CHAPTER 14

Feature Selection

In this chapter the focus will be on an important component of dataset preparation for data science: feature selection. An overused rubric in data science circles is that 80% of the analysis effort is spent on data cleaning and preparation and only 20% is typically spent on modeling. In light of this it may seem strange that this book has devoted more than a dozen chapters to modeling techniques and only a couple to data preparation! However, data cleansing and preparation are things that are better learned through experience and not so much from a book. That said, it is essential to be conversant with the many techniques that are available for these important early process steps. In this chapter the focus will not be on data cleaning, as it was partially covered in Chapter 2, Data Science Process, but rather on reducing a dataset to its essential characteristics or features. This process is known by various terms: feature selection, dimension reduction, variable screening, key parameter identification, attribute weighting or regularization. Regularization was briefly covered in Chapter 5 as applied to multiple linear regression. There it was introduced as a process that helps to reduce overfitting, which is essentially what feature selection techniques implicitly achieve. [Technically, there is a subtle difference between dimension reduction and feature selection. Dimension reduction methods—such as principal component analysis (PCA), discussed in Section 14.2—combine or merge actual attributes in order to reduce the number of attributes of a raw dataset. Feature selection methods work more like filters that eliminate some attributes.]

First a brief introduction to feature selection along with the need for this preprocessing step is given. There are fundamentally two types of feature selection processes: *filter type* and *wrapper type*. Filter approaches work by selecting only those attributes that rank among the top in meeting certain stated criteria (Blum & Langley, 1997; Yu & Liu, 2003). Wrapper approaches work by iteratively selecting, via a feedback loop, only those attributes that improve the performance of an algorithm (Kohavi & John, 1997). Among the filter-type methods, one can further classify based on the data types: numeric versus nominal. The most common wrapper-type methods are the ones

Data Science. DOI: https://doi.org/10.1016/B978-0-12-814761-0.00014-9

associated with multiple regression: stepwise regression, forward selection, and backward elimination. A few numeric filter-type methods will be explored: PCA, which is strictly speaking a dimension reduction method; information gain–based filtering; and one categorical filter-type method: chi-square-based filtering.

14.1 CLASSIFYING FEATURE SELECTION METHODS

IDENTIFYING WHAT MATTERS

Feature selection in data science refers to the process of identifying the few most important variables or attributes that are essential to a model for an accurate prediction. In today's world of big data and high-speed computing, one might be forgiven for asking, why bother? What is the reason to filter any attributes when the computing horsepower exists? For example, some argue that it is redundant trying to fit a model to data; rather one should simply use a fast brute-force approach to sift through data to identify meaningful *correlations* and make decisions based on this (Bollier, 2010).

However, models are still useful for many reasons. Models can improve decision making and help advance knowledge. Blindly relying on correlations to predict future states also has flaws. The now popular "My TiVo thinks I'm gay" example (Zaslow, 2002), illustrated how the TiVo recommendation engine, which works on *large data* and correlations, resulted in a humorous mismatch for a customer. As long as the use of models is needed, feature selection will be an important step in the process. Feature selection serves a couple of purposes: it optimizes the performance of the data science algorithm and it makes it easier for the analyst to interpret the outcome of the modeling. It does this by reducing the number of attributes or features that one must contend with.

There are two powerful technical motivations for incorporating feature selection in the data science process. First, a dataset may contain highly correlated attributes, such as the number of items sold, and the revenue earned by the sales of the item. Typically, there is no new information gained by including both of these attributes. Additionally, in the case of multiple regression–type models, if two or more of the independent variables (or predictors) are correlated, then the estimates of coefficients in a regression model tend to be unstable or counter intuitive. This is the *multicollinearity* discussed in Section 5.1. In the case of algorithms like naïve Bayesian classifiers, the attributes need to be independent of each other. Further, the speed of algorithms is typically a function of the number of attributes. So, by using only one among the correlated attributes the performance is improved.

Second, a dataset may also contain redundant information that does not directly impact the predictions: as an extreme example, a customer ID number has no bearing on the amount of revenue earned from the customer. Such attributes may be filtered out by the analyst before the modeling process begins. However, not all attribute relationships are that clearly known in advance. In such cases, one must resort to computational methods to detect and eliminate attributes that add no new information. The key here is to include attributes that have a strong correlation with the predicted or dependent variable.

So, to summarize, feature selection is needed to *remove* independent variables that may be strongly correlated to one another, and to *keep* independent variables that may be strongly correlated to the predicted or dependent variable.

Feature selection methods can be applied before the modeling process starts and, thus, unimportant attributes will get filtered out, or feature selection methods can be applied iteratively within the flow of the data science process. Depending on the logic, there are two feature selection schemes: filter schemes or wrapper schemes. The filter scheme does not require any learning algorithm, whereas the wrapper type is optimized for a particular learning algorithm. In other words, the filter scheme can be considered unsupervised and the wrapper scheme can be considered a supervised feature selection method. The filter model is commonly used:

1. When the number of features or attributes is really large
2. When computational expense is a criterion

The chart in Fig. 14.1 summarizes a high-level taxonomy of feature selection methods, some of which will be explored in the following sections, as indicated. This is not meant to be a comprehensive taxonomy, but simply a

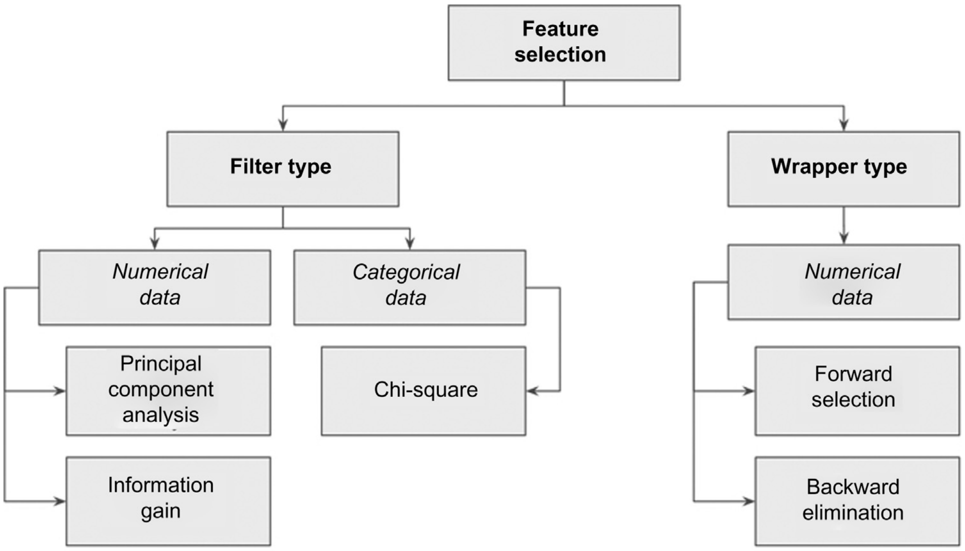


FIGURE 14.1
Taxonomy of common feature selection methods and the sections in this chapter that discuss them.

useful depiction of the techniques commonly employed in data science and described in this chapter.

14.2 PRINCIPAL COMPONENT ANALYSIS

To start with a conceptual introduction to PCA will be provided before the mathematical basis behind the computation is shown. Then a demonstration of how to apply PCA to a sample dataset using RapidMiner will be given.

