimization loop—optimization is discussed for general cases in Chapter 15, Getting Started with RapidMiner.

When this process is run, the new confusion matrix that results is shown in Fig. 5.23. The overall accuracy has not changed much. Note that while the class recall for the Default = Yes class has increased from 57% to 71%, this has come at the price of reducing the class recall for Default = No from 91% to 87%. Is this acceptable? Again, the answer to this comes from examining the actual business costs. More details about interpreting the confusion matrix and evaluating the performance of classification models is provided in Chapter 8, Model Evaluation.

### *Step 5: Applying the Model to an Unseen Dataset*

In RapidMiner, logistic regression is calculated by creating a support vector machine (SVM) with a modified loss function (Fig. 5.24). SVMs were introduced in Chapter 4 on Classification. That's the reason why support vectors are seen at all.

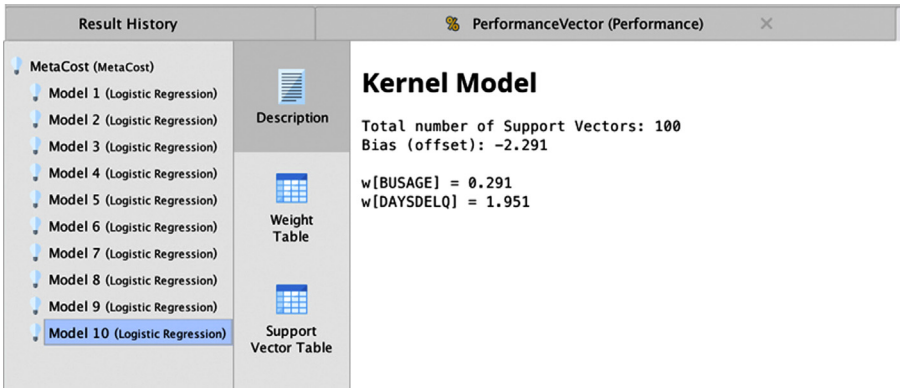



**FIGURE 5.24**
Default logistic regression model in RapidMiner is based on SVM. *SVM*, Support vector machine.

### 5.2.3 Summary Points

- Logistic regression can be considered equivalent to using linear regression for situations where, when the target (or dependent) variable is discrete, that is, not continuous. In principle, the response variable or label is binomial. A binomial response variable has two categories: Yes/No, Accept/Not Accept, Default/Not Default, and so on. Logistic regression is ideally suited for business analytics applications where the target variable is a binary decision (fail/pass, response/no response, etc.).
- Logistic regression comes from the concept of the logit. The logit is the logarithm of the odds of the response, y, expressed as a function of independent or predictor variables, x, and a constant term. That is, for example, log (odds of $y = \text{Yes}) = b_1 x + b_0$.
- This logit gives the odds of the Yes event, however if one wants probabilities, the transformed equation will need to be used:
  $p(y = \text{“Yes”}) = 1/(1 + e^{(-b1_x - b_0)})$
- The predictors can be either numerical or categorical for standard logistic regression. However, in RapidMiner, the predictors can only be numerical, because it is based on the SVM formulation.

## 5.3 CONCLUSION

This chapter explored two of the most common function-fitting methods. Function-fitting methods are one of the earliest data science techniques based on the concept of supervised learning. The multiple linear regression model

works with numeric predictors and a numeric label and is, thus, one of the go-to methods for numeric prediction tasks. The logistic regression model works with numeric or categorical predictors and a categorical (typically, binomial) label. How a simple linear regression model is developed was explained using the methods of calculus and how feature selection impacts the coefficients of a model was discussed. It was explained, how the significance of the coefficients can be interpreted using the *t*-stat and *p*-values and finally several checkpoints that one must follow to build good quality models were laid down. Logistic regression was then introduced, by comparing the similar structural nature of the two function-fitting methods. How a sigmoid curve can better fit predictors to a binomial label was discussed and the concept of logit was introduced, which enables the transformation of a complex function into a more recognizable linear form. How the coefficients of logistic regression can be interpreted was discussed as well as how to measure and improve the classification performance of the model.

## References


Black, K. (2008). Multiple regression analysis. In K. Black (Ed.), *Business statistics* (pp. 601–610). Hoboken, NJ: John Wiley and Sons.

Cramer, J. (2002). *The origins of logistic regression* (pp. 1–15). Tinbergen Institute Discussion Paper.

Eliason, S. (1993). *Maximum likelihood estimation: Logic and practice*. Newbury Park, CA: Sage Publishers.

Fletcher, R. A. (1963). A rapidly convergent descent method for minimization. *The Computer Journal*, *6*(2), 163–168.

Galton, F. (1888). Co-relations and their measurement, chiefly from anthropometric data. *Proceedings of the Royal Society of London*, *45*(273-279), 135–145.

Harrison, D. A. (1978). Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, *5*(1), 81–102.

Hinde, P. (1998). Encyclopedia titanica. Retrieved from <http://www.encyclopedia-titanica.org/>.

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, *11*(2), 431–441.

Stigler, S. (1999). *Statistics on the table: The history of statistical concepts and methods*. Cambridge: Harvard University Press.