CHAPTER 13

Anomaly Detection

Anomaly detection is the process of finding outliers in a given dataset. Outliers are the data objects that stand out amongst other data objects and do not conform to the expected behavior in a dataset. Anomaly detection algorithms have broad applications in business, scientific, and security domains where isolating and acting on the results of outlier detection is critical. For identification of anomalies, algorithms discussed in previous chapters such as classification, regression, and clustering can be used. If the training dataset has objects with known anomalous outcomes, then any of the supervised data science algorithms can be used for anomaly detection. In addition to supervised algorithms, there are specialized (unsupervised) algorithms whose whole purpose is to detect outliers without the use of a labeled training dataset. In the context of unsupervised anomaly detection, algorithms can either measure distance from other data points or density around the neighborhood of the data point. Even clustering techniques can be leveraged for anomaly detection. The outlier usually forms a separate cluster from other clusters because they are far away from other data points. Some of the techniques discussed in previous chapters will be revisited in the context of outlier detection. Before discussing the algorithms, the term outlier or anomaly has to be defined and the reason such data points occur in a dataset will need to be understood.

13.1 CONCEPTS

An outlier is a data object that is markedly different from the other objects in a dataset. Hence, an outlier is always defined in the context of other objects in the dataset. A high-income individual may be an outlier in a middle-class neighborhood dataset, but not in the membership of a luxury vehicle ownership dataset. By nature of the occurrence, outliers are also rare and, hence, they stand out amongst other data points. For example, the majority of computer network traffic is legitimate, and the one malicious network attack would be the outlier.

Data Science. DOI: https://doi.org/10.1016/B978-0-12-814761-0.00013-7

13.1.1 Causes of Outliers

Outliers in the dataset can originate from either error in the data or from valid inherent variability in the data. It is important to understand the provenance of the outliers because it will guide what action, if any, should be performed on the identified outliers. However, pinpointing exactly what caused an outlier is a tedious task and it may be impossible to find the causes of outliers in the dataset. Here are some of the most common reasons why an outlier occurs in the dataset:

Data errors: Outliers may be part of the dataset because of measurement errors, human errors, or data collection errors. For example, in a dataset of human heights, a reading such as 1.70 cm is obviously an error and most likely was wrongly entered into the system. These data points are often ignored because they affect the conclusion of the data science task. Outlier detection here is used as a preprocessing step in algorithms such as regression and neural networks. Data errors due to human mistake could be either intentional introduction of error or unintentional error due to data entry error or significant bias.

Normal variance in the data: In a normal distribution, 99.7% of data points lie within three standard deviations from the mean. In other words, 0.26% or 1 in 370 data points lie outside of three standard deviations from the mean. By definition, they don't occur frequently and yet are a part of legitimate data. An individual earning a billion dollars in a year or someone who is more than 7 ft tall falls under the category of outlier in an income dataset or a human height dataset respectively. These outliers skew some of the descriptive statistics like the mean of the dataset. Regardless, they are legitimate data points in the dataset.

Data from other distribution classes: The number of daily page views for a customer-facing website from a user IP address usually range from one to several dozen. However, it is not unusual to find a few IP addresses reaching hundreds of thousands of page views in a day. This outlier could be an automated program from a computer (also called a bot) making the calls to scrape the content of the site or access one of the utilities of the site, either legitimately or maliciously. Even though they are an outlier, it is quite "normal" for bots to register thousands of page views to a website. All bot traffic falls under the distribution of a different class "traffic from programs" other than traffic from regular browsers that fall under the human user class.

Distributional assumptions: Outlier data points can originate from incorrect assumptions made on the data or distribution. For example, if the data measured is usage of a library in a school, then during term exams there will be an outlier because of a surge in the usage of the library. Similarly, there will be a surge in retail sales during the day after

Thanksgiving in the United States. An outlier in this case is expected and does not represent the data point of a typical measure.

Understanding why outliers occur will help determine what action to perform after outlier detection. In a few applications, the objective is to isolate and act on the outlier as can be seen in credit card transaction fraud monitoring. In this case, credit card transactions exhibiting different behavior from most normal transactions (such as high frequency, high amounts, or very large geographic separation between points of consecutive transactions) has to be isolated, alerted, and the owner of the credit card has to be contacted immediately to verify the authenticity of the transaction. In other cases, the outliers have to be filtered out because they may skew the final outcome. Here outlier detection is used as a preprocessing technique for other data science or analytical tasks. For example, ultra-high-income earners might need to be eliminated in order to generalize a country's income patterns. Here outliers are legitimate data points but are intentionally disregarded in order to generalize conclusions.

DETECTING CLICK FRAUD IN ONLINE ADVERTISING

The rise in online advertising has underwritten successful Internet business models and enterprises. Online advertisements make free Internet services, like web searches, news content, social networks, mobile application, and other services, viable. One of the key challenges in online advertisement is mitigating *click frauds*. Click fraud is a process where an automated program or a person imitates the action of a normal user clicking on an online advertisement, with the malicious intent of defrauding the advertiser, publisher, or advertisement network. Click fraud could be performed by contracting parties or third parties, like competitors trying to deplete advertisement budgets or to tarnish the reputation of the sites. Click fraud distorts the economics of advertising and poses a major challenge for all parties involved in online advertising (Haddadi, 2010). Detecting, eliminating, or discounting click fraud makes the entire marketplace trustworthy and even provides competitive advantage for all the parties.