Assume that there is a dataset with m attributes. These could be for example, commodity prices, weekly sales figures, number of hours spent by assembly line workers, etc.; in short, any business parameter that can have an impact on a performance that is captured by a label or target variable. The question that PCA helps to answer is fundamentally this: Which of these m attributes explain a significant amount of variation contained within the dataset? PCA essentially helps to apply an 80/20 rule: Can a small subset of attributes explain 80% or more of the variation in the data? This sort of variable screening or *feature selection* will make it easy to apply other data science techniques and also make the job of interpreting the results easier.

PCA captures the attributes that contain the greatest amount of variability in the dataset. It does this by transforming the existing variables into a set of principal components or *new variables* that have the following properties (van der Maaten, Postma, & van den Herik, 2009):

1. They are uncorrelated with each other.
2. They cumulatively contain/explain a large amount of variance within the data.
3. They can be related back to the original variables via weighting factors.

The original variables with very low weighting factors in their principal components are effectively removed from the dataset. The conceptual schematic in Fig. 14.2 illustrates how PCA can help in reducing data dimensions with a hypothetical dataset of m variables.

14.2.1 How It Works

The key task is computing the principal components, z_m, which have the properties that were described. Consider the case of just two variables: v_1 and v_2. When the variables are visualized using a scatterplot, something like the one shown in Fig. 14.3 would be observed.

As can be seen, v_1 and v_2 are correlated. But v_1 and v_2 could be transformed into two new variables z_1 and z_2, which meet the guidelines for principal components, by a simple linear transformation. As seen in the chart, this

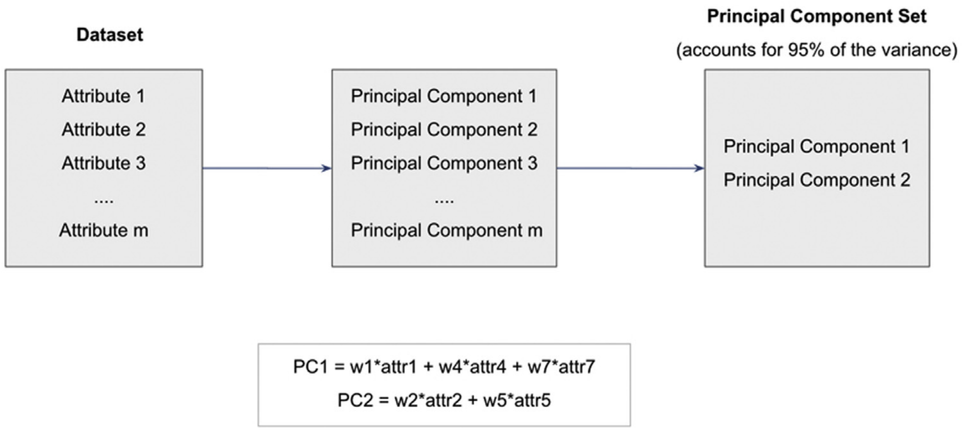


FIGURE 14.2
A conceptual framework illustrating the effectiveness of using PCA for feature selection. The final dataset includes only PC 1 and PC 2. *PCA*, Principal Component Analysis.

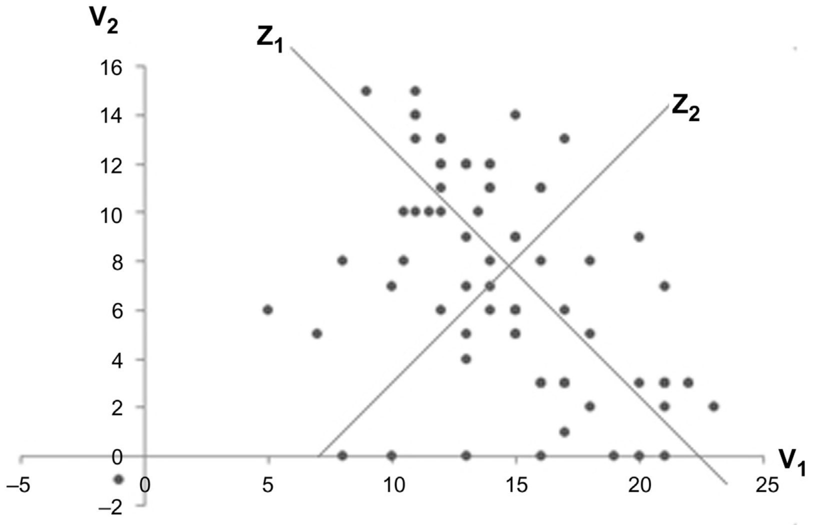


FIGURE 14.3
Transforming variables to a new basis is at the core of PCA. *PCA*, Principal Component Analysis.

amounts to plotting the points along two new axes: z_1 and z_2. Axis z_1 contains the maximum variability, and one can rightly conclude that z_1 explains a significant majority of the variation present in the data and is the first principal component. z_2, by virtue of being orthogonal to z_1, contains the next highest amount of variability. Between z_1 and z_2, 100% of the total variability in the data can be accounted for (in this case of two variables).

Furthermore, z_1 and z_2 are uncorrelated. As the number of variables, v_m, increases it's possible that only the first few principal components are sufficient to express all the data variances. The principal components, z_m, are expressed as a linear combination of the underlying variables, v_m:

$$z_m = \sum w_i \times x_i \tag{14.1}$$

When this logic is extended to more than two variables, the challenge is to find the transformed set of principal components using the original variables. This is easily accomplished by performing an *eigenvalue analysis* of the *covariance matrix* of the original attributes.[1] The eigenvector associated with the largest eigenvalue is the first principal component; the eigenvector associated with the second largest eigenvalue is the second principal component and so on. The covariance explains how two variables vary with respect to their corresponding mean values—if both variables tend to stay on the same side of their respective means, the covariance would be positive, if not it would be negative. (In statistics, covariance is also used in the calculation of correlation coefficient.)

$$\mathrm{Cov}_{ij} = E\left[V_i V_j\right] - E[V_i]E\left[V_j\right] \tag{14.2}$$

where expected value $E[v] = v_k\ P(v = v_k)$. For the eigenvalue analysis, a matrix of such covariances between all pairs of variables v_m is created. For more details behind the eigenvalue analysis refer to standard textbooks on matrix methods or linear algebra (Yu & Liu, 2003).

14.2.2 How to Implement

In this section, with the use of a publicly available dataset,[2] RapidMiner will be used to perform the PCA. Furthermore, for illustrative reasons, non-standardized or non-normalized data will be used. In the next part the data will be standardized and why it may be important to sometimes do so will be explained.

The dataset includes information on ratings and nutritional information on 77 breakfast cereals. There are a total of 16 variables, including 13 numerical parameters (Table 14.1). The objective is to reduce this set of 13 numerical predictors to a much smaller list using PCA.

[1] Let A be an $n\ X\ n$ matrix and x be an $n\ X\ 1$ vector. Then the solution to the vector equation $[A][x] = \lambda[x]$, where λ is a scalar number, involves finding those values of λ for which this equation is satisfied. The values of λ are called eigenvalues and the corresponding solutions for x ($x \neq 0$) are called eigenvectors.

[2] https://www.kaggle.com/jeandsantos/breakfast-cereals-data-analysis-and-clustering/data.

