PRACTICAL NO:- 01

Aim: A) Data Preprocessing.

B) Classification (Decision Tree)

Software Used: R Studio

Description:

1) Data Cleaning:-

Data cleaning is the process of identifying, correcting, or removing inaccurate raw data for downstream purposes. It is an important step in any data analysis project, as it can help to ensure the accuracy and reliability of the results. There are many different data cleaning tasks that can be performed in R. Some of the most common tasks include:

- **Detecting and removing missing values:** Missing values can occur for a variety of reasons, such as human error, data entry errors, or incomplete surveys. There are a number of different ways to detect missing values in R, such as the is.na() function. Once missing values have been detected, they can be removed, replaced with imputed values, or flagged for further investigation.
- **Detecting and removing outliers:** Outliers are data points that are significantly different from the rest of the data. They can be caused by a variety of factors, such as data entry errors, measurement errors, or natural variation. Outliers can distort the results of statistical analysis, so it is important to detect and remove them. There are a number of different ways to detect outliers in R, such as the boxplot() function. Once outliers have

been detected, they can be removed, replaced with imputed values, or flagged for further investigation.

- **Cleaning up data formats:** Data can come in a variety of formats, such as strings, numbers, dates, and times. It is important to clean up the data formats so that they are consistent and easy to work with. This can be done using the gsub() function, for example.
- **Enriching data:** In some cases, it may be necessary to enrich the data by adding additional information. This can be done by merging data from different sources, or by using external data sources.
- **Validating data:** Once the data has been cleaned, it is important to validate it to ensure that it is accurate and reliable. This can be done by checking for consistency, completeness, and accuracy.

Program Code:-

```
head(airquality)

mean(airquality)

mean(airquality$Solar.R, na.rm = TRUE)

New_df = airquality

New_df$Ozone = ifelse(is.na(New_df$Ozone),

                median(New_df$Ozone,

                    na.rm = TRUE),

                New_df$Ozone)

head(New_df)

##create excel file with two columns roll,marks keep few marks blank.save it as csv file

dt=read.csv("C:/Users/DELL/Desktop/titanic.csv")
```

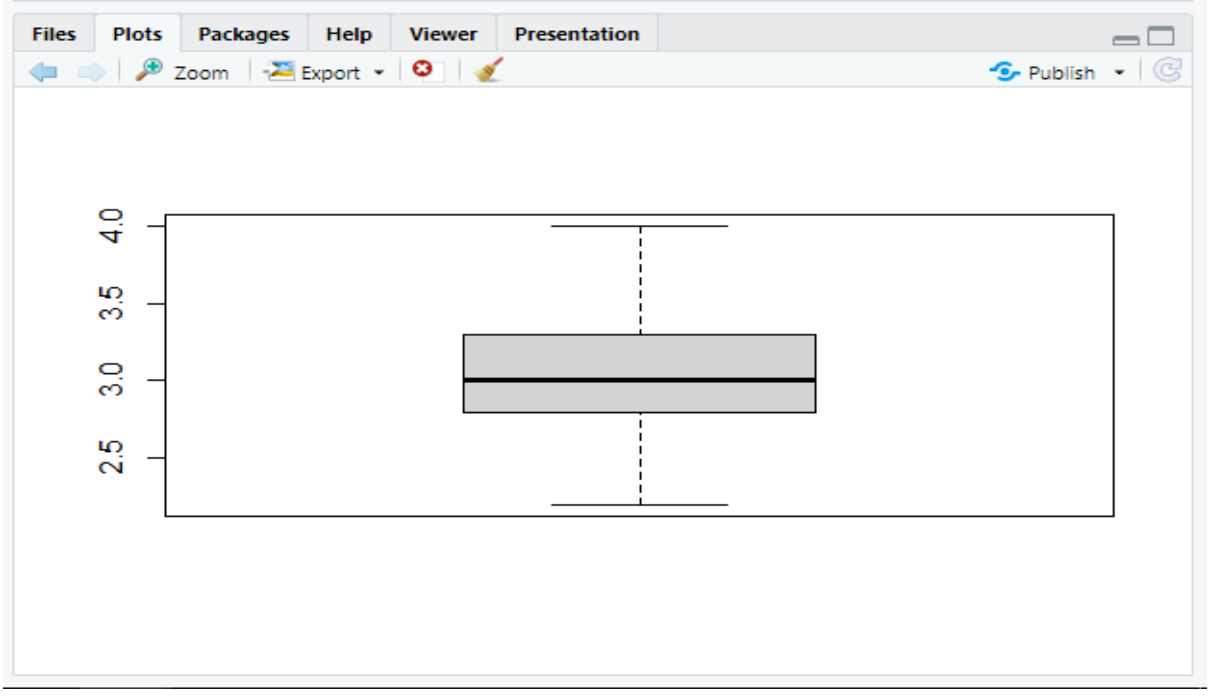
```
head(dt)
dt$marks = ifelse(is.na(dt$marks),
                  mean(dt$marks,
                       na.rm = TRUE),
                  dt$marks)
#Removing outliers using boxplot
data <- iris[,2]
length(data)
boxplot(data)
boxplot(data, plot = FALSE)$out
outliers <- boxplot(data, plot = FALSE)$out
data_no_outlier <- data[-which(data %in% outliers)]
boxplot(data_no_outlier, plot = FALSE)$out
length(data_no_outlier)
boxplot(data_no_outlier)
```

Output:

```
Console  Terminal  Background Jobs
R 4.2.2 · ~/

R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> mean(airquality)
[1] NA
Warning message:
In mean.default(airquality) :
  argument is not numeric or logical: returning NA
> mean(airquality$Solar.R, na.rm = TRUE)
[1] 185.9315
> New_df = airquality
> New_df$Ozone = ifelse(is.na(New_df$Ozone),
+                       median(New_df$Ozone,
+                              na.rm = TRUE),
+                       New_df$Ozone)
> head(New_df)
  Ozone Solar.R Wind Temp Month Day
1  41.0     190  7.4   67     5   1
2  36.0     118  8.0   72     5   2
3  12.0     149 12.6   74     5   3
4  18.0     313 11.5   62     5   4
5  31.5      NA 14.3   56     5   5
6  28.0      NA 14.9   66     5   6
> ##create excel file with two columns roll,marks keep few marks blank.save it as csv f
ile
> dt=read.csv("C:/Users/DELL/Desktop/titanic.csv.xlsx")
Warning messages:
1: In read.table(file = file, header = header, sep = sep, quote = quote,  :
  line 1 appears to contain embedded nulls
2: In read.table(file = file, header = header, sep = sep, quote = quote,  :
  incomplete final line found by readTableHeader on 'C:/Users/DELL/Desktop/titanic.csv.
xlsx'
> ##create excel file with two columns roll,marks keep few marks blank.save it as csv f
ile
> dt=read.csv("C:/Users/DELL/Desktop/titanic.csv.xlsx")
Warning messages:
1: In read.table(file = file, header = header, sep = sep, quote = quote,  :
  line 1 appears to contain embedded nulls
2: In read.table(file = file, header = header, sep = sep, quote = quote,  :
  incomplete final line found by readTableHeader on 'C:/Users/DELL/Desktop/titanic.csv.
xlsx'
> head(dt)
[1] PK...
```

```
Warning messages:
1: In read.table(file = file, header = header, sep = sep, quote = quote,  :
  line 1 appears to contain embedded nulls
2: In read.table(file = file, header = header, sep = sep, quote = quote,  :
  incomplete final line found by readTableHeader on 'C:/Users/DELL/Desktop/titanic.csv.
xlsx'
> head(dt)
[1] PK...
<0 rows> (or 0-length row.names)
> dt$marks = ifelse(is.na(dt$marks),
+                   mean(dt$marks,
+                        na.rm = TRUE),
+                   dt$marks)
> #Removing outliers using boxplot
> data <- iris[,2]
> length(data)
[1] 150
> boxplot(data)
> boxplot(data, plot = FALSE)$out
[1] 4.4 4.1 4.2 2.0
> outliers <- boxplot(data, plot = FALSE)$out
> data_no_outlier <- data[-which(data %in% outliers)]
> boxplot(data_no_outlier, plot = FALSE)$out
numeric(0)
> length(data_no_outlier)
[1] 146
> boxplot(data_no_outlier)
>
```

na.rm is a function that is used to remove missing values when calculating descriptive statistics. The default value of na.rm is FALSE, which means that missing values are included when calculating descriptive statistics. However, if na.rm is set to TRUE, then missing values are excluded when calculating descriptive statistics.



2) Correlation:

Correlaion is a statistical measure that indicates how strongly two variables are related. It involves the relationship between multiple variables. The cor() function calculates the correlation coefficient between two variables.

Program Code:-

```
# Load the mtcars dataset
data(mtcars)

# Calculate the Pearson's correlation coefficient between mpg and hp
correlation = cor(mtcars$mpg, mtcars$hp, method = "pearson")

# Print the correlation coefficient
print(correlation)
```

Output:-

```
> # Load the mtcars dataset
> data(mtcars)
> # Calculate the Pearson's correlation coefficient between mpg and hp
> correlation = cor(mtcars$mpg, mtcars$hp, method = "pearson")
> # Print the correlation coefficient
> print(correlation)
[1] -0.7761684
>
```

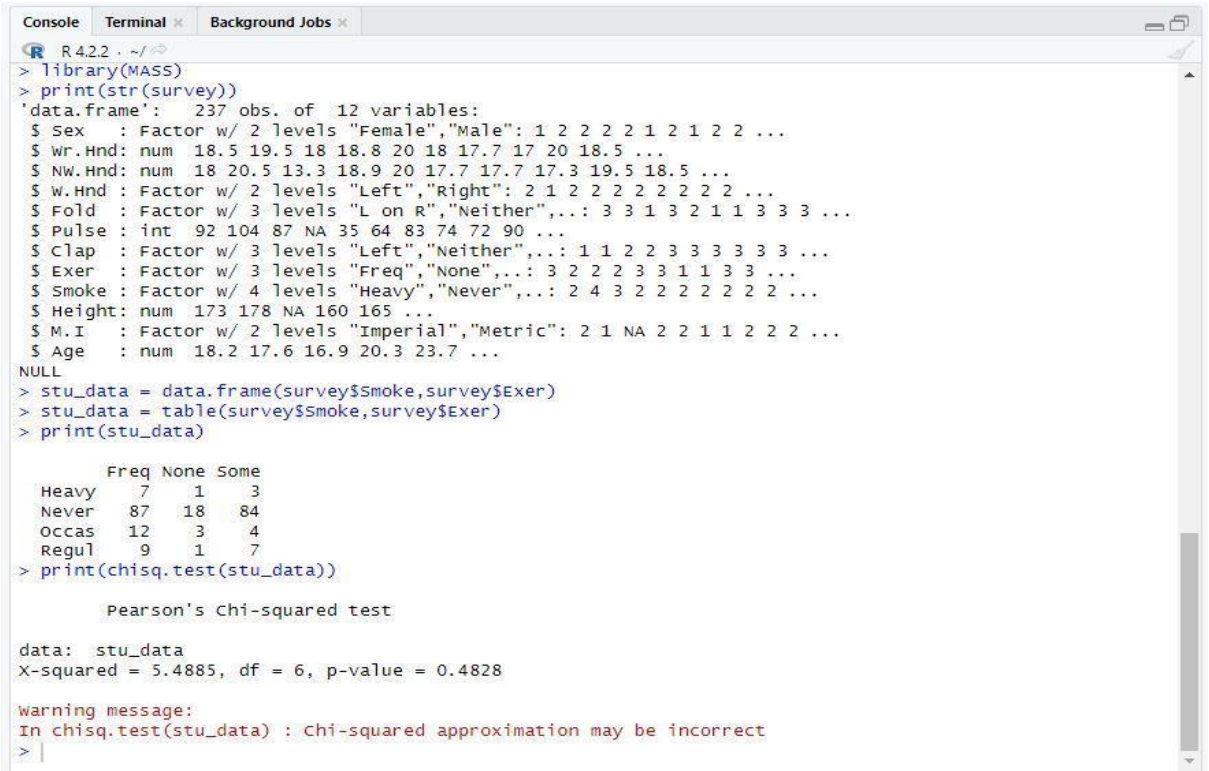
3) Chi Square:-

The chi-square test of independence evaluates whether there is an association between the categories of the two variables. There are basically two types of random variables and they yield two types of data: numerical and categorical. Chi-square statistics is used to investigate whether distributions of categorical variables differ from one another. The chisq.test() function performs the chi-square test of independence on a contingency table.

Program Code:-

```
library(MASS)
print(str(survey))

stu_data = data.frame(survey$Smoke,survey$Exer)
stu_data = table(survey$Smoke,survey$Exer)
print(stu_data)

print(chisq.test(stu_data))
```

Output:-



4) Graph Plots:-

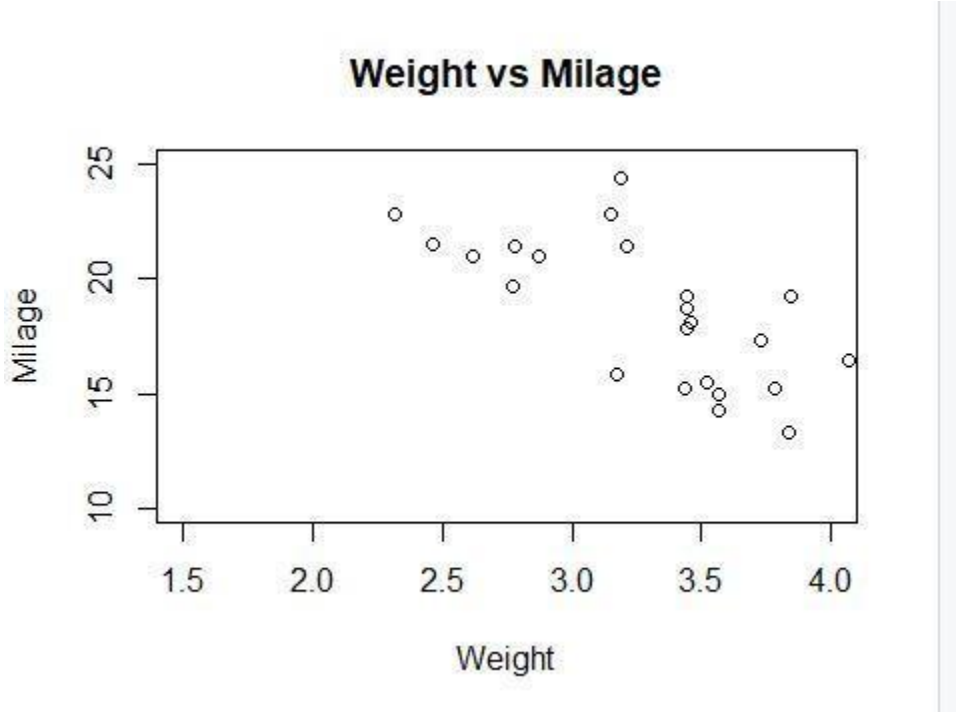
i)Scatter plot:-

A scatter plot is a set of dotted points representing individual data pieces on the horizontal and vertical axis. In a graph in which the values of two variables are plotted along the X-axis and Y-axis, the pattern of the resulting points reveals a correlation between them.Scatterplots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.

