UNIT 16 ADVANCE ANALYSIS USING R

Structure **Page Nos.**

16.0 INTRODUCTION

This unit explores the concepts pertaining to advance level of data analysis and their application in R. The unit explains the theory and the working of the decision tree model and how to run it on R. It further discusses its various types that may fall under the 2 categories based on the target variables. The unit also explores the concept of Random Forest and discusses its application on R. In the subsequent sections, the unit explains the details of classification algorithm and its features and types. It further explains the unsupervised learning technique- Clustering and its application in R programming. It further discusses the 2 types of clustering in R programming including the concept and algorithm of K-Means Clustering. The unit concludes by drawing insights on the theory of the Association Rules and its application in R.

16.1 OBJECTIVES

After going through this Unit, you will be able to:

- explain the concept of Decision Tree- including its types and application in R;
- explain the working of Decision Tree and the factors to consider when choosing a tree in R;
- explain the concept of Random Forest and its application on R;
- explain the concept of Classification algorithm and its features including- classifier, characteristics, binary classification, multi-class classification and multi-label classification;
- explain the types of classifications including- linear classifier and its types, support vector machine and decision tree;
- explain the concept of Clustering and its application in R;
- explain the methods of Clustering and their types including the K-Means clustering and its application in R;
- explain the concept and the theory behind the Association Rule Mining and its further application in R Language.

53

16.2 DECISION TREES

A decision tree is a graph that represents decisions and their results in a tree format. Graph nodes represent events or selections, and graph edges represent decision rules or conditions. It is primarily used in machine learning and data mining applications that use R. Examples of use of decision trees include predicting email as spam or non-spam, predicting cancerous tumours, or predicting credit based on the credit risk of each of these factors. Models are typically built using observational data, also known as training data. Then use a set of validation data to validate and improve the model. R has packages used to build and visualize decision trees. For a new set of predictors, this model is used to reach decisions about the categories of data (yes / no, spam / non-spam).

Installing R Package: Package "party" is used for decision tree. It has a function ctree() which is used to create and analyse decision tree. Figure 16.1 shows the output, when you install the Package using install command.

```
> install.packages("party")
Installing package into 'C:/Users/esha.govil/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
also installing the dependencies 'TH.data', 'libcoin', 'multcomp', 'modeltools', 'coin'

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/TH.data_1.1-0.zip'
Content type 'application/zip' length 8807484 bytes (8.4 MB)
downloaded 8.4 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/libcoin_1.0-9.zip'
Content type 'application/zip' length 1005136 bytes (981 KB)
downloaded 981 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/multcomp_1.4-18.zip'
Content type 'application/zip' length 735384 bytes (718 KB)
downloaded 718 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/modeltools_0.2-23.zip'
Content type 'application/zip' length 208375 bytes (203 KB)
downloaded 203 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/coin_1.4-2.zip'
Content type 'application/zip' length 1439214 bytes (1.4 MB)
downloaded 1.4 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/party_1.3-9.zip'
Content type 'application/zip' length 936057 bytes (914 KB)
downloaded 914 KB

package 'TH.data' successfully unpacked and MD5 sums checked
package 'libcoin' successfully unpacked and MD5 sums checked
package 'multcomp' successfully unpacked and MD5 sums checked
package 'modeltools' successfully unpacked and MD5 sums checked
package 'coin' successfully unpacked and MD5 sums checked
package 'party' successfully unpacked and MD5 sums checked
```

Figure 16.1: Installation of package party used for decision trees

```
> ctree(formula, data)
```

Syntax: Defining tree formula describes the predictor and response variables and data is the name of dataset used. The following are the different types of decision trees that can be created using this package.

- **Decision Stump**: It is used to generate decision trees with only one split and is therefore also known as one-level decision tree. In most cases, known for its low predictive performance due to its simplicity.
- **M5**: It is known for its exact classification accuracy, and the ability to work well with small noisy datasets.
- **ID3 (Iterative Dichroatiser 3):** One of the core and a wide range of decision structures is the best attribute for classifying the specified record with the top-down, greedy search approach via the specified dataset.

- **C4.5**: This type of decision tree, known as a statistical classifier, is derived from its parent ID3. This creates a decision based on the predictor's bundle.
- **C5.0**: As a successor to C4.5, there are two models, the base tree and the rule-based model, whose nodes can only predict category targets.
- **CHAID**: This algorithm is extended as a chi-square automatic interaction detector and basically examines the merged variables and justifies the result of the dependent variable by building a predictive model.
- **MARS**: Extended as a multivariate adaptive regression spline, this algorithm builds a set of piecewise linear models used to model anomalies and interactions between variables. They are known for their ability to process numerical data more efficiently.
- **Conditional inference tree**: This is a type of decision tree that recursively separates response variables using the conditional inference framework. It is known for its flexibility and strong fundamentals.
- **CART**: Expanded as a classification and regression tree, the value of the target variable is predicted if it is continuous. Otherwise, if it is a category, the required class is identified.