Table 14.1 Breakfast Cereals Dataset for Dimension Reduction Using PCA

Name	mfr	Type	Calories	Protein	Fat	Sodium	Fiber	Carbo	Sugars	Potass	Vitamins	Shelf	Weight	Cups	Rating
100%_Bran	N	C	70	4	1	130	10	5	6	280	25	3	1	0.33	68.402973
100%_Natural_Bran	Q	C	120	3	5	15	2	8	8	135	0	3	1	1	33.983679
All-Bran	K	C	70	4	1	260	9	7	5	320	25	3	1	0.33	59.425505
All-Bran_with_Extra_Fiber	K	C	50	4	0	140	14	8	0	330	25	3	1	0.5	93.704912
Almond_Delight	R	C	110	2	2	200	1	14	8	-1	25	3	1	0.75	34.384843
Apple_Cinnamon_Cheerios	G	C	110	2	2	180	1.5	10.5	10	70	25	1	1	0.75	29.509541
Apple_Jacks	K	C	110	2	0	125	1	11	14	30	25	2	1	1	33.174094
Basic_4	G	C	130	3	2	210	2	18	8	100	25	3	1.33	0.75	37.038562

Step 1: Data Preparation

Remove the non-numeric parameters such as Cereal name, Manufacturer, and Type (hot or cold), as PCA can only work with numeric attributes. These are the first three columns in Table 14.1. (In RapidMiner, these can be converted into ID attributes if needed for reference later. This can be done during the import of the dataset into RapidMiner during the next step if needed; in this case these variables will simply be removed. The *Select Attributes* operator may also be used following the *Read Excel* operator to remove these variables.) Read the Excel file into RapidMiner: this can be done using the standard *Read Excel* operator as described in earlier sections.

Step 2: PCA Operator

Type in the keyword PCA in the operator search field and drag and drop the *PCA* operator into the main process window. Connect the output of Read Excel into the ports of the *PCA* operator.

The three available parameter settings for dimensionality reduction are *none, keep variance*, and *fixed number*. Here use *keep variance* and leave the *variance threshold* at the default value of 0.95% or 95% (see Fig. 14.4). The variance threshold selects only those attributes that collectively account for or explain 95% (or any other value set by user) of the total variance in the data. Connect all output ports from the *PCA* operator to the results ports.

Step 3: Execution and Interpretation

By running the analysis as configured above, RapidMiner will output several tabs in the results panel (Fig. 14.5). By clicking on the PCA tab, three PCA related tabs will be seen—Eigenvalues, Eigenvectors, and Cumulative Variance Plot.

Using Eigenvalues, information can be obtained about the contribution to the data variance coming from each principal component individually and cumulatively.

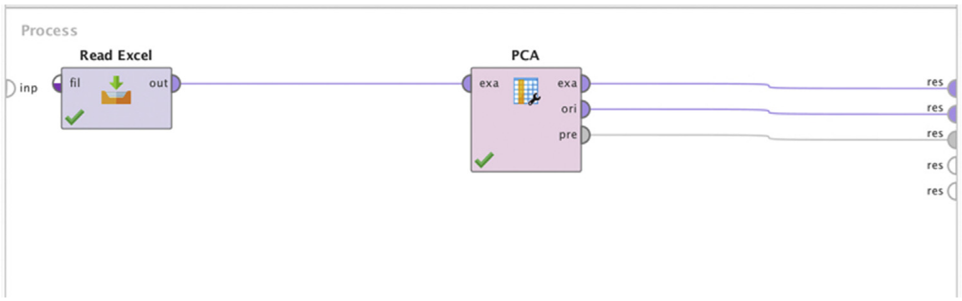
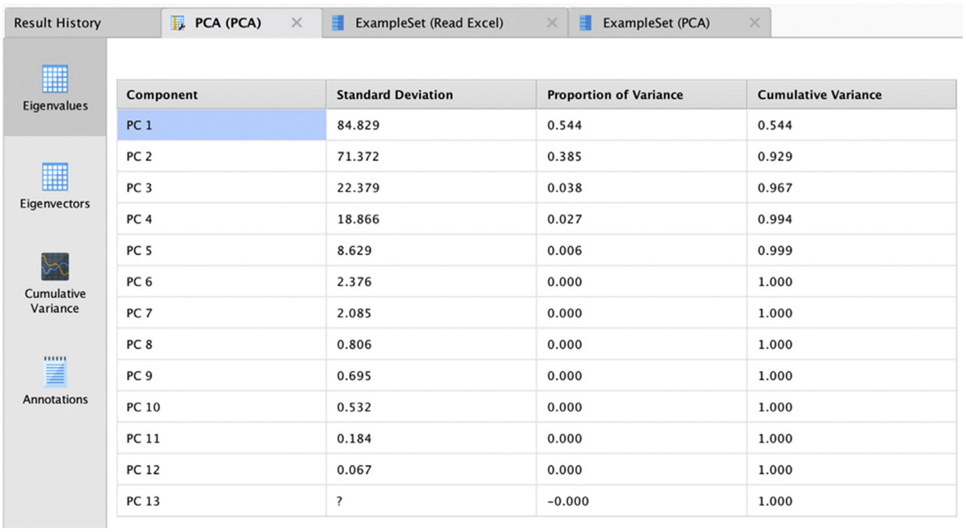


FIGURE 14.4
Configuring the PCA operator. *PCA*, Principal Component Analysis.



Component	Standard Deviation	Proportion of Variance	Cumulative Variance
PC 1	84.829	0.544	0.544
PC 2	71.372	0.385	0.929
PC 3	22.379	0.038	0.967
PC 4	18.866	0.027	0.994
PC 5	8.629	0.006	0.999
PC 6	2.376	0.000	1.000
PC 7	2.085	0.000	1.000
PC 8	0.806	0.000	1.000
PC 9	0.695	0.000	1.000
PC 10	0.532	0.000	1.000
PC 11	0.184	0.000	1.000
PC 12	0.067	0.000	1.000
PC 13	?	-0.000	1.000

FIGURE 14.5
Output from PCA. *PCA*, Principal Component Analysis.

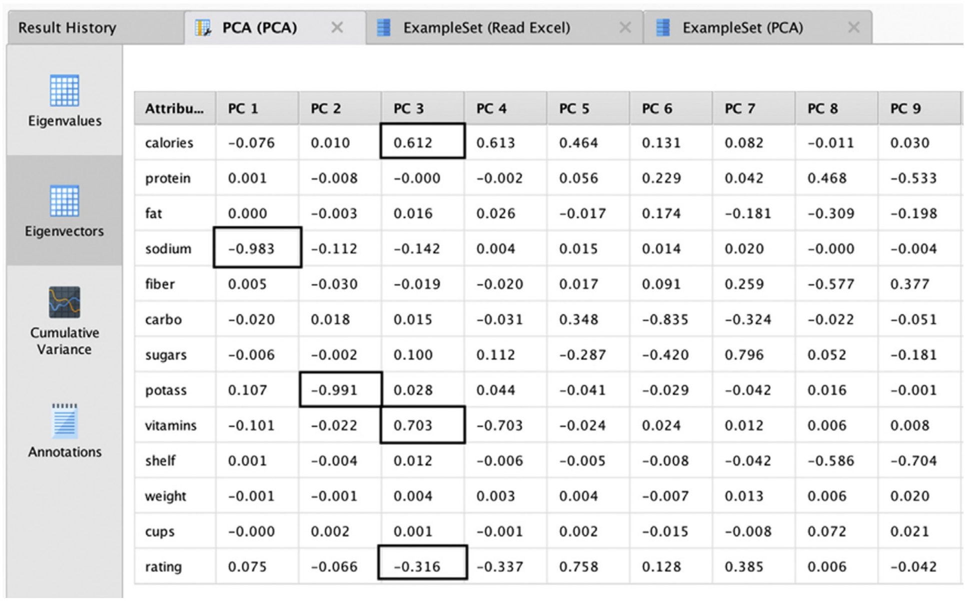
If, for example, our variance threshold is 95%, then PC 1, PC 2, and PC 3 are the only principal components that need to be considered because they are sufficient to explain nearly 97% of the variance. PC 1 contributes to a majority of this variance, about 54%.