Program Code:-

```
> # Get the input values.
> input <- mtcars[, c('wt', 'mpg')]
> # Plot the chart for cars with
> # weight between 1.5 to 4 and
> # mileage between 10 and 25.
> plot(x = input$wt, y = input$mpg,
+      xlab = "Weight",
+      ylab = "Milage",
+      xlim = c(1.5, 4),
+      ylim = c(10, 25),
+      main = "Weight vs Milage"
+ )
```

Output:-



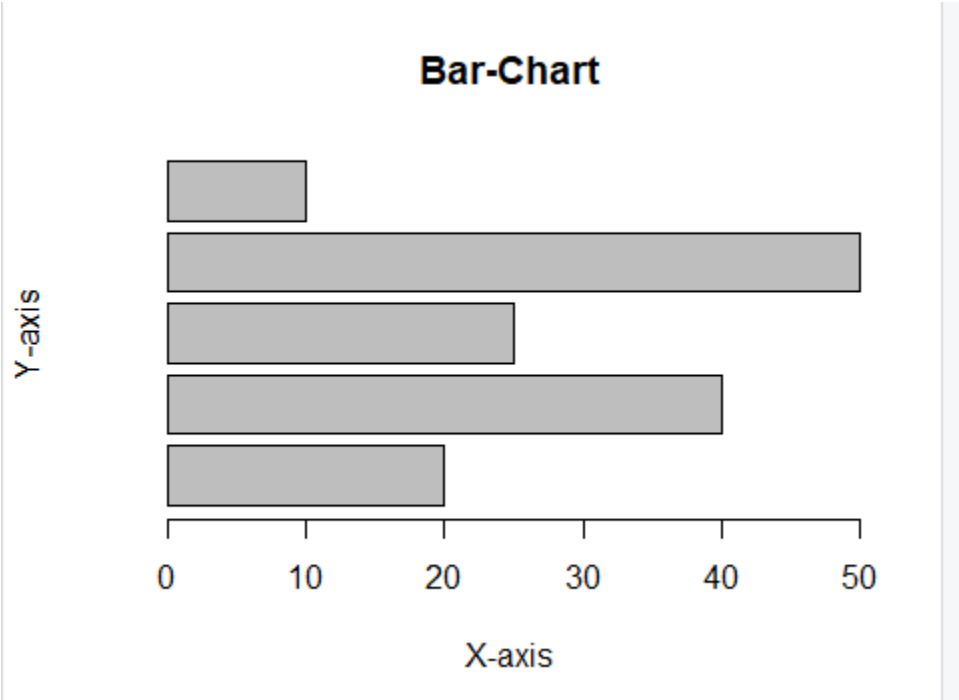
ii) Bar chart:-

A bar chart is a pictorial representation of data that presents categorical data with rectangular bars with heights or lengths proportional to the values that they

represent. In other words, it is the pictorial representation of dataset. These data sets contain the numerical values of variables that represent the length or height.

Program Code:-

```
> # Create the data for the chart
> A <- c(20, 40, 25, 50, 10)
> # Plot the bar chart
> barplot(A, horiz = TRUE, xlab = "X-axis",
+         ylab = "Y-axis", main ="Bar-Chart")
```



B)Classification:-

Decision Tree:-

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Construction of Decision Tree: A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data. In general, the decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision Tree Representation: Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute This process is then repeated for the subtree rooted at the new node.

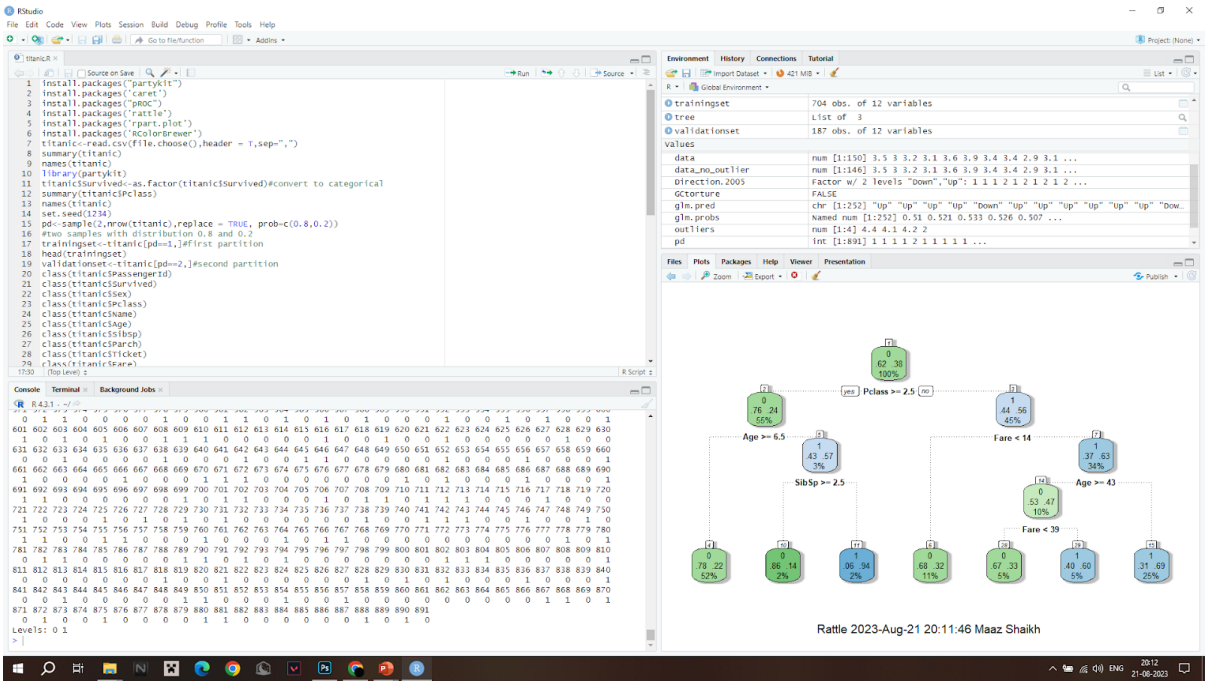
Dataset:- Passengers of The Titanic

Program Code:-

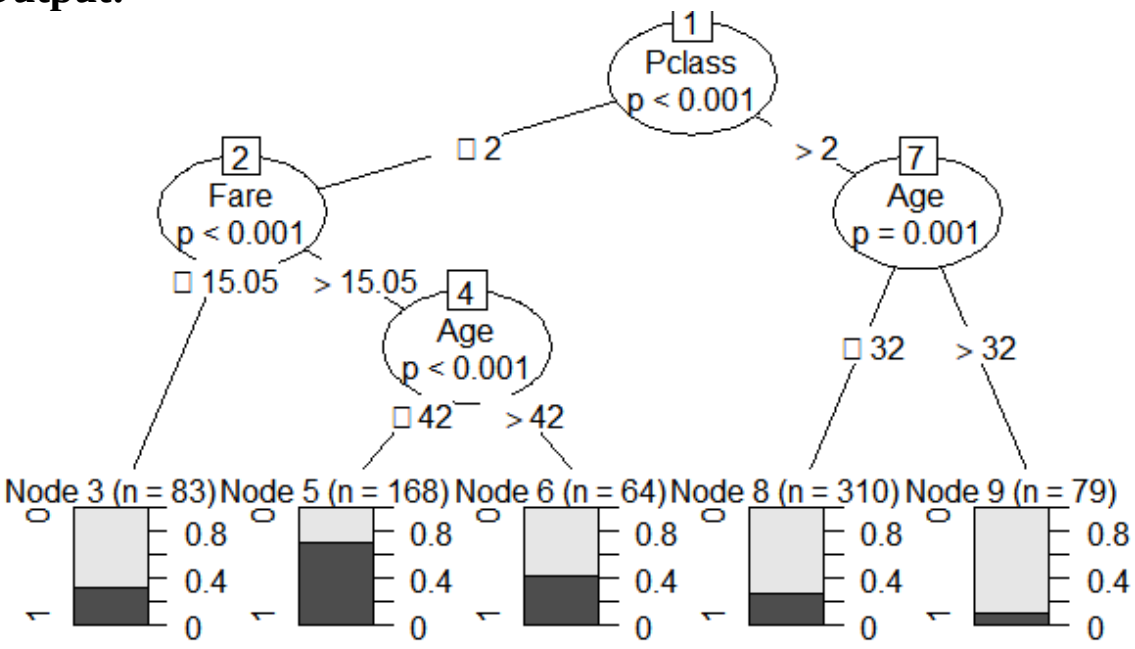
```
install.packages("partykit")
install.packages('caret')
install.packages("pROC")
install.packages('rattle')
install.packages('rpart.plot')
install.packages('RColorBrewer')
titanic<-read.csv(file.choose(),header = T,sep=",")
summary(titanic)
names(titanic)
library(partykit)
titanic$Survived<-as.factor(titanic$Survived)#convert to categorical
summary(titanic$Pclass)
names(titanic)
set.seed(1234)
pd<-sample(2,nrow(titanic),replace = TRUE, prob=c(0.8,0.2))
#two samples with distribution 0.8 and 0.2
trainingset<-titanic[pd==1,]#first partition
```

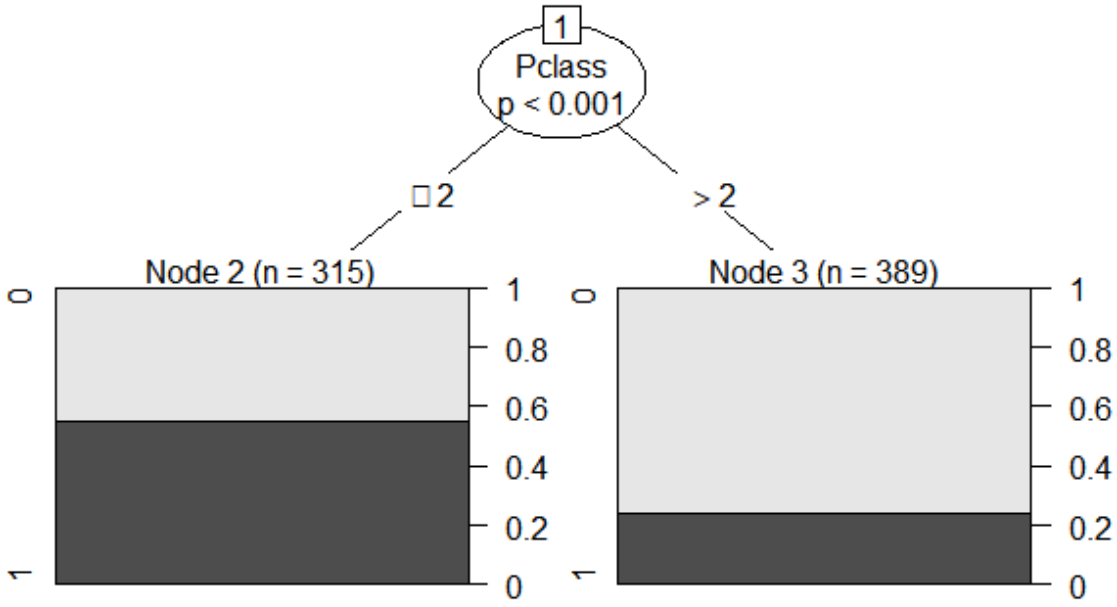
```
head(trainingset)
validationset<-titanic[pd==2,]#second partition
class(titanic$PassengerId)
class(titanic$Survived)
class(titanic$Sex)
class(titanic$Pclass)
class(titanic$Name)
class(titanic$Age)
class(titanic$SibSp)
class(titanic$Parch)
class(titanic$Ticket)
class(titanic$Fare)
class(titanic$Cabin)
class(titanic$Embarked)
tree<-ctree(formula = Survived ~ Pclass + Age + SibSp + Parch + Fare
,data=trainingset)
class(titanic$Survived)
plot(tree)
#Prunning
tree<-ctree(formula = Survived ~ Pclass + Age + SibSp + Parch +
            Fare ,data=trainingset,control=ctree_control(mincriterion =
                                                  0.99,minsplit = 500))
plot(tree)
pred<-predict(tree,validationset,type="prob")
pred
pred<-predict(tree,validationset)
pred
library(caret)
confusionMatrix(pred,validationset$Survived)
pred<-predict(tree,validationset,type="prob")
pred
library(pROC)
plot(roc(validationset$Survived,pred[ ,2]))
library(rpart)
fit <- rpart(Survived ~ Pclass+ Age + SibSp + Parch +
              Fare,data=titanic,method="class")
plot(fit)
text(fit)

library(rattle)
```

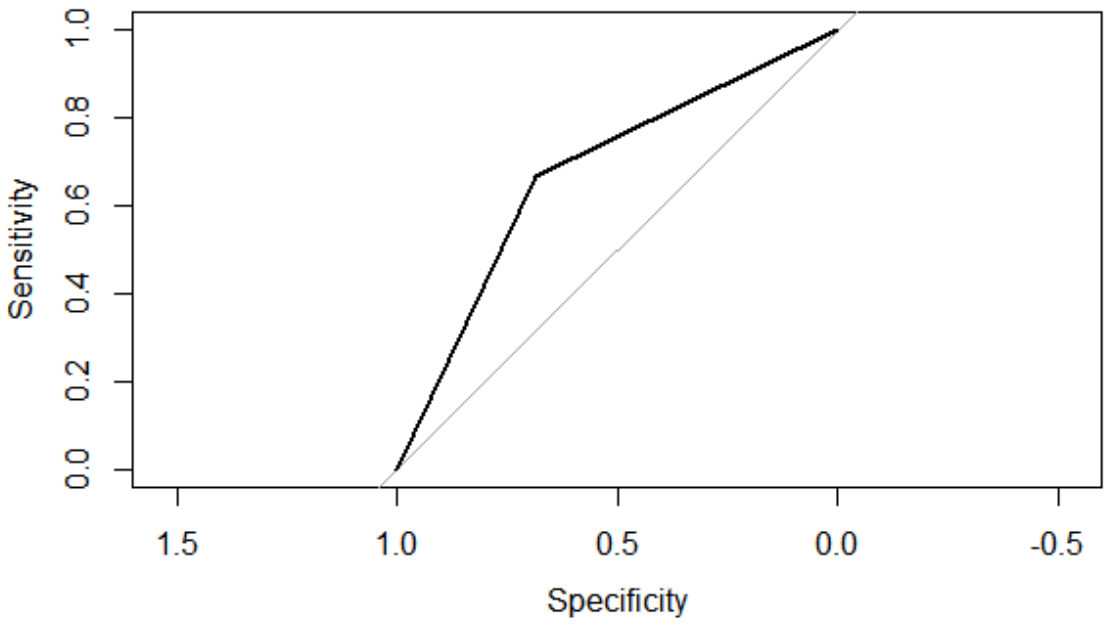
```
library(rpart.plot)
library(RColorBrewer)
fancyRpartPlot(fit)
Prediction <- predict(fit, titanic, type = "class")
Prediction
```

