Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

NAAC
NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL

HANDS ON TRAINING COURSE

ON

DATA MINING USING R TOOL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HANDS ON TRAINING COURSE

ON

DATA MINING USING R TOOL

Date: From 10-07-2019 TO 12-08-2019 (6 Week Course, Only on Saturdays)

COURSE CONTENTS

MODULE -1		
Durations	Topics	Resource Person
Week 1	A Short Introduction to R	Dr.SR. Mugunthan
	Starting with R	
	R Objects	
	Matrices and Arrays	
	Objects, Classes, and Methods	
	Data Frames .	
	Assignment-1	
Week 2	Predicting Algae Blooms	Dr.SR. Mugunthan
	Problem Description and Objectives	
	Loading the Data into R .	
	Multiple Linear Regression	
	Assignment-2	
Week 3	Unsupervised Techniques	Dr.SR. Mugunthan
	Supervised Techniques	
	Precision and Recall	
	Assignment-3	
MODULE -2		
Durations	Topics	Resource Person
Week 4	Model Evaluation and Selection	Dr.SR. Mugunthan
	Reading the Data from the CSV File	
	Getting the Data from the Web	
	Random Forests	
	k-Nearest Neighbors	
	Assignment-4	
MODULE -3		
Durations	Topics	Resource Person

Week 5	Local Outlier Factors (LOF)	Dr.SR. Mugunthan
	Clustering-Based Outlier Rankings (ORh)	
	The Modified Box Plot Rule	
	Assessment -1	
	Conclusion	

A Short Introduction to R

The goal of this section is to provide a brief introduction to the key issues of the R language. We do not assume any familiarity with computer programming. Readers should be able to easily follow the examples presented in this section. Still, if you feel some lack of motivation to continue reading this introductory material, do not worry. You may proceed to the case studies and then return to this introduction as you get more motivated by the concrete applications. R is a functional language for statistical computation and graphics. It can be seen as a dialect of the S language (developed at AT&T) for which John Chambers was awarded the 1998 Association for Computing Machinery (ACM) Software award that mentioned that this language "forever altered how people analyze, visualize and manipulate data". R can be quite useful just by using it in an interactive fashion at its command line. Still, more advanced uses of the system will lead the user to develop his own functions to systematize repetitive tasks, or even to add or change some function a

R Objects

There are two main concepts behind the R language: objects and functions. An object can be seen as a storage space with an associated name. Everything in R is stored in an object. All variables, data, functions, etc. are stored in the memory of the computer in the form of named objects. Functions are a special type of R objects designed to carry out some operation. They usually take some arguments and produce a result by means of executing some set of operations (themselves usually other function calls). R 6 Data Mining with R: Learning with Case Studies already comes with an overwhelming set of functions available for us to use, but as we will see later, the user can also create new functions. Content may be stored in objects using the assignment operator. This operator is denoted by an angle bracket followed by a minus sign (<-):8 > x <- 945 The effect of the previous instruction is thus to store the number 945 on an object named x. By simply entering the name of an object at the R prompt one can see its contents:> x [1] 945 The rather cryptic "[1]" in front of the number 945 can be read as "this line is showing values starting from the first element of the object." This is particularly useful for objects containing several values, like vectors, as we will see later. Below you will find other examples of assignment statements. These examples should make it clear that this is a destructive operation as any object can only have a single content at any time t. This means that by assigning some new content to an existing object, you in effect lose its previous content: > y <- 39 > y [1] 39> y <- 43> y [1] 43 You can also assign numerical expressions to an object. In this case the object will store the result of the expression: > z <- 5> w <- z^2 > w [1] 25 > i <- (z * 2 + 45)/2