One can then deep dive into these three components and identify how they are linearly related to the actual or real parameters from the dataset. At this point only those real parameters can be considered that have significant weight contribution to the each of the first three PCs. These will ultimately form the subset of reduced parameters for further predictive modeling.

The key question is how does one select the real variables based on this information? RapidMiner allows the eigenvectors (weighting factors) to be sorted for each PC and one can decide to choose the two to three highest (absolute) valued weighting factors for PCs 1–3. As seen from Fig. 14.6, the highlighted real attributes have been chosen—calories, sodium, potassium, vitamins, and rating—to form the reduced dataset. This selection was done by simply identifying the top three attributes from each principal component.[3]

For this example, PCA reduces the number of attributes from 13 to 5, a more than 50% reduction in the number of attributes that any model would need

[3] More commonly, only the top three principal components are directly selected for building subsequent models. This route was taken here to explain how PCA, which is a dimension reduction method, can be applied for feature selection.



Attribu...	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6	PC 7	PC 8	PC 9
calories	−0.076	0.010	$\boxed{0.612}$	0.613	0.464	0.131	0.082	−0.011	0.030
protein	0.001	−0.008	−0.000	−0.002	0.056	0.229	0.042	0.468	−0.533
fat	0.000	−0.003	0.016	0.026	−0.017	0.174	−0.181	−0.309	−0.198
sodium	$\boxed{-0.983}$	−0.112	−0.142	0.004	0.015	0.014	0.020	−0.000	−0.004
fiber	0.005	−0.030	−0.019	−0.020	0.017	0.091	0.259	−0.577	0.377
carbo	−0.020	0.018	0.015	−0.031	0.348	−0.835	−0.324	−0.022	−0.051
sugars	−0.006	−0.002	0.100	0.112	−0.287	−0.420	0.796	0.052	−0.181
potass	0.107	$\boxed{-0.991}$	0.028	0.044	−0.041	−0.029	−0.042	0.016	−0.001
vitamins	−0.101	−0.022	$\boxed{0.703}$	−0.703	−0.024	0.024	0.012	0.006	0.008
shelf	0.001	−0.004	0.012	−0.006	−0.005	−0.008	−0.042	−0.586	−0.704
weight	−0.001	−0.001	0.004	0.003	0.004	−0.007	0.013	0.006	0.020
cups	−0.000	0.002	0.001	−0.001	0.002	−0.015	−0.008	0.072	0.021
rating	0.075	−0.066	$\boxed{-0.316}$	−0.337	0.758	0.128	0.385	0.006	−0.042

FIGURE 14.6
Selecting the reduced set of attributes using the Eigenvectors tab from the PCA operator. *PCA*, Principal Component Analysis.

to realistically consider. One can imagine the improvement in performance as one deals with the larger datasets that PCA enables. In practice, PCA is an effective and widely used tool for dimension reduction, particularly when all the attributes are numeric. It works for a variety of real-world applications, but it should not be blindly applied for variable screening. For most practical situations, domain knowledge should be used in addition to PCA analysis before eliminating any of the variables. Here are some observations that explain some of the risks to consider while using PCA.

1. *The results of a PCA must be evaluated in the context of the data*. If the data is extremely noisy, then PCA may end up suggesting that the noisiest variables are the most significant because they account for most of the variation! An analogy would be the total sound energy in a rock concert. If the crowd noise drowns out some of the high-frequency vocals or notes, PCA might suggest that the most significant contribution to the total energy comes from the crowd—and it will be right! But this does not add any clear value if one is attempting to distinguish which musical instruments are influencing the harmonics, for example.
2. *Adding uncorrelated data does not always help. Neither does adding data that may be correlated, but irrelevant*. When more parameters are added to the

dataset, and if these parameters happen to be random noise, effectively the same situation as the first point applies. On the other hand, caution also has to be exercised and spurious correlations have to looked out for. As an extreme example, it may so happen that there is a correlation between the number of hours worked in a garment factory and pork prices (an unrelated commodity) within a certain period of time. Clearly this correlation is probably pure coincidence. Such correlations again can muddy the results of a PCA. Care must be taken to winnow the dataset to include variables that make business sense and are not subjected to many random fluctuations before applying a technique like PCA.

3. *PCA is very sensitive to scaling effects in the data*. If the data in the example is closely examined, it will be observed that the top attributes that PCA helped identify as the most important ones also have the widest range (and standard deviation) in their values. For example, potassium ranges from −1 to 330 and sodium ranges from 1 to 320. Comparatively, most of the other factors range in the single or low double digits. As expected, these factors dominate PCA results because they contribute to the maximum variance in the data. What if there was another factor such as sales volume, which would potentially range in the millions (of dollars or boxes), that were to be considered for a modeling exercise? Clearly it would mask the effects of any other attribute.

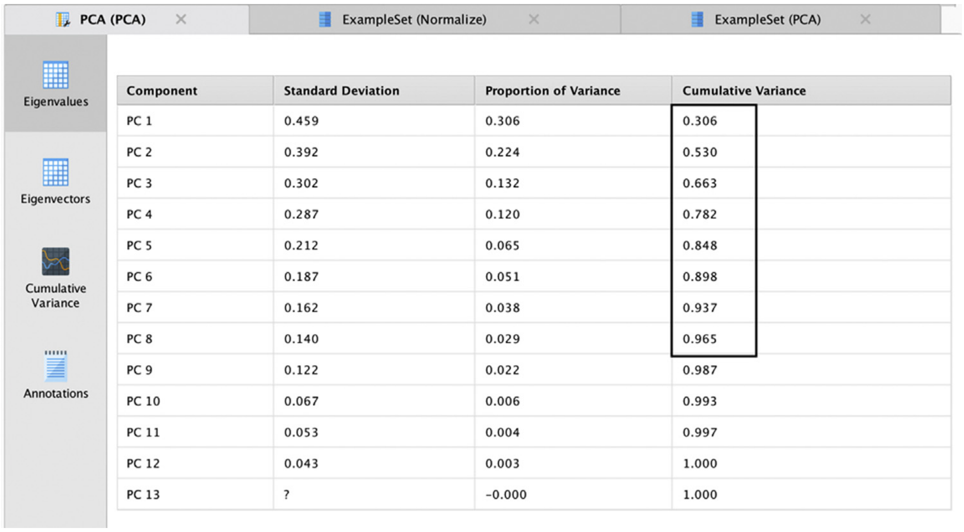
To minimize scaling effects, one can range normalize the data (using for example, the *Normalize* operator). When this data transformation is applied, all the attributes are reduced to a range between 0 and 1 and scale effects will not matter anymore. But what happens to the PCA results?

As Fig. 14.7 shows, eight PCs are now needed to account for the same 95% total variance. As an exercise, use the eigenvectors to filter out the attributes that are included in these eight PCs and it would be observed that (applying the top three rule for each PC as before), none of the attributes would be eliminated!

This leads to the next section on feature selection methods that are not scale sensitive and also work with non-numerical datasets, which were two of the limitations with PCA.