Detecting click fraud takes advantage of the fact that fraudulent traffic exhibits an atypical web browsing pattern when compared with typical clickstream data. Fraudulent traffic often does not follow a logical sequence of actions and contains repetitive actions that would differentiate from other regular traffic (Sadagopan & Li, 2008). For example, most of the fraudulent traffic exhibits either one or many of these characteristics: they have very high click depth (number of web pages accessed deep in the website); the time between each click would be extremely short; a single session would have a high number of clicks on advertisements as compared with normal users; the originating IP address would be different from the target market of the advertisement; there would be very little time spent on the advertiser's target website; etc. It is not one trait that differentiates fraudulent traffic from regular traffic, but a *combination* of the traits. Detecting click fraud is an ongoing and evolving process. Increasingly, the click fraud perpetrators are getting more sophisticated in imitating the characteristics of a normal web browsing user. Hence, click fraud cannot be fully eliminated, however, it can be contained by constantly developing new algorithms to identify fraudulent traffic.

To detect click fraud outliers, first clickstream data would need to be prepared in such a way that detection using

(*Continued*)

(Continued)

data science is easier. A relational column-row dataset can be prepared with each visit occupying each row and the columns being traits like click depth, time between each click, advertisement clicks, total time spent in target website, etc. This multidimensional dataset can be used for outlier detection using data science. Clickstream traits or attributes have to be carefully considered, evaluated, transformed, and added into the dataset. In multidimensional data space, the fraudulent traffic (data point) is distant from other visit records because of their attributes, such as the number of ad clicks in a session. A regular visit usually has one or two ad clicks in a session, while a fraudulent visit would have dozens of ad clicks. Similarly, other attributes can help identify the outlier more precisely. Outlier-detection algorithms reviewed in this chapter assign an outlier score (fraud score) for all the clickstream data points and the records with a higher score are predicted to be outliers.

13.1.2 Anomaly Detection Techniques

Humans are innately equipped to focus on outliers. The news cycle experienced every day is mainly hinged on outlier events. The interest around knowing who is the fastest, who earns the most, and who wins the most medals or scores the most goals is in part due to increased attention to outliers. If the data is in one dimension like taxable income for individuals, outliers can be identified with a simple sorting function. Visualizing data by scatter, histogram, and box-whisker charts can help to identify outliers in the case of single attribute datasets as well. More advanced techniques would fit the data to a distribution model and use data science techniques to detect outliers.

Outlier Detection Using Statistical Methods

Outliers in the data can be identified by creating a statistical distribution model of the data and identifying the data points that do not fit into the model or data points that occupy the ends of the distribution tails. The underlying distribution of many practical datasets fall into the Gaussian (normal) distribution. The parameters for building a normal distribution (i.e., mean and standard deviation) can be estimated from the dataset and a normal distribution curve can be created like the one shown in Fig. 13.1.

Outliers can be detected based on where the data points fall in the standard normal distribution curve. A threshold for classifying an outlier can be specified, say, three standard deviations from the mean. Any data point that is more than three standard deviations is identified as an outlier. Identifying outliers using this method considers only one attribute or dimension at a time. More advanced statistical techniques take multiple dimensions into account and calculate the *Mahalanobis distance* instead of the standard

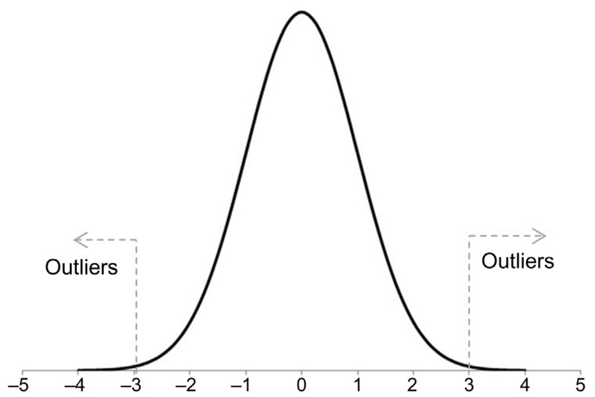

FIGURE 13.1
Standard normal distribution and outliers.

deviations from the mean in a univariate distribution. Mahalanobis distance is the multivariate generalization of finding how many standard deviations away a point is from the mean of the multivariate distribution. Outlier detection using statistics provides a simple framework for building a distribution model and for detection based on the variance of the data point from the mean. One limitation of using the distribution model to find outliers is that the distribution of the dataset is not previously known. Even if the distribution is known, the actual data don't always fit the model.

Outlier Detection Using Data Science

Outliers exhibit a certain set of characteristics that can be exploited to find them. Following are classes of techniques that were developed to identify outliers by using their unique characteristics (Tan, Steinbach, & Kumar, 2005). Each of these techniques has multiple parameters and, hence, a data point labeled as an outlier in one algorithm may not be an outlier to another. Hence, it is prudent to rely on multiple algorithms before labeling the outliers.

> ***Distance-based***: By nature, outliers are different from other data objects in the dataset. In multidimensional Cartesian space they are distant from other data points, as shown in Fig. 13.2. If the average distance of the nearest N neighbors is measured, the outliers will have a higher value than other normal data points. Distance-based algorithms utilize this property to identify outliers in the data.

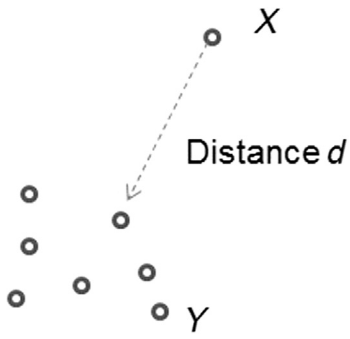

FIGURE 13.2
Distance-based outlier.