Data Frames

Data frames are the data structure most indicated for storing data tables in R. They are similar to matrices in structure as they are also bi-dimensional. However, contrary to matrices, data frames may include data of a different type in each column. In this sense they are more similar to lists, and in effect, for R, data frames are a special class of lists. We can think of each row of a data frame as an observation (or case), being described by a set of variables (the named columns of the data frame). You can create a data frame as follows: > my .dataset <data .frame (site=c('A','B','A','A','B'), +season=c('Winter ','Summer',' Summer', 'Spring' ,'Fall'), + pH = c(7.4,6.3,8.6,7.2,8.9)) > my. Data set 16Because vector elements must have the same type (c.f. Section 1.2.3). Introduction 27 site season pH 1 A Winter 7.4 2 B Summer 6.3 3 A Summer 8.6 4 A Spring 7.2 5 B Fall 8.9 Elements of data frames can be accessed like a matrix: > my. dataset[3, 2] [1] Summer Levels: Fall Spring Summer Winter Note that the "season" column has been coerced into a factor because all its elements are character strings. Similarly, the "site" column is also a factor. This is the default behavior of the data. Frame() function.17 You can use the indexing schemes described in Section 1.2.7 with data frames. Moreover, you can use the column names for accessing full columns of a data frame: > my .dataset $pH [1] 7.4 6.3 8.6 7.2 8.9 You can perform some simple querying of the data in the data frame, taking advantage of the sub-setting possibilities of R, as shown on these examples: > my .dataset[my. Dataset $pH > 7,] site season pH 1 A Winter 7.4 3 A Summer 8.6 4 A Spring 7.2 5 B Fall 8.9> my. Dataset [my. Dataset $site == "A", "pH"] [1] 7.4 8.6 7.2 > my. dataset[my. Dataset $season == "Summer", c("site", "pH")] site pH 2 B 6.3 3 A 8.6

Matrices and Arrays

Data elements can be stored in an object with more than one dimension .This may be useful in several situations. Arrays store data elements in several dimensions. Matrices are a special case of arrays with two single dimensions .Arrays and matrices in R are nothing more than vectors with a particular attribute that is the dimension. Let us see an example. Suppose you have the vector of numbers c(45,23,66,77,33,44,56,12,78,23). The following would "organize" these ten numbers as a matrix:> m <- c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23)> m[1] 45 23 66 77 33 44 56 12 78 23> dim(m) <- c(2, 5)> m[,1] [,2] [,3] [,4] [,5][1,] 45 66 33 56 78[2,] 23 77 44 12 23Notice how the numbers were "spread" through a matrix with two rows and five columns (the dimension we have assigned to m using the dim() function). Actually, you could simply create

the matrix using the simpler instruction:> m <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23), 2,+ 5) You may have noticed that the vector of numbers was spread in the matrix by columns; that is, first fill in the first column, then the second, and so on. You can fill the matrix by rows using the following parameter of the function matrix():> m <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23), 2,+ 5, by row = T)> m[,1] [,2] [,3] [,4] [,5][1,] 45 23 66 77 33[2,] 44 56 12 78 23 As the visual display of matrices suggests, you can access the elements of a matrix through a similar indexing scheme as in vectors, but this time with two indexes (the dimensions of a matrix):> m[2, 3][1] 12 20 Data Mining with R: Learning with Case Studies You can take advantage of the sub-setting schemes described in Section 1.2.7 to extract elements of a matrix, as the following examples show:> m[-2, 1][1] 45> m[1, -c(3, 5)] [1] 45 23 77 Moreover, if you omit any dimension, you obtain full columns or rows of the matrix:> m[1,][1] 45 23 66 77 33> m[, 4][1] 77 78

Objects, Classes, and Methods

One of the design goals of R is to facilitate the manipulation of data so that we can easily perform the data analysis tasks we have. In R, data is store don objects. As mentioned, everything in R is an object, from simple numbers to functions or more elaborate data structures. Every R object belongs to a class. Classes define the abstract characteristics of the objects that belong to them. Namely, they specify the attributes or properties of these objects and also their behaviors (or methods). For instance, the matrix class has specific properties like the dimension of the matrices and it also has specific behavior for some types of operations. In effect, when we ask R the content of a matrix, R will show it with a specific format on the screen. This happens because there is a specific print method associated with all objects of the class matrix. In summary, the class of an object determines (1) the methods that are used by some general functions when applied to these objects, and also (2) the representation of the objects of that class. This representation consists of the information that is stored by the objects of this class.