14.3 INFORMATION THEORY-BASED FILTERING

In Chapter 4, Classification, the concepts of information gain and gain ratio were encountered. Recall that both of these methods involve comparing the *information exchanged* between a given attribute and the target or label

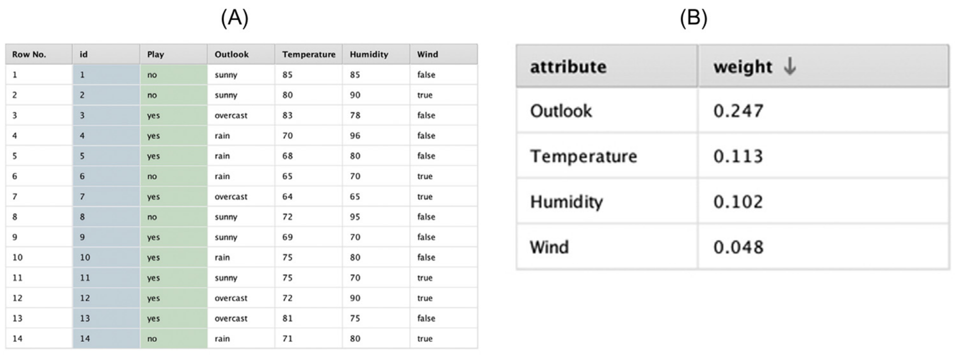


Component	Standard Deviation	Proportion of Variance	Cumulative Variance
PC 1	0.459	0.306	0.306
PC 2	0.392	0.224	0.530
PC 3	0.302	0.132	0.663
PC 4	0.287	0.120	0.782
PC 5	0.212	0.065	0.848
PC 6	0.187	0.051	0.898
PC 7	0.162	0.038	0.937
PC 8	0.140	0.029	0.965
PC 9	0.122	0.022	0.987
PC 10	0.067	0.006	0.993
PC 11	0.053	0.004	0.997
PC 12	0.043	0.003	1.000
PC 13	?	-0.000	1.000

FIGURE 14.7
Interpreting RapidMiner output for principal component analysis.

attribute (Peng, Long, & Ding, 2005). As discussed in Section 14.1, the key to feature selection is to include attributes that have a strong correlation with the predicted or dependent variable. With these techniques, one can rank attributes based on the amount of information gain and then select only those that meet or exceed some (arbitrarily) chosen threshold or simply select the top k (again, arbitrarily chosen) features.

Recall the golf example discussed first in Chapter 4, Classification. The data is presented here again for convenience in Fig. 14.8A. When the information gain calculation methodology that was discussed in Chapter 4, Classification, is applied to compute information gain for all attributes (see Table 4.2), the feature ranking in Fig. 14.8B will be reached, in terms of their respective influence on the target variable Play. This can be easily done using the *Weight by Information Gain* operator in RapidMiner. The output looks almost identical to the one shown in Table 4.2, except for the slight differences in the information gain values for Temperature and Humidity. The reason is that for that dataset, the temperature and humidity had to be converted into nominal values before computing the gains. In this case, the numeric attributes are used as they are. So, it is important to pay attention to the discretization of the attributes before filtering. Use of information gain feature selection is also restricted to cases where the label is nominal. For fully numeric datasets, where the label variable is also numeric, PCA or correlation-based filtering methods are commonly used.


(A)

Row No.	id	Play	Outlook	Temperature	Humidity	Wind
1	1	no	sunny	85	85	false
2	2	no	sunny	80	90	true
3	3	yes	overcast	83	78	false
4	4	yes	rain	70	96	false
5	5	yes	rain	68	80	false
6	6	no	rain	65	70	true
7	7	yes	overcast	64	65	true
8	8	no	sunny	72	95	false
9	9	yes	sunny	69	70	false
10	10	yes	rain	75	80	false
11	11	yes	sunny	75	70	true
12	12	yes	overcast	72	90	true
13	13	yes	overcast	81	75	false
14	14	no	rain	71	80	true

(B)

attribute	weight ↓
Outlook	0.247
Temperature	0.113
Humidity	0.102
Wind	0.048

FIGURE 14.8
(A) Revisiting the golf example for feature selection. (B) Results of information gain—based feature selection.

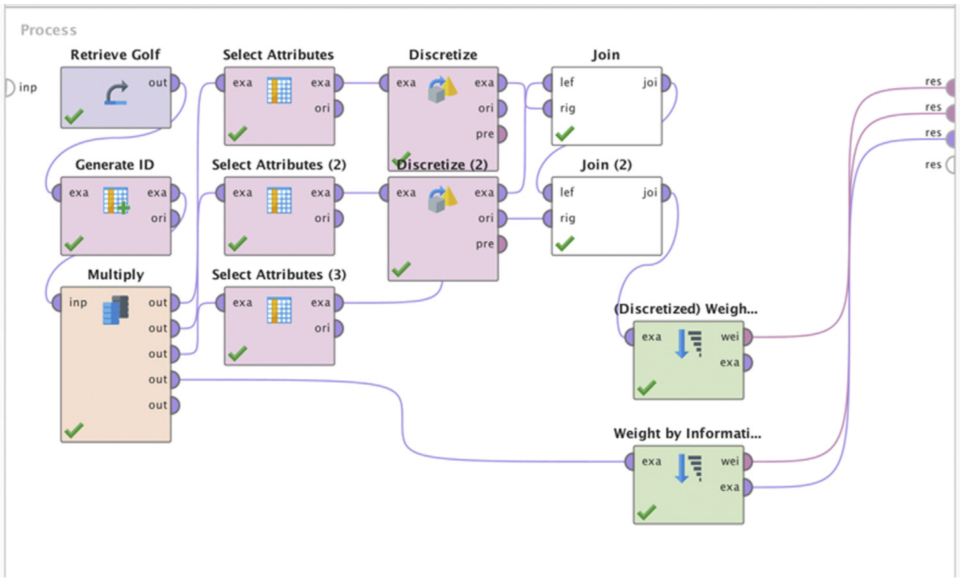


FIGURE 14.9
Process to discretize the numeric Golf dataset before running information gain—based feature selection.

Fig. 14.9 describes a process that uses the sample Golf dataset available in RapidMiner. The various steps in the process convert numeric attributes, Temperature and Humidity, into nominal ones. In the final step, the *Weight by Information Gain* operator is applied to both the original data and the converted dataset in order to show the difference between the gain computed using different data types. The main point to observe is that the gain computation depends not only on the data types, but also on how the nominal

Table 14.2 Results of Information Gain Feature Selection

Attribute	Info Gain Weight (Not Discretized)	Info Gain Weight (Discretized)
Outlook	0.247	0.247
Temperature	0.113	0.029
Humidity	0.102	0.104
Wind	0.048	0.048

