

# PONDICHERRY UNIVERSITY

Public university in Pondicherry



## DATA MINING LAB– CSCE627

DHIVYA. K

21376005

PG Scholar

Master of Technology

Department of Computer Science and Engineering

I – Year & II – Semester

(Date of Submission - 18/06/2022)

Submitted to

**Dr. Pothula Sujatha**

Assistant Professor

Department of Computer Science and Engineering

## **LIST OF EXPERIMENTS:**

### **LAB EXERCISE:**

- 1. HADOOP – WORD COUNT BY MAP REDUCE**
- 2. SPARK WORD COUNT**
- 3. HADOOP – MATRIX MULTIPLICATION**
- 4. APRIORI IN R**
- 5. APRIORI IN WEKA**
- 6. APRIORI IN PYTHON**
- 7. BAYES IN WEKA**
- 8. BAYES CLASSIFIER USING R**
- 9. BAYESIAN IN PYTHON**

### **INTERNAL – 1:**

- 10. Demonstration of Association rule process on dataset contactlenses using apriori algorithm in WEKA**
- 11. Create a dataset in .arff file format. Demonstration of preprocessing using weka dataset**
- 12. Demonstration of Association rule process on any supermarket dataset using Apriori algorithm using R.**
- 13. Demonstration of classification ruleprocess on WEKA dataset using j48 algorithm**
- 14. Demonstration of classification rule process on WEKA dataset using naïve bayes algorithm. Compute the accuracy of the classifier, precision, and recall.**
- 15. Demonstration of clustering rule process on data set iris.arff using simple K.Means**
- 16. Demonstration of classification rule a process on student dataset using KNN algorithm in Python.**
- 17. Demonstration of classification rule a process on student dataset using KNN algorithm in Python.**
- 18. Write a Program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/ Python ML library classes can be used for this problem.**

### **INTERNAL – 2:**

- 19. KMEANS Clustering for IRIS DATASET using R**
- 20. Outliner Detection using K Means Classification algorithm on IRIS dataset using R**
- 21. KNN Algorithm in BodyFat dataset using R.**
- 22. Generate a decision tree using ID3 algorithm for iris dataset using R studio**
- 23. Apply random forest algorithm for bodyfat dataset using R**
- 24. Multiple Linear Regression using bodyfat dataset**

## **LAB EXERCISE:**

### **1. HADOOP – WORD COUNT BY MAP REDUCE**

#### **AIM:**

To find the number of words in a given data using Hadoop.

#### **PROCEDURE:**

**Step 1:** After installing Hadoop, navigate to assigned path of the Hadoop in command prompt.

**Step 2:** Start HDFS (Name node & Datanode) and YARN (Resource Manager and Node Manager)

Start –all.cmd

Start –dfs

Start –yarn

**Step 3:** Run word count Map Reduce Job.

Create a text file with same content c:\file1.txt (Pass this file as input to MapReduce)

Create a directory (input) in HDFS to keep all the text files by the command.

**Hdfs dfs-mkdir input**

Copy the text file (file1.txt) from local disk to the newly created input directory in HDFS.

**Hdfs dfs – copy From Local**

**C:/file1.txt input**

Check content of the copied file

**Hdfs dfs – ls input**

**Step 4:** Run the wordcount Map Reduce job bin\yarn jar share/Hadoop/mapreduce Hadoop-mapreduce-examples – 2.2.0.jar wordcount input output.

**Step 5:** check Output

**Hdfs dfs – cat output/\***

## OUTPUT:

```
[Administrator Command Prompt]
Microsoft Windows [Version 10.0 (9042.1586)]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\Windows\system32>jps
5232 DataNode
10564 NameNode
10490 Jps
10392 ResourceManager
6440 NodeManager

C:\Windows\system32>hadoop fs -mkdir /input1
mkdir: Cannot create directory /input1. Name node is in safe mode.

C:\Windows\system32>cd..

C:\Windows>
C:\Windows><d..>

C:\>cd hadoop
The system cannot find the path specified.

C:\>cd Hadoop-3.3.2
C:\hadoop-3.3.2>cd sbin

C:\hadoop-3.3.2\sbin>hadoop fs -mkdir /input1
C:\hadoop-3.3.2\sbin>hadoop fs -put C:/data1.txt /input1
C:\hadoop-3.3.2\sbin>hadoop fs -put C:/data1.txt /input1

C:\hadoop-3.3.2\sbin>hadoop fs -ls /input1/
Found 1 items
-rw-r--r-- 1 ADMIN supergroup 137 2022-04-06 11:17 /input1/data1.txt

C:\hadoop-3.3.2\sbin>hdfs -cat /input1/data1.txt
'hdfs' is not recognized as an internal or external command,
operable program or batch file.

C:\hadoop-3.3.2\sbin>hadoop dfs -cat /input1/data1.txt
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Hello all
```

```
[Administrator: Command Prompt]
C:\hadoop-3.3.2\sbin>hadoop fs -ls /input1/
Found 1 items
-rw-r--r-- 1 ADMIN supergroup 137 2022-04-06 11:17 /input1/data1.txt

C:\hadoop-3.3.2\sbin>dfs -cat /input1/data1.txt
'dfs' is not recognized as an internal or external command,
operable program or batch file.

C:\hadoop-3.3.2\sbin>hadoop dfs -cat /input1/data1.txt
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Hello all
hi all
This is Data mining session
It is Hadoop session
It is the end
What is data mining
What is Hadoop
thank you all

C:\hadoop-3.3.2\sbin>hadoop jar C:/hadoop-3.3.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
  grep: A map/reduce program that counts the matches of a regex in the input.
  join: A job that effects a join over sorted, equally partitioned datasets.
  multifilewc: A job that counts words from several files.
  pentomino: A map/reduce tile laying program to find solutions to pentomino problems.
  pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
  randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.
  randomwriter: A map/reduce program that writes 10GB of random data per node.
  secondarysort: An example defining a secondary sort to the reduce.
  sort: A map/reduce program that sorts the data written by the random writer.
  sudoku: A sudoku solver.
  teragen: Generate data for the terasort
  terasort: Run the terasort
  teravalidate: Checking results of terasort
  wordcount: A map/reduce program that counts the words in the input files.
  wordmean: A map/reduce program that counts the average length of the words in the input files.
  wordmedian: A map/reduce program that counts the median length of the words in the input files.
  wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.

C:\hadoop-3.3.2\sbin>hadoop jar C:/hadoop-3.3.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar wordcount /input1 /output1
2022-04-06 11:23:06,857 INFO client.DefaultNoHARNFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2022-04-06 11:23:09,084 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ADMIN/.staging/job_1649223789550_0001
2022-04-06 11:23:09,923 INFO input.FileInputFormat: Total input files to process : 1
```

```
C:\>Administrator: Command Prompt
teragen: Generate data for the terasort
terasort: Run the terasort
teravalidate: Checking results of terasort
wordcount: A map/reduce program that counts the words in the input files.
wordmean: A map/reduce program that counts the average length of the words in the input files.
wordmedian: A map/reduce program that counts the median length of the words in the input files.
wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.

C:\>hadoop-3.3.2\bin>hadoop jar C:/hadoop-3.3.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar wordcount /input1 /output1
2022-04-06 11:23:06,857 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at 0.0.0.0:8032
2022-04-06 11:23:09,084 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/job_1649223789550_0001
2022-04-06 11:23:10,139 INFO mapreduce.JobSubmitter: Total input files to process : 1
2022-04-06 11:23:10,139 INFO mapreduce.JobSubmitter: number of splits:1
2022-04-06 11:23:10,724 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1649223789550_0001
2022-04-06 11:23:10,724 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-04-06 11:23:11,309 INFO conf.Configuration: resource-types.xml not found
2022-04-06 11:23:11,314 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-04-06 11:23:12,763 INFO impl.YarnClientImpl: Submitted application application_1649223789550_0001
2022-04-06 11:23:13,733 INFO mapreduce.Job: The url to track the job: http://DESKTOP-A2FP1HT:8088/proxy/application_1649223789550_0001/
2022-04-06 11:23:13,804 INFO mapreduce.Job: Running job: job_1649223789550_0001
2022-04-06 11:23:47,254 INFO mapreduce.Job: Job job_1649223789550_0001 running in uber mode : false
2022-04-06 11:23:47,265 INFO mapreduce.Job: map 0% reduce 0%
2022-04-06 11:24:04,077 INFO mapreduce.Job: map 100% reduce 0%
2022-04-06 11:24:18,486 INFO mapreduce.Job: map 100% reduce 100%
2022-04-06 11:24:24,588 INFO mapreduce.Job: Job job_1649223789550_0001 completed successfully
2022-04-06 11:24:25,936 INFO mapreduce.Job: Counters: 54
    File System Counters
        FILE: Number of bytes read=181
        FILE: Number of bytes written=554093
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=240
        HDFS: Number of bytes written=111
        HDFS: Number of read operations=8
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
        HDFS: Number of bytes read erasure-coded=0
    Job Counters
        Launched map tasks=1
        Launched reduce tasks=1
        Data-local map tasks=1
        Total time spent by all maps in occupied slots (ms)=14136
        Total time spent by all reduces in occupied slots (ms)=11968
        Total time spent by all map tasks (ms)=14136
        Total time spent by all reduce tasks (ms)=11968
        Total vcore-milliseconds taken by all map tasks=14136
        Total vcore-milliseconds taken by all reduce tasks=11968
        Total megabyte-milliseconds taken by all map tasks=14475264
        Total megabyte-milliseconds taken by all reduce tasks=12255232
```

```

Administrator: Command Prompt
Launched map tasks=1
Launched reduce tasks=1
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=14136
Total time spent by all reduces in occupied slots (ms)=11968
Total time spent by all map tasks=14136
Total time spent by all reduce tasks=11968
Total vcore-milliseconds taken by all map tasks=14475264
Total megabyte-milliseconds taken by all reduce tasks=12255232
Map-Reduce Framework
  Map input records=8
  Map output records=27
  Map output bytes=237
  Map output materialized bytes=181
  Input split bytes=103
  Combine input records=27
  Co-partitioned splits=16
  Reduce input groups=16
  Reduce shuffle bytes=181
  Reduce input records=16
  Reduce output records=16
  Spilled Records=32
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map files=1
  GC time elapsed (ms)=191
  CPU time elapsed (ms)=323
  Physical memory (bytes) snapshot=444723200
  Virtual memory (bytes) snapshot=628838400
  Total committed heap usage (bytes)=297795584
  Peak Map Physical memory (bytes)=244936704
  Peak Map Virtual memory (bytes)=35388672
  Peak Reduce Physical memory (bytes)=199786496
  Peak Reduce Virtual memory (bytes)=293449728
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=111
File Output Format Counters
  Bytes Written=111

C:\hadoop-3.3.2\bin>hadoop fs -cat /output1/*
Data 1
Administrator: Command Prompt
CPU time spent (ms)=3323
Physical memory (bytes) snapshot=444723200
Virtual memory (bytes) snapshot=628838400
Total committed heap usage (bytes)=297795584
Peak Map Physical memory (bytes)=244936704
Peak Map Virtual memory (bytes)=35388672
Peak Reduce Physical memory (bytes)=199786496
Peak Reduce Virtual memory (bytes)=293449728
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=137
File Output Format Counters
  Bytes Written=111

C:\hadoop-3.3.2\bin>hadoop fs -cat /output1/*
Data 1
Hadoop 2
Hello 1
It 2
This 1
all 3
data 1
end 1
hi 1
is 5
mining 2
session 2
thank 1
the 1
what 2
you 1

C:\hadoop-3.3.2\bin>

```

The screenshot shows a web-based HDFS browser interface. At the top, there's a header bar with tabs for 'All Applications' and 'Browsing HDFS'. Below the header is a navigation bar with links for 'Hadoop', 'Overview', 'Datanodes', 'Datanode Volume Failures', 'Snapshot', 'Startup Progress', and 'Utilities'. The main content area is titled 'Browse Directory' and shows a table of file entries. The table columns include 'Permission', 'Owner', 'Group', 'Size', 'Last Modified', 'Replication', 'Block Size', and 'Name'. The entries listed are:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	ADMIN	supergroup	0 B	Apr 06 11:16	0	0 B	input
drwxr-xr-x	ADMIN	supergroup	0 B	Apr 06 11:17	0	0 B	input1
drwxr-xr-x	ADMIN	supergroup	0 B	Apr 02 14:53	0	0 B	out
drwxr-xr-x	ADMIN	supergroup	0 B	Apr 06 11:24	0	0 B	output1
drwx----	ADMIN	supergroup	0 B	Apr 02 14:52	0	0 B	tmp

Below the table, it says 'Showing 1 to 5 of 5 entries'. There are 'Previous' and 'Next' buttons. At the bottom, it says 'Hadoop, 2022.'

This screenshot shows two overlapping 'File information' pop-ups. The left pop-up is for 'data1.txt' and the right one is for 'part-r-00000'. Both pop-ups have sections for 'Block Information' (specifically Block 0), 'File contents', and 'Availability'.

**File information - data1.txt**

- Block ID: 1073741837
- Block Pool ID: BP-1944207902-192.168.2.1-1648624867201
- Generation Stamp: 1013
- Size: 137
- Availability:
  - DESKTOP-A2FPIHT

**File contents**

```
Hello all
hi all
This is Data mining session
It is Hadoop session
It is the end
what is data mining
what is Hadoop
thank you all
```

**File information - part-r-00000**

- Block ID: 1073741844
- Block Pool ID: BP-1944207902-192.168.2.1-1648624867201
- Generation Stamp: 1020
- Size: 111
- Availability:
  - DESKTOP-A2FPIHT

**File contents**

```
Data 1
Hadoop 2
Hello 1
It 2
This 1
all 3
data 1
end 1
```

## RESULT:

Thus, the word count has successfully implemented using Hadoop.

## 2. SPARK WORD COUNT

## **AIM:**

To find the number of words in complex word file using SPARK

### **PROCEDURE:**

## **STEP 1: Install SPARK in Windows**

**STEP 2:** Create one-word file and save it in C:\data.txt

**STEP 3:** Open New terminal and Type the Following command line by line

1. spark-shell
  2. val data=sc.textFile("c:/data.txt")
  3. data.collect;
  4. val splitdata = data.flatMap(line => line.split(" "));
  5. splitdata.collect;
  6. val mapdata = splitdata.map(word => (word,1));
  7. mapdata.collect;
  8. val reducedata = mapdata.reduceByKey(\_+\_);
  9. reducedata.collect;
  10. stop-all (after completing stop SPARK to run at background)