Density-based: The density of a data point in a neighborhood is inversely related to the distance to its neighbors. Outliers occupy low-density areas while the regular data points congregate in high-density areas. This is derived from the fact that the relative occurrence of an outlier is low compared with the frequency of normal data points.
Distribution-based: Outliers are the data points that have a low probability of occurrence and they occupy the tail ends of the distribution curve. So, if one tries to fit the dataset in a statistical distribution, these anomalous data points will stand out and, hence, can be identified. A simple normal distribution can be used to model the dataset by calculating the mean and standard deviation.
Clustering: Outliers by definition are not similar to normal data points in a dataset. They are rare data points far away from regular data points and generally do not form a tight cluster. Since most of the clustering algorithms have a minimum threshold of data points to form a cluster, the outliers are the lone data points that are not clustered. Even if the outliers form a cluster, they are far away from other clusters.
Classification techniques: Nearly all classification techniques can be used to identify outliers, if previously known classified data are available. In classification techniques for detecting outliers, a known test dataset is needed where one of the class labels should be called "Outlier." The outlier-detection classification model that is built based on the test dataset can predict whether the unknown data is an outlier or not. The challenge in using a classification model is the availability of previously labeled data. Outlier data may be difficult to source because they are rare. This can be partially solved by stratified sampling where the outlier records are oversampled against normal records.

Supervised classification methods have been discussed in past chapters and unsupervised outlier-detection methods will be discussed in the coming sections. The focus will mainly be placed on distance and density-based detection techniques in the coming sections.

13.2 DISTANCE-BASED OUTLIER DETECTION

Distance or proximity-based outlier detection is one of the most fundamental algorithms for anomaly detection and it relies on the fact that outliers are distant from other data points. The proximity measures can be simple Euclidean distance for real values and cosine or Jaccard similarity measures for binary and categorical values. For the purpose of this discussion, consider a dataset with numeric attributes and Euclidean distance as the proximity measure. Fig. 13.3 shows a two-dimensional scatterplot of a sample dataset.

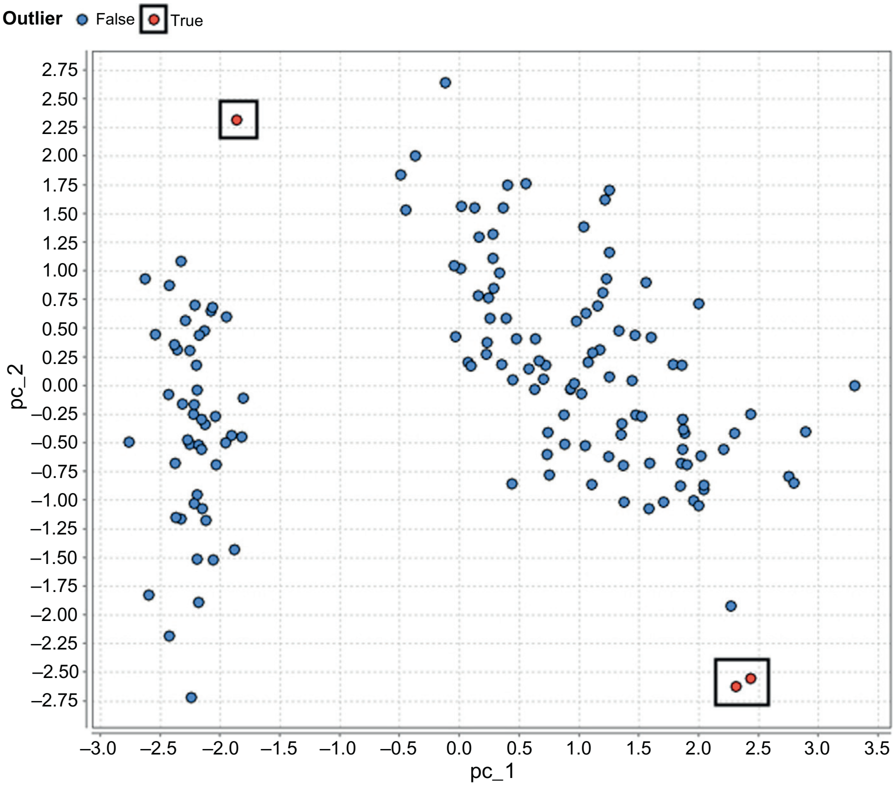

FIGURE 13.3
Dataset with outliers.

Outliers are the data points marked as gray and can be visually identified away from the groups of data. However, when working with multidimensional data with more attributes, visual techniques show limitations quickly.

13.2.1 How It Works

The fundamental concept of distance-based outlier detection is assigning a *distance score* for all the data points in the dataset. The distance score should reflect how far a data point is separated from other data points. A similar concept was reviewed in the k-nearest neighbor (k-NN) classification technique in Chapter 4, Classification. A distance score can be assigned for each data object that is the distance to the k^{th}-nearest data object. For example, a distance score can be assigned for every data object that is the distance to the third-nearest data object. If the data object is an outlier, then it is far away from other data objects; hence, the distance score for the outlier will be higher than for a normal data object. If the data objects are sorted by distance score, then the objects with the highest scores are potentially outlier(s). As with k-NN classification or any algorithm that uses distance measures, it is important to normalize the numeric attributes, so an attribute with a higher absolute scale, such as income, does not dominate attributes with a lower scale, such as credit score.

In distance-based outlier detection, there is a significant effect based on the value of k, as in the k-NN classification technique. If the value of $k = 1$, then two outliers next to each other but far away from other data points are not identified as outliers. On the other hand, if the value of k is large, then a group of normal data points which form a cohesive cluster will be mislabeled as outliers, if the number of data points is less than k and the cluster is far away from other data points. With a defined value of k, once the distance scores have been calculated, a distance threshold can be specified to identify outliers or pick the top n objects with maximum distances, depending on the application and the nature of the dataset. Fig. 13.4 shows the results of two different outlier-detection algorithms based on distance for the Iris dataset. Fig. 13.4A shows the outlier detection with $k = 1$ and Fig. 13.4B shows the detection of the same dataset with $k = 5$.