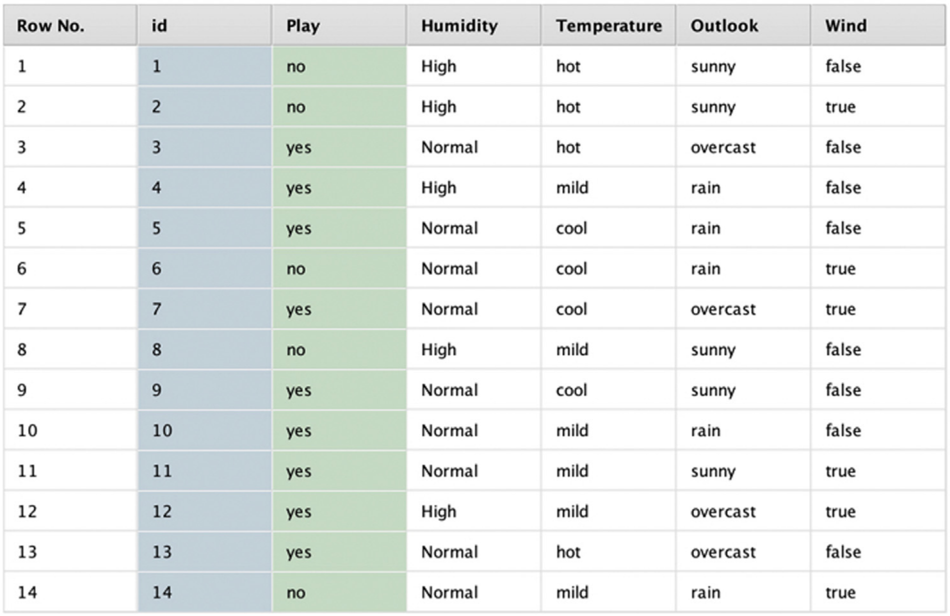
data is discretized. For example, one gets slightly different gain values (see Table 14.2) if Humidity is divided into three bands (high, medium, and low) as opposed to only two bands (high and low). These variants can be tested easily using the process described. In conclusion, the top-ranked attributes are selected. In this case, they would be Outlook and Temperature if one chose the non-discretized version, and Outlook and Humidity in the discretized version.

14.4 CHI-SQUARE-BASED FILTERING

In many cases the datasets may consist of only categorical (or nominal) attributes. In this case, what is a good way to distinguish between high influence attributes and low or no influence attributes?

A classic example of this scenario is the gender selection bias. Suppose one has data about the purchase of a big-ticket item like a car or a house. Can the influence of gender on purchase decisions be verified? Are men or women the primary decision makers when it comes to purchasing big-ticket items? For example, is gender a factor in color preference of a car? Here attribute 1 would be gender and attribute 2 would be the color. A chi-square test would reveal if there is indeed a relationship between these two attributes. If there are several attributes and one wishes to rank the relative influence of each of these on the target attribute, the chi-square statistic can still be used.

Back to the golf example in Fig. 14.10—this time all numeric attributes have been converted into nominal ones. Chi-square analysis involves *counting* occurrences (number of sunny days or windy days) and *comparing* these variables to the target variable based on the frequencies of occurrences. The chi-square test checks if the frequencies of occurrences across any pair of attributes, such as Outlook = overcast and Play = yes, are correlated. In other words, for the given Outlook type, overcast, what is the probability that Play = yes (existence of a strong correlation)? The multiplication law of probabilities states that if event A happening is independent of event B, then the probabilities of A and B happening together is simply $p_A \times p_B$. The next step



Row No.	id	Play	Humidity	Temperature	Outlook	Wind
1	1	no	High	hot	sunny	false
2	2	no	High	hot	sunny	true
3	3	yes	Normal	hot	overcast	false
4	4	yes	High	mild	rain	false
5	5	yes	Normal	cool	rain	false
6	6	no	Normal	cool	rain	true
7	7	yes	Normal	cool	overcast	true
8	8	no	High	mild	sunny	false
9	9	yes	Normal	cool	sunny	false
10	10	yes	Normal	mild	rain	false
11	11	yes	Normal	mild	sunny	true
12	12	yes	High	mild	overcast	true
13	13	yes	Normal	hot	overcast	false
14	14	no	Normal	mild	rain	true

FIGURE 14.10
Converting the golf example set into nominal values for chi-square feature selection.

Table 14.3 Contingency Table of Observed Frequencies for Outlook and the Label Attribute, Play

Outlook	Sunny	Overcast	Rain	Total
Play = no	3	0	2	**5**
Play = yes	2	4	3	**9**
Total	**5**	**4**	**5**	**14**

is to convert this joint probability into an expected frequency, which is given by $p_A \times p_B \times N$, where N is the sum of all occurrences in the dataset.

For *each* attribute, a table of observed frequencies, such as the one shown in Table 14.3, is built. This is called a *contingency table*. The last column and row (the margins) with heading 'Totals' are simply the sums in the corresponding rows or columns as can be verified. Using the contingency table, a corresponding *expected frequency table* can be built using the expected frequency definition ($p_A \times p_B \times N$) from which the chi-square statistic is then computed by comparing the difference between the observed frequency and expected frequency for *each* attribute. The expected frequency table for *Outlook* is shown in Table 14.4.

Table 14.4 Expected Frequency Table

Outlook	Sunny	Overcast	Rain	Total
Play = no	1.785714	1.428571	1.785714	**5**
Play = yes	3.214286	2.571429	3.214286	**9**
Total	**5**	**4**	**5**	**14**

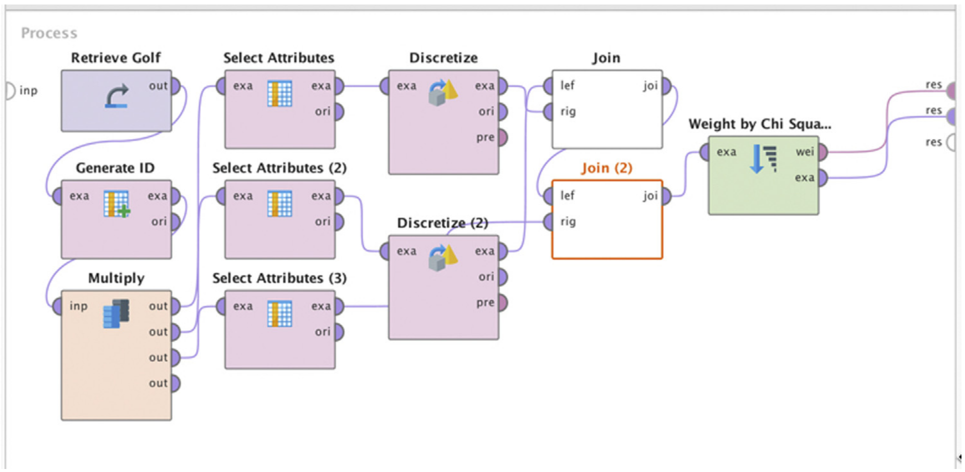


FIGURE 14.11
Process to rank attributes of the Golf dataset by the chi-square statistic.

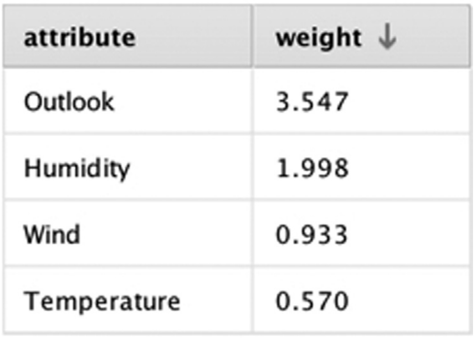
The expected frequency for the event [Play = no and Outlook = sunny] is calculated using the expected frequency formula: (5/14 × 5/14 × 14) = 1.785 and is entered in the first cell as shown. Similarly, the other expected frequencies are calculated. The formula for the chi-square statistic is the summation of the square of the differences between observed and expected frequencies, as given in Eq. (14.3):

$$\chi^2 = \sum\sum \frac{(f_o - f_e)^2}{f_e} \tag{14.3}$$

where f_o is the observed frequency and f_e is the expected frequency. The test of independence between any two parameters is done by checking if the observed chi-square is less than a critical value which depends on the confidence level chosen by the user (Black, 2007). In this case of feature weighting, all the observed chi-square values are simply gathered and used to rank the attributes. The ranking of attributes for the golf example is generated using the process described in Fig. 14.11 and is shown in the table of observed chi-square values in Fig. 14.14. Just like in information gain feature selection, most of the operators shown in the process are simply transforming the data into nominal values to generate it in the form shown in Fig. 14.10.