Problem Description and Objectives

High concentrations of certain harmful algae in rivers constitute a serious ecological problem with a strong impact not only on river life forms, but al soon water quality. Being able to monitor and perform an early forecast of algae blooms is essential to improving the quality of rivers .With the goal of addressing this prediction problem, several water samples were collected in different European rivers at different times during a period of approximately 1 year. For each water sample, different chemical properties were measured as well as the frequency of occurrence of seven harmful algae. Some other characteristics of the water collection process were also stored ,such as the season of the year, the river size, and the river speed .One of the main motivations behind this application lies in the fact that chemical monitoring is cheap and easily automated, while the biological anal- y sis of the samples to identify the algae that are present in the water involves microscopic examination, requires trained manpower, and is therefore both expensive and slow. As such, obtaining models that are able to accurately predict the algae frequencies based on chemical properties would facilitate the creation of cheap and automated systems for monitoring harmful algae blooms.

Loading the Data into R

We will consider two forms of getting the data into R: (1) one by simply taking advantage of the package accompanying the book that includes data frames with the datasets ready for use; and (2) the other by going to the book Website, downloading the text files with the data, and then loading them into R. The former is obviously much more practical. We include information on the second alternative for illustrative purposes on how to load data into R from text files .If you want to follow the easy path, you simply load the book package,3 and you immediately have a data frame named algae available for use. This data frame contains the first set of 200 observations mentioned above.> library(DM w R) > head(algae) season size speed mxPHmnO2Cl NO3 NH4 oPO4 PO4 Chla1 winter small medium 8.00 9.8 60.800 6.238 578.000 105.000 170.000 50.0 2 spring small medium 8.35 8.0 57.750 1.288 370.000 428.750 558.750 1.3 3 autumn small medium 8.10 11.4 40.020 5.330 346.667 125.667 187.057 15.6 4 spring small medium 8.07 4.8 77.364 2.302 98.182 61.182 138.700 1.4 5 autumn small medium 8.06 9.0 55.350 10.416 233.700 58.222 97.580 10.5 6 winter small high 8.25 13.1 65.750 9.248 430.000 18.250 56.667 28.4a1 a2 a3 a4 a5 a6 a7 1 0.0 0 .0 0 .0 0.0 34.2 8.3 0.02 1.4 7.6 4.8 1.9 6.7 0.0 2.1 3 3.3 53.6 1.9 0.0 0 .0 0.0 9.7 4 3.1 41.0 18.9 0.0 1.4 0.0 1.4 5 9.2 2.9 7.5 0.0 7.5 4.1 1.0 6 15.1 14.6 1.4 0.0 22.5 12.6 2.9

Multiple Linear Regression

Multiple linear regression is among the most used statistical data analysis techniques. These models obtain an additive function relating a target variable to a set of predictor variables. This additive function is a sum of terms of the form $\beta i \times Xi$, where Xi is a predictor variable and β is a number .As mentioned before, there is no predefined way of handling missing values for this type of modeling technique. As such, we will use the data resulting from applying the method of exploring the similarities among the training cases to fill in the unknowns (see Section 2.5.4). Nevertheless, before we apply this method, we will remove water samples number 62 and 199 because, as mentioned before, they have six of the eleven predictor variables missing. The following code obtains a data frame without missing values:> data(algae)> algae <- algae[-many NAs(algae),]> clean. algae <- k n n Imputation(algae, k = 10)