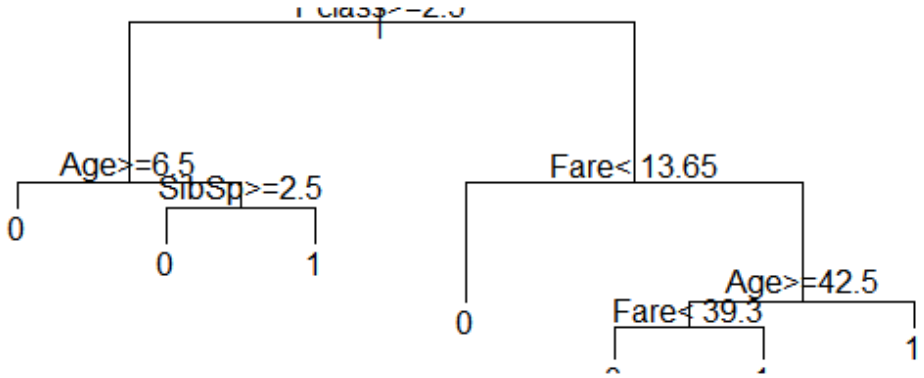


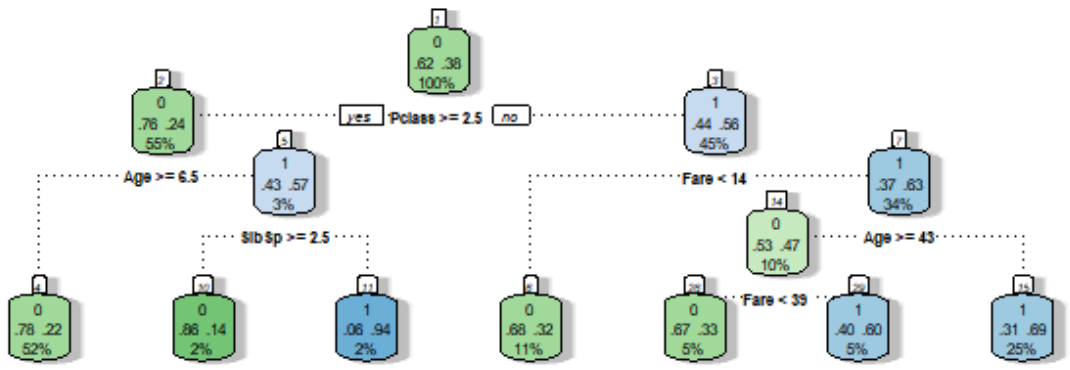
Output:


1
Pclass
p < 0.001
□ 2
> 2
2
Fare
p < 0.001
7
Age
p = 0.001
□ 15.05
> 15.05
4
Age
p < 0.001
□ 32
> 32
□ 42
> 42
Node 3 (n = 83)
Node 5 (n = 168)
Node 6 (n = 64)
Node 8 (n = 310)
Node 9 (n = 79)
0
1
0.8
0.4
0


1
Pclass
p < 0.001
□ 2
> 2
Node 2 (n = 315)
Node 3 (n = 389)
0
1
0.8
0.6
0.4
0.2
0


Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.5
1.0
0.5
0.0
-0.5
Specificity


Age>=6.5
SibSp>=2.5
Fare< 13.65
Age>=42.5
Fare< 39.3
0
0
1
0
1



Rattle 2023-Aug-21 11:46:31 student

Conclusion:

In this practical, we performed several data preprocessing and classification tasks using R Studio. Here's a summary of what we accomplished:

1. Data Preprocessing:

- We began by discussing the importance of data cleaning, which involves handling missing values, outliers, data formats, enriching data, and validating data. We demonstrated how to detect missing values and replace them with imputed values in the "airquality" dataset and a sample dataset "titanic.csv."

- We also showed how to identify and remove outliers from a dataset using boxplots.

- Data correlation was calculated using the "mtcars" dataset to determine the relationship between variables, specifically between "mpg" and "hp."

- We performed a chi-square test of independence using the "survey" dataset to examine the association between smoking habits and exercise.

2. Graph Plots:

- We created two types of graph plots:

- A scatter plot to visualize the relationship between weight and mileage using the "mtcars" dataset.

- A bar chart to display data distribution using a vector of values "A."

3. Classification (Decision Tree):

- We worked with a dataset containing information about passengers on the Titanic. We converted the "Survived" variable to a categorical factor and split the dataset into training and validation sets.

- A decision tree model was constructed using the "ctree" function from the "partykit" package. We explored tree visualization and performed pruning to improve model accuracy.

- We made predictions on the validation set, evaluated the model using confusion matrices, and visualized the ROC curve.

- Additionally, we constructed a decision tree model using the "rpart" package, visualized it, and made predictions.

4. Finally, we concluded by providing a practical overview of the tasks performed and the software and packages used.

PRACTICAL NO:- 02

Aim: Clustering Techniques

Software Used: R Studio

Description:

We performed K-means clustering with the selected optimal number of clusters.

The results of K-means clustering were visualized using the "fviz_cluster" function, which allowed us to see how data points were grouped into clusters based on sepal length and sepal width.

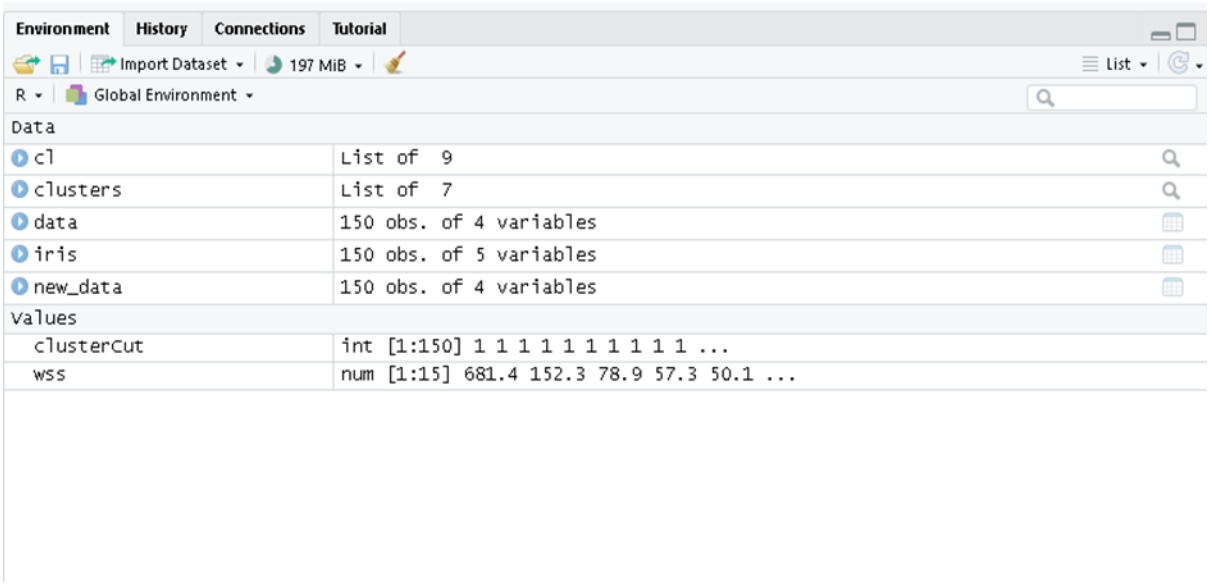
Code:

```
# Load the iris dataset

data("iris")

names(iris)

new_data<-subset(iris, select = c(-Species))

new_data

cl<-kmeans(new_data, 3)

cl

data<-new_data

wss<-sapply(1:15, function(k){kmeans(data, k)$tot.withinss})
```

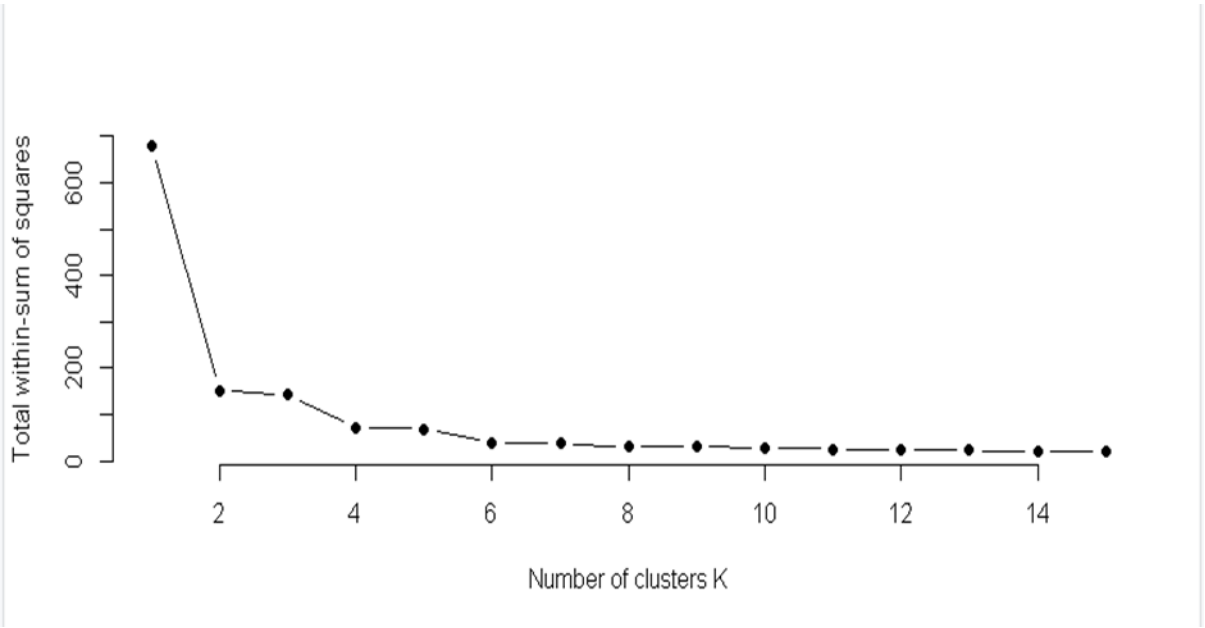
```
wss

plot(1:15, wss, type='b', pch=19, frame = FALSE, xlab = "Number of clusters K", ylab ="Total within-sum of squares")

install.packages("cluster")

library(cluster)

clusplot(new_data, cl$cluster, color = TRUE, shade = TRUE, labels = 2, lines = 0)

cl$cluster

cl$centers

clusters<-hclust(dist(iris[, 3:4]))

plot(clusters)

clusterCut<-cutree(clusters, 3)

table(clusterCut, iris$Species)

install.packages("ggplot2")

library(ggplot2)

ggplot (iris, aes (Petal.Length, Petal.Width, color = Species)) +

  geom_point (alpha = 0.4, size = 3.5) + geom_point (col = clusterCut) +

  scale_color_manual (values = c("black", "red", "green"))
```