Compare the output of the chi-square ranking to the information gain—based ranking (for the nominalized or discretized attributes) and it will be evident that the ranking is identical (see Fig. 14.12).

Note that the Normalize weights option is sometimes also used, which is a range normalization onto the interval 0 to 1.



attribute	weight ↓
Outlook	3.547
Humidity	1.998
Wind	0.933
Temperature	0.570

FIGURE 14.12
Results of the attribute weighting by the chi-square method.

14.5 WRAPPER-TYPE FEATURE SELECTION

In this section of the chapter, wrapper scheme feature reduction methods will briefly be introduced using a linear regression example. As explained earlier, the wrapper approach iteratively chooses features to add or to remove from the current attribute pool based on whether the newly added or removed attribute improves the accuracy.

Wrapper-type methods originated from the need to reduce the number of attributes that are needed to build a high-quality regression model. A thorough way to build regression models is something called the "all possible regressions" search procedure. For example, with three attributes, $v1$, $v2$, and $v3$, one could build the different regression models in Table 14.5.

In general, if a dataset contains k different attributes, then conducting all possible regression searches implies that one builds $2^k - 1$ separate regression

Table 14.5 All Possible Regression Models With Three Attributes

Model	Independent Variables Used
1	*v*1 alone
2	*v*2 alone
3	*v*3 alone
4	*v*1 and *v*2 only
5	*v*1 and *v*3 only
6	*v*2 and *v*3 only
7	*v*1, *v*2, and *v*3 all together

models and picks the one that has the best performance. Clearly this is impractical.

A better way, from a computational resource consumption point of view, to do this search would be to start with one variable, say *v*1, and build a baseline model. Then add a second variable, say *v*2, and build a new model to compare with the baseline. If the performance of the new model, for example, the R^2 (see Chapter 5: Regression Methods), is better than that of the baseline, this model can be made to be the new baseline, add a third variable, *v*3, and proceed in a similar fashion. If, however, the addition of the second attribute, *v*2, did not improve the model significantly (over some arbitrarily prescribed level of improvement in performance), then a new attribute *v*3 can be chosen, and a new model built that includes *v*1 and *v*3. If this model is better than the model that included *v*1 and *v*2, proceed to the next step, where the next attribute *v*4 can be considered, and a model built that includes *v*1, *v*3, and *v*4. In this way, one steps forward selecting attributes one by one until the desired level of model performance is achieved. This process is called *forward selection*.[4]

A reverse of this process is where the baseline model with all the attributes is started, *v*1, *v*2, ..., *vk* and for the first iteration, one of the variables, *vj*, is removed and a new model is constructed. However, how does one select which *vj* to remove? Here, it is typical to start with a variable that has the lowest *t*-stat value, seen in the case study described below.[5] If the new model is better than the baseline, it becomes the new baseline and the search continues to remove variables with the lowest *t*-stat values until some stopping

[4] Forward selection is considered a "greedy" approach, and does not necessarily yield the globally optimum solution.
[5] RapidMiner typically tries removing attributes one after the other. Vice versa for forward selection: first it tries out all models having just one attribute. It selects the best, then adds another variable, again trying out every option.

criterion is met (usually if the model performance is not significantly improved over the previous iteration). This process is called *backward elimination*.

As observed, the variable selection process wraps around the modeling procedure, hence, the name for these classes of feature selection. A case study will now be examined, using data from the Boston Housing dataset first introduced in Chapter 5, Regression Methods, to demonstrate how to implement the backward elimination method using RapidMiner. Recall that the data consists of 13 predictors and 1 response variable. The predictors include physical characteristics of the house (such as the number of rooms, age, tax, and location) and neighborhood features (schools, industries, zoning), among others. The response variable is the median value (MEDV) of the house in thousands of dollars. These 13 independent attributes are considered to be predictors for the target or label attribute. The snapshot of the data table is shown again in Table 14.6 for continuity.

14.5.1 Backward Elimination

The goal here is to build a high-quality multiple regression model that includes as few attributes as possible, without compromising the predictive ability of the model.

The logic used by RapidMiner for applying these techniques is not linear, but a nested logic. The graphic in Fig. 14.13 explains how this nesting was used in setting up the training and testing of the *Linear Regression* operator for the analysis done in Chapter 5, Regression Methods, on the Boston Housing data. The arrow indicates that the training and testing process was nested within the *Split Validation* operator.

In order to apply a wrapper-style feature selection method such as backward elimination, the training and testing process will need to be tucked inside another subprocess, a learning process. The learning process is now nested inside the *Backward Elimination* operator. Therefore, a double nesting as schematically shown in Fig. 14.13 is now obtained. Next, the Backward Elimination operator can be configured in RapidMiner. Double clicking on the *Backward Elimination* operator opens up the learning process, which can now accept the *Split Validation* operator that has been used many times.

The *Backward Elimination* operator can now be filled in with the *Split Validation* operator and all the other operators and connections required to build a regression model. The process of setting these up is exactly the same as discussed in Chapter 5, Regression Methods, and, hence, is not repeated here. Now the configuration of the *Backward Elimination* operator will be examined. Here one can specify several parameters to enable feature

Table 14.6 Sample View of the Boston Housing Dataset

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	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	14.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	14.43	22.9
0.14455	14.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1

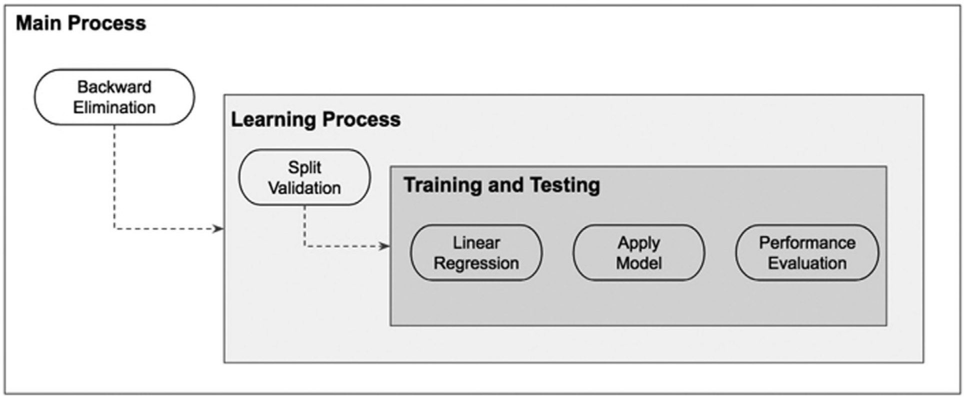


FIGURE 14.13
Wrapper function logic used by RapidMiner.

selection. The most important one is the *stopping behavior*. The choices are "with decrease," "with decrease of more than," and "with significant decrease." The first choice is highly parsimonious—a decrease from one iteration to the next will stop the process. But if the second choice is chosen, one now has to now indicate a "maximal relative decrease." In this example, a 10% decrease has been indicated. Finally, the third choice is very stringent and requires achieving some desired statistical significance by allowing one to specify an alpha level. But it has not been said by how much the performance parameter should decrease yet! This is specified deep inside the nesting: all the way at the *Performance* operator that was selected in the Testing window of the *Split Validation* operator. In this example, the performance criterion was squared correlation. For a complete description of all the other *Backward Elimination* parameters, the RapidMiner help can be consulted.