Predictions for the Seven Algae

In this section we will see how to obtain the predictions for the seven algae on the 140 test samples. Section 2.7 described how to proceed to choose the best models to obtain these predictions. The used procedure consisted of obtaining unbiased estimates of the NMSE for a set of models on all seven predictive task s, by means of a cross-validation experimental process .The main goal in this data mining problem is to obtain seven predictions for each of the 140 test samples. Each of these seven predictions will be obtained using the model that our cross-validation process has indicated as being the "best" for that task. This will be one of either the models shown by our call to the best Scores() function in the previous section. Namely, it will be one of either "cv.rf.v3", "cv.rf.v2", "cv.rf.v1", or "cv.rpart.v3". Let us start by obtaining these models using all the available training data so that we can apply them to the test set. Notice that, for simplicity, we will grow the regression tree using the clean .algae data frame that had the NA

values substituted by a k nearest neighbor imputation process. This can be avoided for regression trees as they incorporate their own method for handling unknown values. Random forests, on the contrary, do not include such a method so they will need to be learned using the clean. algae data frame. The following code obtains all seven models:> best Models Names <-s apply (best Scores(res .all),+ function(x) x[' n m se', 'system'])> learners <- c(r f='random Forest' ,r part ='r par t X se')> fun c s <- learners[s apply(s t r split(best Models Names,'\\.'),+ function(x) x[2])] > par Sett s <- l apply(best Models Names, + function(x) get Variant(x, res .all)@pars) > best Models <- list()> for(a in 1:7) {92 Data Mining with R: Learning with Case Studies + form <- as. formula(paste(names(clean .algae)[11+a],'~ .')) + best Model s[[a]] <- do .call(fun c s[a], + c(list(form, clean .algae[,c(1:11,11+a)]),par Sett s[[a]]))+ }

Reading the Data from the CSV File

As we have mentioned before, at the book Web site you can find different sources containing the data to use in this case study. If you decide to use the CSV file, you will download a file whose first lines look like this: "Index" "Open" "High" "Low" "Close" "Volume" "AdjClose"1970-01-02 92.06 93.54 91.79 93 8050000 93 1970-01-05 93 94.25 92.53 93.46 11490000 93.46 1970-01-06 93.46 93.81 92.13 92.82 11460000 92.821970-01-07 92.82 93.38 91.93 92.63 10010000 92.63 1970-01-08 92.63 93.47 91.99 92.68 10670000 92.68 1970-01-09 92.68 93.25 91.82 92.4 9380000 92.4 1970-01-12 92.4 92.67 91.2 91.7 8900000 91.7 102 Data Mining with R: Learning with Case Studies Assuming you have downloaded the file and have saved it with the name “sp500.csv” on the current working directory of your R session, you can load it into R and create an x t s object with the data, as follows:> GSPC <- as.xts(read.zoo("sp500.csv", header = T)) The function read.zoo() of package zoo8 reads a CSV file and transforms the data into a zoo object assuming that the first column contains the time tags. The function as.xts() coerces the resulting object into an object of class x t s.

Getting the Data from the Web

Another alternative way of getting the S&P 500 quotes is to use the free service provided by Yahoo finance, which allows you to download a CSV file with the quotes you want. The t series (Trap let i and Horn I k, 2009) R package9 includes the function get .his t quote() that can be used to download the quotes into a zoo object. The following is an example of the use of this function to get the quotes of S&P 500:> library(t series) > GSPC <- as.xts(get .h I s t .quote("^GSPC" ,start="1970-01-02",quote=c("Open", "High", "Low", "Close ","Volume" ,"A d j Close")))).....> head(GSPC) Open High Low Close Volume A d j Close 1970-01-02 92.06 93.54 91.79 93.00 8050000 93.00 1970-01-05 93.00 94.25 92.53 93.46 11490000 93.46 1970-01-06 93.46 93.81 92.13 92.82 11460000 92.821970-01-07 92.82 93.38 91.93 92.63 10010000 92.63 1970-01-08 92.63 93.47 91.99 92.68 10670000 92.68 1970-01-09 92.68 93.25 91.82 92.40 9380000 92.40 As the function get. H is t .quote() returns an object of class zoo, we have again used the function as.xts() to coerce it to x t s.