There are many types of decision tree but all of them fall under two main categories based on the target variable.

- **Categorical Variable:** It refers to the variables whose target variables has definite set of values and belong to a group.
- **Continuous Variable**: It refers to the variables whose target variables can choose value from the wide range of data types.

Input Data:

Use R's built-in dataset "readingSkills" to build a decision tree. If you know the age, shoe size, score (raw score on reading test), it represents the person's reading literacy score and also if the person is native speaker or not. Figure 16.2 shows this data.

```
> library("party")
Loading required package: grid
Loading required package: mvtnorm
Loading required package: modeltools
Loading required package: stats4
Loading required package: strucchange
Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric

Loading required package: sandwich
> print(head(readingSkills))
  nativeSpeaker age shoeSize    score
1           yes   5 24.83189 32.29385
2           yes   6 25.95238 36.63105
3            no  11 30.42170 49.60593
4           yes   7 28.66450 40.28456
5           yes  11 31.88207 55.46085
6           yes  10 30.07843 52.83124
>
```

Figure 16.2: Sample data for decision tree

55

Let's use **ctree()** function on the above data set to create decision tree and its graph.

```
library(party)

# Create the input data frame.
input.data <- readingSkills[c(1:105),]

# Give the chart file a name.
png(file = "decision_tree.png")

# Create the tree.
output.tree <- ctree(
  nativeSpeaker ~ age + shoeSize + score,
  data = input.data)

# Plot the tree.
plot(output.tree)

# Save the file.
dev.off()
```

Figure 16.3: Making decision tree

Output: The ellipse in the diagram represents a node of decision tree. It shows the name of the variable and the calculated p-value. The links are marked with the cut-off values on which the decision is taken. From the decision tree of Figure 16.4, we can conclude that people whose reading skills is less than 38.306 and age more than 6 is not a native speaker. The black rectangles states that they are native speakers and the grey one shows they aren't the native speakers. The reading score greater than 38.306 determines that the probability of determining 0.6+ is "yes" (native speaker) and the remaining probability is "no" (not a native speaker). People whose age is less than 6 and reading skills greater than 30.766 are native speakers and reading skills less than equal to 30.766 are not native speakers.

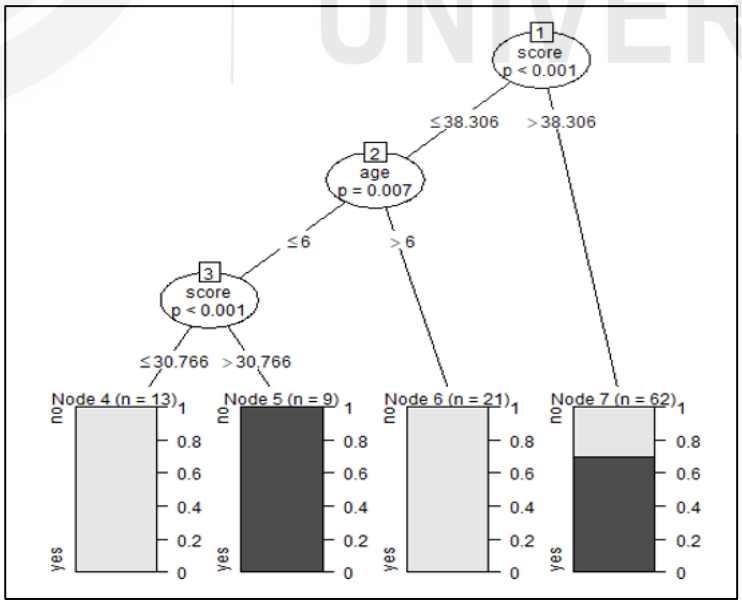


Figure 16.4: The Decision Tree for the example

Working of Decision Tree:

- **Partitioning:** Refers to the process of partitioning a dataset into subsets. The decision to make a strategic split has a significant impact on the accuracy of the tree. Many algorithms are used in the tree to divide a node into sub-nodes. As a result, the overall clarity of the node with respect to the target variable is improved. To do this, various algorithms such as chi-square and Gini coefficient are used and the most efficient algorithm is selected.

- **Pruning:** This refers to the process of turning a branch node into a leaf node and shortening the branches of a tree. The essence behind this idea is that most complex classification trees fit well into training data, but do not do the compelling task of classifying new values, thus avoiding overfitting with simpler trees. That is.

- **Tree selection:** The main goal of this process is to select the smallest tree that fits your data for the reasons described in the pruning section.