13.2.2 How to Implement

Commercial data science tools offer specific outlier-detection algorithms and solutions as part of the package either in the modeling or data cleansing sections. In RapidMiner, unsupervised outlier-detection operator can be found in Data Transformation > Data Cleansing > Outlier Detection > Detect Outlier Distance. The example set used in this process is the Iris dataset with four numerical attributes and 150 examples.

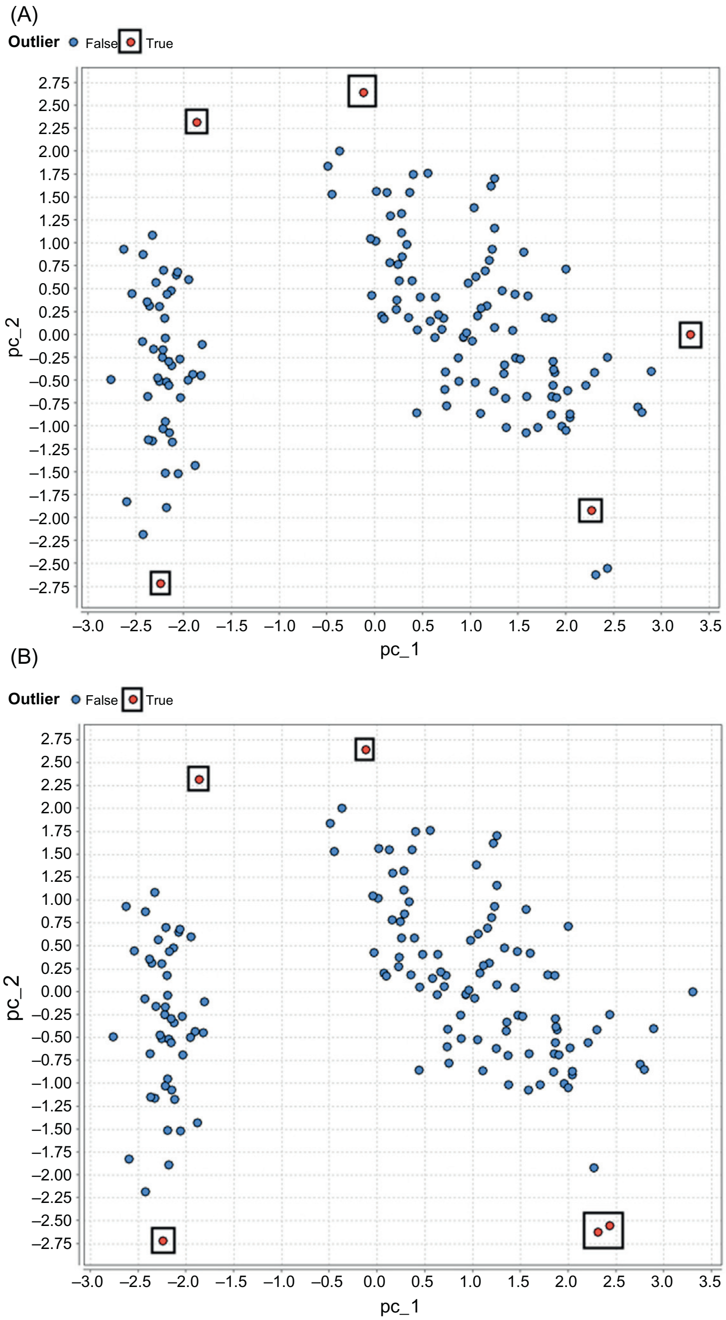

FIGURE 13.4

Top five outliers of Iris dataset when (A) $k = 1$ and (B) $k = 5$.

Step 1: Data Preparation

Even though all four attributes of the Iris dataset measure the same quantity (length) and are measured on the same scale (centimeters), a normalization step is included as a matter of best practice for techniques that involve distance calculation. The *Normalize* operator can be found in Data Transformation > Value modification > Numerical. The attributes are converted to a uniform scale of mean 0 and standard deviation 1 using Z-transformation.

For the purposes of this demonstration, a two-dimensional scatterplot with two attributes will be helpful to visualize outliers. However, the Iris dataset has four attributes. To aid in this visualization objective, the four numerical attributes will be reduced to two attributes (principal components) using the *principal component analysis (PCA)* operator. Please note that the use of the *PCA* operator is optional and not required for outlier detection. The results of the outlier detection with or without *PCA* in most cases will be unchanged. But visualization of the results will be easy with two-dimensional scatterplots. *PCA* will be discussed in detail in Chapter 14, Feature Selection. In this process a variance threshold has been specified for the *PCA* operator of 0.95. Any principal component that has a variance threshold more than 0.95 is removed from the result set. The outcome of the *PCA* operator has two principal components.

Step 2: Detect Outlier Operator

The *Detect Outlier (Distances)* operator has a data input port and outputs data with an appended attribute called *outlier*. The value of the output outlier attribute is either true or false. The *Detect Outlier (Distances)* operator has three parameters that can be configured by the user.

> *Number of neighbors*: This is the value of k in the algorithm. The default value is 10. If the value is made lower, the process finds smaller outlier clusters with less data points.
> *Number of outliers*: The individual outlier score is not visible to the users. Instead the algorithm finds the data points with the highest outlier scores. The number of data points to be found can be configured using this parameter.
> *Distance function*: As in the k-NN algorithm, the distance measurement function needs to be specified. Commonly used functions are Euclidean and cosine (for document vectors).