Unsupervised Techniques

In the reports that were not inspected, the column In sp is in effect irrelevant as it carries no information. For these observations we only have descriptors of the transactions. This means that these sales reports are only described by a set of independent variables. This is the type of data used by unsupervised learning techniques. These methods are named this way because their goal is not to learn some “concept ”with the help of a“ teacher ”as in supervised methods. The data used by these latter methods are examples of the concepts being learned (e.g., the concept of fraud or normal transaction). This requires that the data is pre classified (labeled) by a domain expert into one of the target concepts. This is not the case for the set of reports with unknown inspection results. We are thus facing a descriptive data mining task as opposed to predictive tasks, which are the goal of supervised methods .Clustering is an example of a descriptive data mining technique. Clustering methods try to find the “natural” groupings of a set of observations by forming clusters of cases that are similar to each other. The notion of similarity usually requires the definition of a metric over the space defined by the variables that describe the observations. This metric is a distance function that measures how far an observation is from another. Cases that are near each other are usually considered part of the same natural group of data.

Supervised Techniques

The set of transactions that were labeled normal or fraudulent (i.e., have been inspected) can be used with other types of modeling approaches. Supervised learning methods use this type of labeled data. The goal of these approaches is to obtain a model that relates a target variable (the concept being learned) with a set of independent variables (predictors, attributes). This model can be regarded as an approximation of an unknown function Y = f(X1, X2, · · · ,X p) that describes the relationship between the target variable Y and the predictor s X1, X2, · · · , X p. The task of the modeling technique is to obtain the model parameters that optimize a certain selected criterion, for example, minimize the prediction error of the model. This search task is carried out with the help of a sample of observations of the phenomena under study, that is, it is based on a dataset containing examples of the concept being learned. These examples are particular instances of the variables X1, X2, · · · , Xp, Y . If the target variable Y is continuous, we have a (multiple) regression problem. If Y is a nominal variable, we have a classification problem.

Precision and Recall

In this application a successful model should obtain a ranking that includes all known frauds at the top positions of the ranking. Fraudulent reports are a minority in our data. Given a number k of reports that our resources allow to inspect, we would like that among the k top-most positions of the obtained ranking, we only have either frauds or non-inspected reports. Moreover, we would like to include in these k positions all of the known fraud cases that exist in the test set. Random Forests Random forests (Breiman, 2001) are an example of an ensemble model, that is, a model that is formed by a set of simpler models. In particular, random Classifying Microarray Samples 255 forests consist of a set of decision trees, either classification or regression trees, depending on the problem being addressed. The user decides the number of trees in the

ensemble. Each tree is learned using a bootstrap sample obtained by randomly drawing N cases with replacement from the original dataset, where N is the number of cases in that dataset. With each of these training sets, a different tree is obtained. Each node of these trees is chosen considering only a random subset of the predictors of the original problem. The size of these subsets should be much smaller than the number of predictors in the dataset. The trees are fully grown, that is, they are obtained without any post-pruning step. More details on how tree-based models are obtained appear in Section 2.6.2 (page 71).The predictions of these ensembles are obtained by averaging over the predictions of each tree. For classification problems this consists of a voting mechanism. The class that gets more votes across all trees is the prediction of the ensemble. For regression, the values predicted by each tree are averaged to obtain the random forest prediction. In R, random forests are implemented in the package random Forest. We have already seen several examples of the use of the functions provided by this package throughout the book, namely, for feature selection. Further readings on random forests The reference on Random Forests is the original work by Breiman (2001). Further in for-mation can also be obtained at the site http://stat-www.berkeley.edu/users/breiman/