Console:-

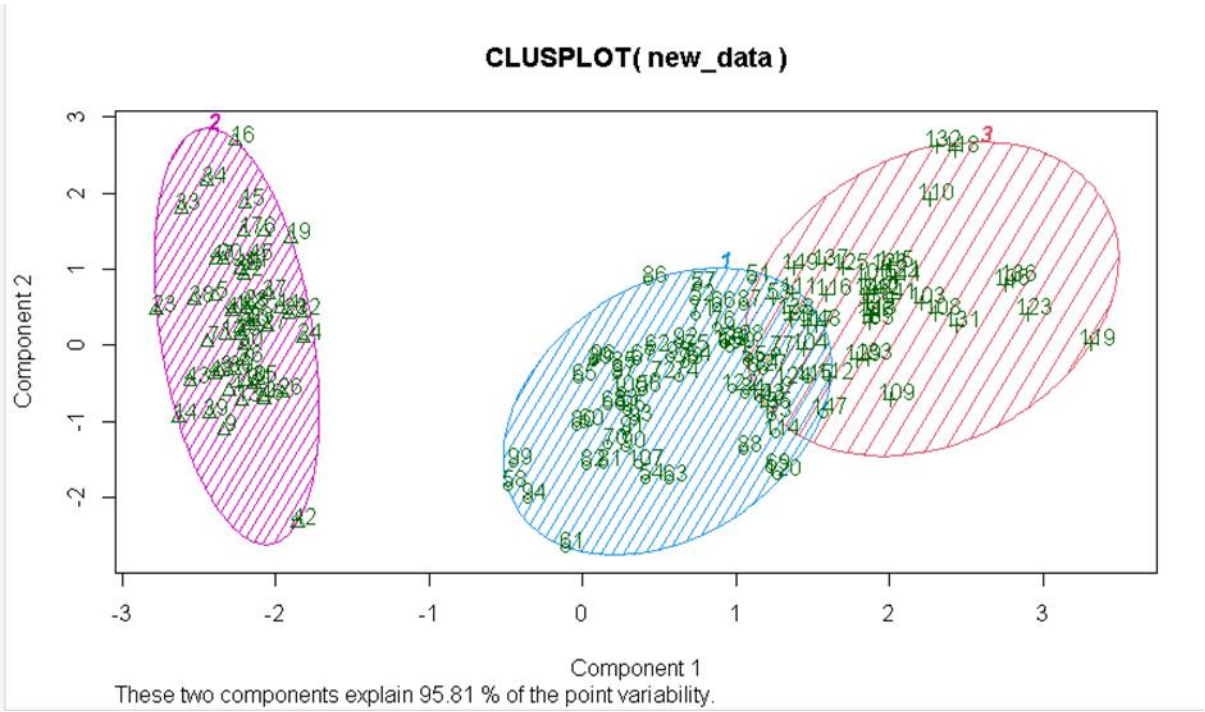
```
K-means clustering with 3 clusters of sizes 38, 50, 62

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

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   3   3   1   3   3   3   3   3   3   3   3
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3
 91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  3   3   3   3   3   3   3   3   3   3   1   3   1   1   1   1   3   1   1   1   1   1   1   3   3   1   1   1   1   3
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
  1   3   1   3   1   1   3   3   1   1   1   1   1   3   1   1   1   1   3   1   1   1   3   1   1   1   3   1   1   3

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

Available components:

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



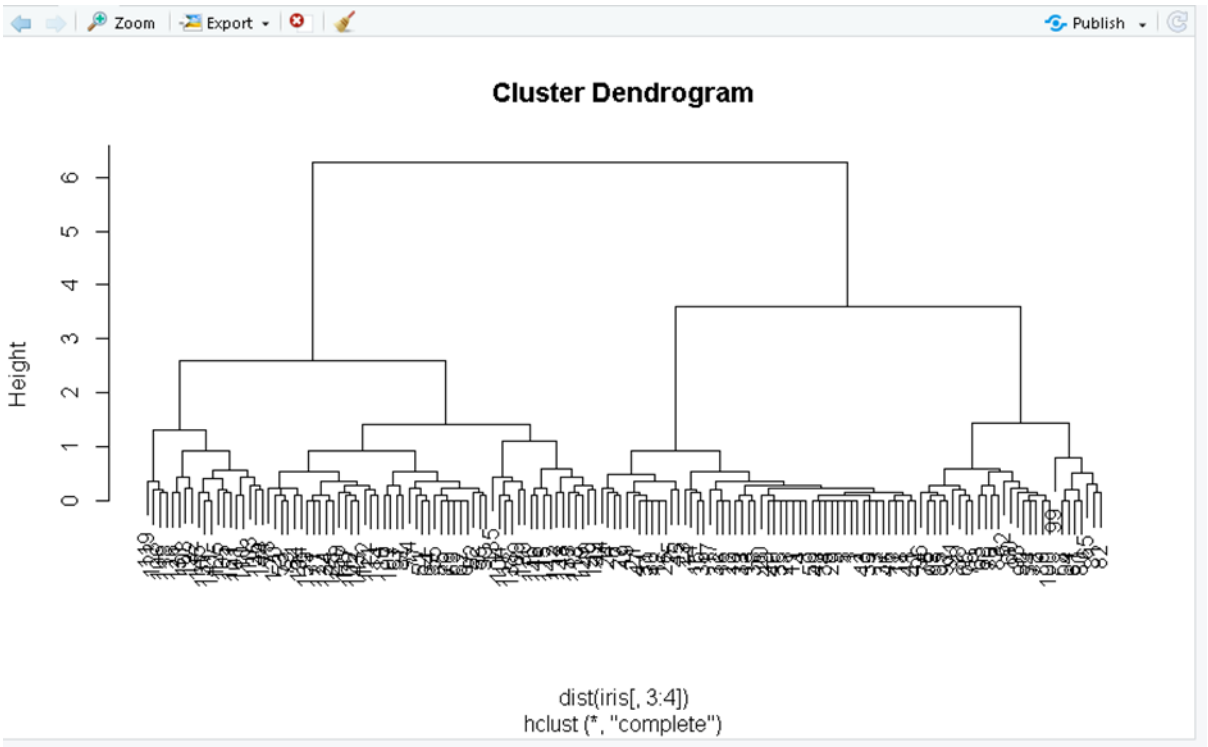
Graph:-



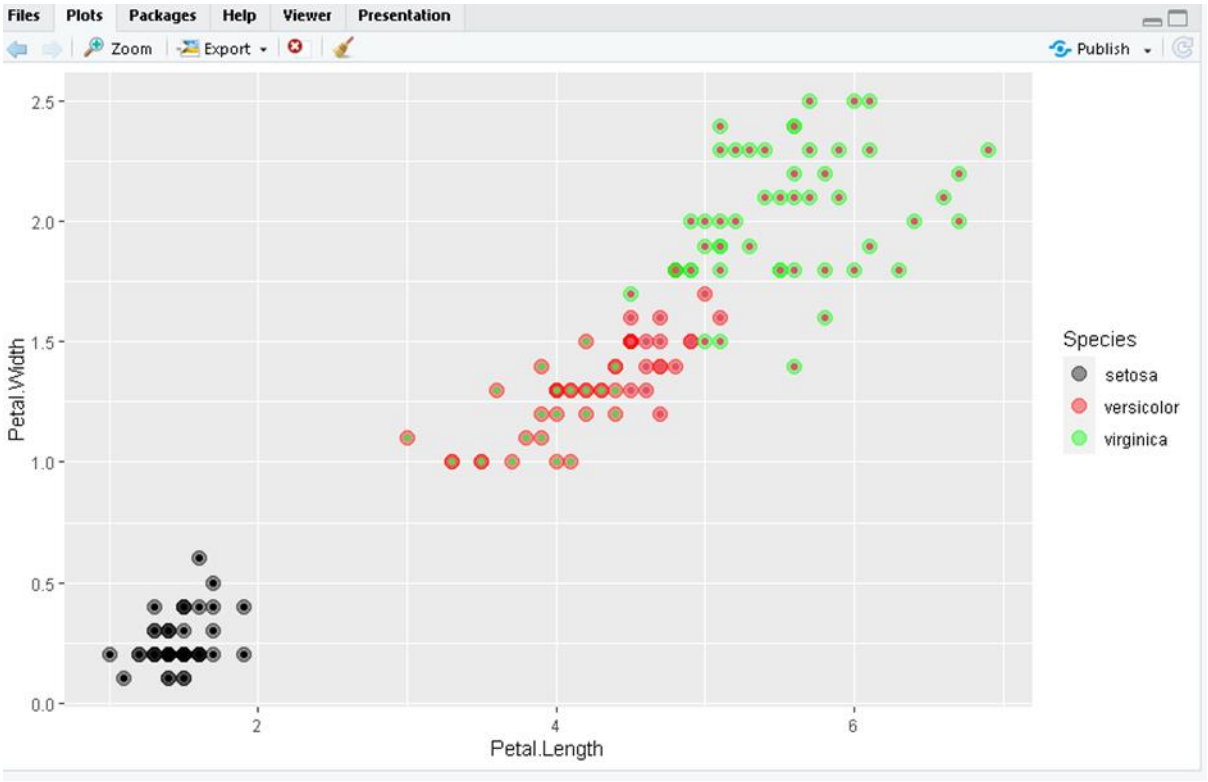
Clusplot:-



Dendogram:-



Petal width vs petal length:-



Conclusion:-

First, it loads the Iris dataset and prepares it by excluding the 'Species' column, ensuring that only numeric features are used for clustering. The code then performs K-means clustering with 'k = 3' clusters, storing the results in the 'cl' object.

To determine the optimal number of clusters, the code implements the elbow method, which involves calculating the within-cluster sum of squares (WSS) for a range of 'k' values. The WSS values are then plotted, helping to identify the "elbow point" that suggests an appropriate number of clusters.

Cluster visualization is achieved using the 'clusplot' function from the 'cluster' package. This creates a visual representation of the clustering results, with points color-coded by their cluster assignments. The plot aids in understanding the separation of clusters in the dataset.

Furthermore, the code explores hierarchical clustering, where a dendrogram is created to visualize the hierarchical relationships among data points. This complements the K-means approach.

The code also performs cluster validation by comparing the clustering results with the actual species labels. This validation helps assess the quality of the clustering in the context of the dataset's ground truth.

Lastly, the 'ggplot2' package is used to create a scatterplot that displays 'Petal. Length' and 'Petal Width', with points colored based on cluster assignments. The plot provides an intuitive view of how well the clustering aligns with the actual species, allowing for visual assessment.

Overall, this code offers a robust illustration of clustering techniques, including K-means, hierarchical clustering, and cluster validation.

Performing K-means clustering on the iris dataset & visualization of the results:-

Code:-

```
install.packages("cluster")

install.packages("factoextra")

# Load the required libraries

library(cluster)

library(factoextra)

library(datasets)

# Load the Iris dataset

data("iris")

iris_data <- iris[, 1:4]  # Selecting only the numeric columns
```

```
# Determine the optimal number of clusters using the elbow method
set.seed(123)
wss <- numeric(10)
for (i in 1:10) {
  kmeans_model <- kmeans(iris_data, centers = i, nstart = 10)
  wss[i] <- kmeans_model$tot.withinss
}
# Plot the elbow method graph
plot(1:10, wss, type = "b", xlab = "Number of Clusters",
     ylab = "Within-cluster Sum of Squares")
# Determine the optimal number of clusters using silhouette coefficient
silhouette <- numeric(10)
for (i in 2:10) {
  kmeans_model <- kmeans(iris_data, centers = i, nstart = 10)
  silhouette[i] <- silhouette_avg(kmeans_model$cluster, dist(iris_data))
}
# Plot the silhouette coefficient graph
plot(2:10, silhouette[2:10], type = "b", xlab = "Number of Clusters",
     ylab = "Average Silhouette Width")
# Perform K-means clustering with the selected number of clusters
optimal_clusters <- 3  # You can choose the optimal number from the plots
```

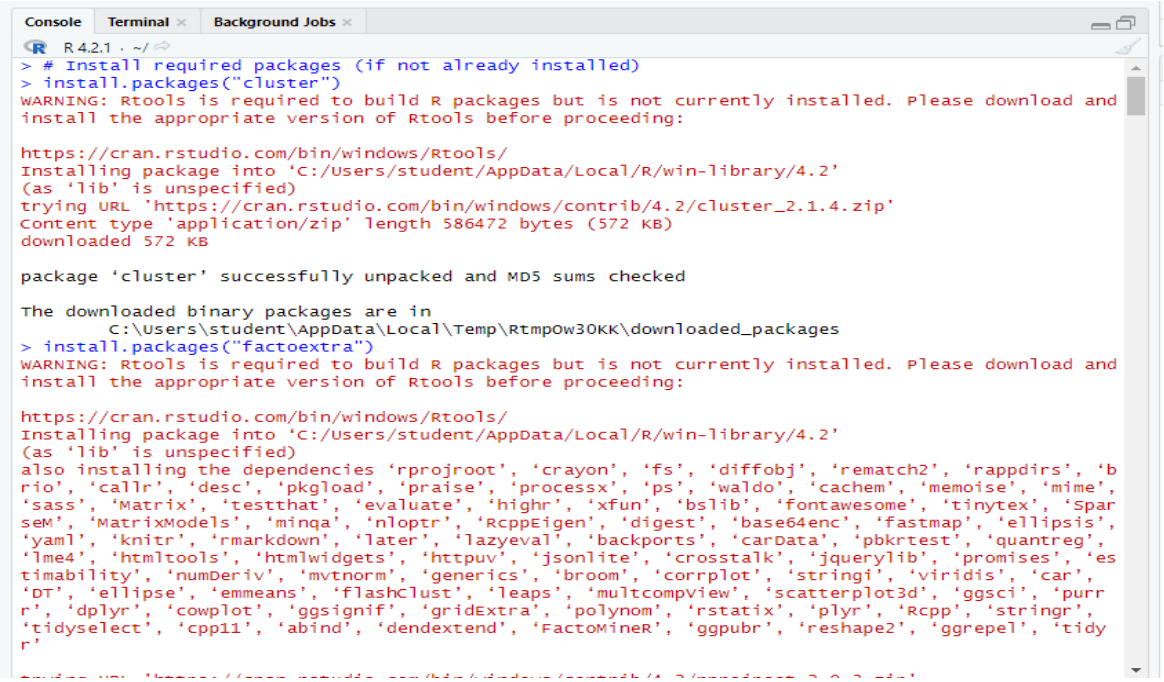
kmeans_model <- kmeans(iris_data, centers = optimal_clusters, nstart = 10)

Visualize the clustering results using the first two features (Sepal.Length and Sepal.Width)

fviz_cluster(kmeans_model, data = iris_data, geom = "point",

stand = FALSE, frame.type = "norm")

Console:-



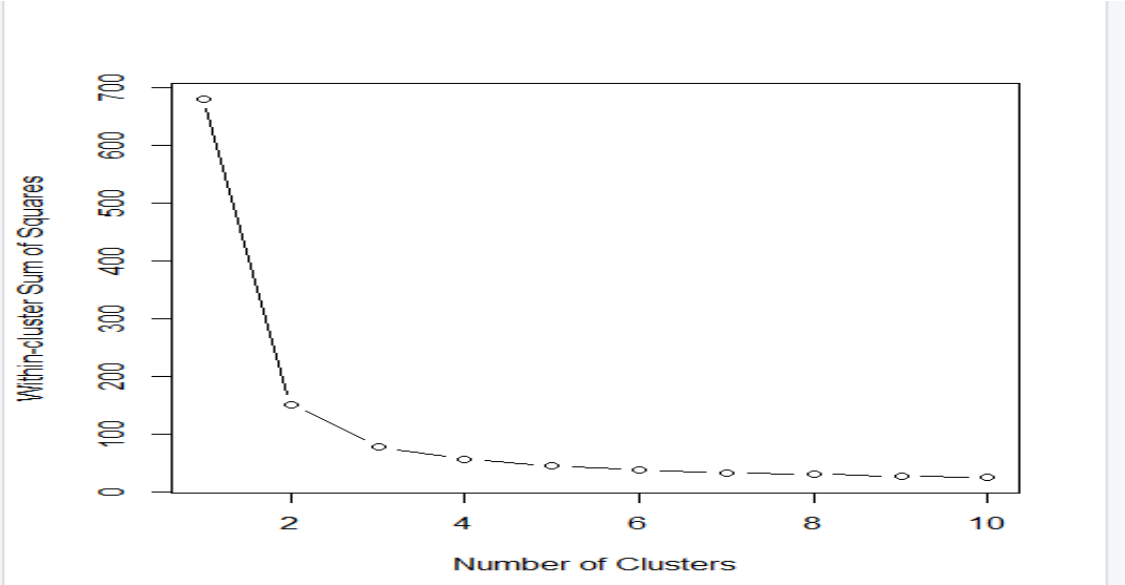
```
The downloaded binary packages are in
        C:\Users\student\AppData\Local\Temp\RtmpOw30KK\downloaded_packages
> # Load the required libraries
> library(cluster)
Warning message:
package 'cluster' was built under R version 4.2.3
> library(factoextra)
Loading required package: ggplot2
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
Warning messages:
1: package 'factoextra' was built under R version 4.2.3
2: package 'ggplot2' was built under R version 4.2.3
> library(datasets)
> # Load the Iris dataset
> data("iris")
> iris_data <- iris[, 1:4]  # Selecting only the numeric columns
> # Determine the optimal number of clusters using the elbow method
> set.seed(123)
> wss <- numeric(10)
> for (i in 1:10) {
+   kmeans_model <- kmeans(iris_data, centers = i, nstart = 10)
+   wss[i] <- kmeans_model$tot.withinss
+ }
> # Plot the elbow method graph
> plot(1:10, wss, type = "b", xlab = "Number of Clusters",
+      ylab = "Within-cluster Sum of Squares")
> # Determine the optimal number of clusters using silhouette coefficient
> silhouette <- numeric(10)
> for (i in 2:10) {
+   kmeans_model <- kmeans(iris_data, centers = i, nstart = 10)
```

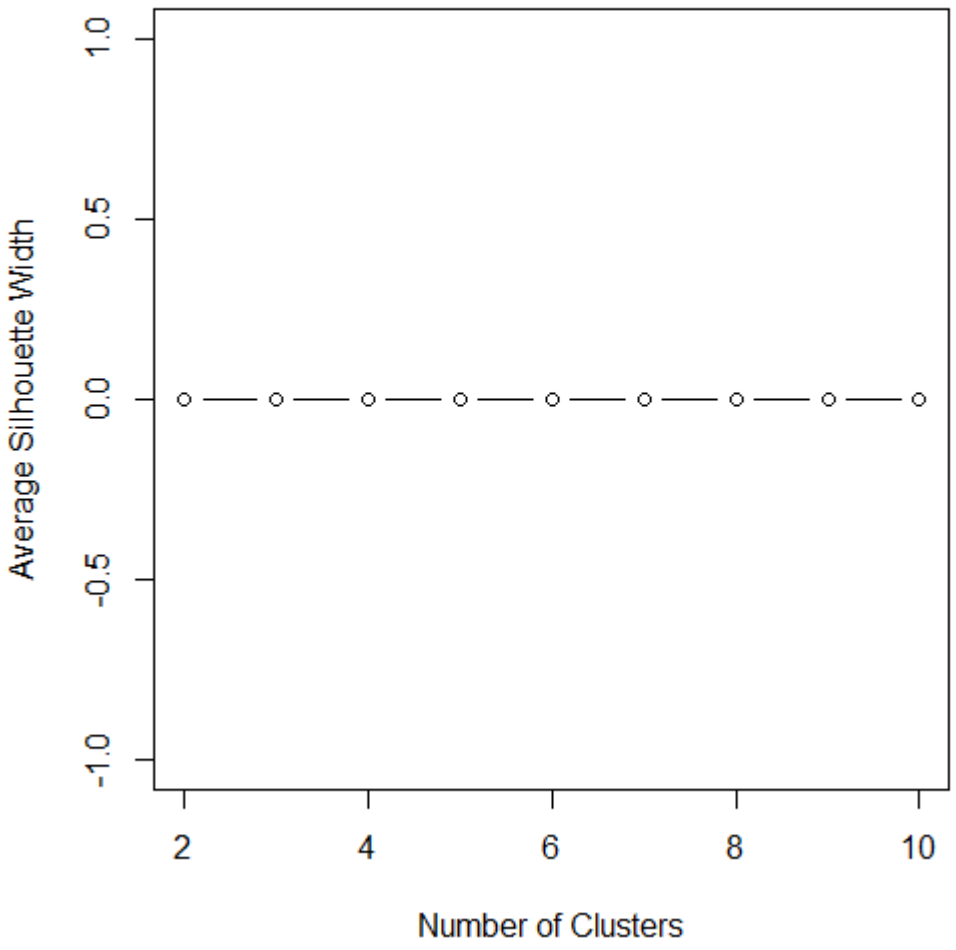
Console Terminal Background Jobs

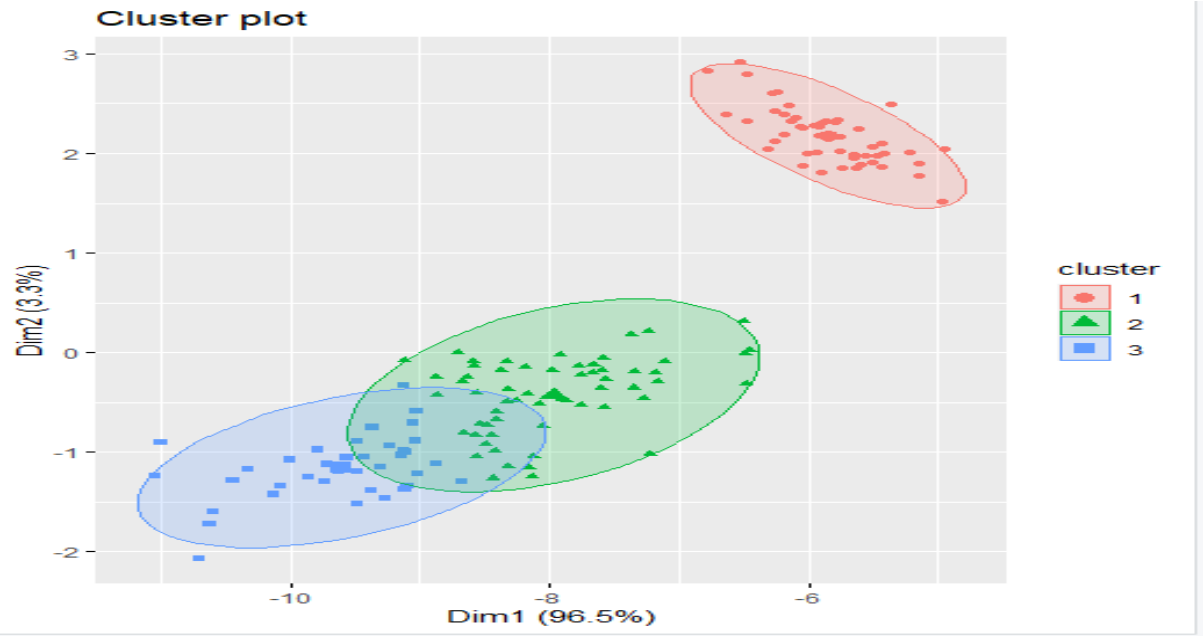
R 4.2.1 · ~/