Important factors to consider when choosing a tree in R

- **Entropy:** Mainly used to determine the uniformity of a particular sample. If the sample is perfectly uniform, the entropy is 0, and if it is evenly distributed, the entropy is 1. The higher the entropy, the harder it is to draw conclusions from this information.

- **Information Gain:** Statistical property that measures how well the training samples are separated based on the target classification. The main idea behind building a decision tree is to find the attributes that provide the minimum entropy and maximum information gain. It is basically a measure of the decrease in total entropy and is calculated by taking the difference between the undivided entropy of the dataset and the average entropy after it is divided, based on the specified attribute value.

16.3 RANDOM FOREST

Random forests are a set of decision trees that are used in supervised learning algorithms for classification and regression analysis, but primarily for classification. This classification algorithm is non-linear. To achieve more accurate predictions and forecasts, Random Forest creates and combines numerous decision trees together. However, when utilised alone, each decision tree model is used. In the cases where the tree is not built, error estimation is performed. This method is termed as the out-of-bag percent error estimation.

The “Random Forest” is named ‘random’ since the predictors are chosen at random during training. It is termed as ‘forest’ because a Random Forest makes decisions based on the findings of several trees. Since multiple uncorrelated trees (models) that operate as committees are always better than individual composition models, therefore the random forests are considered to be better than the decision trees.

Random forest attempts to develop a model using samples from observations and random beginning variables (columns).

57

The random forest algorithm is:

- Draw size n bootstrap random samples (randomly select n samples from the training data).
- Build a decision tree from the bootstrap sample. Randomly select features on each tree node.
- Split the node using a feature (variable) that provides the best split according to the objective function. One such example is to maximise the information gain.
- Repeat the first two steps "k" number of times, where k represents the number of trees that you will create from subset of the sample.
- Aggregate the predictions from each tree of new data points and assign a class label by majority vote. Select the selected group in the most trees and assign new data points to that group.

Install R package

```
install.packages("randomForest")
```

Syntax:

```
randomForest(formula, data)
```

Formula is the formula which describes the variables i.e. predictor and response.

Data is the name of the dataset used.

Input Data: Use R's built-in dataset "readingSkills" to build a decision tree. If you know the age, shoe size, score (raw score on reading test), it represents the person's reading literacy score and also if the person is native speaker or not.

```
# Load the party package. It will automatically load other
# required packages.
library(party)

# Print some records from data set readingSkills.
print(head(readingSkills))
```

Output:

```
> print(head(readingSkills))
  nativeSpeaker age shoeSize    score
1           yes   5 24.83189 32.29385
2           yes   6 25.95238 36.63105
3            no  11 30.42170 49.60593
4           yes   7 28.66450 40.28456
5           yes  11 31.88207 55.46085
6           yes  10 30.07843 52.83124
```

Figure 16.5: Sample data for random forest

You can now create the random forest by applying the syntax given above and print the results

```
# Create the forest.
output.forest <- randomForest(nativeSpeaker ~ age + shoeSize + score,
                              data = readingSkills,importance=TRUE)

# View the forest results.
print(output.forest)

# Importance of each predictor.
out.importance <- round(importance(output.forest), 2)
print(out.importance )
```

Figure 16.6: Sequence of commands using R for random forest

The output of random forest is in the form of confusion matrix. Therefore, before showing the output, let us discuss about confusion matrix.

Confusion matrix:

A confusion matrix is a performance measurement for machine learning classification problems where output can be two or more classes.

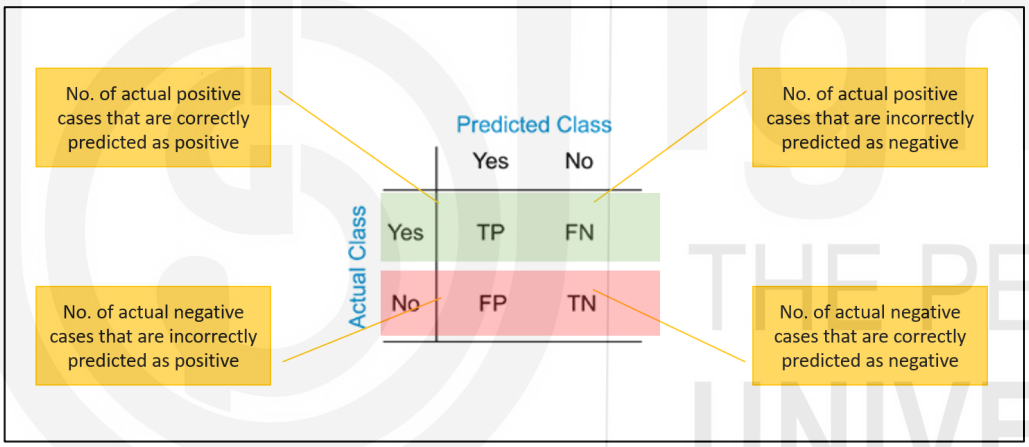


Figure 16.7: The Confusion Matrix

Confusion Matrix Example:

	Positive (Predicted)	Negative (Predicted)
Positive (Actual)	100	50
Negative (Actual)	150	9700

Figure 16.8: Example of Confusion Matrix

Please note the following for the confusion matrix of Figure 16.8

1. **Total number of observations** = (100 + 50 + 150 + 9700) = 10000

59

2. **Total number of positive cases (Actual) =** (100 + 50) = 150
3. **Total number of negative cases (Actual)** = (150 + 9700) = 9850
4. **TP = 100, i.e.**100 out of 150 positive cases are correctly predicted as positive.
5. **FN = 50, i.e**. 50 out of 150 positive cases are incorrectly predicted as negative.
6. **FP = 150, i.e.**150 out of 9850 negative cases are incorrectly predicted as positive.
7. **TN = 9700, i.e.**9700 out of 9850 negative cases are correctly predicted as negative**.**

Output:

```
Call:
 randomForest(formula = nativeSpeaker ~ age + shoeSize + score,      da
ta = readingSkills)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 1.5%
Confusion matrix:
    no yes class.error
no  99   1        0.01
yes  2  98        0.02
```

```
> # Importance of each predictor.
> out.importance <- round(importance(output.forest), 2)
> print(out.importance )
              no   yes MeanDecreaseAccuracy MeanDecreaseGini
age        36.85 33.37                49.33            13.07
shoeSize   17.85 15.73                25.54            18.88
score      80.29 72.63                92.63            57.80
> |
```

Figure 16.9: Output of Random Forest

Where, no is not a native speaker and yes is a native speaker. Mean Decrease Accuracy express how much accuracy a model loses excluding each variable and MeanDecreaseGini tells how each variable contributes to the homogeneity of the nodes.

From the Random Forest above, we can conclude that shoe size and score are important factors in determining if someone is native. As the values of MeanDecreaseGini is lower that means higher the purity and hence the 2 independent variables turns out to be important. Also, the model error is only 1%. This means that you can predict with 99% accuracy.

Check Your Progress 1

1. Can Random Forest Algorithm be used both for Continuous and Categorical Target Variables?

……

……

2.What is Out-of-Bag Error?

……

……

3. What does random refer to in 'Random Forest'?

……

……

16.4 CLASSIFICATION

The idea of a classification algorithm is very simple. Predict the target class by analysing the training dataset. Use the training dataset to get better boundary conditions that you can use to determine each target class. Once the constraints are determined, the next task is to predict the target class. This entire process is called classification.

The classification algorithm has some important points.

- **Classifier:** This is an algorithm that assigns input data to a specific category. Classification model. The classification model attempts to draw some conclusions from the input values given to the training. This inference predicts the class label / category of new data.
- **Characteristic**: This is an individually measurable property of the observed event.
- **Binary classification**: This is a classification task with two possible outcomes. For example, a gender classification with only two possible outcomes i.e. Men and women.
- **Multi-class classification:** This is a classification task where classification is done in three or more classes. Here is an example of a multiclass classification: A classifier can recognise a digit only as one of the digit classes say 0 or 1 or 2…or 9.
- **Multi-label classification:** This is a classification task where each sample is assigned a set of target labels. Here is an example of a multi-label classification: A news article that can be classified by a multi-label classifier as *people*, *places*, and *sports* at the same time.

In R, classification algorithms can be broadly divided into the following types.

Linear classifier

In machine learning, the main task of statistical classification is to use the properties of an object to find the class to which the object belongs. This task is solved by determining the classification based on the value of the linear combination of features. R has two linear classification algorithms:

- Logistic regression
- Naive Bayes classifier

61

Support vector machine

Support vector machines are supervised learning algorithms that analyse the data used for classification and regression analysis. In SVM, each data element is represented as a value for each attribute, that is, a point in n-dimensional space with a value at a particular coordinate. The least squares support vector machine is the most used classification algorithm in R.

Decision tree

The decision tree is a supervised learning algorithm used for classification and regression tasks. In R, the decision tree classifier is implemented using the R machine learning cullet package. The Random Forest algorithm is the most used decision tree algorithm in R.

16.5 CLUSTERING

Clustering in the R programming language is an unsupervised learning technique that divides a dataset into multiple groups and is called a cluster because of its similarities. After segmenting the data, multiple data clusters are generated. All objects in the cluster have common properties. Clustering is used in data mining and analysis to find similar datasets.

Clustering application in the R programming language

- **Marketing**: In R programming, clustering is useful for marketing. This helps identify market patterns and, therefore, find potential buyers. By identifying customer interests through clustering and displaying the same products of interest, you can increase your chances of buying a product.
- **Internet**: Users browse many websites based on their interests. Browsing history can be aggregated and clustered, and a user profile is generated based on the results of the clustering.
- **Games**: You can also use clustering algorithms to display games based on your interests.
- **Medicine**: In the medical field, every day there are new inventions of medicines and treatments. Occasionally, new species are discovered by researchers and scientists. Those categories can be easily found by using a clustering algorithm based on their similarity.