Random Forests

Random forests (Breiman, 2001) are an example of an ensemble model, that is, a model that is formed by a set of simpler models. In particular, random forests consist of a set of decision trees, either classification or regression trees, depending on the problem being addressed. The user decides the number of trees in the ensemble. Each tree is learned using a bootstrap sample obtained by randomly drawing N cases with replacement from the original dataset, where N is the number of cases in that dataset. With each of these training sets, a different tree is obtained. Each node of these trees is chosen considering only a random subset of the predictors of the original problem. The size of these subsets should be much smaller than the number of predictors in the dataset. The trees are fully grown, that is, they are obtained without any post-pruning step.

k-Nearest Neighbors

The k-nearest neighbors algorithm belongs to the class of so-called lazy learners. These types of techniques do not actually obtain a model from the training data. They simply store this dataset. Their main work happens at prediction time. Given a new test case, its prediction is obtained by searching for similar cases in the training data that was stored. The k most similar training cases are used to obtain the prediction for the given test case. In classification problems, this prediction is usually obtained by voting and thus an odd number for k is desirable. However, more elaborate voting mechanisms that take into account the distance of the test case to each of the k neighbors are also possible. For regression, instead of voting we have an average of the target variable values of the k neighbors. This type of model is strongly dependent on the notion of similarity between cases. This notion is usually defined with the help of a metric over the input space defined by the predictor variables. This metric is a distance function that can calculate a number representing the “difference” between any two observations. There are many distance functions, but a rather frequent selection is the Euclidean distance function that is defined as 256 Data Mining with R: Learning with Case Studies d(xi, x j =v u u t X p k=1(X I ,k – X j, k)2 (5.3)

where p is the number of predictors, and xi and xj are two observations. These methods are thus very sensitive to both the selected metric and also to the presence of irrelevant variables that may distort the notion of similarity .Moreover, the scale of the variables should be uniform; otherwise we might underestimate some of the differences in variables with lower average values. The choice of the number of neighbors (k) is also an important parameter of these methods. Frequent values include the numbers in the set {1, 3, 5, 7, 11},but obviously these are just heuristics. However, we can say that larger values of k should be avoided because there is the risk of using cases that are already far away from the test case. Obviously, this depends on the density of the training data. Too sparse datasets incur more of this risk. As with any learning model, the "ideal" parameter settings can be estimated through some experimental methodology. In R, the package class (V enables and Ripley, 2002) includes the function k n n () that implements this idea. Below is an illustrative example of its use on the iris dataset:> library(class)> data(iris) > I d x <- sample(1:nrow(iris), as. integer(0.7 * n row(iris)))> t r <- iris[I d x,] > t s <- iris[-I d x,]> preds <- knn(tr[, -5], ts[, -5], tr[, 5], k = 3)> table(preds, ts[, 5]) preds setosaversi color virginic setosa 14 0 0versicolor 0 14 2virginica 0 1 14 As you see, the function k n n () uses a nonstandard interface. The first argument is the training set with the exception of the target variable column. The second argument is the test set, again without the target. The third argument includes the target values of the training data. Finally, there are several other parameters controlling the method, among which the parameter k determines the number of neighbors. We can create a small function that enables the use of this method in a more standard formula-type interface:> k N N <- function(form, train, test, norm = T, norm .stats = NULL,+ ...) {+require(class,quietlyTRUE)+tgtColwhich(colnames(train)==as.character(form[[2]]))+ if (norm) {Classifying Microarray Samples 257+ if (is.null(norm.stats)) + tmp <- scale(train[, -t g t Col], center = T, scale = T)+ else t mp <- scale(train[, -t g t Col], center = norm .stats[[1]], + scale = norm .stats[[2]]) + train[, -t g tCol] <- t m p+ ms <- a t t r (t m p, "scaled :center")+ s s <- a t t r(t mp, "scaled :scale")+ test[, -t g t Col] <- scale(test[, -t g tCol], center = ms,+ scale = ss) + k n n(train[, -tgtCol], test[, -tgtCol], train[, t g t Col],+ ...)+ } > preds . norm <- k N N(Species ~ ., tr, ts, k = 3) > table(preds .norm, t s[, 5]) preds . norm set osaversicolor virginica setosa 14 0 0 versicolor 0 14 3 virgin ica 0 1 13 > preds.notNorm <- kNN(Species ~ ., tr, ts, norm = F, k = 3) > table(preds.notNorm, ts[, 5]) preds .not Norm setosa versicolor virginica setosa 14 0 0 versicolor 0 14 2 virginica 0 1 14