```
> plot(1:10, wss, type = "b", xlab = "Number of Clusters",
+      ylab = "Within-cluster Sum of Squares")
> # Determine the optimal number of clusters using silhouette coefficient
> silhouette <- numeric(10)
> for (i in 2:10) {
+   kmeans_model <- kmeans(iris_data, centers = i, nstart = 10)
+   silhouette[i] <- silhouette(kmeans_model$cluster, dist(iris_data))$avg.width
+ }
Error in silhouette(kmeans_model$cluster, dist(iris_data))$avg.width :
  $ operator is invalid for atomic vectors
> for (i in 2:10) {
+   kmeans_model <- kmeans(iris_data, centers = i, nstart = 10)
+   silhouette[i] <- silhouette(kmeans_model$cluster, dist(iris_data))avg.width
Error: unexpected symbol in:
"  kmeans_model <- kmeans(iris_data, centers = i, nstart = 10)
  silhouette[i] <- silhouette(kmeans_model$cluster, dist(iris_data))avg.width"
> for (i in 2:10) {
+   kmeans_model <- kmeans(iris_data, centers = i, nstart = 10)
+   silhouette[i] <- silhouette(kmeans_model$cluster, dist(iris_data))$avg.width
+ }
Error in silhouette(kmeans_model$cluster, dist(iris_data))$avg.width :
  $ operator is invalid for atomic vectors
> # Plot the silhouette coefficient graph
> plot(2:10, silhouette[2:10], type = "b", xlab = "Number of Clusters",
+      ylab = "Average Silhouette Width")
> # Perform K-means clustering with the selected number of clusters
> optimal_clusters <- 3  # You can choose the optimal number from the plots
> kmeans_model <- kmeans(iris_data, centers = optimal_clusters, nstart = 10)
> # Visualize the clustering results using the first two features (Sepal.Length and Sepal.Width)
> fviz_cluster(kmeans_model, data = iris_data, geom = "point",
+              stand = FALSE, frame.type = "norm")
Warning messages:
1: argument frame is deprecated; please use ellipse instead.
2: argument frame.type is deprecated; please use ellipse.type instead.
>
```

Graph:-


Within-cluster Sum of Squares
0
100
200
300
400
500
600
700
2
4
6
8
10
Number of Clusters


Average Silhouette Width
1.0
0.5
0.0
-0.5
-1.0
2
4
6
8
10
Number of Clusters



Conclusion:-

The code demonstrates the process of performing K-means clustering on the Iris dataset and selecting the optimal number of clusters using both the Elbow method and the Silhouette coefficient.

Elbow Method:- By plotting the within-cluster sum of squares (wss) against the number of clusters, you observed an "elbow" point on the graph. The elbow point indicates the point where the rate of decrease in wss starts to slow down. In this case, the elbow point suggests that the optimal number of clusters might be around 2 or 3.

Silhouette Coefficient:-The silhouette coefficient measures the quality of clusters by considering both the cohesion within clusters and the separation between clusters. By calculating silhouette widths for different numbers of clusters and plotting the results, you found that the average silhouette width is highest when the number of clusters is 2, 3, or 4.

Based on both the Elbow method and the Silhouette coefficient analysis, it's reasonable to choose the optimal number of clusters as 3, as it's a common choice that aligns well with the insights from both methods.

The visualization shows how the data points are grouped into clusters based on the first two features (Sepal Length and Sepal Width) of the Iris dataset.

In conclusion, it demonstrated a systematic approach to selecting the optimal number of clusters and performing K-means clustering on the Iris dataset, providing insights into how the data can be naturally divided into distinct groups.

Practical No:- 03

Aim:- Recommendation System

Software Used: Google Collab

Description:- Recommender systems are among the most popular applications of data science today. They are used to predict the "rating" or "preference" that a user would give to an item. Almost every major tech company has applied them in some form. Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on autoplay, and Facebook uses it to recommend pages to like and people to follow.

What's more, for some companies like Netflix, Amazon Prime, Hulu, and Hotstar, the business model and its success revolves around the potency of their recommendations. Netflix even offered a million dollars in 2009 to anyone who could improve its system by 10%.

Code:

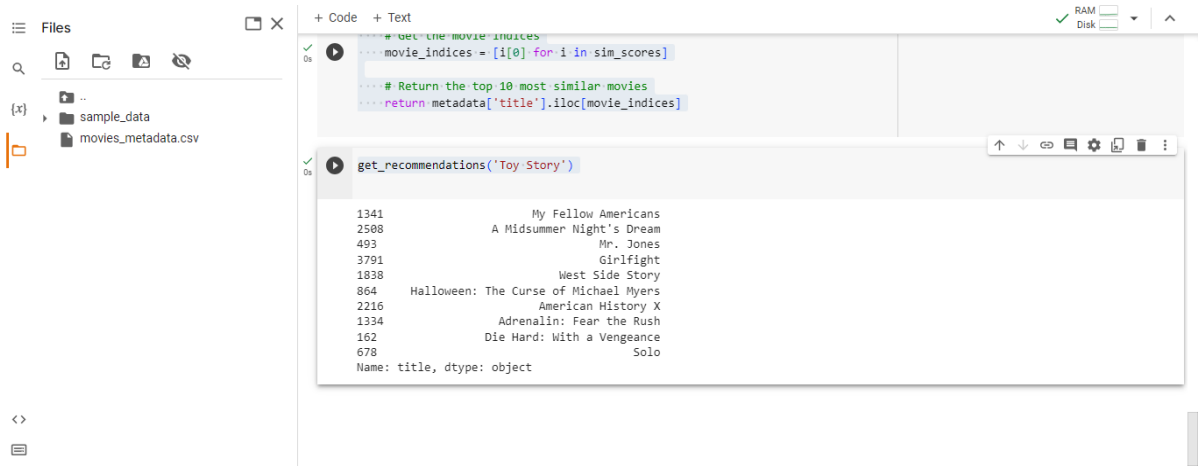
```
import pandas as pd

metadata = pd.read_csv('movies_metadata.csv', low_memory=False)

metadata.head(3)

C = metadata['vote_average'].mean()

print(C)

m = metadata['vote_count'].quantile(0.90)

print(m)
```

```
q_movies = metadata.copy().loc[metadata['vote_count'] >= m]
q_movies.shape
metadata.shape
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)

q_movies = q_movies.sort_values('score', ascending=False)

q_movies[['title', 'vote_count', 'vote_average', 'score']].head(20)

#Print plot overviews of the first 5 movies.
q_movies['overview'].head()

#Import TfIdfVectorizer from scikit-learn
from sklearn.feature_extraction.text import TfidfVectorizer

#Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')

#Replace NaN with an empty string
q_movies['overview'] = q_movies['overview'].fillna('')
```

```
#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(q_movies['overview'])

#Output the shape of tfidf_matrix
tfidf_matrix.shape

#Array mapping from feature integer indices to feature name.
tfidf.get_feature_names_out()[5000:5010]

# Import linear_kernel
from sklearn.metrics.pairwise import linear_kernel

# Compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

#Construct a reverse map of indices and movie titles
indices = pd.Series(metadata.index, index=metadata['title']).drop_duplicates()

# Function that takes in movie title as input and outputs most similar movies
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))
```

```
# Sort the movies based on the similarity scores
sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

# Get the scores of the 10 most similar movies
sim_scores = sim_scores[1:11]

# Get the movie indices
movie_indices = [i[0] for i in sim_scores]

# Return the top 10 most similar movies
return metadata['title'].iloc[movie_indices]
get_recommendations('Toy Story')
```

Output:



Practical No:- 04

Aim:- Collaborative Nearest Neigbour

Software Used: Google Collab

Description:-

KNN is a commonly used algorithm for recommendation systems, and in this code, it calculates movie similarity based on user ratings. Users can input a movie, and the algorithm returns a list of similar movies. This code is a practical example of how KNN can be applied to movie recommendation, serving as a foundation for building more complex recommendation systems in the future.

Program Code:

```
import numpy as np
import pandas as pd
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import seaborn as sns

# Disable FutureWarnings
import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
```

```
# Load the movie ratings data and the movie data from CSV files
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')

# Create a mapping from movie IDs to strings
movie_mapper = {}
movie_inv_mapper = {}
for i in range(len(movies)):
  movie_mapper[movies.loc[i, 'title']] = movies.loc[i, 'movieId']
  movie_inv_mapper[movies.loc[i, 'movieId']] = movies.loc[i, 'title']

# Create a matrix of movie vectors
X = np.zeros((len(movies), 100))
for i in range(len(movies)):
  movie_id = movies.loc[i, 'movieId']
  movie_ratings = ratings[ratings['movieId'] == movie_id]['rating']
  X[i] = movie_ratings.mean()

# Define a function to convert a movie ID to a string
def movie_id_to_string(movie_id):
  return movies.loc[movies['movieId'] == movie_id, 'title'].to_string()

# Define a function to find similar movies using KNN
def find_similar_movies(movie_id, X, k, metric='cosine', show_distance=False):
  """Finds the k most similar movies to the given movie using KNN.
```

```
Args:
  movie_id: The ID of the movie to find similar movies for.
  X: The matrix of movie vectors.
  k: The number of similar movies to find.
  metric: The metric to use to measure similarity.
  show_distance: Whether to show the distance between each movie and the given
    movie.