Clustering method

There are two types of clustering in R programming.

- **Hard clustering**: With this type of clustering, data points are assigned to only one cluster, whether they belong entirely to the cluster. The algorithm used for hard clustering is k-means clustering.
- **Soft clustering**: Soft clustering assigns the probabilities or possibilities of data points in a cluster rather than placing each data point in a cluster. All data points have a certain probability of being present in all clusters. The algorithm used for soft clustering is fuzzy clustering or soft k-means.

K-Means clustering in the R programming language

K-Means is an iterative hard clustering technique that uses an unsupervised learning algorithm. The total number of clusters is predefined by the user, and the data points are clustered based on the similarity of each data point. This algorithm also detects the center of gravity of the cluster.

Algorithm: Specifying the number of clusters (k): Let's look at an example of k = 2 and 5 data points. Randomly assign each data point to the cluster. Assume, the yellow and blue colors show two clusters with their respective random data points assigned. Calculate the centroid of a cluster: Remap each data point to the nearest cluster centroid. The blue data point is assigned to the yellow cluster because it is close to the center of gravity of the yellow cluster. Refactor the cluster centroid fuzzy clustering method or soft-k-means.

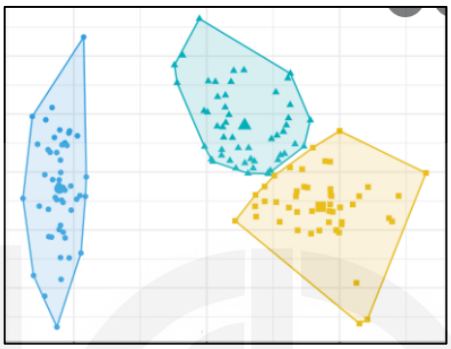


Figure 16.10: Clustering

Syntax: kmeans(x, centers, nstart)
where,

- x represents numeric matrix or data frame object
- centers represents the K value or distinct cluster centers
- nstart represents number of random sets to be chosen

Input Data and loading the necessary packages in R. For the clustering, we are using iris data set. The dataset contains 3 classes each of around 50 instances and the class refers to a type of iris plant.

```
data(iris)
str(iris)

# Installing Packages
install.packages("ClusterR")
install.packages("cluster")

# Loading package
library(ClusterR)
library(cluster)
```

Fitting the K means clustering model

63

```
# Removing initial label of
# Species from original dataset
iris_df <- iris[, -5]

# Fitting K-Means clustering Model
# to training dataset
set.seed(240) # Setting seed
kmeans.res <- kmeans(iris_df, centers = 3, nstart = 20)
kmeans.res
```

Result:

```
K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
 [53] 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3
[105] 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

The 3 clusters are made which are of 50, 62, and 38 sizes respectively. Within the cluster, the sum of squares is 88.4%.

Confusion Matrix:

```
> # Confusion Matrix
> cmax <- table(iris$Species, kmeans.re$cluster)
> cmax

            1  2  3
  setosa     50  0  0
  versicolor  0 48  2
  virginica   0 14 36
```

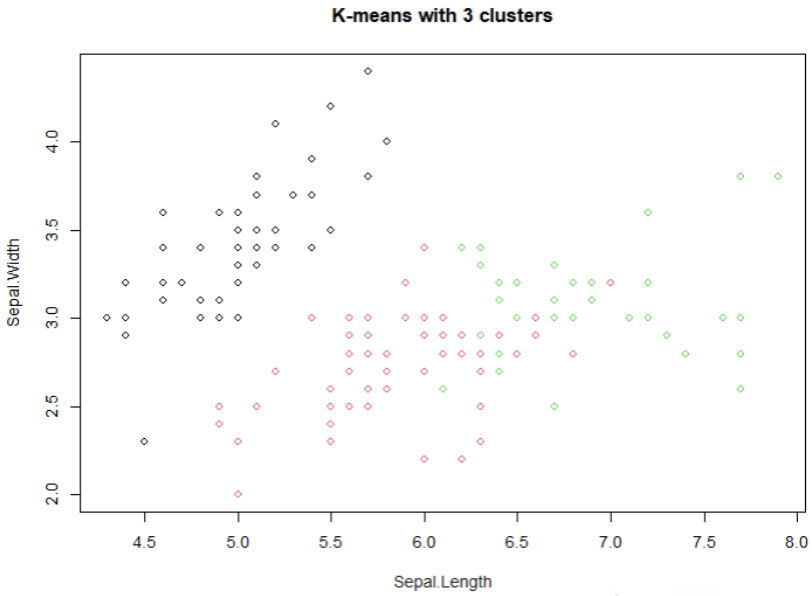
Confusion matrix suggests that 50 setosa are correctly classified as setosa. Out of 62 versicolor, 14 are incorrectly classified as virginica and 48 correctly as versicolor. Out of 38 virginica, 2 are incorrectly classified as versicolor and 36 correctly classified as virginica.

Model Evaluation and Visualization

Code:

```
# Model Evaluation and visualization
plot(iris_df[c("Sepal.Length", "Sepal.Width")])
plot(iris_df[c("Sepal.Length", "Sepal.Width")],
     col = kmeans.re$cluster)
plot(iris_df[c("Sepal.Length", "Sepal.Width")],
     col = kmeans.re$cluster,
     main = "K-means with 3 clusters")
```

Output:



The plot shows 3 cluster plots with 3 different colors.

Plotting Cluster Centres

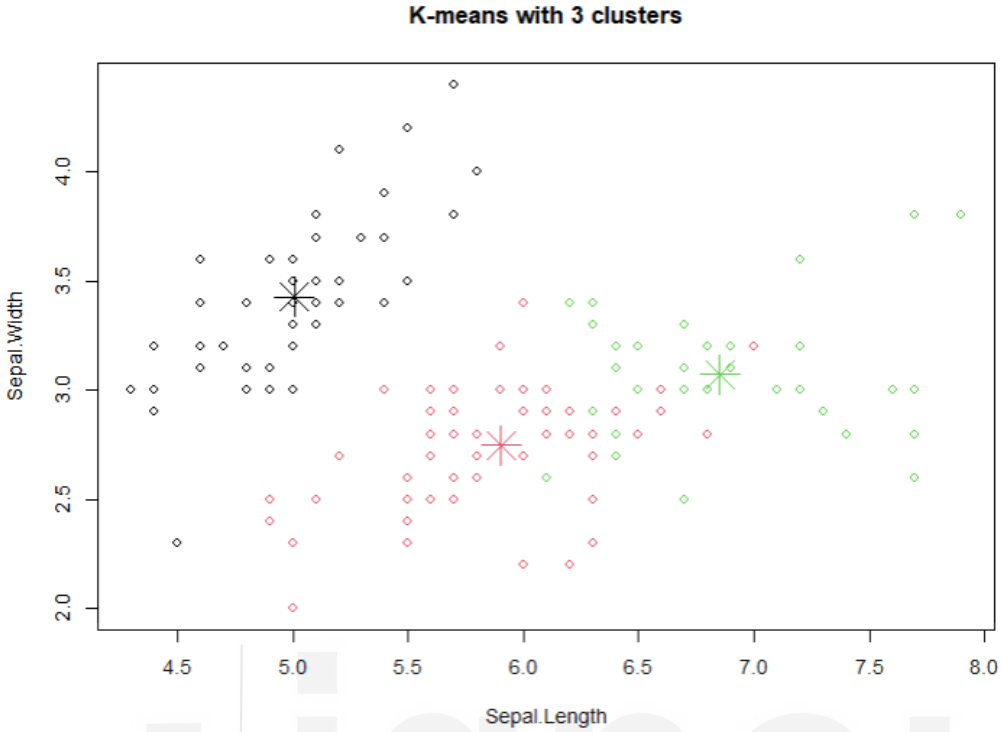
Code:

```
## Plotiing cluster centers
kmeans.res$centers
kmeans.res$centers[, c("Sepal.Length", "Sepal.Width")]