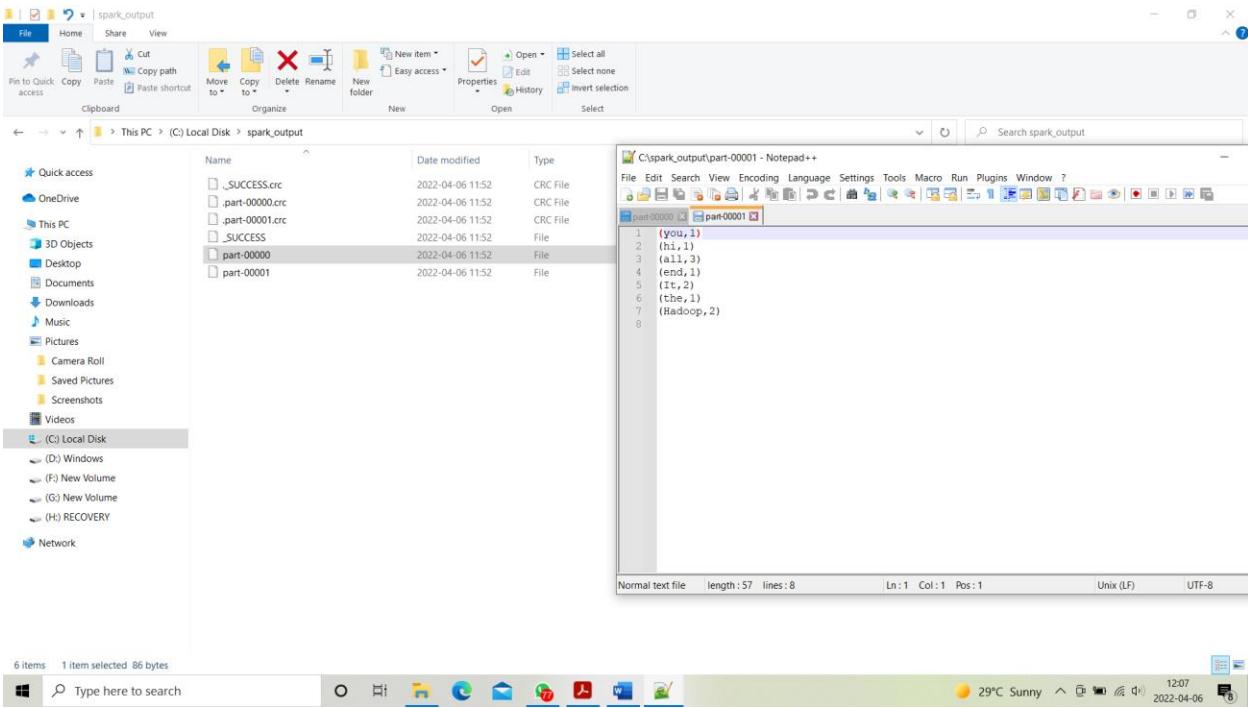
## OUTPUT:

```
Administrator: Command Prompt - spark-shell
scala> text.collect
res1: Array[String] = Array(Hello all, hi all, This is Data mining session, It is Hadoop session, It is the end, what is data mining, what is Hadoop, thank you all)
scala> val counts= text.flatMap(line =>(use greater than size here) line.split(" "))
<console>:1: error: ')' expected but '.' found
    val counts= text.flatMap(line =>(use greater than size here) line.split(" "))
                                         ^
<console>:1: error: ';' expected but ')' found.
    val counts= text.flatMap(line =>(use greater than size here) line.split(" "))
                                         ^
scala> val counts= text.flatMap(line => line.split(" "))
counts: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:25
scala> counts.collect
res2: Array[String] = Array(Hello, all, hi, all, This, is, Data, mining, session, It, is, Hadoop, session, It, is, the, end, what, is, data, mining, what, is, Hadoop, thank, you, all)
scala> val mapf = counts.map(word => (word,1))
mapf: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:25
scala> mapf.collect
res3: Array[(String, Int)] = Array((Hello,1), (all,1), (hi,1), (all,1), (This,1), (is,1), (Data,1), (mining,1), (session,1), (It,1), (is,1), (Hadoop,1), (session,1), (It,1), (is,1), (the,1), (end,1), (what,1), (is,1), (data,1), (mining,1), (what,1), (is,1), (Hadoop,1), (thank,1), (you,1), (all,1))
scala> val reducef=mapf.reduceByKey(+)
<console>:25: error: not found: value +
    val reducef=mapf.reduceByKey(+)
                                         ^
scala> val reducef=mapf.reduceByKey(_+_)
reducef: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:25
scala> reducef.collect
res4: Array[(String, Int)] = Array((mining,2), (is,5), (Hello,1), (thank,1), (session,2), (data,1), (This,1), (what,2), (Data,1), (you,1), (hi,1), (all,3), (end,1), (It,2), (the,1), (Hadoop,2))
scala> reducef.saveAsTextFile("C:/spark_output")
scala>
```

29°C Sunny 12:01 2022-04-06

The screenshot shows a Windows desktop environment. In the center, there is a file explorer window titled 'spark\_output' located at 'C:\spark\_output\part-00000'. The file list includes several files: '\_SUCCESS.crc', 'part-00000.crc', 'part-00001.crc', '\_SUCCESS', 'part-00000', and 'part-00001'. The '\_SUCCESS' file was selected. To the right of the file explorer is a Notepad+ window titled 'C:\spark\_output\part-00000 - Notepad+'. The text content of the file is displayed in the Notepad+ window, showing the same data as the Scala command prompt: Hello all, hi all, This is Data mining session, It is Hadoop session, It is the end, what is data mining, what is Hadoop, thank you all. The Notepad+ status bar indicates it's a 'Normal text file' with 86 length, 10 lines, and position 87.

6 items 1 item selected 86 bytes 29°C Sunny 12:07 2022-04-06



## RESULT:

Thus, the word count has successfully implemented using SPARK.

### 3. HADOOP – MATRIX MULTIPLICATION

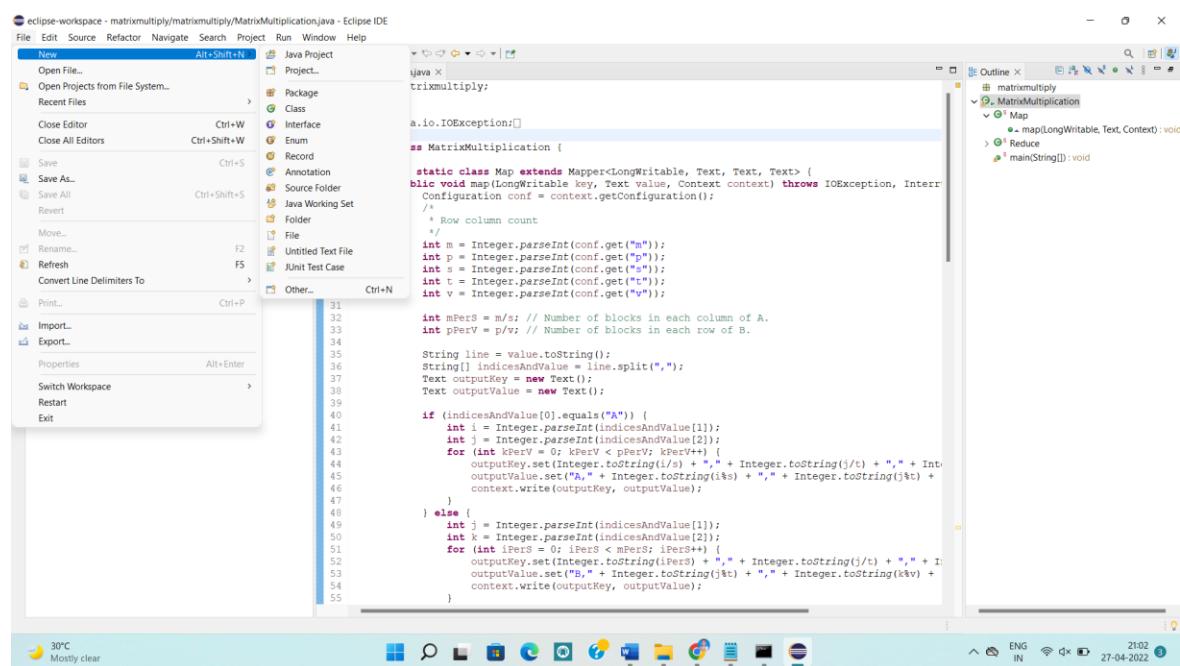
#### AIM:

To implement matrix multiplication in Hadoop map reduce

#### PROCEDURE:

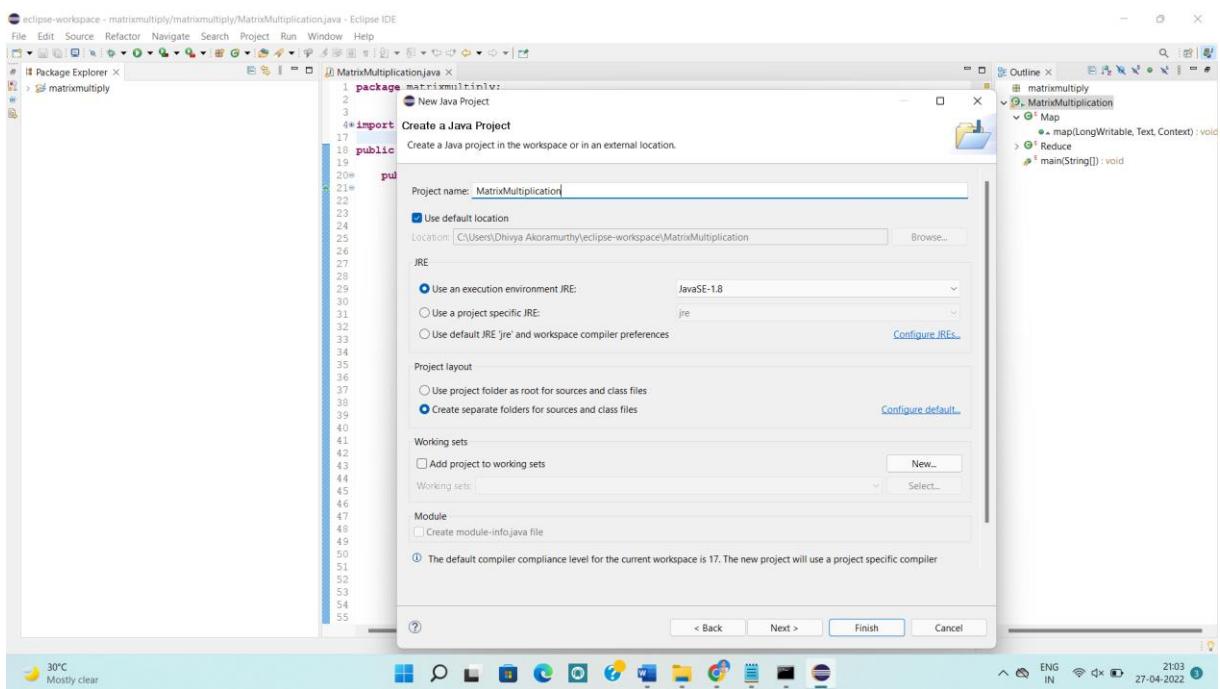
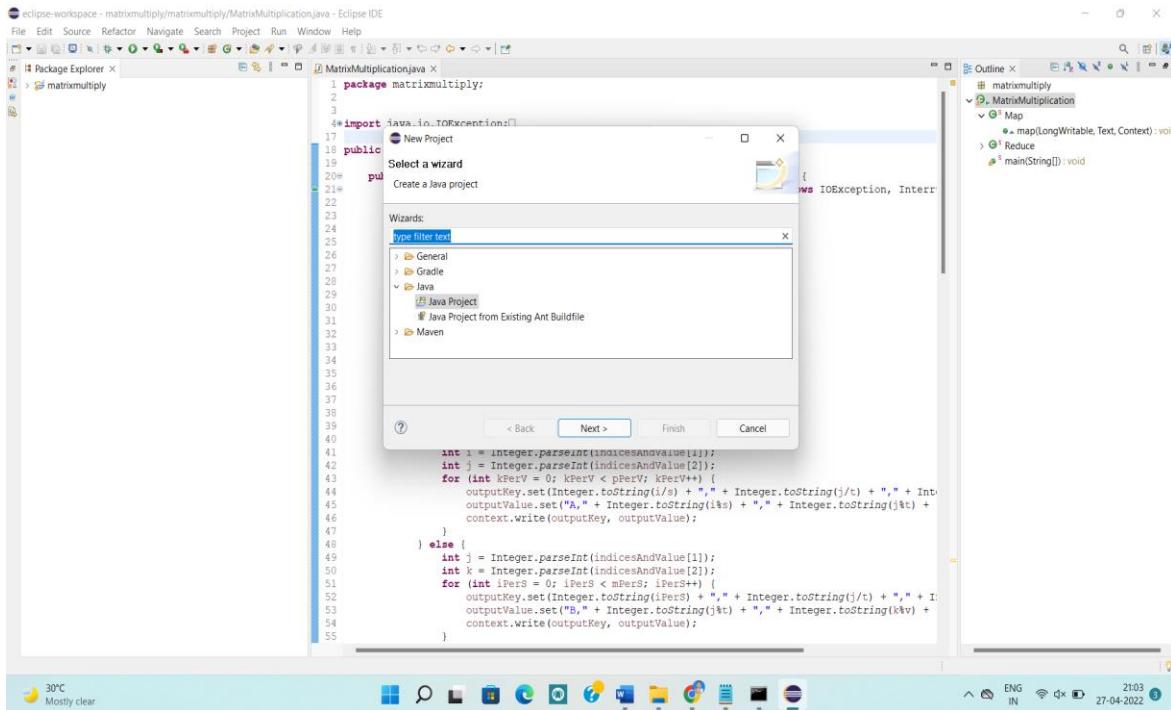
In Eclipse workspace – java file name as MatrixMultiplication.java

Go to: File – new – project – java project – project name(Path name)



The screenshot shows the Eclipse IDE interface with the following details:

- File Menu:** New (Alt+Shift+N), Open File..., Open Projects from File System..., Recent Files, Close Editor, Close All Editors, Save, Save As..., Save All, Revert, Move..., Rename..., Refresh, Convert Line Delimiters To, Print..., Import..., Export..., Properties, Switch Workspace, Restart, Exit.
- Java Project View:** Java Project, Package, Class, Interface, Enum, Record, Annotation, Source Folder, Java Working Set, Folder, File, Untitled Text File, JUnit Test Case, Other... (Ctrl+N).
- Code Editor:** Shows the `MatrixMultiplication.java` file content. The code implements a Mapper for matrix multiplication. It reads input lines, parses them into indices and values, and then performs the multiplication based on the indices. The code uses Java's `LongWritable`, `Text`, and `Context` classes from the Hadoop API.
- Outline View:** Shows the class structure: `matrixmultiply` contains a `Map` implementation with a `map` method and a `main` method.
- Bottom Status Bar:** Shows the weather (30°C, Mostly clear), system icons (ENG IN, battery, signal), and the date/time (27-04-2022, 21:02).



The screenshot shows the Eclipse IDE interface with the following details:

- File Path:** eclipse-workspace - matrixmultiply/matrixmultiply/MatrixMultiplication.java - Eclipse IDE
- Editor View:** The main window displays the Java code for `MatrixMultiplication`. The code implements a MapReduce job for matrix multiplication. It includes imports for `java.io.IOException`, `java.util.*`, `org.apache.hadoop.fs.Path`, `org.apache.hadoop.conf.*`, `org.apache.hadoop.io.*`, `org.apache.hadoop.mapreduce.*`, `org.apache.hadoop.mapreduce.lib.input.FileInputFormat`, `org.apache.hadoop.mapreduce.lib.input.TextInputFormat`, `org.apache.hadoop.mapreduce.lib.output.FileOutputFormat`, and `org.apache.hadoop.mapreduce.lib.output.TextOutputFormat`.
- Outline View:** The right-hand panel shows the class structure:
  - `matrixmultiply`
  - `MatrixMultiplication` (selected)
    - `Map`
      - `map(LongWritable, Text, Text, Text)`
    - `Reduce`
    - `main(String[])`
- System Tray:** At the bottom, it shows the date (27-04-2022), time (21:02), battery level (ENG IN), signal strength, and weather information (30°C, Mostly clear).