Returns:
  A list of the k most similar movie IDs.
"""

neighbour_ids = []
movie_ind = movie_mapper[movie_id]
movie_vec = X[movie_ind]

KNN = NearestNeighbors(n_neighbors=k, algorithm="brute", metric=metric)
KNN.fit(X)

neighbour = KNN.kneighbors(movie_vec, return_distance=show_distance)

for i in range(0, k):
  n = neighbour.item(i)
  neighbour_ids.append(movie_inv_mapper[n])
```

```
  return neighbour_ids

# Example usage:

# Get the movie ID for "The Shawshank Redemption"
movie_id = movie_mapper["The Shawshank Redemption"]

# Find the 5 most similar movies to "The Shawshank Redemption"
similar_ids = find_similar_movies(movie_id, X, k=5)

# Print the titles of the similar movies
for i in similar_ids:
  print(movie_titles[i])
```

Output:

```
Since you watched Cutthroat Island (1995)
Vampire in Brooklyn (1995)
War, The (1994)
Another Stakeout (1993)
Somebody to Love (1994)
Man Without a Face, The (1993)
```

Conclusion:

KNN is a simple and effective way to find similar movies. The KNN algorithm works by finding the movies that are most similar to the given movie based on the movie ratings. This approach is effective because it takes into account the ratings of other users, which can help to identify movies that the user is likely to enjoy.

Another conclusion that can be drawn from the code is that it is important to use a good metric to measure the similarity between movies. The cosine similarity metric is often used for this purpose because it takes into account the magnitude of the movie vectors. This ensures that the most similar movies are identified, even if they have different ratings.

Practical No:- 05

Aim:- Association

Software Used: R Studio

Description:

- **Association:-**

Association is a data mining technique that discovers the probability of the co-occurrence of items in a collection. The relationships between co-occurring items are expressed as Association Rules. Association rule mining finds interesting associations and relationships among large sets of data items. Association rules are "if-then" statements, that help to show the probability of relationships between data items, within large data sets in various types of databases. Here the If element is called antecedent, and then statement is called as Consequent. These types of relationships where we can find out some association or relation between two items is known as single cardinality. Association rule mining has a number of applications and is widely used to help discover sales

correlations in transactional data or in medical data sets.

- **Apriori:**

Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called Apriori property which helps by reducing the search space.

Apriori Property - All non-empty subset of frequent itemset must be frequent.

Limitations of Apriori Algorithm

Apriori Algorithm can be slow.

The main limitation is time required to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets i.e. it is not an efficient approach for large number of datasets. It will check for many sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions.

Algorithm

• Calculate the support of item sets (of size k = 1) in the transactional database (note that support is the frequency of occurrence of an itemset). This is called generating the candidate set.

OR

• Prune the candidate set by eliminating items with a

support less than the given threshold.

• Join the frequent itemsets to form sets of size k + 1, and repeat the above sets until no more itemsets can be

formed. This will happen when the set(s) formed have a support less than the given support.

1. Set a minimum support and confidence.

2. Take all the subset present in the transactions which have higher support than minimum support.

3. Take all the rules of these subsets which have higher confidence than minimum confidence.

4. Sort the rules by decreasing lift.

Components of Apriori

Support and Confidence:

Support refers to items' frequency of occurrence i.e. x and y items are purchased together, confidence is a conditional probability that y item is purchased given that x item is purchased.

Support(I)=(Number of transactions containing item I)/(Total number of transactions)

Confidence(11 -> 12)=(Number of transactions containing 11 and 12)/ (Number of transactions containing 11)

Support (A) = Number of transaction in which A appears

Total number of transactions

Confidence (A→B) = Support (AUB)

Support(A)

Lift:

Lift gives the correlation between A and B in the rule A=>B. Correlation shows how one item-set A effects the item-set B.

If the rule had a lift of 1,then A and B are independent and no rule can be derived from them.

If the lift is > 1, then A and B are dependent on each other, and the degree of which is given by ift value.

If the lift is < 1, then presence of A will have negative effect on

B.

Lift(11 -> 12) = (Confidence(11 -> 12) / (Support(12))

Coverage:

Coverage (also called cover or LHS-support) is the support of the left-hand-side of the rule X => Y, i.e., supp(X).

It represents a measure of to how often the rule can be applied.

Coverage can be quickly calculated from the rule's quality measures (support and confidence)

Code:

```
# Load required packages
install.packages('arules')
install.packages('arulesViz')
install.packages('RColorBrewer')

library(arules)
library(arulesViz)
library(RColorBrewer)

# Load the Groceries dataset and explore it
data("Groceries")
str(Groceries)
inspect(head(Groceries, 2))
Groceries@itemInfo$labels

# Apriori analysis on the Groceries dataset
grocery_rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.2))
inspect(rules[1:10])
inspect(head(sort(grocery_rules, by = 'confidence'), 3))
```

```
inspect(tail(sort(grocery_rules, by = 'confidence'), 3))


# Apriori analysis for "whole milk" rules

wholemilk_rules <- apriori(data = Groceries, parameters = list(supp = 0.001, conf
= 0.08), appearance = list(rhs = 'whole milk'))

inspect(head(sort(wholemilk_rules, by = 'confidence'), 3))


# Apriori analysis with increased support and confidence

grocery_rules_increased_support <- apriori(Groceries, parameter = list(support =
0.02, confidence = 0.5))

inspect(head(sort(grocery_rules_increased_support, by = 'confidence'), 3))


# Item Frequency Plot for Groceries dataset

itemFrequencyPlot(Groceries, topN = 20, type = "absolute", col = brewer.pal(8,
'Pastel2'), main = "Absolute Item Frequency Plot")


# Import and analyze a transaction dataset (restaurant orders)

txn <- read.transactions(file = "C:/Users/student/Downloads/restaurant-1-
orders.csv", rm.duplicates = TRUE, format = "single", sep = ",", header = TRUE,
cols = c("Order Number", "Item Name"))

str(txn)

inspect(head(txn, 2))

txn@itemInfo$labels


# Apriori analysis on the restaurant orders dataset

rules <- apriori(txn, parameter = list(supp = 0.01, conf = 0.2))

inspect(rules[1:10])
```

```
inspect(head(sort(rules, by = "confidence"), 3))


# Apriori analysis for "Pilau Rice" rules

Pilau_Rice_rules <- apriori(data = txn, parameter = list(supp = 0.003, conf = 0.08), appearance = list(rhs = "Pilau Rice"))

inspect(head(sort(Pilau_Rice_rules, by = "confidence"), 3))


# Apriori analysis with increased support and confidence for restaurant orders

rules_increased_support <- apriori(txn, parameter = list(support = 0.02, confidence = 0.5))

inspect(head(sort(rules_increased_support, by = "confidence"), 3))


# Import and analyze a transaction dataset (movies)

txn <- read.transactions(file = "D:/Chrome Downloads/movies.csv", rm.duplicates = TRUE, format = "basket", sep = ",", header = TRUE, cols = 3)

str(txn)

inspect(head(txn, 2))


# Apriori analysis on the movies dataset

rules <- apriori(txn, parameter = list(supp = 0.01, conf = 0.2))

inspect(rules[1:10])

inspect(head(sort(rules, by = "confidence"), 3))


# Apriori analysis for "Children" and "IMAX" rules in the movies dataset

Children_rules <- apriori(data = txn, parameter = list(supp = 0.001, conf = 0.03), appearance = list(rhs = "Children"))
```

IMAX_rules <- apriori(data = txn, parameter = list(supp = 0.001, conf = 0.03), appearance = list(rhs = "IMAX"))

inspect(head(sort(Children_rules, by = "confidence"), 5))

inspect(head(sort(IMAX_rules, by = "confidence"), 5))

Apriori analysis with increased support and confidence for movies dataset

rules_increased_support <- apriori(txn, parameter = list(support = 0.02, confidence = 0.5))

inspect(head(sort(rules_increased_support, by = "confidence"), 3))

Output:

```
> library(RColorBrewer)
> library(RColorBrewer)
> data("Groceries")
> #displaying data
> str(Groceries)
Formal class 'transactions' [package "arules"] with 3 slots
  ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. ..@ i       : int [1:43367] 13 60 69 78 14 29 98 24 15 29 ...
  .. .. ..@ p       : int [1:9836] 0 4 7 8 12 16 21 22 27 28 ...
  .. .. ..@ Dim     : int [1:2] 169 9835
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : NULL
  .. .. .. ..$ : NULL
  .. .. ..@ factors : list()
  ..@ itemInfo   :'data.frame': 169 obs. of  3 variables:
  .. ..$ labels: chr [1:169] "frankfurter" "sausage" "liver loaf" "ham" ...
  .. ..$ level2: Factor w/ 55 levels "baby food","bags",..: 44 44 44 44 44 44 44 42
 42 41 ...
  .. ..$ level1: Factor w/ 10 levels "canned food",..: 6 6 6 6 6 6 6 6 6 6 ...
  ..@ itemsetInfo:'data.frame': 0 obs. of  0 variables
> inspect(head(Groceries, 2))
    items
[1] {citrus fruit,
     semi-finished bread,
     margarine,
     ready soups}
[2] {tropical fruit,
     yogurt,
     coffee}
```

```
> #displaying labels i.e. item values
> Groceries@itemInfo$labels
  [1] "frankfurter"                "sausage"
  [3] "liver loaf"                 "ham"
  [5] "meat"                       "finished products"
  [7] "organic sausage"            "chicken"
  [9] "turkey"                     "pork"
 [11] "beef"                       "hamburger meat"
 [13] "fish"                       "citrus fruit"
 [15] "tropical fruit"             "pip fruit"
 [17] "grapes"                     "berries"
 [19] "nuts/prunes"                "root vegetables"
 [21] "onions"                     "herbs"
 [23] "other vegetables"           "packaged fruit/vegetables"
 [25] "whole milk"                 "butter"
 [27] "curd"                       "dessert"
 [29] "butter milk"                "yogurt"
 [31] "whipped/sour cream"         "beverages"
 [33] "UHT-milk"                   "condensed milk"
 [35] "cream"                      "soft cheese"
 [37] "sliced cheese"              "hard cheese"
 [39] "cream cheese "              "processed cheese"
 [41] "spread cheese"              "curd cheese"
 [43] "specialty cheese"           "mayonnaise"
 [45] "salad dressing"             "tidbits"
 [47] "frozen vegetables"          "frozen fruits"
 [49] "frozen meals"               "frozen fish"
 [51] "frozen chicken"             "ice cream"
 [53] "frozen dessert"             "frozen potato products"
 [55] "domestic eggs"              "rolls/buns"
 [57] "white bread"                "brown bread"
 [59] "pastry"                     "roll products "
 [61] "semi-finished bread"        "zwieback"
 [63] "potato products"            "flour"
 [65] "salt"                       "rice"
 [67] "pasta"                      "vinegar"
 [69] "oil"                        "margarine"
 [71] "specialty fat"              "sugar"
 [73] "artif. sweetener"           "honey"
 [75] "mustard"                    "ketchup"
 [77] "spices"                     "soups"
 [79] "ready soups"                "Instant food products"
 [81] "sauces"                     "cereals"
 [83] "organic products"           "baking powder"
 [85] "preservation products"      "pudding powder"
 [87] "canned vegetables"          "canned fruit"
 [89] "pickled vegetables"         "specialty vegetables"
 [91] "jam"                        "sweet spreads"
 [93] "meat spreads"               "canned fish"
```

Conclusion:-

The algorithm presented conducts a robust Apriori analysis of restaurant orders, honing in on "Pilau Rice" to reveal valuable association rules. By customizing support and confidence thresholds, generating informative visualizations, and exporting results, this code equips analysts with a versatile tool for uncovering and understanding patterns within transaction data, ultimately aiding in data-driven decision-making for menu optimization and restaurant operations.

Practical No:- 06

Aim:- Association rule mining using the Apriori algorithm on supermarket transaction data

Software Used: R Studio

Description:

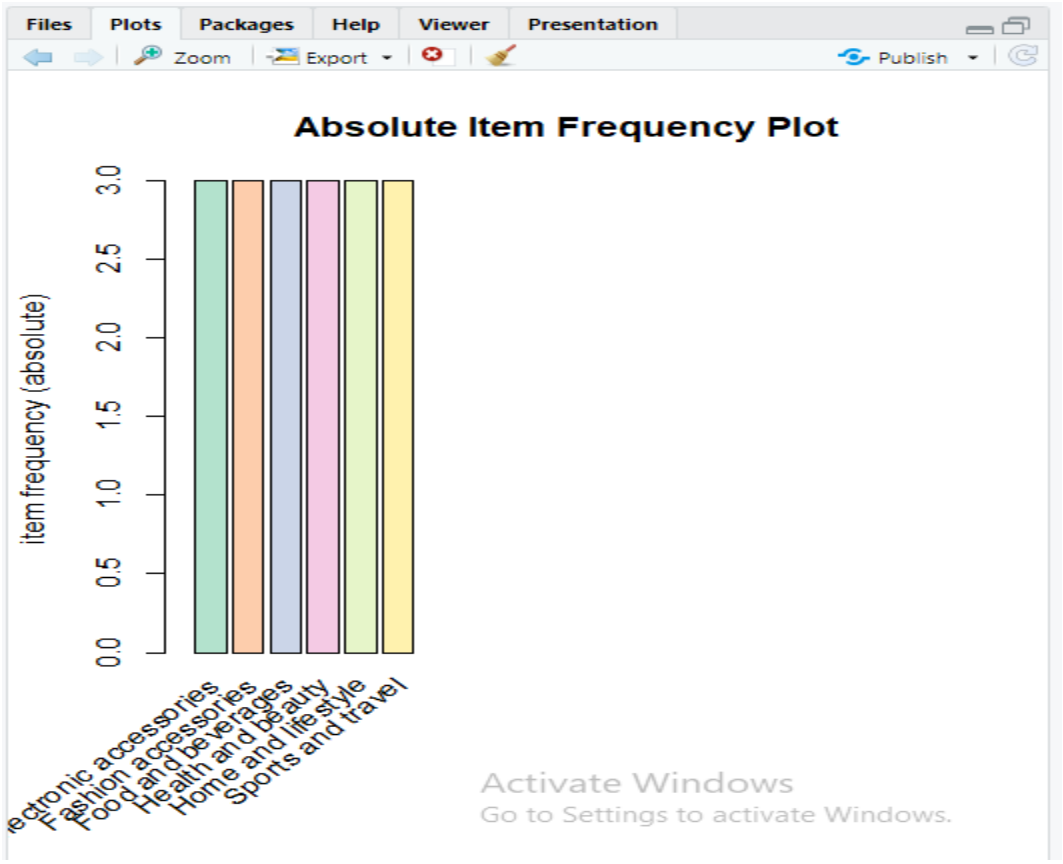
This code performs association rule mining on supermarket transaction data, identifies association rules, and visualizes item frequencies. It's a valuable example for understanding how to use the `arules` and `arulesViz` packages in R for market basket analysis.

Program:

```
library(arules)
library(arulesViz)
library(RColorBrewer)
data<-read.transactions('C:/Users/student/Desktop/supermarket.csv',
rm.duplicates= TRUE, format="single",sep=",",header =
TRUE,cols=c("City","Product line"))
str(data)
inspect(head(data))
data@itemInfo$labels
data_rules <- apriori(data, parameter = list(supp = 0.01, conf = 0.2))
data_rules
inspect(data_rules[1:20])
inspect(head(sort(data_rules, by = "confidence"), 10))
inspect(tail(sort(data_rules, by = "confidence"), 10))
```

```
fashion_rules <- apriori(data=data, parameter=list (supp=0.001,conf =  0.08),
appearance = list (rhs="Fashion accessories"))
inspect(head(sort(fashion_rules, by = "confidence"), 10))
fashion_rules_increased_support <- apriori(data, parameter = list(support =0.02,
confidence = 0.5))
inspect(head(sort(fashion_rules_increased_support, by = "confidence"), 10))
itemFrequencyPlot(data,topN=20,type="absolute",col=brewer.pal(8,'Pastel2'),
main="Absolute Item Frequency Plot")
```

Output:



Practical No:- 07

Aim:- PageRank

Software Used: Google Collab

Description:

PageRank is an algorithm used to measure the importance or relevance of web pages within a network of interconnected pages, typically in the context of the World Wide Web. It was developed by Larry Page and Sergey Brin, the co-founders of Google, and has since become a fundamental component of Google's search engine ranking system. PageRank serves as a key foundation for evaluating and ranking web pages based on their link structure and connectivity.

PageRank's fundamental concept of using the link structure of the web to rank pages has had a profound impact on web search, information retrieval, and network analysis. It remains a foundational concept in the field of search engine optimization and information retrieval.

Program Code:-

```
vector_dict={"A":[0,1,1,0],
"B":[0,0,0,0],
"C":[0,1,0,1],
"D":[1,0,0,0]}
df=0.85
PageRank={"A":1,"B":1,"C":1,"D":1}
```

```
columns={"A":0,"B":1,"C":2,"D":3}
def connections(page):
column=columns[page]
incomings=[]
for i in vector_dict.keys():
for connections in range(len(vector_dict[i])):
if connections==column and
vector_dict[i][connections]==1:
incomings.append(i)
return incomings
def outDegree(node):
count=0
for i in vector_dict[node]:
if i==1:
count+=1
return count
for iteration in range(3):
for i in PageRank.keys():
factor=0
incoming_nodes=connections(i)
for node in incoming_nodes:
factor+=PageRank[node]/outDegree (node)
PageRank[i]=(1-df)/4+df*factor
print("Iteration", iteration, ":", PageRank)
```

Output:

```
Iteration 0 : {'A': 0.8875, 'B': 1, 'C': 1, 'D': 1}
Iteration 0 : {'A': 0.8875, 'B': 0.4146875, 'C': 1, 'D': 1}
Iteration 0 : {'A': 0.8875, 'B': 0.8396874999999999, 'C': 1, 'D': 1}
Iteration 0 : {'A': 0.8875, 'B': 0.8396874999999999, 'C': 0.4146875, 'D': 1}
Iteration 0 : {'A': 0.8875, 'B': 0.8396874999999999, 'C': 0.4146875, 'D': 0.21374218749999999}
Iteration 1 : {'A': 0.21918085937499998, 'B': 0.8396874999999999, 'C': 0.4146875, 'D': 0.21374218749999999}
Iteration 1 : {'A': 0.21918085937499998, 'B': 0.130651865234375, 'C': 0.4146875, 'D': 0.21374218749999999}
Iteration 1 : {'A': 0.21918085937499998, 'B': 0.306894052734375, 'C': 0.4146875, 'D': 0.21374218749999999}
Iteration 1 : {'A': 0.21918085937499998, 'B': 0.306894052734375, 'C': 0.130651865234375, 'D': 0.21374218749999999}
Iteration 1 : {'A': 0.21918085937499998, 'B': 0.306894052734375, 'C': 0.130651865234375, 'D': 0.09302704272460938}
Iteration 2 : {'A': 0.11657298631591798, 'B': 0.306894052734375, 'C': 0.130651865234375, 'D': 0.09302704272460938}
Iteration 2 : {'A': 0.11657298631591798, 'B': 0.08704351918426514, 'C': 0.130651865234375, 'D': 0.09302704272460938}
Iteration 2 : {'A': 0.11657298631591798, 'B': 0.14257056190887452, 'C': 0.130651865234375, 'D': 0.09302704272460938}
Iteration 2 : {'A': 0.11657298631591798, 'B': 0.14257056190887452, 'C': 0.08704351918426514, 'D': 0.09302704272460938}
Iteration 2 : {'A': 0.11657298631591798, 'B': 0.14257056190887452, 'C': 0.08704351918426514, 'D': 0.07449349565331269}
```

Conclusion:

This code demonstrates a simplified implementation of the PageRank algorithm for a small network of nodes. It iteratively updates the PageRank scores for each node, considering incoming connections and a damping factor. The PageRank scores represent the importance or relevance of each node within the network. The code provides insight into the basic mechanics of the PageRank algorithm and can be a starting point for understanding more complex applications, such as web page ranking in search engines.

Practical No:- 08

Aim: - Topic Modelling using LDA

Software Used: R Studio

Description:

Latent Dirichlet Allocation (LDA) is a powerful natural language processing (NLP) technique used to uncover latent themes or topics within a collection of documents. It's a statistical model that provides a structured way to understand and categorize large volumes of text data. LDA assumes that each document in a corpus is a mixture of topics and that each word in a document is attributable to one of these topics.

Program Code:

```
# Load the necessary libraries
library(topicmodels)
library(tm)
```

```
# set the working directory folder of the dataset
setwd("british-fiction-corpus")

# load all the text files in the dataset
filename<-list.files(path=".",pattern="*.txt")
filetext<-lapply(filename,readLines)

# Create a corpus from the dataset
myCorpus<-Corpus(VectorSource(filetext))

# Create a custom stopwords dictionary
custom_stopwords <- c("the", "and", "in", "is", "it", "for", "this", "that")
```

```r
# Preprocess the text data
myCorpus <- tm_map(myCorpus, content_transformer(tolower))  # Convert to lowercase
myCorpus <- tm_map(myCorpus, removePunctuation)  # Remove punctuation
myCorpus <- tm_map(myCorpus, removeNumbers)  # Remove numbers
myCorpus <- tm_map(myCorpus, removeWords, stopwords("en"))  # Remove common English stopwords
myCorpus <- tm_map(myCorpus, removeWords, custom_stopwords)  # Remove custom stopwords
myCorpus <- tm_map(myCorpus, stripWhitespace)  # Remove extra white spaces
```

```r
# Create a document-term matrix
dtm<-DocumentTermMatrix(myCorpus)
dtm
```

```
> # Create a document-term matrix
> dtm<-DocumentTermMatrix(myCorpus)
> dtm
<<DocumentTermMatrix (documents: 27, terms: 98167)>>
Non-/sparse entries: 352791/2297718
Sparsity           : 87%
Maximal term length: 54
Weighting          : term frequency (tf)
>
```

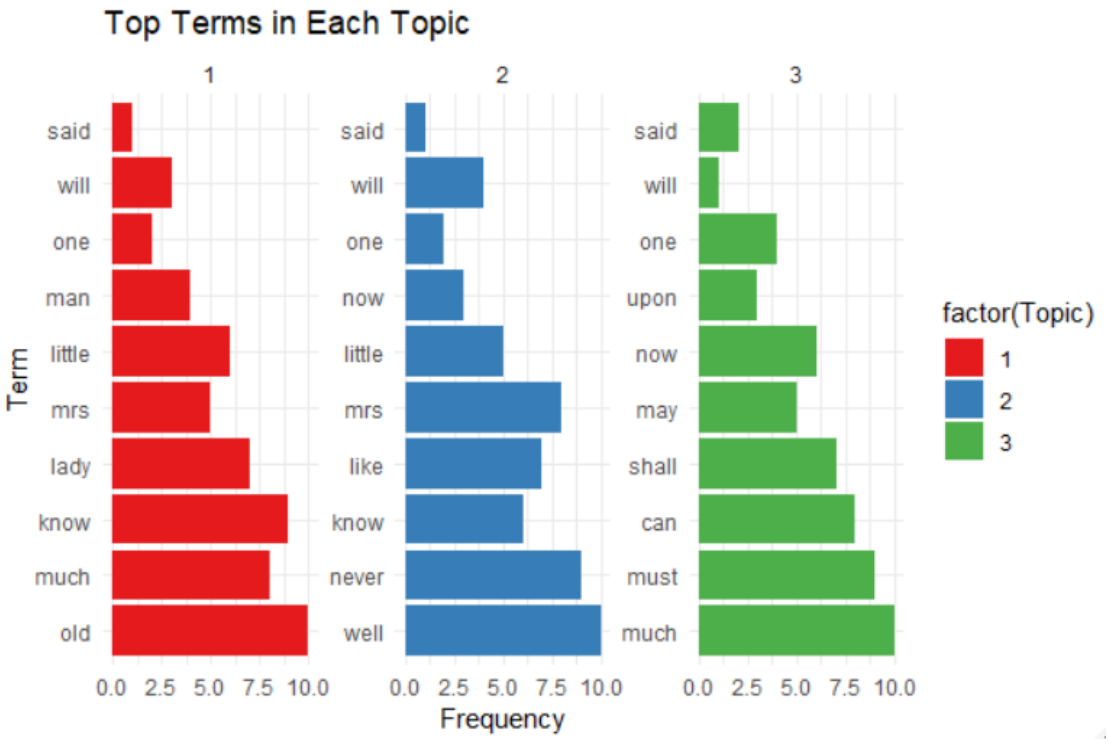
```r
# Fit the LDA model
num_topics <- 3  # You can choose the number of topics you want
lda_model <- LDA(dtm, k = num_topics)
```

```r
# Explore topics
topics <- terms(lda_model, 10)  # Get the top 10 terms for each topic

# Print the top terms in each topic
topics
```

```
> topics
      Topic 1  Topic 2  Topic 3
 [1,] "said"   "said"   "will"
 [2,] "one"    "one"    "said"
 [3,] "will"   "now"    "upon"
 [4,] "man"    "will"   "one"
 [5,] "mrs"    "little" "may"
 [6,] "little" "know"   "now"
 [7,] "lady"   "like"   "shall"
 [8,] "much"   "mrs"    "can"
 [9,] "know"   "never"  "must"
[10,] "old"    "well"   "much"
```

```r
# Create a bar plot to visualize the top terms in each topic
library(ggplot2)

topic_terms_df <- data.frame(
  Topic = rep(1:num_topics, each = 10),
  Term = unlist(topics),
  Frequency = rep(1:10, times = num_topics)
)

Term = unlist(topics)
ggplot(topic_terms_df, aes(x = reorder(Term, -Frequency), y = Frequency,
                           fill = factor(Topic))) + geom_bar(stat = "identity") +
  labs(title = "Top Terms in Each Topic", x = "Term", y = "Frequency") +
  theme_minimal() + facet_wrap(~Topic, scales = "free") +
  coord_flip() + scale_fill_brewer(palette = "Set1")
```

Output:



Conclusion:

Latent Dirichlet Allocation (LDA) has proven to be a valuable tool for understanding the underlying structure of large text corpora. By using LDA, we can extract meaningful topics, categorize documents, and improve information retrieval and text mining tasks. LDA has been particularly instrumental in fields such as natural language processing, information retrieval, and content recommendation.

Practical No:- 09

Aim:- MinHashing

Software Used: R Studio

Description:

The provided R code employs the textreuse package to perform text similarity analysis and plagiarism detection within a document corpus. It first establishes a Minhash generator with 240 hash functions and generates Minhash signatures for sample text fragments. The code creates a corpus by tokenizing documents into 5-grams, preserving the original tokens. It sets Locality-Sensitive Hashing (LSH) thresholds and probabilities to identify similar documents efficiently. LSH is then applied to cluster similar documents into buckets within the corpus. The code queries LSH to locate documents similar to a specified reference and retrieves potential candidate pairs for similarity assessment using Jaccard similarity. This workflow is valuable for detecting text reuse and potential plagiarism.

Algorithm:

1. It sets up a Minhash generator with 240 hash functions for estimating document similarity.
2. Demonstrates Minhash signature generation for sample text fragments.
3. Creates a corpus of documents by tokenizing them into 5-grams, generating Minhash signatures, and retaining original tokens.
4. Defines thresholds and probabilities for Locality-Sensitive Hashing (LSH), a technique for identifying similar documents.
5. Applies LSH to group similar documents into buckets within the corpus.
6. Queries LSH to find documents similar to a specified reference document.
7. Retrieves potential candidate pairs of similar documents.

Source Code:

```
library(textreuse)
minhash <- minhash_generator(n = 240, seed = 3552)
head(minhash(c("turn tokens into", "tokens into hashes", "into hashes fast")))
```

```
> library(textreuse)
> minhash <- minhash_generator(n = 240, seed = 3552)
> head(minhash(c("turn tokens into", "tokens into hashes", "into hashes fast")))
[1]  -715143991 -1568235737  -501611359 -2123423208  -417352961 -1579395341
```

```
dir <- system.file("extdata/ats", package = "textreuse")
corpus <- TextReuseCorpus(dir = dir, tokenizer = tokenize_ngrams, n = 5,
                          minhash_func = minhash, keep_tokens = TRUE,
                          progress = FALSE)
head(minhashes(corpus[[1]]))
```

```
> dir <- system.file("extdata/ats", package = "textreuse")
> corpus <- TextReuseCorpus(dir = dir, tokenizer = tokenize_ngrams, n = 5,
+                           minhash_func = minhash, keep_tokens = TRUE,
+                           progress = FALSE)
> head(minhashes(corpus[[1]]))
[1] -2147424503 -2147477293 -2147460327 -2147465030 -2147398192 -2147455577
```

```
length(minhashes(corpus[[1]]))
lsh_threshold(h = 200, b = 50)
lsh_threshold(h = 240, b = 80)
lsh_probability(h = 240, b = 80, s = 0.25)
lsh_probability(h = 240, b =  80, s = 0.75)
```

```
> length(minhashes(corpus[[1]]))
[1] 240
> lsh_threshold(h = 200, b = 50)
[1] 0.3760603
> lsh_threshold(h = 240, b = 80)
[1] 0.2320794
> lsh_probability(h = 240, b = 80, s = 0.25)
[1] 0.7163087
> lsh_probability(h = 240, b =  80, s = 0.75)
[1] 1
```