# cex is font size, pch is symbol
points(kmeans.res$centers[, c("Sepal.Length", "Sepal.Width")],
       col = 1:3, pch = 8, cex = 3)
```

Output:

65



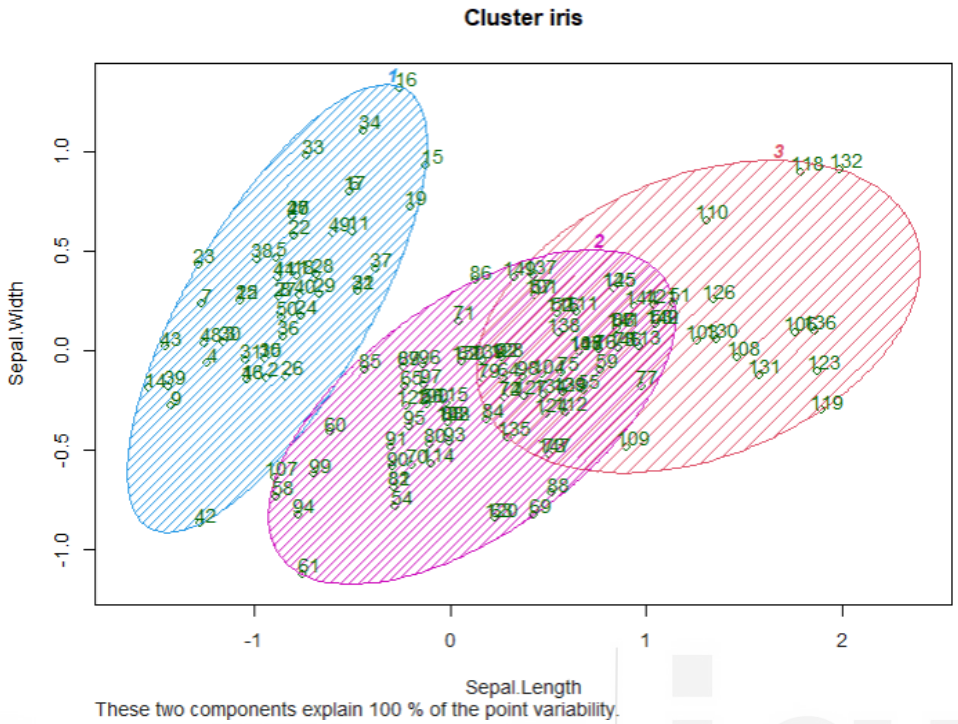
The plot shows the center of the clusters which are marked as cross sign with same color of the cluster.

Visualizing Clusters

Code:

```
## Visualizing clusters
y_kmeans <- kmeans.res$cluster
clusplot(iris_df[, c("Sepal.Length", "Sepal.Width")],
         y_kmeans,
         lines = 0,
         shade = TRUE,
         color = TRUE,
         labels = 2,
         plotchar = FALSE,
         span = TRUE,
         main = paste("Cluster iris"),
         xlab = 'Sepal.Length',
         ylab = 'Sepal.Width')
```

Output:



The plot showing 3 clusters formed with varying sepal length and sepal width.

Check Your Progress 2

1. What is the difference between Classification and Clustering?

   ……

   ……

2. Can decision trees be used for performing clustering?

   ……

   ……

3. What is the minimum no. of variables/ features required to perform clustering?

   ……

   ……

16.6 ASSOCIATION RULES

Association Rule Mining in R Language is an Unsupervised Non-linear algorithm to discover how any item is associated with other. Frequent Mining shows which items appear together in a transaction. Major usage is in Retail, grocery stores, an online platform i.e. those having a large transactional database. The same way when any online social media or e-commerce websites know what you buy next using recommendations engines. The recommendations you get on item, while you check out the order is because of Association rule mining boarded on past user data. There are three common ways to measure association:

- Support

67

- Confidence
- Lift

Theory
In association rule mining, Support, Confidence, and Lift measure association.

[E1] Buy Product A => [E2] Buy Product B
Support (Rule) = P(E1 and E2) = Probability of Buying both the products A and B.
Confidence (Rule) = P(E2|E1) = Probability of buying the product B given that product A has already been bought.

Interpreting Support & Confidence of a Rule:

Computer => Antivirus software [support = 2%, confidence = 60%]
Computer: Antecedent**& Antivirus Software:** Consequence

Support: 2% of all the transaction under analysis show that computer and antivirus software are purchased together.
Confidence: 60% of the customers who purchased a computer also bought the software.

Lift: A measure of Association i.e. the occurrence of itemset A is independent of the occurrence of itemset B if
P(A and B) = P(A).P(B)

$$\frac{P(A \text{ and } B)}{P(A).P(B)} = 1$$

If <1, if buying A & buying B are negatively associated.
If =1, if buying A & buying B are not associated.
If >1, if buying A & buying B are positively associated.

Packages in R:
Installing Packages
install.packages("arules")
install.packages("arulesViz")

Syntax:
associa_rules = apriori(data = dataset, parameter = list(support = x,
confidence = y))

Installing the relevant packages and the dataset

```
install.packages("arules")
install.packages("arulesViz")
library(arules)
library(arulesViz)

data("Groceries")
```

The first 2 transactions and the items involved in each transaction can be observed below.

```
> inspect(head(Groceries, 2))
    items
[1] {citrus fruit, semi-finished bread, margarine, ready soups}
[2] {tropical fruit, yogurt, coffee}
```

```
> grocery_rules <- apriori(Groceries, parameter = list(support = 0.01, confidence = 0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.5    0.1    1 none FALSE            TRUE       5    0.01      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 98

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [15 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The algorithm generated 15 rules with the given constraints.
Minval is the minimum value of support of an itemset which should be satisfied to be a part of a rule.
Smax: maximum support value.
Arem is additional rule evaluation parameter. We have fed Support and confidence as the constraints. There are several other rules using the arem parameter.