We have to save java file in below location

**C:\Users\DHIVYA\Akoramurthy\eclipse-workspace\matrixmultiply\matrixmultiply\MatrixMultiplication.java (example)**

## MatrixMultiplication.java

NOTE: class name and file name of java should be same

```
package matrixmultiply;

import java.io.IOException;
import java.util.*;
import java.util.AbstractMap.SimpleEntry;
import java.util.Map.Entry;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
```

```

public class MatrixMultiplication {

    public static class Map extends Mapper<LongWritable, Text, Text, Text> {
        public void map(LongWritable key, Text value, Context context) throws IOException,
        InterruptedException {
            Configuration conf = context.getConfiguration();
            /*
             * Row column count
             */
            int m = Integer.parseInt(conf.get("m"));
            int p = Integer.parseInt(conf.get("p"));
            int s = Integer.parseInt(conf.get("s"));
            int t = Integer.parseInt(conf.get("t"));
            int v = Integer.parseInt(conf.get("v"));

            int mPerS = m/s; // Number of blocks in each column of A.
            int pPerV = p/v; // Number of blocks in each row of B.

            String line = value.toString();
            String[] indicesAndValue = line.split(",");
            Text outputKey = new Text();
            Text outputValue = new Text();

            if (indicesAndValue[0].equals("A")) {
                int i = Integer.parseInt(indicesAndValue[1]);
                int j = Integer.parseInt(indicesAndValue[2]);
                for (int kPerV = 0; kPerV < pPerV; kPerV++) {
                    outputKey.set(Integer.toString(i/s) + "," + Integer.toString(j/t) + "," +
                    Integer.toString(kPerV));
                    outputValue.set("A," + Integer.toString(i%s) + "," + Integer.toString(j%t) + "," +
                    indicesAndValue[3]);
                    context.write(outputKey, outputValue);
                }
            } else {
                int j = Integer.parseInt(indicesAndValue[1]);
                int k = Integer.parseInt(indicesAndValue[2]);
                for (int iPerS = 0; iPerS < mPerS; iPerS++) {
                    outputKey.set(Integer.toString(iPerS) + "," + Integer.toString(j/t) + "," +
                    Integer.toString(k/v));

```

```

        outputValue.set("B," + Integer.toString(j%t) + "," + Integer.toString(k%v) + "," +
indicesAndValue[3]);
        context.write(outputKey, outputValue);
    }
}
}

public static class Reduce extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
        String[] value;
        ArrayList<Entry<String, Float>> listA = new ArrayList<Entry<String, Float>>();
        ArrayList<Entry<String, Float>> listB = new ArrayList<Entry<String, Float>>();
        for (Text val : values) {
            value = val.toString().split(",");
            if (value[0].equals("A")) {
                listA.add(new SimpleEntry<String, Float>(value[1] + "," + value[2],
Float.parseFloat(value[3])));
            } else {
                listB.add(new SimpleEntry<String, Float>(value[1] + "," + value[2],
Float.parseFloat(value[3])));
            }
        }
        String[] iModSAndJModT;
        String[] jModTAndKModV;
        float a_ij;
        float b_jk;
        String hashKey;
        HashMap<String, Float> hash = new HashMap<String, Float>();
        for (Entry<String, Float> a : listA) {
            iModSAndJModT = a.getKey().split(",");
            a_ij = a.getValue();
            for (Entry<String, Float> b : listB) {
                jModTAndKModV = b.getKey().split(",");
                b_jk = b.getValue();
                if (iModSAndJModT[1].equals(jModTAndKModV[0])) {
                    hashKey = iModSAndJModT[0] + "," + jModTAndKModV[1];
                    if (hash.containsKey(hashKey)) {
                        hash.put(hashKey, hash.get(hashKey) + a_ij*b_jk);
                    }
                }
            }
        }
    }
}

```

```

        } else {
            hash.put(hashKey, a_ij*b_jk);
        }
    }
}

String[] blockIndices = key.toString().split(",");
String[] indices;
String i;
String k;
Configuration conf = context.getConfiguration();
int s = Integer.parseInt(conf.get("s"));
int v = Integer.parseInt(conf.get("v"));
Text outputValue = new Text();
for (Entry<String, Float> entry : hash.entrySet()) {
    indices = entry.getKey().split(",");
    i = Integer.toString(Integer.parseInt(blockIndices[0])*s + Integer.parseInt(indices[0]));
    k = Integer.toString(Integer.parseInt(blockIndices[2])*v +
    Integer.parseInt(indices[1]));
    outputValue.set(i + "," + k + "," + Float.toString(entry.getValue()));
    context.write(null, outputValue);
}
}
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    // A is an m-by-n matrix; B is an n-by-p matrix.
    conf.set("m", "2");
    conf.set("n", "5");
    conf.set("p", "3");
    conf.set("s", "2"); // Number of rows in a block in A.
    conf.set("t", "5"); // Number of columns in a block in A = number of rows in a block in B.
    conf.set("v", "3"); // Number of columns in a block in B.

    Job job = new Job(conf, "Multiplication");
    job.setJarByClass(MatrixMultiplication.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
}

```

```
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```

**data.txt**

A,0,1,1.0  
A,0,2,2.0  
A,0,3,3.0  
A,0,4,4.0  
B,3,1,10.0  
B,3,2,11.0  
A,1,0,5.0  
A,1,1,6.0  
A,1,2,7.0  
A,1,3,8.0  
A,1,4,9.0  
B,0,1,1.0  
B,0,2,2.0  
B,1,0,3.0  
B,1,1,4.0  
B,1,2,5.0  
B,2,0,6.0  
B,2,1,7.0  
B,2,2,8.0  
B,3,0,9.0  
B,4,0,12.0  
B,4,1,13.0  
B,4,2,14.0

## In Browser:

Name node: Port number – 9870

Check the file is in Hadoop or not

The screenshot shows a web browser window titled "Browsing HDFS" with the URL "localhost:9870/explorer.html#/input". The browser has tabs for WhatsApp and other applications. The main content area is titled "Browse Directory" and shows a table of files in the "/input" directory. The table columns are: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, Name, and a delete icon. There is one entry: "data.txt" owned by "Dhivya" in "supergroup" with a size of 256 B, last modified on Apr 27 16:45, replication factor of 1, and a block size of 128 MB. The browser status bar at the bottom shows the URL "localhost:9870/explorer.html#", the weather "30°C Mostly clear", and the date/time "27-04-2022 21:35".

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
	-rw-r--r--	Dhivya	supergroup	256 B	Apr 27 16:45	1	128 MB	data.txt	trash

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2022.

localhost:9870/explorer.html#  
30°C Mostly clear  
27-04-2022 21:35

**In Command Prompt, open cmd as administrator**

```
>Start-all  
>hdfs dfs -mkdir /input2  
>hdfs dfs -put C:\hadoopprj\data.txt /input2  
>hdfs dfs -cat C:\hadoopprj\data.txt /input2  
>hdfs dfs -cat /input2/data.txt  
A,0,1,1.0  
A,0,2,2.0  
A,0,3,3.0  
A,0,4,4.0  
B,3,1,10.0  
B,3,2,11.0  
A,1,0,5.0  
A,1,1,6.0  
A,1,2,7.0  
A,1,3,8.0  
A,1,4,9.0  
B,0,1,1.0  
B,0,2,2.0  
B,1,0,3.0  
B,1,1,4.0  
B,1,2,5.0  
B,2,0,6.0  
B,2,1,7.0  
B,2,2,8.0  
B,3,0,9.0  
B,4,0,12.0  
B,4,1,13.0  
B,4,2,14.0  
>hadoop jar C:\hadoopprj\multiply.jar matrixmultiply.MatrixMultiplication /input2 put23  
>hdfs dfs -cat put23/*  
0,090.0  
1,0240.0  
0,1100.0  
1,1275.0  
0,2110.0  
1,2310.0  
>stop-all (after completing we need to close hadoop)
```

## OUTPUT SCREENSHOT:

```

Administrator: Command Prompt
Microsoft Windows [Version 10.0_22000.556]
(c) Microsoft Corporation. All rights reserved.

C:\windows\system32>start-all
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\windows\system32>hdfs dfs -mkdir /input
mkdir: '/input': File exists

C:\windows\system32>hdfs dfs -mkdir /input2
C:\windows\system32>hdfs dfs -put C:\hadoopprj\data.txt /input2

C:\windows\system32>hdfs dfs -cat C:\hadoopprj\data.txt /input2
Usage: hdfs dfs [generic options]
      [-appendToFile <localsrc> ... <dst>]
      [-atomic [ -ignorecrc ] <src> ...]
      [-checksum [ -v ] <src> ...]
      [-chgrp [ -R ] GROUP PATH...]
      [-chmod [ -R ] OWNER|[GROUP] PATH...]
      [-chown [ -R ] OWNER|[GROUP] PATH...]
      [-concat <target path> <src path> <src path> ...]
      [-copyFromLocal [ -p ] [ -t ] [ -d ] [ -q <thread count> ] [ -q <thread pool queue size> ] <localsrc> ... <dst>]
      [-copyToLocal [ -p ] [ -t ] [ -d ] [ -q <thread count> ] [ -q <thread pool queue size> ] <src> ... <localdst>]
      [-creat [ -q ] [ -t ] [ <storage type> ] [ -x ] [ -s ] [ -z ] <path> ...
      [-cp [ -f ] [ -p ] [ -p | -pt | -px ] [ -d ] [ -t <thread count> ] [ -q <thread pool queue size> ] <src> ... <dst>]
      [-createSnapshot <snapshotDir> [<snapshotName>]]
      [-deleteSnapshot <snapshotDir> <snapshotName>]
      [-df [ -h ] [ <path> ...]]
      [-du [ -s ] [ -h ] [ -v ] [ -x ] <path> ...]
      [-expunge [ -immediate ] [ -fs <path> ]]
      [-find [ -f ] [ <expression> ... ]
      [-get [ -f ] [ -p ] [ -l ] [ -ignorecrc ] [ -t <thread count> ] [ -q <thread pool queue size> ] <src> ... <localdst>]
      [-getfacl [ -R ] <path>]
      [-getfattr [ -R ] [ -n name | -d ] [ -e en ] <path>]
      [-getmerge [ -n1 ] [ -skipEmptyFile ] <src> <localdst>]
      [-head <file>]
      [-help [ <cmd> ...]]
      [-ls [ -d ] [ -h ] [ -q ] [ -R ] [ -t ] [ -S ] [ -r ] [ -u ] [ -e ] [ <path> ...]]
      [-mdir [ -p ] <path> ...]
      [-moveFromLocal [ -f ] [ -p ] [ -d ] <localsrc> ... <dst>]
      [-moveToLocal <src> <localdst>]
      [-my <src> ... <dst>]
      [-pe [ -f ] [ -p ] [ -d ] [ -t <thread count> ] [ -q <thread pool queue size> ] <localsrc> ... <dst>]
      [-renameSnapshot <snapshotDir> <oldName> <newName>]
      [-rm [ -f ] [ -r ] [ -R ] [ -skiptrash ] [ -safely ] <src> ...]
      [-rmdir [ -e ] [ -ignore-fail-on-non-empty ] <dir> ...]
  
```

```

Administrator: Command Prompt
[-t <TIMESTAMP> (yyyy-MM-dd HH:mm:ss) ] [-c] <path> ...
[-touchz <path> ...]
[-truncate [ -u ] <length> <path> ...]
[-usage [ <cmd ... ]]

Generic options supported are:
--conf <configuration file>           specify an application configuration file
-D <property>[=value]                  define a value for a given property
--fs <file:///hdfs://namenode>[port]  specify default filesystem URL to use, overrides 'fs.defaultFS' property from configurations.
--jt <jobtracker>[<resource manager>]  specify a ResourceManager
--files <file>...                      specify a comma-separated list of files to be copied to the map reduce cluster
--libjars <jar1>,...                   specify a comma-separated list of jar files to be included in the classpath
--archives <archive1>,...              specify a comma-separated list of archives to be unarchived on the compute machines

The general command line syntax is:
command [genericOptions] [commandOptions]

Usage: hadoop fs [generic options] -cat [-ignorecrc] <src> ...

C:\windows\system32>hdfs dfs -cat /input2
cat: '/input2': Is a directory

C:\windows\system32>hdfs dfs -cat /input2/data.txt
A,0,1,1.0
A,0,2,2.0
A,0,3,3.0
A,0,4,4.0
A,0,5,5.0
A,0,6,6.0
A,0,7,7.0
A,0,8,8.0
A,0,9,9.0
A,1,0,10.0
A,0,2,11.0
A,1,0,12.0
A,1,1,13.0
A,1,2,14.0
A,1,3,15.0
A,1,4,16.0
A,1,5,17.0
A,1,6,18.0
A,1,7,19.0
A,1,8,20.0
A,1,9,21.0
A,2,0,22.0
A,2,1,23.0
A,2,2,24.0
A,2,3,25.0
A,2,4,26.0
A,2,5,27.0
A,2,6,28.0
A,2,7,29.0
A,2,8,30.0
A,2,9,31.0
A,3,0,32.0
A,3,1,33.0
A,3,2,34.0
A,3,3,35.0
A,3,4,36.0
A,3,5,37.0
A,3,6,38.0
A,3,7,39.0
A,3,8,40.0
A,3,9,41.0
A,4,0,42.0
A,4,1,43.0
A,4,2,44.0
A,4,3,45.0
A,4,4,46.0
A,4,5,47.0
A,4,6,48.0
A,4,7,49.0
A,4,8,50.0
A,4,9,51.0
A,5,0,52.0
A,5,1,53.0
A,5,2,54.0
A,5,3,55.0
A,5,4,56.0
A,5,5,57.0
A,5,6,58.0
A,5,7,59.0
A,5,8,60.0
A,5,9,61.0
A,6,0,62.0
A,6,1,63.0
A,6,2,64.0
A,6,3,65.0
A,6,4,66.0
A,6,5,67.0
A,6,6,68.0
A,6,7,69.0
A,6,8,70.0
A,6,9,71.0
A,7,0,72.0
A,7,1,73.0
A,7,2,74.0
A,7,3,75.0
A,7,4,76.0
A,7,5,77.0
A,7,6,78.0
A,7,7,79.0
A,7,8,80.0
A,7,9,81.0
A,8,0,82.0
A,8,1,83.0
A,8,2,84.0
A,8,3,85.0
A,8,4,86.0
A,8,5,87.0
A,8,6,88.0
A,8,7,89.0
A,8,8,90.0
A,8,9,91.0
A,9,0,92.0
A,9,1,93.0
A,9,2,94.0
A,9,3,95.0
A,9,4,96.0
A,9,5,97.0
A,9,6,98.0
A,9,7,99.0
A,9,8,100.0
A,9,9,101.0
C:\windows\system32>hadoop jar C:\hadoopprj\matrixmultiply.jar matrixmultiply.MatrixMultiplication /input2
2022-04-27 20:48:42,730 INFO client.DefaultHttpAgnFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2022-04-27 20:48:43,243 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
  
```

```
Administrator: Command Prompt
0,4,1,13,0
0,4,2,14,0
C:\Windows\system32\hadoop jar C:\hadoop\bin\multiplay.jar matrixmultiply.MatrixMultiplication /input2 pu
2022-04-27 20:48:43,230 INFO client.DefaultHttpAclService$ProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2022-04-27 20:48:43,243 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the ToolRunner to remedy this.
2022-04-27 20:48:43,274 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Dhivya/.staging/job_1651072035602_0001
2022-04-27 20:48:43,568 INFO mapreduce.JobSubmitter: Total input files to process : 1
2022-04-27 20:48:43,588 INFO mapreduce.JobSubmitter: number of splits:1
2022-04-27 20:48:43,682 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1651072035602_0001
2022-04-27 20:48:43,682 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-04-27 20:48:43,841 INFO conf.Configuration: resource-types.xml not found
2022-04-27 20:48:43,851 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-04-27 20:48:44,251 INFO impl.YarnClientImpl: Application application_1651072035602_0001
2022-04-27 20:48:44,350 INFO mapreduce.Job: The url to track the job: http://LAPTOP-RAOHV1KF:8088/proxy/application_1651072035602_0001/
2022-04-27 20:48:44,352 INFO mapreduce.Job: Running job: job_1651072035602_0001
2022-04-27 20:48:51,596 INFO mapreduce.Job: Job map 100% reduce 0%
2022-04-27 20:48:51,598 INFO mapreduce.Job: map 100% reduce 0%
2022-04-27 20:49:01,835 INFO mapreduce.Job: map 100% reduce 0%
2022-04-27 20:49:02,043 INFO mapreduce.Job: Counters: 54
File System
    FILE: Number of bytes read=425
    FILE: Number of bytes written=556237
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=358
    HDFS: Number of bytes written=59
    HDFS: Number of read operations=8
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
    HDFS: Number of bytes read erasure-coded=0
Job Counters
    Launched map tasks=1
    Launched reduce tasks=1
    Data-local map tasks=1
    Total time spent by all maps in occupied slots (ms)=2328
    Total time spent by all reduces in occupied slots (ms)=2628
    Total time spent by all map tasks (ms)=2328
    Total time spent by all reduce tasks (ms)=2628
    Total vcore-milliseconds taken by all map tasks=2328
    Total vcore-milliseconds taken by all reduce tasks=2628
    Total megabyte-milliseconds taken by all map tasks=2383872
    Total megabyte-milliseconds taken by all reduce tasks=2691072
Map-Reduce Framework
    Map input records=23
    Map output records=23
    Map output bytes=373
30°C
Mostly clear
ENG IN 21:31 27-04-2022
```