In this example, $k = 1$, number of outliers = 10, and the distance function is set to Euclidian. The output of this operator is the example set with an appended outlier attribute. Fig. 13.5 provides the RapidMiner process with

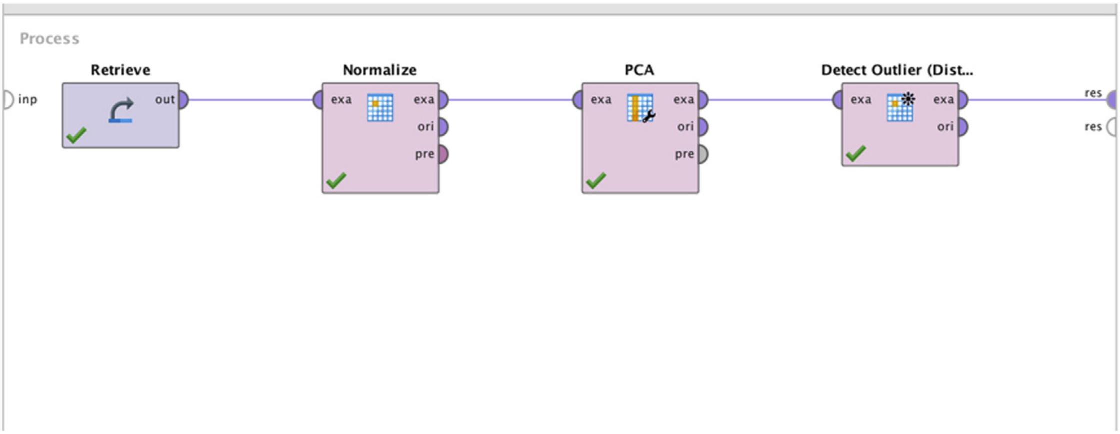

FIGURE 13.5
Process to detect outliers based on distance.

data extraction, PCA dimensional reduction, and outlier-detection operators. The process can now be saved and executed.

Step 3: Execution and Interpretation

The result dataset can be sorted by outlier attribute, which has either a true or false value. Since 10 outliers have been specified in the parameter of the *Detect outlier* operator, that number of outliers can be found in the result set. An efficient way of exploring the outliers is to look at the scatterplot in the Chart view of results set. The *X*- and *Y*-axes can be specified as the principal components and the color as the outlier attribute. The output scatterplot shows the outlier data points along with all the normal data points as shown in Fig. 13.6.

Distance-based outlier detection is a simple algorithm that is easy to implement and widely used when the problem involves many numeric variables. The execution becomes expensive when the dataset involves a high number of attributes and records, because the algorithm has to calculate distances with other data points in high-dimensional space.

13.3 DENSITY-BASED OUTLIER DETECTION

Outliers, by definition, occur less frequently compared to normal data points. This means that in the data space outliers occupy low-density areas and normal data points occupy high-density areas. Density is a count of data points in a normalized unit of space and is inversely proportional to the

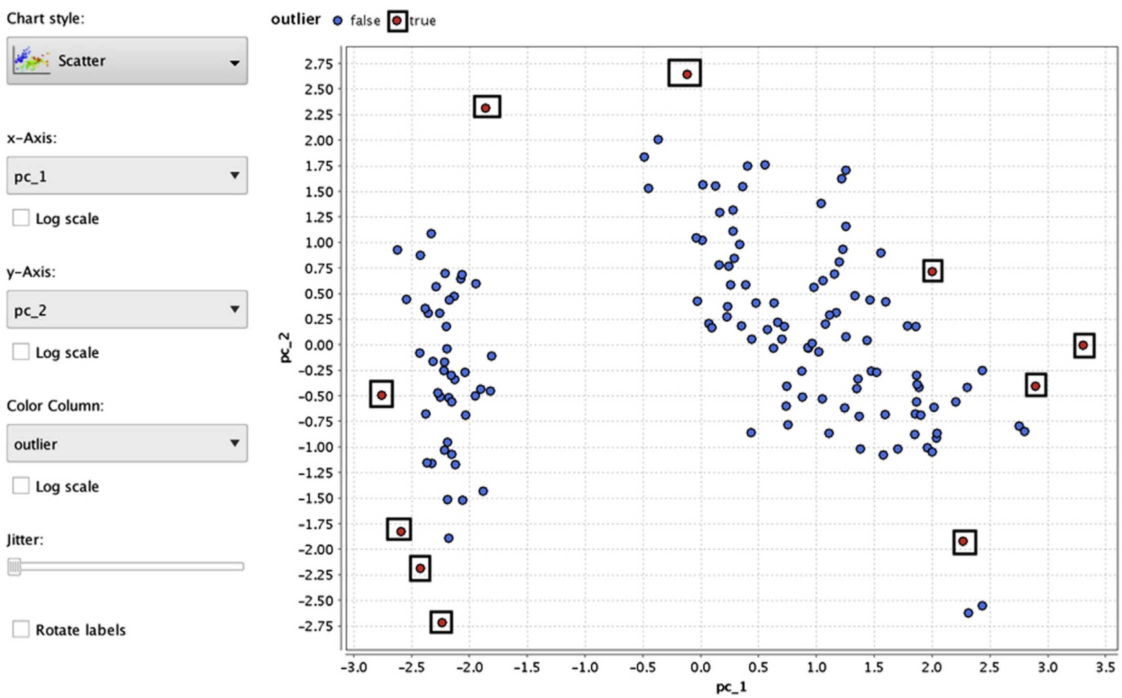

FIGURE 13.6
Outlier detection output.

distances between data points. The objective of a density-based outlier algorithm is to identify those data points from low-density areas. There are a few different implementations to assign an outlier score for the data points. The inverse of the average distance of all k neighbors can be found. The distance between data points and density are inversely proportional. Neighborhood density can also be calculated by calculating the number of data points from a normalized unit distance. The approach for density-based outliers is similar to the approach discussed for density-based clustering and for the k-NN classification algorithm.