The top 3 rules sorted by confidence are shown below:

```
> inspect(head(sort(grocery_rules, by = "confidence"), 3))
    lhs                            rhs             support    confidence coverage   lift     count
[1] {citrus fruit, root vegetables} => {other vegetables} 0.01037112 0.5862069  0.01769192 3.029608 102
[2] {tropical fruit, root vegetables} => {other vegetables} 0.01230300 0.5845411  0.02104728 3.020999 121
[3] {curd, yogurt}                  => {whole milk}       0.01006609 0.5823529  0.01728521 2.279125  99
```

16.7 SUMMARY

This unit explores the concepts of advance data analysis and their application in R. It explains about the Decision Tree model and its application in R. It represents decisions and their results in a tree format. There are various types of decision trees that may fall under the 2 categories based on the target variables, i.e. categorical and continuous variable. Partitioning, pruning and the tree selection are the steps involved in the working of the decision tree. The other factors to consider when choosing a tree in R, are- entropy and information gain. The unit further explains in detail the concept of Random and its application in R. Random forests are a set of decision trees that are used in supervised learning algorithms for classification and regression analysis, but primarily for classification. It is a non-linear classification algorithm and takes samples from observations and random initial variables (columns) and attempts to build a model. In the subsequent sections, the unit explains the details of classification algorithm and its features and types. In R, the types of classification algorithms are- Linear classifier, Support vector machine and the decision tree. The linear classification algorithms can further be of 2 types- logistic regression and Naive Bayes classifier. It further explains the about Clustering and its application in R programming. Clustering in the R programming language is an unsupervised learning technique that divides a dataset into multiple groups and is called a

69

cluster because of its similarities. It is used in data mining and analysis to find similar datasets ad has application in marketing, internet, gaming, medicine, etc. There are two types of clustering in R programming- namely, hard clustering and soft clustering. The unit also discusses K-Means clustering in the R programming language which is an iterative hard clustering technique that uses an unsupervised learning algorithm. The unit discusses the theory of the Association Rules and its application in R. It is an Unsupervised Non-linear algorithm to discover how any item is associated with other and has a major usage in Retail, grocery stores, an online platform i.e. those having a large transactional database. In association rule mining, support, confidence, and lift measure the association.

16.8 ANSWERS

Check Your Progress 1

1. Yes, Random Forest can be used for both continuous as well as categorical target (dependent) variables.
   A random forest i.e., the combination of decision trees, the classification model refers to the categorical dependent variable, and the regression model refers to the numeric or continuous dependent variable. But random forest is mainly used for Classification problems.

2. Out of Bag supports validation or test data. Random forests do not require a separate test dataset to validate the results. It is calculated internally during the execution of the algorithm in the following way: Because the forest is built on training data, each tree is tested on 1/3 (36.8%) of the samples that were not used to build that tree (similar to the validation dataset). This is known as the out-of-bag error estimation. This is simply an internal error estimate for the random forest you are building.

3. Random forest is one of the most popular and widely used machine learning algorithms in classification problems. It can also be used for regression problem statements, but it works mostly well with classification models.
   Improvements to the predictive model have become a deadly weapon for modern data scientists. The best part of the algorithm is that there are very few assumptions involved. Therefore, preparing the data is not too difficult and saves time.

Check Your Progress 2

1. **Classification** is taking data and putting it into **pre-defined categories** and in **Clustering** the set of categories, that you want to group the data into, **is not known** beforehand.

2.True, Decision trees can also be used to for clusters in the data, but clustering often generates natural clusters and is not dependent on any objective function.

3. At least a single variable is required to perform clustering analysis. Clustering analysis with a single variable can be visualized with the help of a histogram.

16.9 REFERENCES AND FURTHER READINGS

1. De Vries, A., & Meys, J. (2015). *R for Dummies*. John Wiley & Sons.
2. Peng, R. D. (2016). *R programming for data science* (pp. 86-181). Victoria, BC, Canada: Leanpub.
3. Schmuller, J. (2017). *Statistical Analysis with R For Dummies*. John Wiley & Sons.
4. Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. Sage publications.
5. Lander, J. P. (2014). *R for everyone: Advanced analytics and graphics*. Pearson Education.
6. Lantz, B. (2019). *Machine learning with R: expert techniques for predictive modeling*. Packt publishing ltd.
7. Heumann, C., & Schomaker, M. (2016). *Introduction to statistics and data analysis*. Springer International Publishing Switzerland.
8. Davies, T. M. (2016). *The book of R: a first course in programming and statistics*. No Starch Press.
9. https://www.tutorialspoint.com/r/index.html
10. https://www.guru99.com/r-decision-trees.html
11. https://codingwithfun.com/p/r-language-random-forest-algorithm/
12. https://towardsdatascience.com/association-rule-mining-in-r-ddf2d044ae50
13. https://www.geeksforgeeks.org/k-means-clustering-in-r-programming/

71