```
Administrator: Command Prompt
Total vcore-milliseconds taken by all map tasks=2328
Total vcore-milliseconds taken by all reduce tasks=2628
Total megabyte-milliseconds taken by all map tasks=2383872
Total megabyte-milliseconds taken by all reduce tasks=2691072
Map-Reduce Framework
    Map input records=23
    Map output records=23
    Map output bytes=373
    Map output materialized bytes=425
    Input split bytes=102
    Input split bytes=102
    Combine input records=0
    Combine output records=0
    Reduce input groups=1
    Reduce shuffle bytes=425
    Reduce input records=23
    Reduce output records=6
    Spilled Records=46
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=79
    CPU time spent (ms)=245
    Physical memory (bytes) snapshot=595968000
    Virtual memory (bytes) snapshot=825311232
    Total committed heap usage (bytes)=457703424
    Peak Map Physical memory (bytes)=360071168
    Peak Map Virtual memory (bytes)=460787712
    Peak Reduce Physical memory (bytes)=235896832
    Peak Reduce Virtual memory (bytes)=364597248
Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
File Input Format Counters
    Bytes Read=256
File Output Format Counters
    Bytes Written=59
C:\Windows\system32\dfs -cat pu/*
0,0,90,0
1,0,240,0
0,1,100,0
1,1,275,0
0,2,110,0
1,2,310,0
30°C
Mostly clear
ENG IN 21:32 27-04-2022
```

## RESULT:

Thus, the implementation of Matrix multiplication using Hadoop is shown above.

#### **4. APRIORI IN R:**

##### **AIM:**

To implement Apriori algorithm in R.

##### **PROCEDURE:**

**Step 1:** The first part of any analysis is to bring in the dataset. We will be using an inbuilt dataset “Groceries” from the ‘arules’ package to simplify our analysis.

**Step 2:** All stores and retailers store their information of transactions in a specific type of dataset called the “Transaction” type dataset.

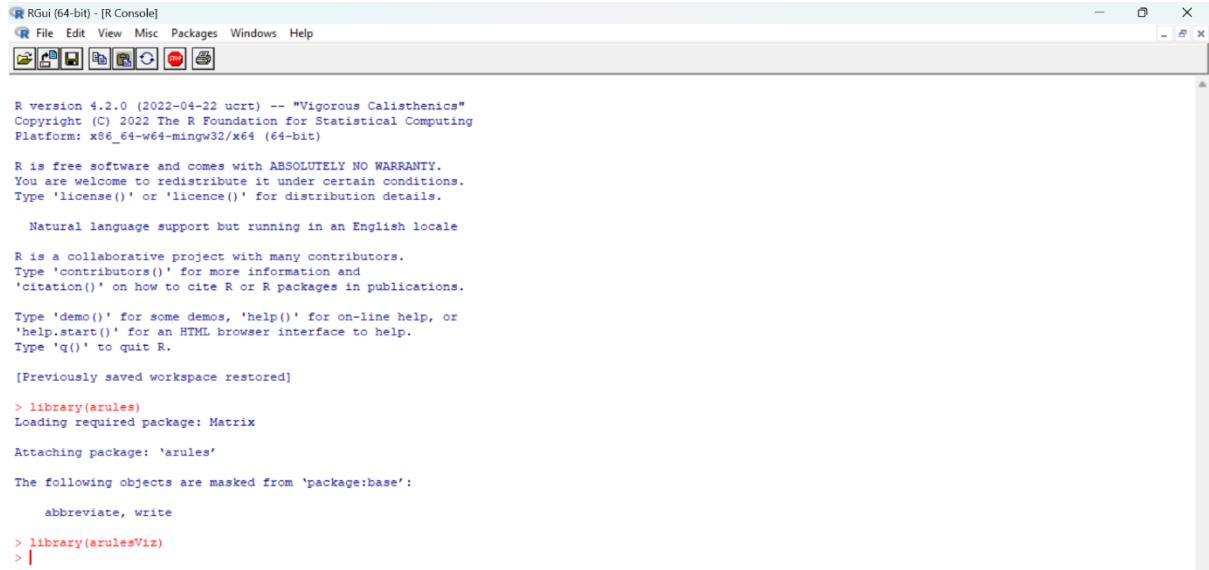
**Step 3:** The ‘pacman’ package is an assistor to help load and install the packages. we will be using pacman to load the arules package.

**Step 4:** The p\_load() function from “pacman” takes names of packages as arguments.

**Step 5:** If your system has those packages, it will load them and if not, it will install and load them.

##### **PROGRAM AND OUTPUT SCREENSHOT:**

```
library(arules)
library(arulesViz)
```



```
R version 4.2.0 (2022-04-22 ucrt) -- "Vigorous Calisthenics"
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.

Natural language support but running in an English locale

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.

[Previously saved workspace restored]

> library(arules)
Loading required package: Matrix

Attaching package: 'arules'

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

abbreviate, write

> library(arulesViz)
> |
```

```
data(Groceries)
```

```
data <- list(c("a","b","c"),
c("a","b"),
c("a","b","d"),
c("b","e"),
c("b","c","e"),
c("a","d","e"),
c("a","c"),
c("a","b","d"),
c("c","e"),
c("a","b","d","e"),
c("a",'b','e','c')
)
data <- as(data, "transactions")
inspect(data)
tl <- as(data, "tidLists")
inspect(tl)
```

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

Attaching package: 'arules'

The following objects are masked from 'package:base':
  abbreviate, write

> library(arulesViz)
> data(Groceries)
> data <- list(c("a","b","c"),
+ c("a","b"),
+ c("a","b","d"),
+ c("b","e"),
+ c("b","c","e"),
+ c("a","d","e"),
+ c("a","c"),
+ c("a","b","d"),
+ c("a","b","d","e"),
+ c("a","b','e','c')
+ )
> data <- as(data, "transactions")
> inspect(data)
  items
[1] {a, b, c}
[2] {a, b}
[3] {a, b, d}
[4] {b, e}
[5] {b, c, e}
[6] {a, d, e}
[7] {a, c}
[8] {a, b, d}
[9] {c, e}
[10] {a, b, d, e}
[11] {a, b, c, e}
> |
```

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> data <- list(c("a","b","c"),
+ c("a","b"),
+ c("a","b","d"),
+ c("b","e"),
+ c("b","c","e"),
+ c("a","d","e"),
+ c("a","c"),
+ c("a","b","d"),
+ c("c","e"),
+ c("a","b","d","e"),
+ c("a","b','e','c')
+ )
> data <- as(data, "transactions")
> inspect(data)
  items
[1] {a, b, c}
[2] {a, b}
[3] {a, b, d}
[4] {b, e}
[5] {b, c, e}
[6] {a, d, e}
[7] {a, c}
[8] {a, b, d}
[9] {c, e}
[10] {a, b, d, e}
[11] {a, b, c, e}
> tl <- as(data, "tidLists")
> inspect(tl)
  items transactionIDs
1 a      {1,2,3,6,7,8,10,11}
2 b      {1,2,3,4,5,8,10,11}
3 c      {1,5,7,9,11}
4 d      {3,6,8,10}
5 e      {4,5,6,9,10,11}
> |
```

summary(Groceries)

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

[8] {a, b, d}
[9] {c, e}
[10] {a, b, d, e}
[11] {a, b, c, e}
> tl <- as(data, "tidLists")
> inspect(tl)
  items transactionIDs
1 a      {1,2,3,6,7,8,10,11}
2 b      {1,2,3,4,5,8,10,11}
3 c      {1,5,7,9,11}
4 d      {3,6,8,10}
5 e      {4,5,6,9,10,11}
> summary(Groceries)
transactions as itemMatrix in sparse format with
9835 rows (elements/itemsets/transactions) and
169 columns (items) and a density of 0.02609146

most frequent items:
  whole milk other vegetables      rolls/buns        soda      yogurt      (Other)
      2513       1903          1809        1715       1372       34055

element (itemset/transaction) length distribution:
sizes
   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  31  32
2159 1643 1299 1005  858  645  545  438  350  246  182  117  78  77  55  46  29  14  14  9  11  4  6  1  1  1  3  1

Min. 1st Qu. Median  Mean 3rd Qu.  Max.
1.000  2.000  3.000  4.409  6.000 32.000

includes extended item information - examples:
  labels level2      level1
1 frankfurter sausage meat and sausage
2 sausage sausage meat and sausage
3 liver loaf sausage meat and sausage
> |
```

rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.80))

```
RGUI (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

element (itemset/transaction) length distribution:
sizes
 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  31  32
2159 1643 1299 1005 855 645 545 438 350 246 182 117 78 77 55 46 29 14 14 9 11 4 6 1 1 1 3 1

Min. 1st Qu. Median Mean 3rd Qu. Max.
1.000 2.000 3.000 4.409 6.000 32.000

includes extended item information - examples:
  labels level2 level1
1 frankfurter sausage meat and sausage
2   sausage sausage meat and sausage
3 liver loaf sausage meat and sausage
> rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.80))
Apriori

Parameter specification:
confidence minval maxval arem aval originalSupport maxtime support minlen maxlen target ext
      0.8       0.1     1 none FALSE           TRUE      5  0.001     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: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.03s].
writing ... [410 rule(s)] done [0.00s].
creating 54 object ... done [0.00s].
> |
```

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

```

NGUI (64-bit) - [R Console]
File Edit View Msc Packages Windows Help

2 sausage sausage meat and sausage
3 liver loaf sausage meat and sausage
> rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.80))
Apriori

Parameter specification:
confidence minval smart arcm aval originalSupport maxtime support minlen maxlen target ext
0.8 0.1 1 none FALSE TRUE 5 0.001 1 10 rules TRUE

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

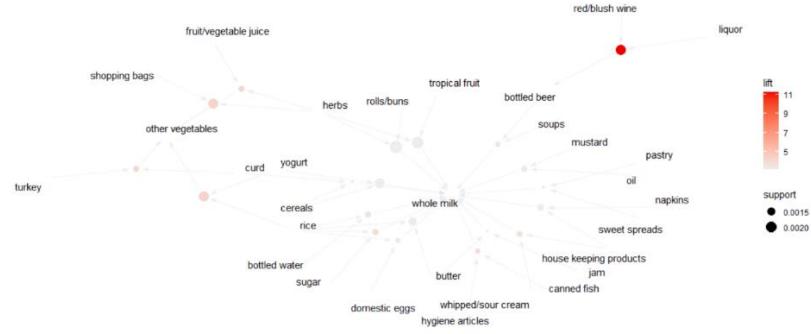
Absolute minimum support count: 9

set item appearances ... [0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and reencoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.03s].
writing ... [410 rule(s)] done [0.00s].
creating 54 object ... done [0.00s].
> inspect(rules[1:10])
#> #> #> #> #>
#>          lhs          rhs      support confidence coverage lift      count
[1] (liquor, red/blend wine) => (bottled beer) 0.001931876 0.81267419 11.23513523 11
[2] (curd, cereals)        => (whole milk) 0.001016777 0.8090509 0.001118454 3.557863 10
[3] (yogurt, cereals)     => (whole milk) 0.001728521 0.8095238 0.002135231 3.168192 17
[4] (butter, jam)         => (whole milk) 0.001016777 0.8333333 0.001220132 3.261374 12
[5] (soups, bottled beer) => (whole milk) 0.001118454 0.9166667 0.001220132 3.587512 11
[6] (napkins, house keeping products) => (whole milk) 0.001321810 0.8125000 0.0016266843 3.179940 13
[7] (whipped/sour cream, house keeping products) => (whole milk) 0.0012020132 0.9230769 0.001321810 3.612599 12
[8] (pastry, sweet spreads)    => (whole milk) 0.001016777 0.8090509 0.001118454 3.557863 10
[9] (turkey, curd)           => (other vegetables) 0.0012020132 0.8000000 0.001525165 4.134524 12
[10] (rice, sugar)          => (whole milk) 0.001220132 1.0000000 0.001220132 3.913649 12
> 
```

```
plot(rules[1:20], method = "graph", control = list(type = "items"))
```

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
[|] [X] [?] [?] [?] [?] [?]

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.03s].
writing ... [410 rule(s)] done [0.00s].
creating 34 object ... done [0.00s].
> inspect(rules[1:10])
#>
#>   lhs          rhs          support    confidence coverage lift count
[1] (liquor, red/bleush wine) => (bottled beer) 0.001931876 0.9047619 0.002135231 11.235269 19
[2] (curd, cereals)      => (whole milk)   0.001016777 0.9050909 0.001118454 3.557863 10
[3] (yogurt, cereals)   => (whole milk)   0.001728521 0.8095238 0.002135231 3.168192 17
[4] (butter, jam)       => (whole milk)   0.001016777 0.8333333 0.001220132 3.261374 10
[5] (soups, bottled beer) => (whole milk)   0.001118454 0.9166667 0.001220132 3.587512 11
[6] (napkins, house keeping products) => (whole milk) 0.001321810 0.8125000 0.001426843 3.179840 13
[7] (whipped/sour cream, house keeping products) => (whole milk) 0.001220132 0.9230769 0.001321810 3.612559 12
[8] (pastry, sweet spreads)     => (whole milk)   0.001016777 0.9090909 0.001118454 3.557863 10
[9] (turkey, curd)           => (other vegetables) 0.001220132 0.8000000 0.001825165 4.134524 12
[10] (rice, sugar)          => (whole milk)   0.001220132 1.0000000 0.001220132 3.913649 12
> plot(rules[1:10],
+ method = "graph",
+ control = list(type = "items"))
Warning: Unknown control parameters: type
Available control parameters (with default values):
layout    = stress
circular   = FALSE
graphdots  = NULL
edges      = <environment>
nodes      = <environment>
nodeText   = <environment>
colors     = c("#FF0000FF", "#FFFFFFFF")
engine     = ggplot2
max        = 100
verbose    = FALSE
> |
```



## RESULT:

Thus, the Apriori Algorithm is successfully executed in R language.

## 5. APRIORI IN WEKA

### AIM

To implement apriori algorithm in weka.

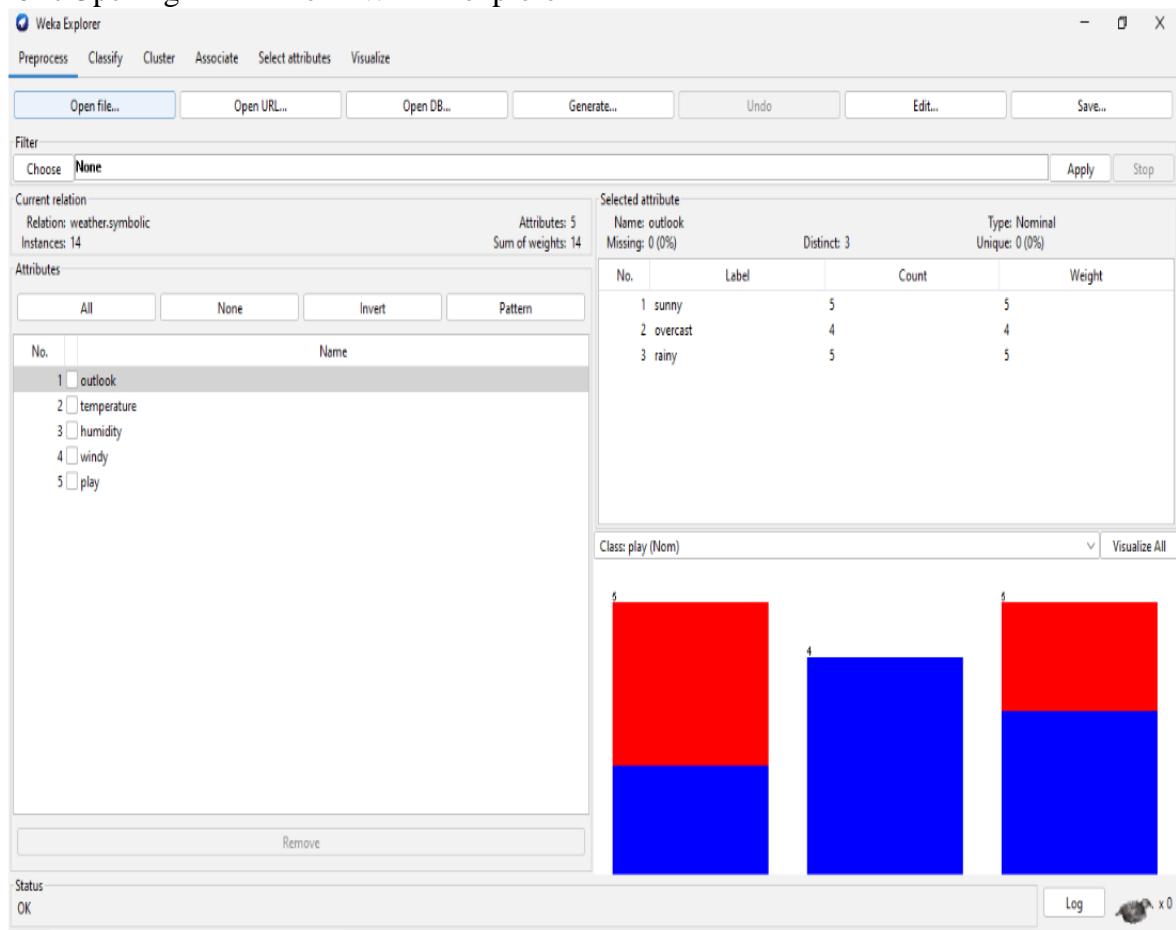
### PROCEDURE

**Step 1:** The first part of analysis is to bring in the dataset. We will be using a dataset weather.numeric.arff to simplify our analysis.

**Step 2:** Opening ARFF file in WEKA explorer.

**Step 3:** Apply the apriori algorithm

**Figure 1.** Opening ARFF file in WEKA explorer



**Figure 2** Apply the apriori algorithm

The screenshot shows the Weka Explorer interface with the 'Associate' tab selected. In the 'Choose' dropdown, 'Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S 1.0 -c 1' is selected. The 'Start' and 'Stop' buttons are visible. The 'Result list (right-click for ...)' pane shows '21:37:52 - Apriori'. The 'Associate output' pane displays the following log:

```

Apriori
=====
Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4   <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3   <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3   <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3   <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3   <conf:(1)> lift:(1.56) lev:(0.09) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3   <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2   <conf:(1)> lift:(2) lev:(0.07) [1] conv:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2   <conf:(1)> lift:(2.8) lev:(0.09) [1] conv:(1.29)

```

The status bar at the bottom shows 'OK'.

**Figure 3** Associator output

The screenshot shows the Weka Explorer interface with the 'Associate' tab selected. In the 'Choose' dropdown, 'FilteredAssociator -F "weka.filters.MultiFilter -F \weka.filters.unsupervised.attribute.ReplaceMissingValues \ -S 1" -c -1 -W weka.associations.Apriori -- -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S 1.0 -c 1' is selected. The 'Start' and 'Stop' buttons are visible. The 'Result list (right-click for op...)' pane shows '21:37:52 - Apriori', '21:40:31 - Apriori', and '21:48:26 - FilteredAssociator'. The 'Associate output' pane displays the following run information:

```

--- Run information ---
Scheme:      weka.associations.FilteredAssociator -F "weka.filters.MultiFilter -F \weka.filters.unsupervised.attribute.ReplaceMissingValues \ -S 1" -c -1 -W weka.associations.Apriori -- -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S 1.0 -c 1
Relation:     weather
Instances:   14
Attributes:  5
outlook
temperature
humidity
windy
play

```

The status bar at the bottom shows 'See error log'.

## RESULT:

Thus, the Apriori Algorithm is successfully executed in Weka.

## 6. APRIORI IN PYTHON

### AIM:

To implement apriori algorithm in python.

### PROCEDURE:

Step 1: Importing Libraries

Step 2: Importing the Dataset

Step 3: Data Preprocessing

Step 4: Using Apriori algorithm.

### OUTPUT:

```
▷   import numpy as np
    import plotly.express as px
    import pandas as pd
    from apyori import apriori
[1]

    data = pd.read_csv('D:\\DM LAB_APRIORI\\Groceries_dataset.csv')
    print("Data Dimension:", data.shape)
    data.head()
[2]

... Data Dimension: (38765, 3)

</>   Member_number      Date itemDescription
0        1808  21-07-2015    tropical fruit
1        2552  05-01-2015     whole milk
2        2300  19-09-2015      pip fruit
3        1187  12-12-2015  other vegetables
4        3037  01-02-2015     whole milk

▷   data.isnull().any()
[3]

... Member_number      False
Date          False
itemDescription  False
dtype: bool

    print("Total number of unique products are:", len(data['itemDescription'].unique()))
[4]

... Total number of unique products are: 167

    print("Top 10 frequently sold products(Tabular Representation)")
    x = data['itemDescription'].value_counts().sort_values(ascending=False)[:10]
    x
[5]

... Top 10 frequently sold products(Tabular Representation)

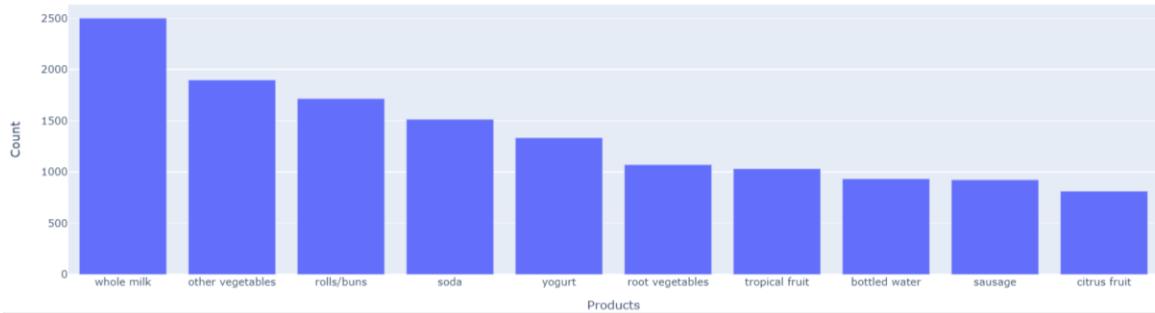
whole milk      2502
other vegetables 1898
rolls/buns      1716
soda             1514
yogurt           1334
root vegetables  1071
tropical fruit    1032
bottled water     933
sausage            924
citrus fruit       812
Name: itemDescription, dtype: int64
```

```

fig = px.bar(x= x.index, y= x.values)
fig.update_layout(title_text="Top 10 frequently sold products (Graphical Representation)", xaxis_title= "Products", yaxis_title="Count")
fig.show()

```

[6] Top 10 frequently sold products (Graphical Representation)



```

data["Year"] = data['Date'].str.split("-").str[-1]
data["Month-Year"] = data['Date'].str.split("-").str[1] + "-" + data['Date'].str.split("-").str[-1]
data.head()

```

[7]

	Member_number	Date	itemDescription	Year	Month-Year
0	1808	21-07-2015	tropical fruit	2015	07-2015
1	2552	05-01-2015	whole milk	2015	01-2015
2	2300	19-09-2015	pip fruit	2015	09-2015
3	1187	12-12-2015	other vegetables	2015	12-2015
4	3037	01-02-2015	whole milk	2015	02-2015

```

fig1 = px.bar(data["Month-Year"].value_counts(ascending=False),
               orientation="v",
               color = data["Month-Year"].value_counts(ascending=False),
               labels={'value':'Count', 'index':'Date','color':'Meter'})

```

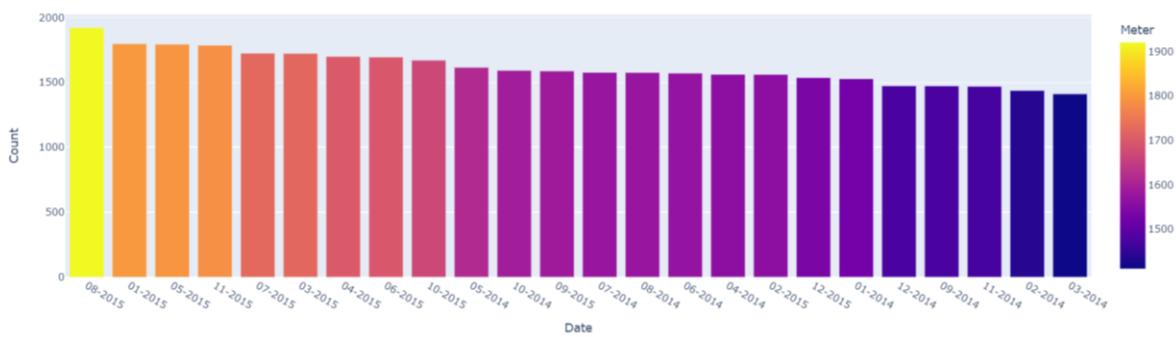
```
fig1.update_layout(title_text="Exploring higher sales by the date")
```

```
fig1.show()
```

Python

[8]

Exploring higher sales by the date





```

x = data1.values
x = [sub[(sub==0)].tolist() for sub in x if sub[sub != 0].tolist()]
transactions = x
transactions[0:10]

[14] Python

...[['whole milk', 'yogurt', 'sausage', 'semi-finished bread'],
['whole milk', 'pastry', 'salty snack'],
['canned beer', 'misc. beverages'],
['sausage', 'hygiene articles'],
['soda', 'pickled vegetables'],
['frankfurter', 'curd'],
['whole milk', 'rolls/buns', 'sausage'],
['whole milk', 'soda'],
['beef', 'white bread'],
['frankfurter', 'soda', 'whipped/sour cream']]

rules = apriori(transactions, min_support = 0.00030, min_confidence = 0.05, min_lift = 3, max_length = 2, target = "rules")
association_results = list(rules)
print(association_results[0])

[15] Python
...RelationRecord(items=frozenset({'liver loaf', 'fruit/vegetable juice'}), support=0.00040098910646260775, ordered_statistics=[OrderedStatistic(items_base=frozenset({'liver loaf'})), items_add=frozenset({'fruit/vegetable juice'}), confidence=0.12, lift=3.5276227897838903])

for item in association_results:

    pair = item[0]
    items = [x for x in pair]

    print("Rule : ", items[0], " -> " + items[1])
    print("Support : ", str(item[1]))
    print("Confidence : ", str(item[2][0][2]))
    print("Lift : ", str(item[2][0][3]))

    print("=====")

[16] Python
Rule : liver loaf -> fruit/vegetable juice
Support : 0.00040098910646260775
Confidence : 0.12
Lift : 3.5276227897838903
=====
Rule : pickled vegetables -> ham
Support : 0.0005346521419501437
Confidence : 0.05970149253731344
Lift : 3.4895055970149254
=====
Rule : roll products -> meat
Support : 0.0003341575887188398
Confidence : 0.06097560975609757
Lift : 3.620547812620984
=====
Rule : misc. beverages -> salt
Support : 0.0003341575887188398
Confidence : 0.05617977528089888
Lift : 3.5619405827461437
=====
Rule : misc. beverages -> spread cheese
Support : 0.0003341575887188398
Confidence : 0.05
Lift : 3.170127118644068
=====
Rule : soups -> seasonal products
Support : 0.0003341575887188398
Confidence : 0.10416666666666667
Lift : 14.704205974842768
=====
Rule : sugar -> spread cheese
Support : 0.00040098910646260775
Confidence : 0.06
Lift : 3.3878490566037733
=====
```

## RESULT:

Thus, Apriori algorithm is successfully executed using Python.

## 7. BAYES IN WEKA

### AIM:

To implement Bayes classifier in Weka.

### PROCEDURE:

The following is how to apply simple classifier and visualization with Weka to a Naive Bayes classifier.

#### Loading the Diabetes Data

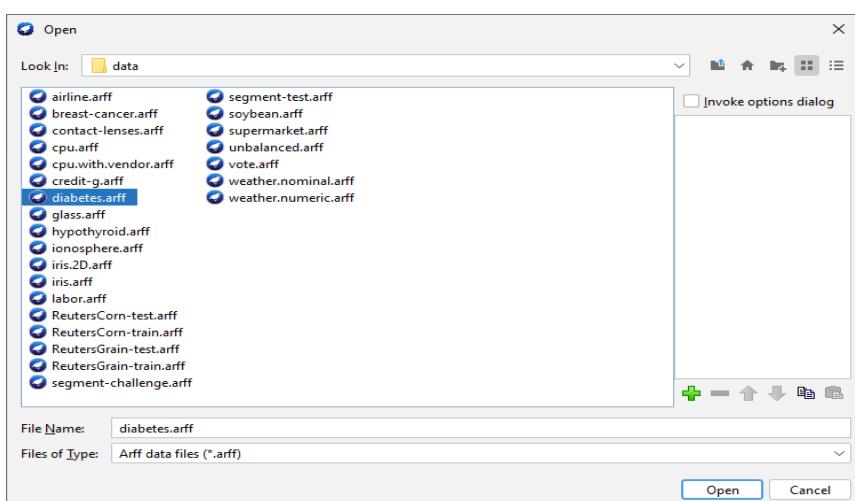
**Step 1:** Start Weka

**Step 2:** Press the Explorer Button:



**Step 3:** Download the diabetes data file, e.g. from Weka folder.

**Step 4:** In the “Preprocess” tab of the Weka Explorer window, click the “Open file...” button, and select the diabetes data file from your download location. The window should then look like this:

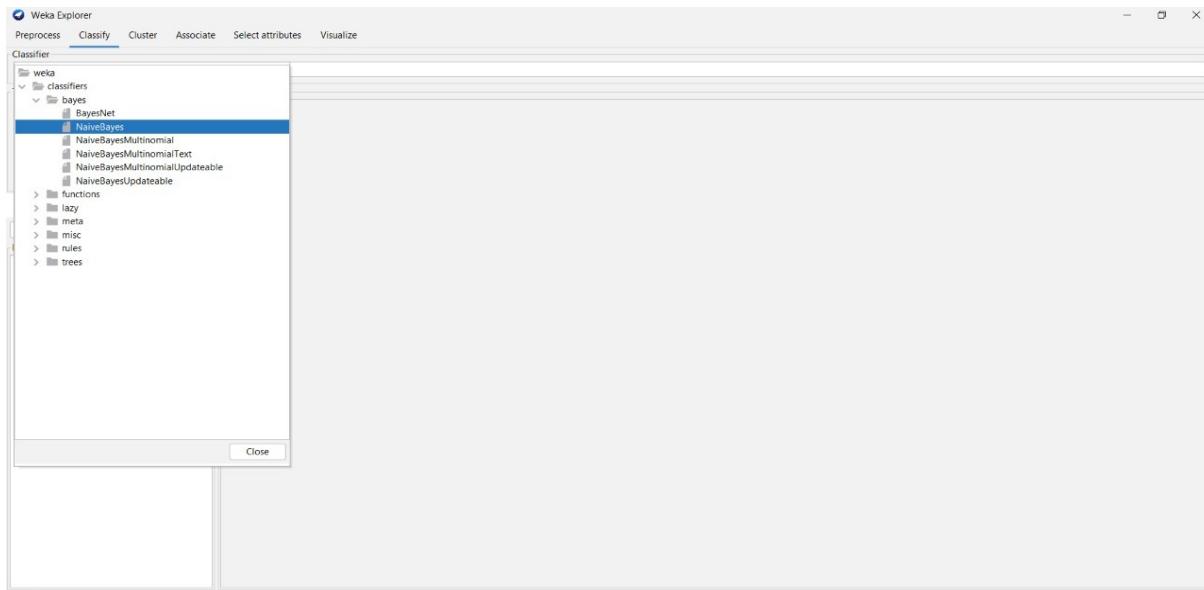


## **Naive Bayes classifier:**

We will experiment here to see how well the **Naive Bayes classifier** algorithm classifies the data according to the original class labels.

**Step 1:** Click the “Classify” tab at the top of the Weka Explorer.

**Step 2:** Click the Classify “Choose” button and select “Naive Bayes”.



**Step 3:** Click “Cross-validation” in “Test options” and set “folds as 10”.

**Step 4:** Press Start to begin Naive Bayes classifier and evaluation.