Local Outlier Factors (LOF)

Outlier ranking is a well-studied research topic. Breunig et al. (2000) have developed the local outlier factor (LOF) system that is usually considered a state-of-the-art outlier ranking method. The main idea of this system is to try to obtain an outlyingness score for each case by estimating its degree of isolation with respect to its local neighborhood. The method is based on the notion of the local density of the observations. Cases in regions with very low density are considered outliers. The estimates of the density are obtained using the distances between cases. The authors defined a few concepts that drive the algorithm used to calculate the outlyingness score of each point. These are the (1) concept of core distance of a point p, which is defined as its distance to its kth nearest neighbor, (2) concept of reach ability distance between the case p1 and p2, which is given by the maximum of the core distance of p1 and the distance between both cases, and (3) local reach ability distance of a point, which is inversely proportional to the average reach ability

distance of its k neighbors. The LOF of a case is calculated as a function of its local reach ability distance.

Clustering-Based Outlier Rankings (ORh)

The next outlier ranking method we consider is based on the results of a clustering algorithm. The ORh method (Tor go, 2007) uses a hierarchical agglomerative clustering algorithm to obtain a den drogram of the given data. Den drograms are visual representations of the merging process of these clustering methods. Cutting these trees at different height levels produces different clusterings of the data. At the lowest level we have a solution with as many groups as there are observations on the given training data. This is the initial solution of the iterative algorithm used by these methods. The next steps of this algorithm decide which two groups from the previous step should be merged into a single cluster. This merging process is guided by some criterion that tries to put together observations that are more similar to each other. The iterative process is stopped when the last two groups are merged into a single cluster with all observations. The dendrogram describes the entire merging process. The function hclust() of the base package stats implements several variants of this type of clustering. The object returned by this function includes a data structure (merge) that includes information on which cases are involved in each merging step. The ORh method uses the information in this data structure as the basis for the following outlier ranking method. Clustering-Based Outlier Rankings (ORh) The next outlier ranking method we consider is based on the results of a clustering algorithm. The ORh method (Torgo, 2007) uses a hierarchical agglomerative clustering algorithm to obtain a dendrogram of the given data. Dendrograms are visual representations of the merging process of these clustering methods. Cutting these trees at different height levels produces different clusterings of the data. At the lowest level we have a solution with as many groups as there are observations on the given training data. This is the initial solution of the iterative algorithm used by these methods. The next steps of this algorithm decide which two groups from the previous step should be merged into a single cluster. This merging process is guided by some criterion that tries to put together observations that are more similar to each other. The iterative process is stopped when the last two groups are merged into a single cluster with all observations. The dendrogram describes the entire merging process. The function hclust() of the base package stats implements several variants of this type of clustering. The object returned by this function includes a data structure (merge) that includes information on which cases are involved in each merging step. The ORh method uses the information in this data structure as the basis for the following outlier ranking method.

Data Mining Applications

Data mining is beneficial in various domains. Let’s review some applications below:

- **Banking:** Data mining helps detect fraud cases in the credit card industry, fraud profiling, and fraud investment cases in banks.
- **Manufacturing:** With mining, it is easy to predict products that should increase or reduce their manufacturing based on market data.
- **Insurance:** Data mining helps insurance companies optimize the price of their most profitable products and predict which offers to promote to different types of customers.
- **E-commerce:** E-commerce sites, products price, product recommendations, and offers are all possible because of data mining.
- **Education:** Data mining can detect learning patterns in students using their background and achievement records to indicate who needs more attention and who should focus on a particular field.