There is one more step that may be helpful to complete before running this model. Simply connecting the *Backward Elimination* operator ports to the output will not show the final regression model equation. To be able to see that, one needs to connect the example port of the *Backward Elimination* operator to another *Linear Regression* operator in the main process. The output of this operator will contain the model, which can be examined in the Results perspective. The top level of the final process is shown in Fig. 14.14.

Comparing the two regression equations (Fig. 14.15 and in Chapter 5: Regression Methods, see Fig. 5.6A) it is evident that nine attributes have been eliminated. Perhaps the 10% decrease was too aggressive. As it happens, the R^2 for the final model with only three attributes was only 0.678. If the stopping criterion was changed to a 5% decrease, one would end up with an

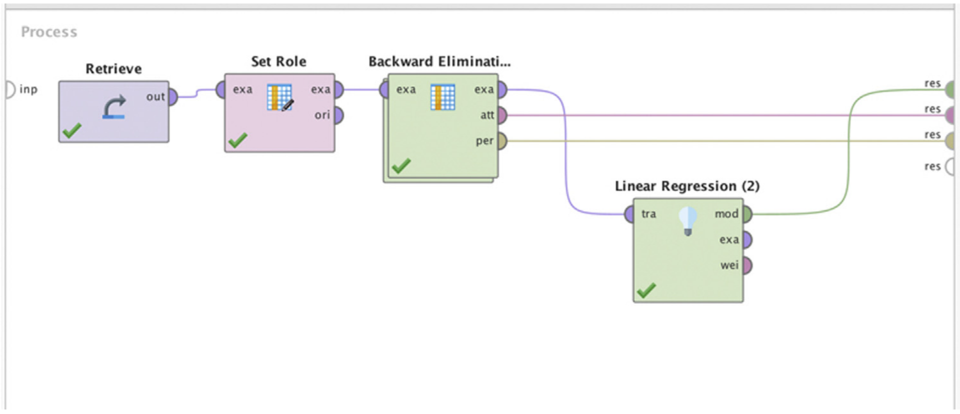
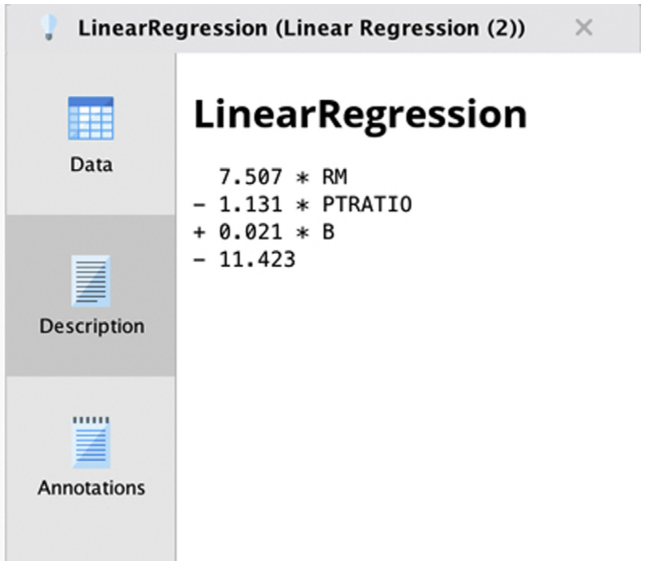


FIGURE 14.14
Final setup of the backward elimination wrapper process.

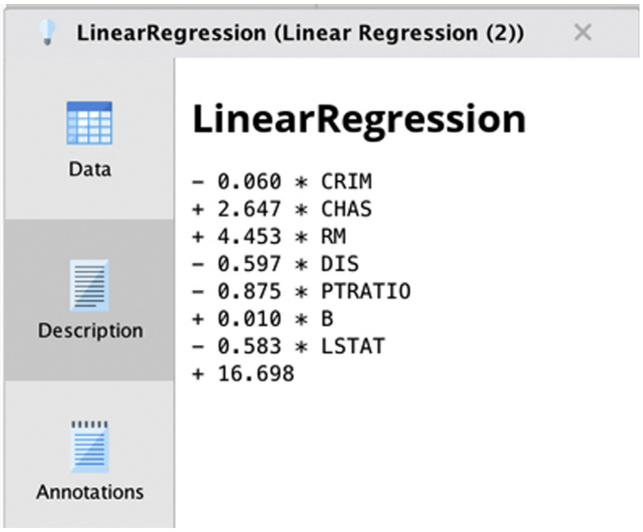


LinearRegression

```
  7.507 * RM
- 1.131 * PTRATIO
+ 0.021 * B
- 11.423
```

FIGURE 14.15
Aggressive feature selection. Maximal relative decrease = 10%.

R^2 of 0.812 and now have 8 of the 13 original attributes (Fig. 14.16). It is also evident that the regression coefficients for the two models are different as well. The final judgment on what is the right criterion and its level can only be made with experience with the dataset and of course, good domain knowledge.



LinearRegression

```
- 0.060 * CRIM
+ 2.647 * CHAS
+ 4.453 * RM
- 0.597 * DIS
- 0.875 * PTRATIO
+ 0.010 * B
- 0.583 * LSTAT
+ 16.698
```

FIGURE 14.16
A more permissive feature selection with backward elimination. Maximal relative decrease = 5%.

Each iteration using a regression model either removes or introduces a variable, which improves model performance. The iterations stop when a preset stopping criterion or no change in performance criterion (such as adjusted r^2 or RMS error) is reached. The inherent advantage of wrapper-type methods are that multicollinearity issues are automatically handled. However, no prior knowledge about the actual relationship between the variables will be gained. Applying forward selection is similar and is recommended as an exercise.

14.6 CONCLUSION

This chapter covered the basics of an important part of the overall data science paradigm: feature selection or dimension reduction. A central hypothesis among all the feature selection methods is that good feature selection results in attributes or features that are highly correlated with the class, yet uncorrelated with each other (Hall, 1999). A high-level classification of feature selection techniques were presented and each of them were explored in some detail. As stated at the beginning of this chapter, dimension reduction is best understood with real practice. To this end, it is recommended that one applies all the techniques described in this chapter on all the datasets provided. The same technique can yield quite different results based on the selection of analysis parameters. This is where data visualization can play an important role. Sometimes, examining a correlation plot between the various attributes, like in a scatterplot matrix, can provide valuable clues about which attributes are likely redundant and which ones can be strong predictors of the label variable. While there is usually no substitute for domain knowledge, sometimes data are simply too large or mechanisms are unknown. This is where feature selection can actually help.

References

Black, K. (2007). *Business statistics*. New York: John Wiley and Sons.

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*(1–2), 245–271.

Bollier, D. (2010). *The promise and perils of big data*. Washington, D.C.: The Aspen Institute.

Hall, M.A. (1999). *Correlation based feature selection for machine learning* (Ph.D. thesis). University of Waikato: New Zealand.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97* (1–2), 273–324.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information: Criteria of max-dependency, max-relevance and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(8), 1226–1238.

van der Maaten, L. J. P., Postma, E. O., & van den Herik, H. J. (2009). Dimensionality reduction: A comparative review. In: *Tilburg University technical report*. *TiCC-TR*.

Yu, L., Liu, H. (2003). Feature selection for high dimensional data: A fast correlation based filter solution. In: *Proceedings of the twentieth international conference on machine learning (ICML-2003)*. Washington, DC.

Zaslow, J. (December 4, 2002). Oh No! My TiVo thinks I'm gay. *Wall Street Journal*.