13.3.1 How It Works

Since distance is the inverse of density, the approach of a density-based outlier can be explained with two parameters, distance (d) and proportion of data points (p). A point X is considered an outlier if at least p fraction of points lie more than d distance from the point (Knorr & Ng, 1998). Fig. 13.7 provides a visual illustration of outlier detection. By the given definition, the point X occupies a low-density area. The parameter p is specified

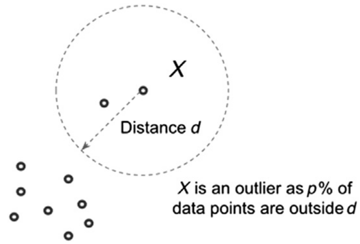

FIGURE 13.7
Outlier detection based on distance and propensity.

as a high value, above 95%. One of the key issues in this implementation is specifying distance. It is important to normalize the attributes so that the distance makes sense, particularly when attributes involve different measures and units. If the distance is specified too low, then more outliers will be detected, which means normal points have the risk of being labeled as outliers and vice versa.

13.3.2 How to Implement

The RapidMiner process for outlier detection based on density is similar to outlier detection by distance, which was reviewed in the previous section. The process developed for previous distance-based outliers can be used, but the *Detect Outlier (Distances)* operator would be replaced with the *Detect Outlier (Densities)* operator.

Step 1: Data Preparation

Data preparation will condition the data so the *Detect Outlier (Densities)* operator returns meaningful results. As with the outlier detection by distance technique, the Iris dataset will be used with normalization and the *PCA* operator so that the number of attributes is reduced to two for easy visualization.

Step 2: Detect Outlier Operator

The *Detect Outlier (Densities)* operator can be found in Data Transformation > Data Cleansing > Outlier Detection, and has three parameters:

Distance (d): Threshold distance used to find outliers. For this example, the distance is specified as 1.
Proportion (p): Proportion of data points outside of radius *d* of a point, beyond which the point is considered an outlier. For this example, the value specified is 95%.
Distance measurement: A measurement parameter like Euclidean, cosine, or squared distance. The default value is Euclidean.

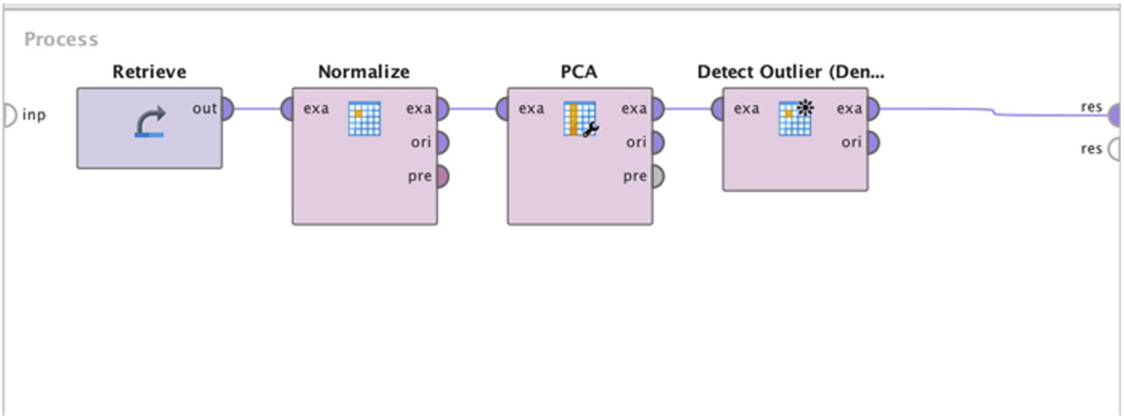

FIGURE 13.8
Process to detect outliers based on density.

Any data point that has more than 95% of other data points beyond distance *d* is considered an outlier. Fig. 13.8 shows the RapidMiner process with the *Normalization*, *PCA*, and *Detect Outlier* operators. The process can be saved and executed.

Step 3: Execution and Interpretation

The process adds an outlier attribute to the example set, which can be used for visualization using a scatterplot as shown in Fig. 13.9. The outlier attribute is Boolean and indicates whether the data point is predicted to be an outlier or not. In the scatterplot, a few data points marked as outliers can be found. The parameters *d* and *p* of the *Detect Outlier* operator can be tuned to find the desired level of outlier detection.

Density-based outlier detection is closely related to distance-based outlier approaches and, hence, the same pros and cons apply. As with distance-based outlier detection, the main drawback is that this approach does not work with varying densities. The next approach, local outlier factor (LOF) is designed for such datasets. Specifying the parameter distance (*d*) and proportion (*p*) is going to be challenging, particularly when the characteristics of the data are not previously known.

13.4 LOCAL OUTLIER FACTOR

The LOF technique is a variation of density-based outlier detection, and addresses one of its key limitations, detecting the outliers in varying density. Varying density is a problem in simple density-based methods, including