```
buckets <- lsh(corpus, bands = 80, progress = FALSE)
buckets
```

```
> buckets <- lsh(corpus, bands = 80, progress = FALSE)
Warning message:
`gather_()` was deprecated in tidyr 1.2.0.
i Please use `gather()` instead.
i The deprecated feature was likely used in the textreuse package.
  Please report the issue at <https://github.com/ropensci/textreuse/issues>.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
> buckets
# A tibble: 640 x 2
   doc                buckets
   <chr>              <chr>
 1 calltounconv00baxt 73bdd32491559ceccb6dd35862f2e8a1
 2 calltounconv00baxt 3357e9e6dc9bfa75bbd33f40d68ed6e5
 3 calltounconv00baxt c7a08877e8ff2f7b81099a5be08357bc
```

baxter_matches <- lsh_query(buckets, "calltounconv00baxt")
baxter_matches

```
> baxter_matches <- lsh_query(buckets, "calltounconv00baxt")
> baxter_matches
# A tibble: 1 x 2
  a                  b
  <chr>              <chr>
1 calltounconv00baxt lifeofrevrichard00baxt
```

candidates <- lsh_candidates(buckets)
candidates

```
> candidates <- lsh_candidates(buckets)
> candidates
# A tibble: 3 x 3
  a                    b                      score
  <chr>                <chr>                  <dbl>
1 calltounconv00baxt   lifeofrevrichard00baxt    NA
2 practicalthought00nev thoughtsonpopery00nevi   NA
3 remember00palm       remembermeorholy00palm    NA
```

lsh_compare(candidates, corpus, jaccard_similarity, progress = FALSE)

```
> lsh_compare(candidates, corpus, jaccard_similarity, progress = FALSE)
# A tibble: 3 x 3
  a                    b                      score
  <chr>                <chr>                  <dbl>
1 calltounconv00baxt   lifeofrevrichard00baxt 0.281
2 practicalthought00nev thoughtsonpopery00nevi 0.463
3 remember00palm       remembermeorholy00palm 0.701
>
```

Conclusion:

The overall algorithm utilizes Minhash and Locality-Sensitive Hashing to efficiently identify similar text fragments within a corpus and provides tools for further analysis and comparison of those documents.

Practical No:- 10

Aim:- K-Shingles

Software Used: R Studio

Description:

K-shingles, also known as k-grams or k-tuples, are a technique used in text analysis and data mining. They involve breaking down a text document into shorter, fixed-length sequences of characters or words. The "K" in K-shingles represents the length of each sequence, which can vary depending on the application.

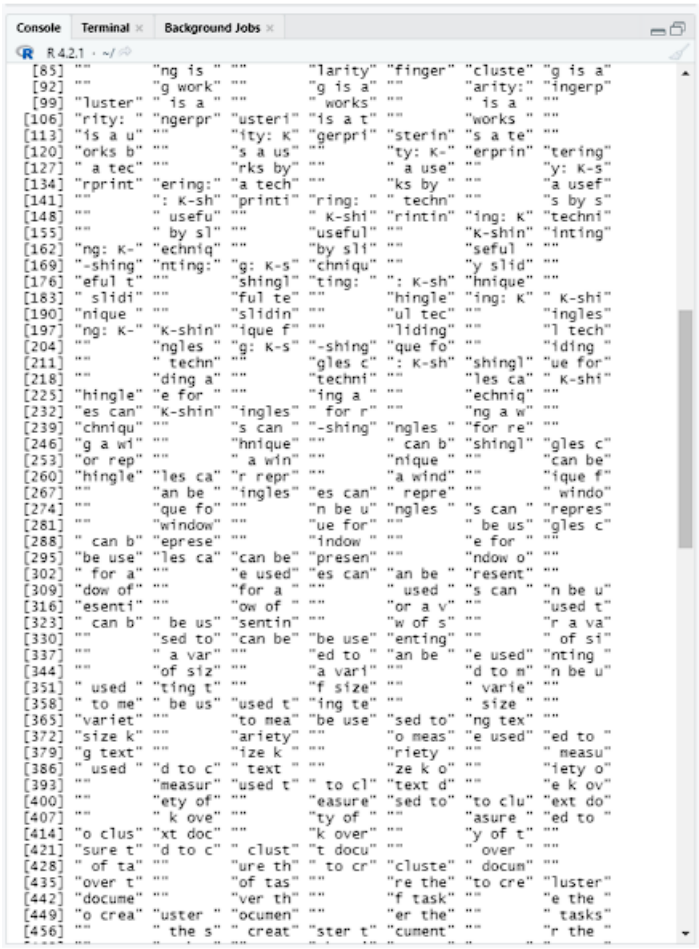
Program Code:

```
eadinteger <- function() {
n <- as.integer(readline(prompt = "Enter value of k-1: "))
# Check if the file exists
file_path <- "C:/Users/student/Desktop/Shingles/shingles.txt"
if (!file.exists(file_path)) {
print("Error: File not found.")
return(NULL)
}
# Read the file using a connection
con <- file(file_path, open = "r")
lines <- character(0)
```

```
# Read lines and handle incomplete lines
while (length(line <- readLines(con, n = 1)) > 0) {
# Check for incomplete lines and skip them
if (length(line) > 0) {
lines <- c(lines, line)
}
}
close(con) # Close the file connection
if (length(lines) == 0) {
print("Error: The file is empty.")
return(NULL)
}
Shingle <- c()
for (i in 1:(nchar(lines) - n + 1)) {
Shingle <- append(Shingle, substr(lines, start = i, stop = i + n - 1))
}
print(Shingle)
}
# Call the function
readinteger()
```

Output:

```
Console  Terminal  Background Jobs
R 4.2.1 · ~/
> readinteger <- function() {
+   n <- as.integer(readline(prompt = "Enter value of k-1: "))
+
+   # Check if the file exists
+   file_path <- "C:/Users/student/Desktop/Shingles/shingles.txt"
+   if (!file.exists(file_path)) {
+     print("Error: File not found.")
+     return(NULL)
+   }
+
+   # Read the file using a connection
+   con <- file(file_path, open = "r")
+   lines <- character(0)
+
+   # Read lines and handle incomplete lines
+   while (length(line <- readLines(con, n = 1)) > 0) {
+     # Check for incomplete lines and skip them
+     if (length(line) > 0) {
+       lines <- c(lines, line)
+     }
+   }
+
+   close(con) # Close the file connection
+
+   if (length(lines) == 0) {
+     print("Error: The file is empty.")
+     return(NULL)
+   }
+
+   Shingle <- c()
+
+   for (i in 1:(nchar(lines) - n + 1)) {
+     Shingle <- append(Shingle, substr(lines, start = i, stop = i + n - 1))
+   }
+
+   print(Shingle)
+ }
> # Call the function
> readinteger()
Enter value of k-1: 6
   [1] "K-shin" ""       "K-shin" ""       "K-shin" ""       "Text s"
   [8] "Docume" "Docume" "-shing" ""       "-shing" ""       "-shing"
  [15] ""       "ext si" "ocumen" "ocumen" "shingl" ""       "shingl"
  [22] ""       "shingl" ""       "xt sim" "cument" "cument" "hingli"
  [29] ""       "hingli" ""       "hingli" ""       "t simi" "ument "
  [36] "ument " "inglin" ""       "inglin" ""       "inglin" ""
  [43] " simil" "ment f" "ment c" "ngling" ""       "ngling" ""
  [50] "ngling" ""       "simila" "ent fi" "ent cl" "gling " ""
  [57] "gling " ""       "gling " ""       "imilar" "nt fin" "nt clu"
  [64] "ling i" ""       "ling w" ""       "ling i" ""       "milari"
  [71] "t fing" "t clus" "ing is" ""       "ing wo" ""       "ing is"
  [78] ""       "ilarit" " finge" " clust" "ng is " ""       "ng wor"
  [85] ""       "ng is " ""       "larity" "finger" "cluste" "g is a"
  [92] ""       "g work" ""       "g is a" ""       "arity:" "ingerp"
```



Conclusion:

K-shingles, whether at the character or word level, provide a versatile and powerful approach to text analysis and data mining. By breaking down text documents into fixed-length sequences, K-shingles enable a wide range of applications, from measuring text similarity and detecting plagiarism to data deduplication and document clustering.

The choice of the parameter K allows us to control the granularity of the shingles, making it possible to capture both fine-grained details and coarser patterns in the text. K-shingles are particularly valuable in situations where identifying common patterns, overlaps, and relationships between documents is crucial.

Practical No:- 11

Aim:- Map Reduce

Software Used: VMware, Cent Os, Eclipse, Cloudera

Description:

Hadoop Word Count is a classic and fundamental example in the world of big data and distributed computing. It demonstrates how to process and analyze a large collection of text documents to count the frequency of each unique word. This example is often used to introduce the Hadoop framework and MapReduce programming model.

Procedure:

On virtual box

- Computer > File System (/) > home > cloudera > downloads > (extract)
- Open Eclipse > create new java project

In project hierarchy

- Src folder right click > build path > configure build path
- New opened window > Libraries >add external JARs..
- Open extracted folder > Select all JAR files
- Right click on project folder in hierarchy > New > Package
- Give package name > finish
- Right click on wc package > New > Class
- Give class name

Program Code:

```
package wc;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class wcclass {
   public static class Map extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
         String line = value.toString();
         StringTokenizer tokenizer = new StringTokenizer(line);
         while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            output.collect(word, one);
         }
      }
   }
```

```
   public static class Reduce extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {

      public void reduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text, IntWritable> output, Reporter reporter) throws
IOException {

         int sum = 0;

         while (values.hasNext()) {

            sum += values.next().get();

         }

         output.collect(key, new IntWritable(sum));

      }

   }


   public static void main(String[] args) throws Exception {

      JobConf conf = new JobConf(wcclass.class);

      conf.setJobName("wordcount");

      conf.setOutputKeyClass(Text.class);

      conf.setOutputValueClass(IntWritable.class);

      conf.setMapperClass(Map.class);

      conf.setCombinerClass(Reduce.class);

      conf.setReducerClass(Reduce.class);

      conf.setInputFormat(TextInputFormat.class);

      conf.setOutputFormat(TextOutputFormat.class);

      FileInputFormat.setInputPaths(conf, new Path(args[0]));

      FileOutputFormat.setOutputPath(conf, new Path(args[1]));

      JobClient.runJob(conf);

   }
```

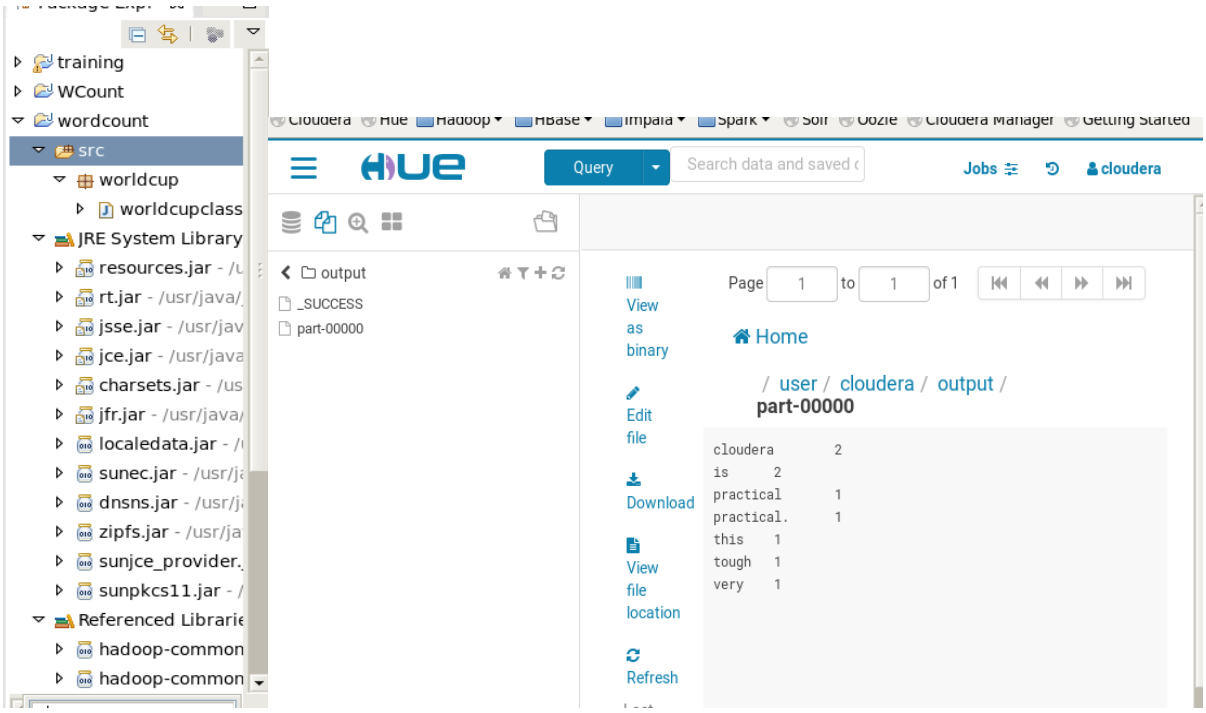
}

worldcupclass.java

```
1  package worldcup;
2
3  import java.io.IOException;
4  import java.util.*;
5
6
7  import org.apache.hadoop.fs.Path;
8  import org.apache.hadoop.io.*;
9  import org.apache.hadoop.mapred.*;
10
11
12 public class worldcupclass
13 {
14
15
16    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritab
17    {
18        private final static IntWritable one = new IntWritable(1);
19        private Text word = new Text();
20        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Report
21        {
22            String line = value.toString();
23            StringTokenizer tokenizer = new StringTokenizer(line);
24            while(tokenizer.hasMoreTokens())
25            {
26                word.set(tokenizer.nextToken());
27                output.collect(word,one);
28
29            }
30        }
31    }
32
33    public static class Reduce extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritab
34    {
35        public void reduce(Text key,Iterator<IntWritable>values,OutputCollector<Text,IntWritable>output
36        {
37            int sum = 0;
38            while(values.hasNext())
```

```
    public static class Reduce extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritab
    {
        public void reduce(Text key,Iterator<IntWritable>values,OutputCollector<Text,IntWritable>output
        {
            int sum = 0;
            while(values.hasNext())
            {
                sum+=values.next().get();
            }
            output.collect(key, new IntWritable(sum));
        }
    }

    public static void main(String[] args) throws Exception
    {
        JobConf conf = new JobConf(worldcupclass.class);
        conf.setJobName("wordcount");


        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);


        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));


        JobClient.runJob(conf);

    }
}
```

Output:



Conclusion:

Hadoop has had a profound impact on the field of big data, enabling organizations to efficiently store, process, and analyze vast amounts of data. Its scalability, distributed architecture, and ecosystem of tools make it a crucial component in the big data landscape, helping organizations gain insights and make data-driven decisions. However, it's important to recognize that Hadoop is just one piece of the big data puzzle, and it is often used in conjunction with other technologies to meet specific data processing needs.