```

Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier Choose NaiveBayes
Test options
 Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66
More options...
(Nom) class Start Stop
Result list (right-click for options)
12:50:02 - bayes.NaiveBayes

Classifier output
precision 0.0045 0.0045

age
mean 31.2494 37.0808
std. dev. 11.6059 10.5146
weight sum 500 268
precision 1.1765 1.1765

Time taken to build model: 0.02 seconds
*** Stratified cross-validation ***
*** Summary ***

Correctly Classified Instances 586 76.3021 %
Incorrectly Classified Instances 182 23.6979 %
Kappa statistic 0.4664
Mean absolute error 0.2841
Root mean squared error 0.4168
Relative absolute error 62.5020 %
Root relative squared error 87.4349 %
Total Number of Instances 768

*** Detailed Accuracy By Class ***

      TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.844 0.388 0.802 0.844 0.823 0.468 0.819 0.892 tested_negative
0.612 0.156 0.678 0.612 0.643 0.468 0.819 0.671 tested_positive
Weighted Avg. 0.763 0.307 0.759 0.763 0.760 0.468 0.819 0.815

*** Confusion Matrix ***

a b <-- classified as
422 79 | a = tested_negative
104 164 | b = tested_positive

```

```

Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier Choose NaiveBayes
Test options
 Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66
More options...
(Nom) class Start Stop
Result list (right-click for options)
12:50:02 - bayes.NaiveBayes

Classifier output
*** Run information ***
Scheme: weka.classifiers.bayes.NaiveBayes
Relation: pima_diabetes
Instances: 768
Attributes: 9
preg
plas
pres
skin
insu
mass
pedi
age
class
Test mode: 10-fold cross-validation

*** Classifier model (full training set) ***

Naive Bayes Classifier

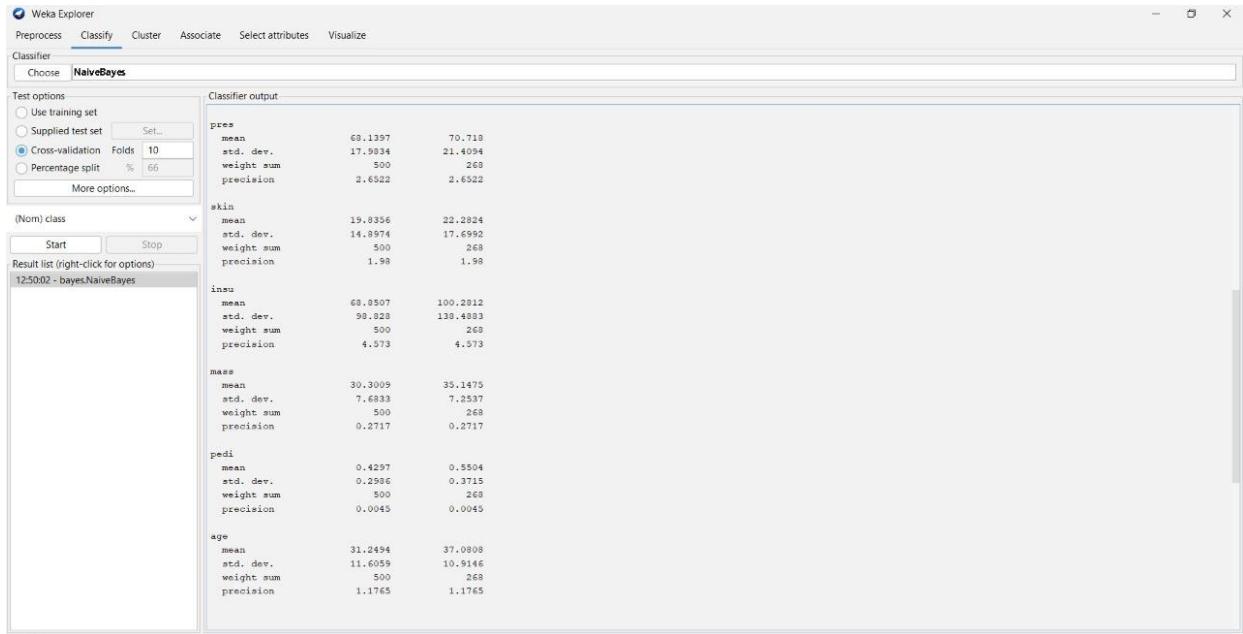
      Class
Attribute tested_negative tested_positive
(preg (0.65) (0.35)

preg
mean 3.4234 4.9795
std. dev. 3.0166 3.6027
weight sum 500 268
precision 1.0625 1.0625

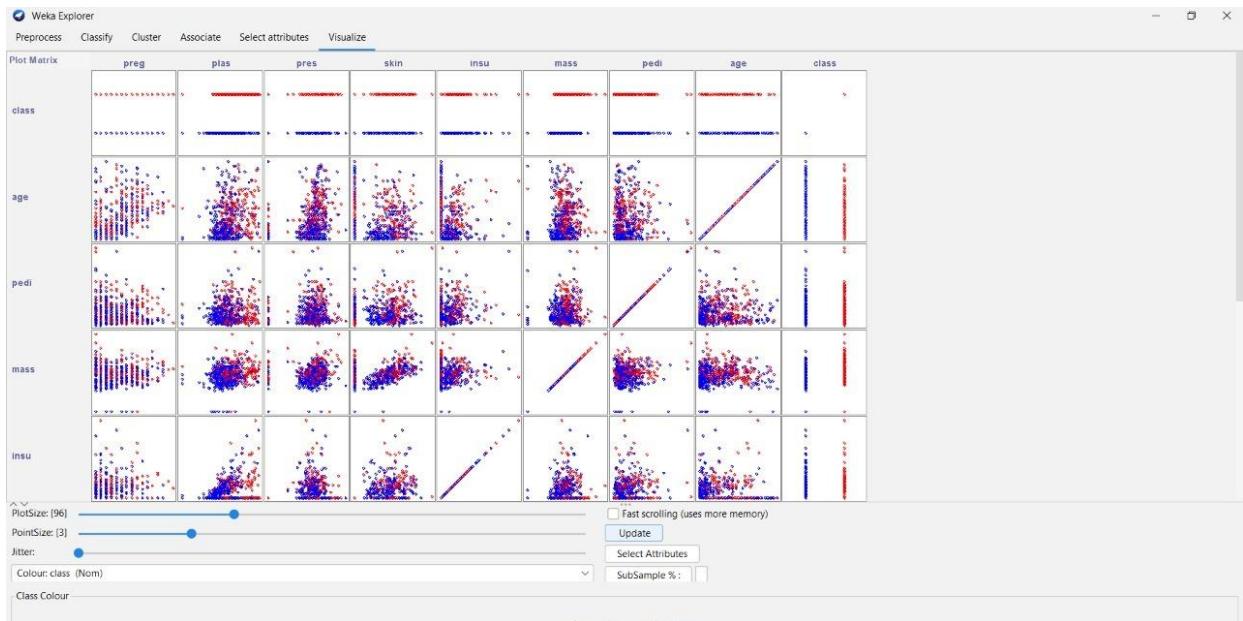
plas
mean 109.9041 141.2581
std. dev. 26.1114 31.8728
weight sum 500 268
precision 1.4741 1.4741

pres

```



**Step 5:** Click on the “Visualize” tab.



## RESULT:

Thus, Bayes classifier is successfully executed using Weka.

## 8. BAYES CLASSIFIER USING R

### AIM:

To implement Bayes classifier using R.

### PROCEDURE:

- **Step 1:** The first part of any analysis is to bring in the dataset. We will be using an inbuilt dataset “Iris”, caTools, caret and e1071 packages are loaded for our analysis.
- **Step 2:** The iris data are now trained and tested by using the below commands.

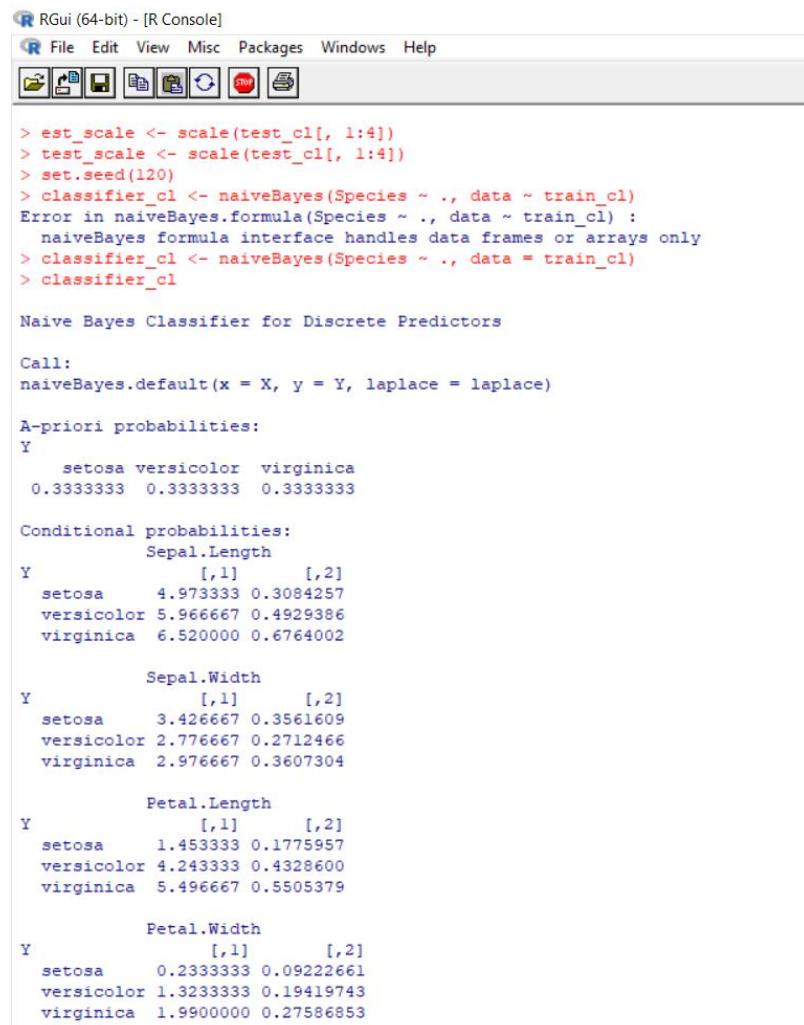
```
> data(iris)
> str(iris)
>library(caTools)
> library(caret)
> library(e1071)
> split <- sample.split(iris, SplitRatio=0.7)
> train_cl <- subset(iris, split=="TRUE")
> test_cl <- subset(iris, split=="FALSE")
> train_scale <- scale(train_cl[, 1:4])
> est_scale <- scale(test_cl[, 1:4])
> test_scale <- scale(test_cl[, 1:4])
> set.seed(120)

> utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this session ---
Error in contrib.url(repos, type) :
  trying to use CRAN without setting a mirror
> data(iris)
> str(iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
> library(caTools)
> library(caret)
Loading required package: ggplot2
Loading required package: lattice
> library(e1071)
>
>
> library(caTools)
> split <- sample.split(iris, SplitRatio=0.7)
> train_cl <- subset(iris, split=="TRUE")
> test_cl <- subset(iris, split=="FALSE")
> train_scale <- scale(train_cl[, 1:4])
> est_scale <- scale(test_cl[, 1:4])
> test_scale <- scale(test_cl[, 1:4])
> set.seed(120)
```

**Step 3:** Now Bayes classifier is implemented.

```
> classifier_cl <- naiveBayes(Species ~ ., data = train_cl)

> classifier_cl
```



The screenshot shows the RGui interface with the title bar "RGui (64-bit) - [R Console]". The menu bar includes File, Edit, View, Misc, Packages, Windows, and Help. Below the menu is a toolbar with various icons. The main console area displays the R code and its output. The output shows the creation of a Naive Bayes Classifier for Discrete Predictors, detailing A-priori probabilities and Conditional probabilities for Sepal.Length, Sepal.Width, Petal.Length, and Petal.Width across three species: setosa, versicolor, and virginica.

```
> est_scale <- scale(test_cl[, 1:4])
> test_scale <- scale(test_cl[, 1:4])
> set.seed(120)
> classifier_cl <- naiveBayes(Species ~ ., data ~ train_cl)
Error in naiveBayes.formula(Species ~ ., data ~ train_cl) :
  naiveBayes formula interface handles data frames or arrays only
> classifier_cl <- naiveBayes(Species ~ ., data = train_cl)
> classifier_cl

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
  setosa versicolor virginica
0.3333333 0.3333333 0.3333333

Conditional probabilities:
  Sepal.Length
Y      [,1]      [,2]
setosa 4.973333 0.3084257
versicolor 5.966667 0.4929386
virginica 6.520000 0.6764002

  Sepal.Width
Y      [,1]      [,2]
setosa 3.426667 0.3561609
versicolor 2.776667 0.2712466
virginica 2.976667 0.3607304

  Petal.Length
Y      [,1]      [,2]
setosa 1.453333 0.1775957
versicolor 4.243333 0.4328600
virginica 5.496667 0.5505379

  Petal.Width
Y      [,1]      [,2]
setosa 0.2333333 0.09222661
versicolor 1.3233333 0.19419743
virginica 1.9900000 0.27586853
```

## RESULT:

Thus, Bayes classifier is successfully executed in R.

## 9. BAYESIAN IN PYTHON

### AIM:

Implementing Naive Bayes algorithm using Python.

### PROCEDURE:

#### Naive Bayes algorithm:

**Step 1:** Data Pre-processing step

**Step 2:** Fitting Naive Bayes to the Training Set

**Step 3:** Prediction of the test set result

**Step 4:** Creating Confusion Matrix

### OUTPUT:

```
In [1]: import numpy as np

In [2]: def fit(X_train,Y_train):
    result = {}
    class_values = set(Y_train)
    for curr_value in class_values:
        result[curr_value] = {}
        result["total_data"] = len(Y_train)
        curr_class_rows = (Y_train == curr_value)
        X_train_curr = X_train[curr_class_rows]
        Y_train_curr = Y_train[curr_class_rows]
        num_features = X_train.shape[1]
        result[curr_value]["total_count"] = len(Y_train_curr)
        for j in range(1,num_features+1):
            result[curr_value][j] = {}
            all_possible_values = set(X_train[:,j-1])
            for this_value in all_possible_values:
                result[curr_value][j][this_value] = (X_train_curr[:,j-1]==this_value).sum()
    return result

In [3]: def probability(dictionary,x,current_class):
    output= np.log(dictionary[current_class][ "total_count"])-np.log(dictionary[ "total_data"])
    num_features = len(dictionary[current_class].keys())-1;
    for j in range(1,num_features+1):
        xj = x[j-1]
        count_current_class_with_value_xj = dictionary[current_class][j][xj] + 1
        count_current_class = dictionary[current_class][ "total_count"] + len(dictionary[current_class][j].keys())
        current_xj_prob = np.log(count_current_class_with_value_xj) -np.log(count_current_class)
        output = output + current_xj_prob
    return output
```

```
In [4]: def predictSinglePoint(dictionary,x):
    classes = dictionary.keys()
    best_p = -1000
    best_class = -1
    first_run = True
    for current_class in classes:
        if(current_class == "total_data"):
            continue
        p_curr_class = probability(dictionary,x,current_class)
        if(first_run or p_curr_class > best_p):
            best_p = p_curr_class
            best_class = current_class
        first_run = False
    return best_class
```

```
In [5]: def predict(dictionary,X_test):
    Y_pred = []
    for x in X_test:
        x_class = predictSinglePoint(dictionary,x)
        Y_pred.append(x_class)
    return Y_pred
```

```
In [6]: def makelabelled(column):
    second_limit = column.mean()
    first_limit = 0.5 * second_limit
    third_limit = 1.5 * second_limit
    for i in range(0,len(column)):
        if(column[i]<first_limit):
            column[i] = 0
        elif(column[i] < second_limit):
            column[i] = 1
        elif(column[i]<third_limit):
            column[i] = 2
        else:
            column[i] = 3
    return column
```

```
In [7]: from sklearn import datasets
iris = datasets.load_iris()
x = iris.data
y = iris.target
```

```
In [8]: for i in range(0,x.shape[-1]):
    x[:,i] = makelabelled(x[:,i])
```

```
In [9]: from sklearn import model_selection
X_train,X_test,Y_train,Y_test = model_selection.train_test_split(x,y,test_size=0.25,random_state=0)
```

```
In [10]: dictionary = fit(X_train,Y_train)
```

```
In [10]: dictionary = fit(X_train,Y_train)

In [11]: Y_pred = predict(dictionary,X_test)

In [12]: from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(Y_test,Y_pred))
print(confusion_matrix(Y_test,Y_pred))

precision    recall   f1-score   support

          0       1.00      1.00      1.00      13
          1       0.94      1.00      0.97      16
          2       1.00      0.89      0.94       9

   accuracy                           0.97      38
  macro avg       0.98      0.96      0.97      38
weighted avg       0.98      0.97      0.97      38