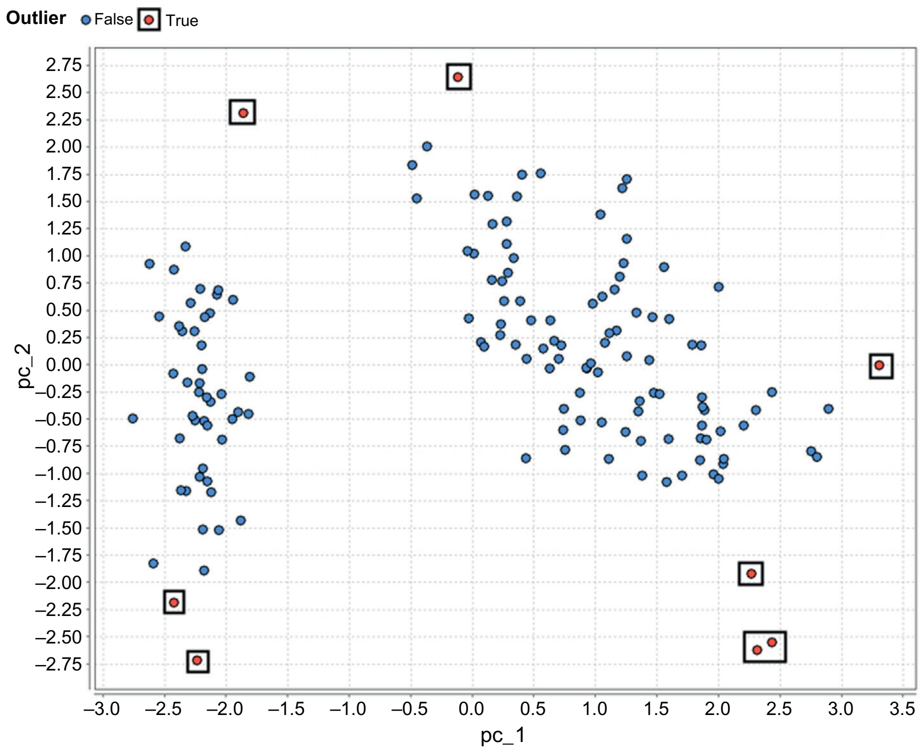

FIGURE 13.9
Output of density-based outlier detection.

DBSCAN clustering (see Chapter 7: Clustering). The LOF technique was proposed in the paper *LOF: Identifying Density-Based Local Outliers* (Breunig, Kriegel, Ng, & Sander, 2000).

13.4.1 How it Works

LOF takes into account the density of the data point and the *density of the neighborhood* of the data point as well. A key feature of the LOF technique is that the outlier score takes into account the relative density of the data point. Once the outlier scores for data points are calculated, the data points can be sorted to find the outliers in the dataset. The core of the LOF lies in the calculation of the relative density. The relative density of a data point X with k neighbors is given by the following equation:

$$\text{Relative density of } X = \frac{\text{Density of } X}{\text{Average density of all data points in the neighborhood}} \tag{13.1}$$

where the density of *X* is the inverse of the average distance for the nearest *k* data points. The same parameter *k* also forms the locality of the neighborhood. By comparing the density of the data point and density of all the data points in the neighborhood, whether the density of the data point is lower than the density of the neighborhood can be determined. This scenario indicates the presence of an outlier.

13.4.2 How to Implement

A LOF-based data science process is similar to the other outlier processes explained in RapidMiner. The *Detect Outlier (LOF)* operator is available in Data Transformation > Data Cleansing > Outlier Detection. The output of the *LOF* operator contains the example set along with a numeric outlier score. The LOF algorithm does not explicitly label a data point as an outlier; instead the score is exposed to the user. This score can be used to visualize a comparison to a threshold, above which the data point is considered an outlier. Having the raw score means that the data science practitioner can "tune" the detection criteria, without having to rerun the scoring process, by changing the threshold for comparison.

Step 1: Data Preparation

Similar to the distance- and density-based outlier-detection processes, the dataset has to be normalized using *Normalize* operator. The *PCA* operator is

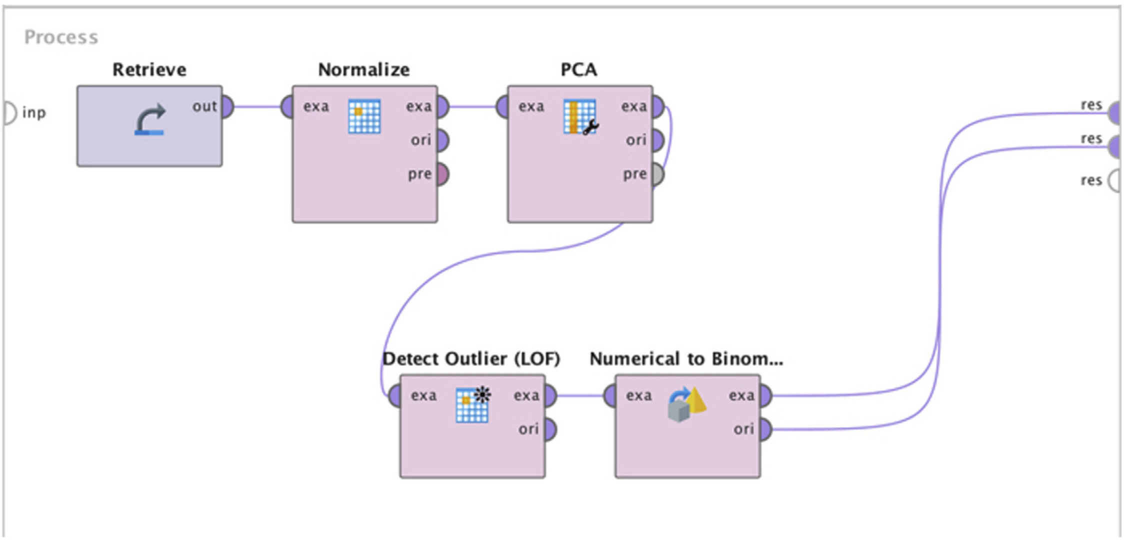

FIGURE 13.10
RapidMiner process for LOF outlier detection. *LOF*, Local outlier factor.

used to reduce the four-dimensional Iris dataset to two dimensions, so that the output can be visualized easily.

Step 2: Detect Outlier Operator

The *LOF* operator has minimal points (*MinPts*) lower bound and upper bound as parameters. The *MinPts* lower bound is the value of *k*, the neighborhood number. The LOF algorithm also takes into account a *MinPts* upper bound to provide more stable results (Breunig et al., 2000). Fig. 13.10 shows the RapidMiner process.

Step 3: Results Interpretation

After using the *Detect Outlier* operator, the outlier score is appended to the result dataset. Fig. 13.11 shows the result set with outlier score represented as the color of the data point. In the results window, the outlier score can be used to color the data points. The scatterplot indicates that points closer to the blue spectrum (left side of the outlier scale in the chart legend) are predicted to be regular data points and points closer to the red spectrum

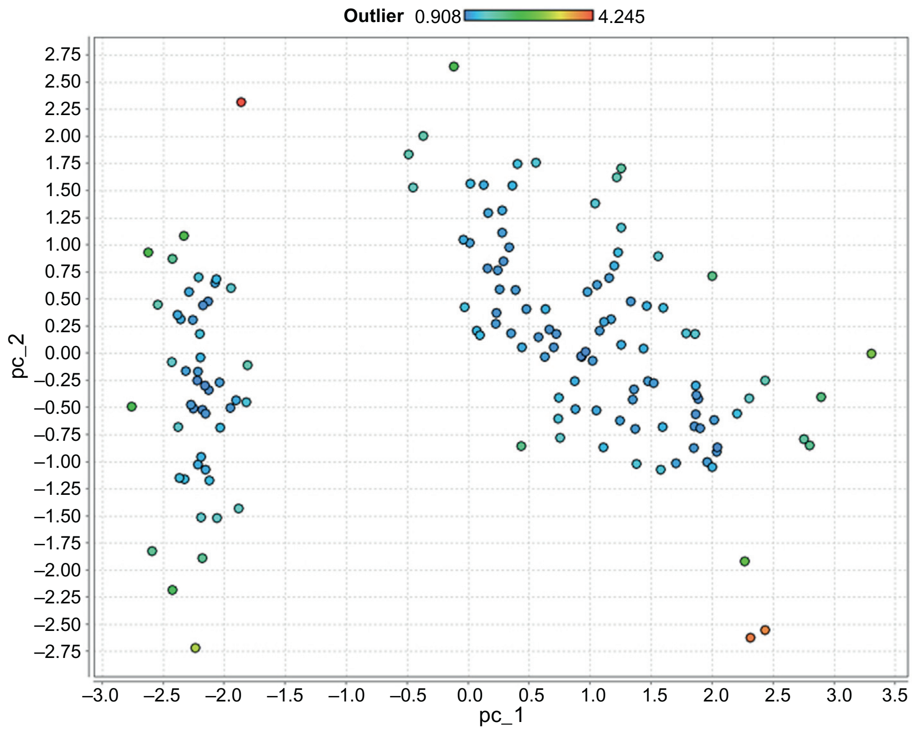

FIGURE 13.11

Output of LOF outlier detection. *LOF*, Local outlier factor.

(right side of the outlier scale in the chart legend) are predicted to be outliers. If an additional Boolean flag indicating whether a data point is an outlier or not is needed, a *Numeric to Binominal* operator can be added to the result dataset. The *Numeric to Binominal* operator converts the numeric outlier score to a binominal true or false based on the threshold specification in the parameter of the operator and to the score output from the *LOF* operator.

In addition to the three data science techniques discussed for outlier detection, the RapidMiner Anomaly *Detection extension* (RapidMiner Extension: Anomaly Detection, 2014) offers more algorithms to identify outliers. RapidMiner extensions can be installed by accessing Help > Updates and Extensions.

13.5 CONCLUSION

In theory, any classification algorithm can be used for outlier detection, if a previously classified dataset is available. A generalized classification model tries to predict outliers the same way it predicts the class label of the data point. However, there is one key issue in using classification models. Since the probability of occurrence of an outlier is really low, say, less than 0.1%, the model can just "predict" the class as "regular" for all the data points and still be 99.9% accurate! This method clearly does not work for outlier detection, since the *recall measure* (see Chapter 8: Model Evaluation for details about recall) is 0%. In practical applications, like detecting network intrusion or fraud prevention in high-volume transaction networks, the cost of not detecting an outlier is very high. The model can even have an acceptable level of false alarms, that is, labeling a regular data point as an outlier. Therefore, special care and preparation is required to improve the detection of the outliers.

Stratified sampling methods can be used to increase the frequency of occurrence of outlier records in the training set and reduce the relative occurrence of regular data points. In a similar approach, the occurrence of outliers and regular records can be sampled with replacements so that there are an equal number of records in both classes. Stratified sampling boosts the number of outlier records in the test dataset with respect to regular records in an attempt to increase both the accuracy and recall of outlier detection. In any case, it is important to know the biases in any algorithm that might be used to detect outliers and to specially prepare the training dataset in order to make the resulting model effective. In practical applications, outlier-detection models have to be updated frequently as the characteristics of an outlier changes over time, and hence, the relationship between outliers and normal records changes as well. In constant real time data streams, outlier detection creates additional challenges because of the dynamic distribution of the data and dynamic relationships within the data (Sadik & Gruenwald, 2013). Outlier detection remains one of the most profound applications of data science as it impacts the majority of the population through financial transaction monitoring, fraud prevention, and early identification of anomalous activity in the context of security.

References

Breunig, M. M., Kriegel, H., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD 2000 international conference on management of data* (pp. 1–12).

Haddadi, H. (2010). Fighting online click-fraud using bluff ads. *ACM SIGCOMM Computer Communication Review*, *40*(2), 21–25.

Knorr, E. M., & Ng, R. T. (1998) Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th VLDB conference* (pp. 392–403). New York, USA.

RapidMiner Extension: Anomaly Detection. (2014). *German research center for artificial intelligence*. DFKI GmbH. Retrieved from <http://madm.dfki.de/rapidminer/anomalydetection>.

Sadagopan, N., & Li, J. (2008) Characterizing typical and atypical user sessions in clickstreams. In *Proceeding of the 17th international conference on World Wide Web—WWW '08 885*. <https://doi.org/10.1145/1367497.1367617>.

Sadik, S., & Gruenwald, L. (2013). Research issues in outlier detection for data streams. *ACM SIGKDD Explorations Newsletter*, *15*(1), 33–40.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Anomaly detection. Introduction to data mining* (pp. 651–676). Boston, MA: Addison Wesley.