[[13  0  0]
 [ 0 16  0]
 [ 0  1  8]]
```

## RESULT:

Thus, Bayes classifier is successfully executed using Python.

## **INTERNAL – 1**

### **10.Demonstration of Association rule process on dataset contactlenses using apriori algorithm in WEKA**

#### **AIM:**

To find Association rule process on dataset contactlenses using apriori algorithm in Weka

#### **PROCEDURE:**

Step1: Open the data file in Weka Explorer. It is presumed that the required data fields have been discretized. In this example it is age attribute

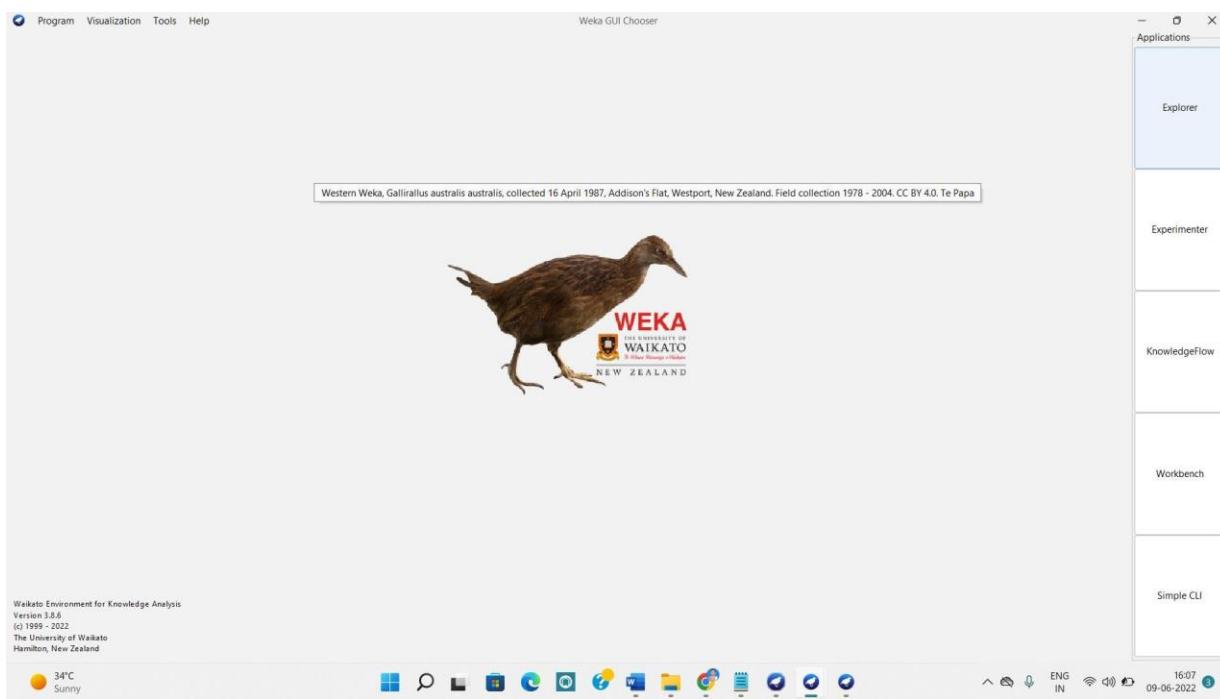
Step2: Clicking on the associate tab will bring up the interface for association rule algorithm

Step3: We will use apriori algorithm. This is the default algorithm

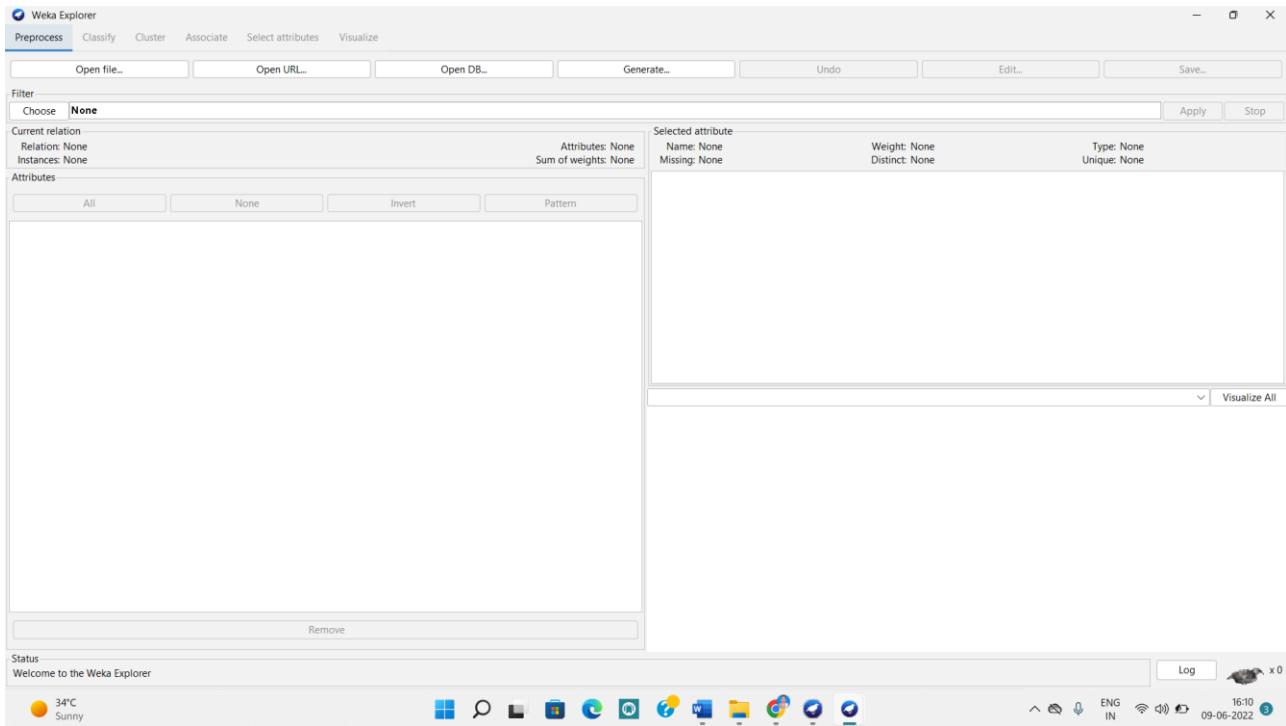
Step4: In order to change the parameters for the run (example support, confidence etc) we click on the text box immediately to the right of the choose button

#### **OUTPUT SCREENSHOT:**

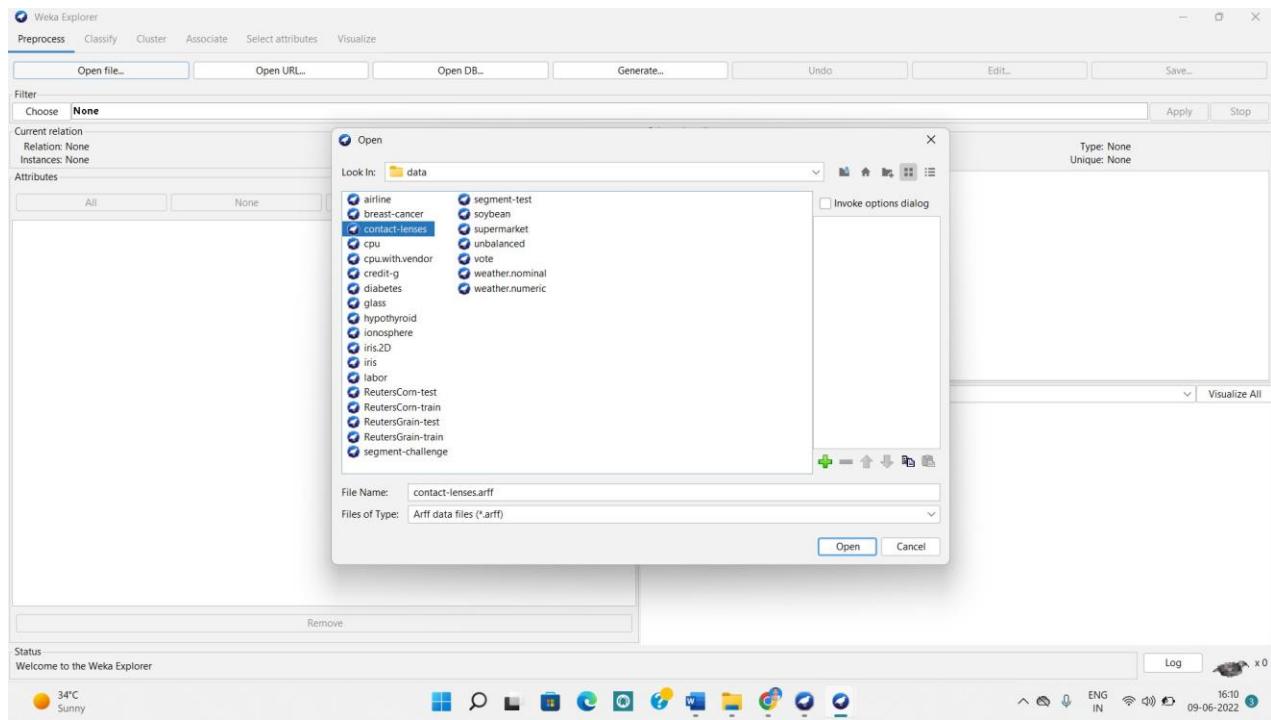
Open Weka → Click Explore



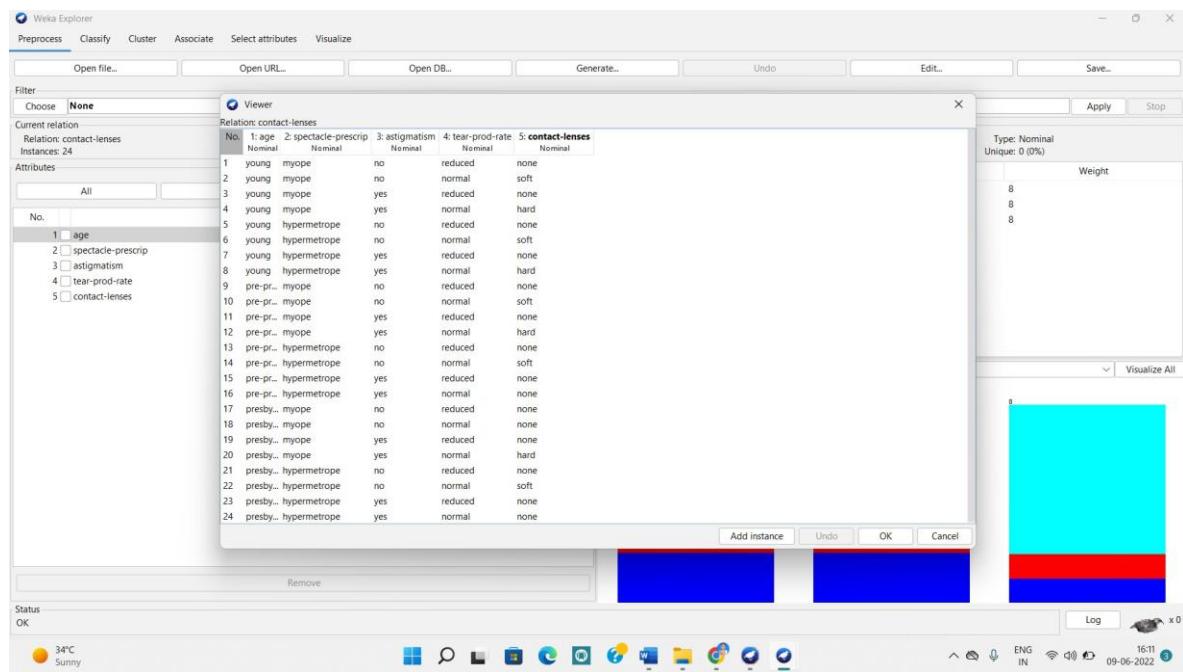
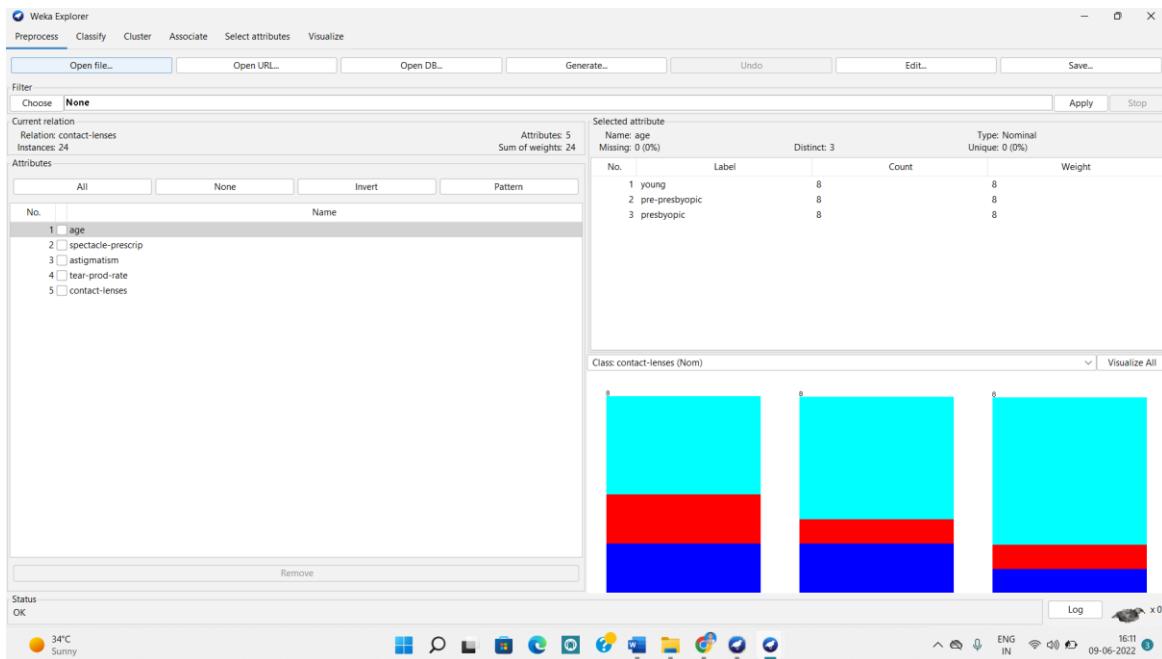
In preprocess → Click open file



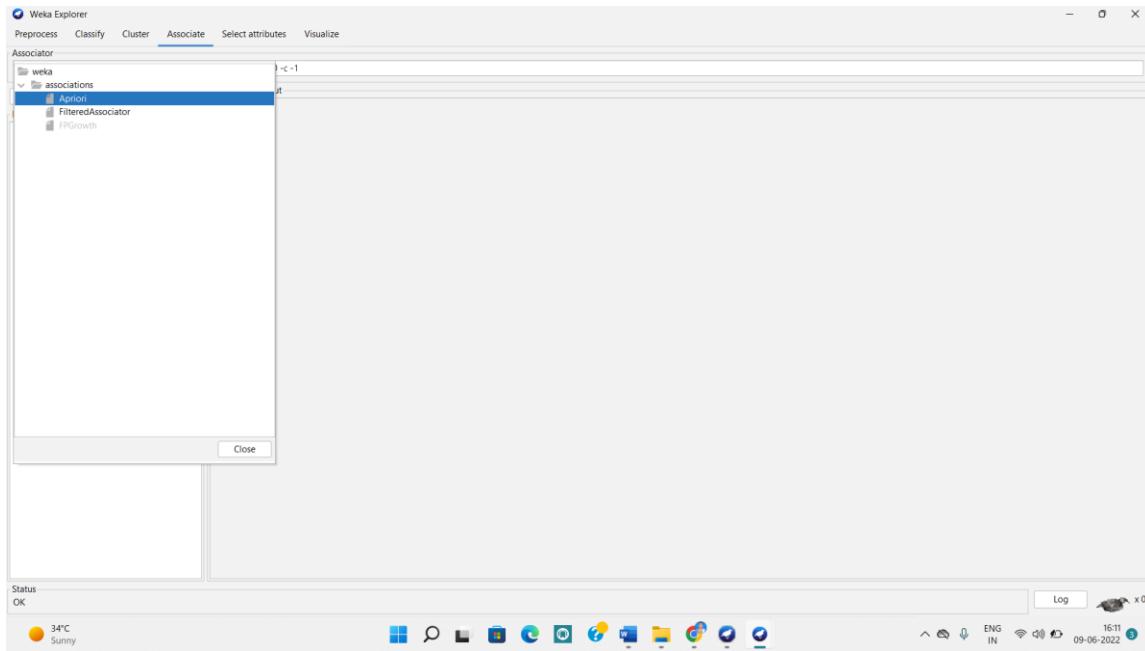
Select weka → data → (Dataset)contactlenses → click open



It will show the pre-processed data of contact lens



## Select Associate → Select Apriori



It will show minimum support, minimum matrix, no. of cycles in this figure

```
Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Associate
Choose: Apriori -N 10 -T 0 -C 0.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start Stop
Result list (right-click for ...)
161208 - Apriori
161252 - Apriori
Associate output
==== Run information ====
Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation: contact-lenses
Instances: 24
Attributes: 5
age
spectacle-prescrip
astigmatism
tear-prod-rate
contact-lenses
==== Associator model (full training set) ====
Apriori
=====
Minimum support: 0.2 (5 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16
Generated sets of large itemsets:
Size of set of large itemsets L(1): 11
Size of set of large itemsets L(2): 21
Size of set of large itemsets L(3): 6
Best rules found:
1. tear-prod-rate=reduced 12 ==> contact-lenses=none 12 <conf:(1)> lift:(1.6) lev:(0.19) [4] conv:(4.5)
2. spectacle-prescrip=normal tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
3. spectacle-prescrip=hypermetropic tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
4. astigmatism=none tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
5. astigmatism=type tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
6. contact-lenses=soft 5 ==> astigmatism=none 5 <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
=====
Status OK
Log x0
34°C Sunny
17:17 09-06-2022
```

## RESULT:

Thus, the Association rule process on dataset contactlenses using apriori algorithm in WEKA is shown.

## **11. Create a dataset in .arff file format. Demonstration of preprocessing using weka dataset**

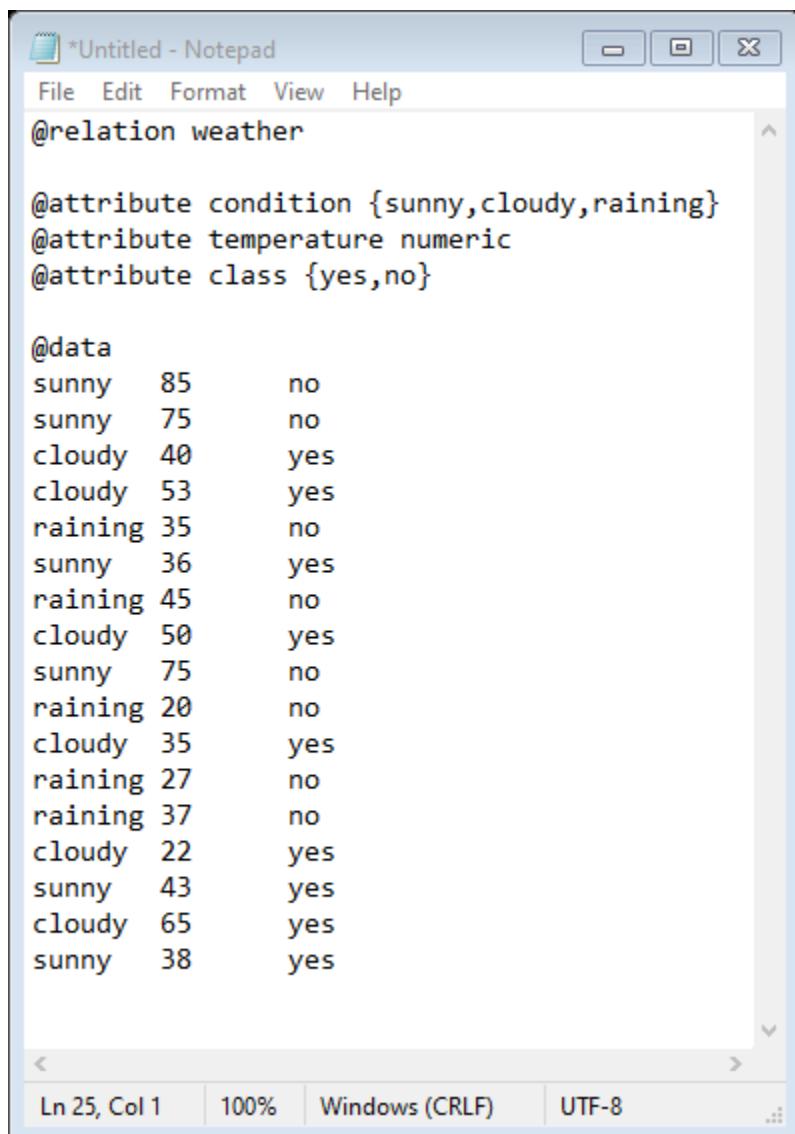
### **AIM:**

To create a dataset in .arff file format and demonstrate preprocessing using WEKA dataset.

### **PROCEDURE:**

**STEP-1:** Open NOTEPAD.

**STEP-2:** Type your dataset.



The screenshot shows a Microsoft Notepad window titled "\*Untitled - Notepad". The window contains an ARFF (Attribute-Relationship File Format) dataset for weather forecasting. The code is as follows:

```
@relation weather

@attribute condition {sunny,cloudy,raining}
@attribute temperature numeric
@attribute class {yes,no}

@data
sunny 85 no
sunny 75 no
cloudy 40 yes
cloudy 53 yes
rainning 35 no
rainning 36 yes
rainning 45 no
cloudy 50 yes
sunny 75 no
rainning 20 no
cloudy 35 yes
rainning 27 no
rainning 37 no
cloudy 22 yes
sunny 43 yes
cloudy 65 yes
sunny 38 yes
```

The Notepad window includes standard menu options: File, Edit, Format, View, Help. At the bottom, it shows "Ln 25, Col 1" and zoom levels "100%", "Windows (CRLF)", and "UTF-8".

Here,

Firstly, we need to say type of relation/name as follows

“@relation weather”

Secondly, we have to define the attributes as follows,

“@attribute attribute\_name type/type of arguments it has”

In our dataset we are taking three attribute namely –

condition, temperature and class

In our dataset,

condition has three arguments – sunny, raining and cloudy.

Temperature is taking numbers so it is a numeric type

Class having – yes, no arguments in the dataset

Finally, we enter our dataset data by following “@data”

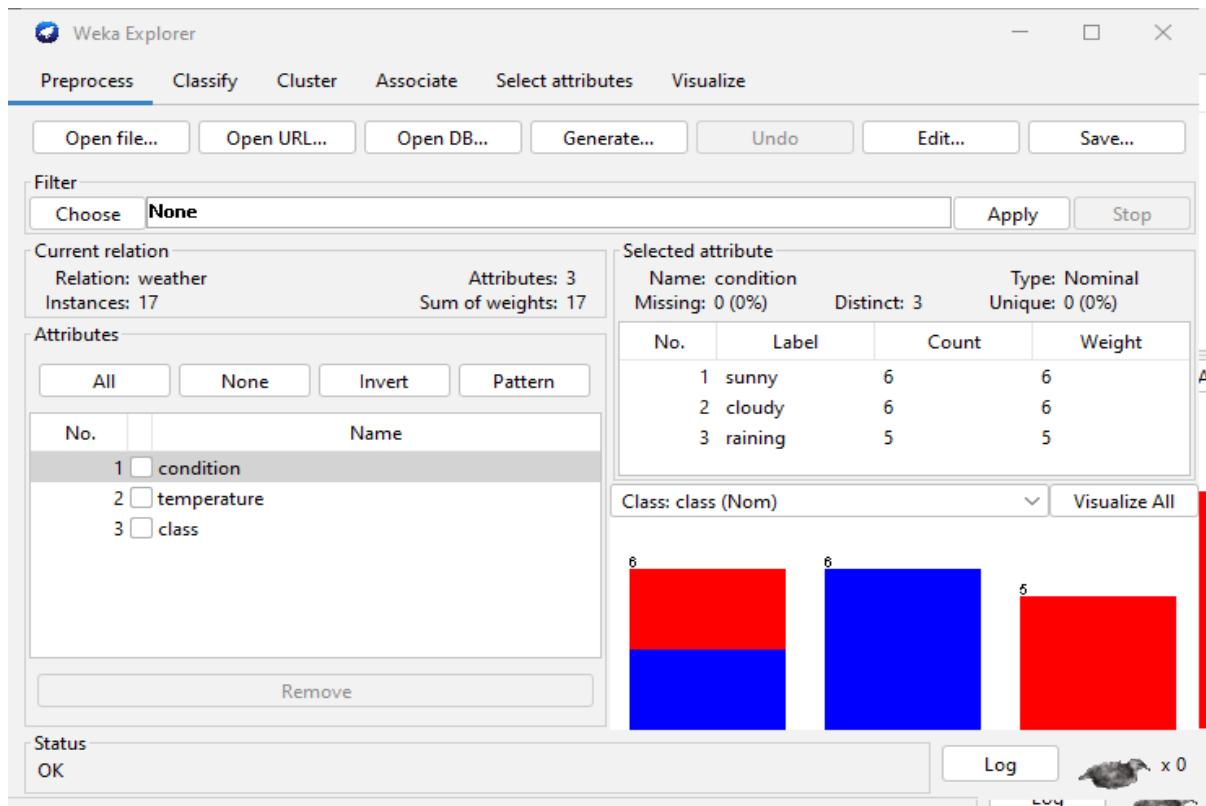
These are the requirements then only .arff file can be read by weka. Otherwise, it will throw an error.

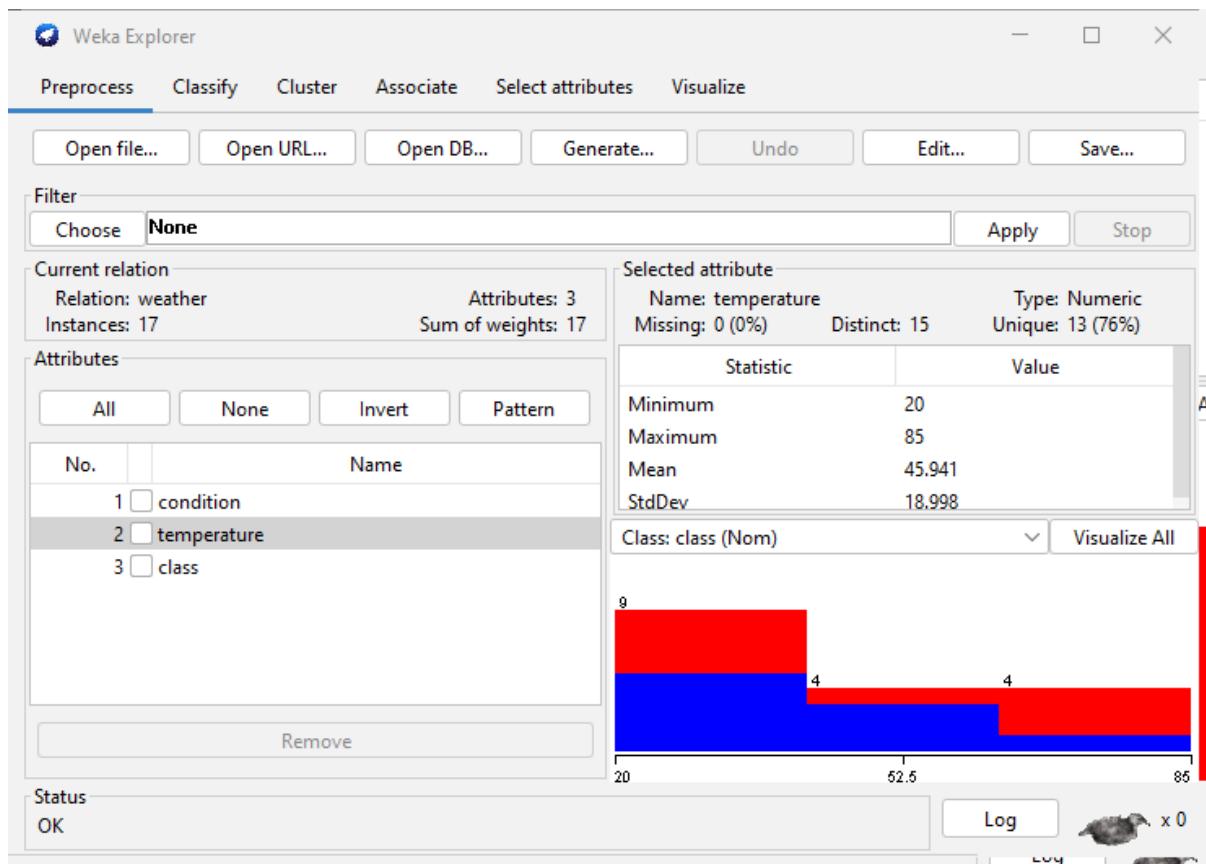
**STEP-3:** Finally we SAVE our dataset file as file\_name.arff i.e.; with .arff

file extention. In our case we saved as weather.arff

## Demonstration of preprocessing on dataset using weka

**STEP-1:** Open weather.arff file in weka tool in PREPROCESS.





**Weka Explorer**

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter Choose **None** Apply Stop

Current relation  
Relation: weather Attributes: 3  
Instances: 17 Sum of weights: 17

Selected attribute  
Name: class Type: Nominal  
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count	Weight
1	yes	9	9
2	no	8	8

Attributes  
All None Invert Pattern

No.	Name
1	<input type="checkbox"/> condition
2	<input type="checkbox"/> temperature
3	<input type="checkbox"/> class

Remove

Status OK Log

Class: class (Nom) Visualize All

9 8 0 0

Viewer

Relation: weather

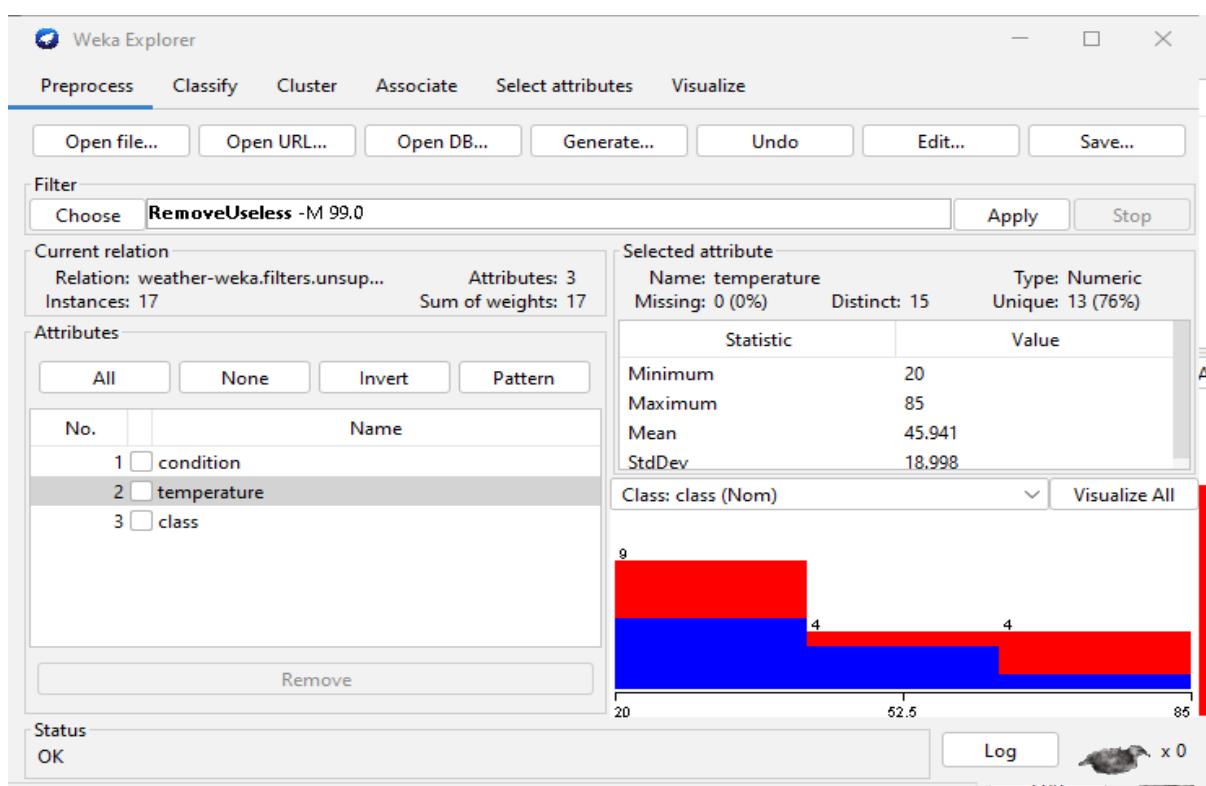
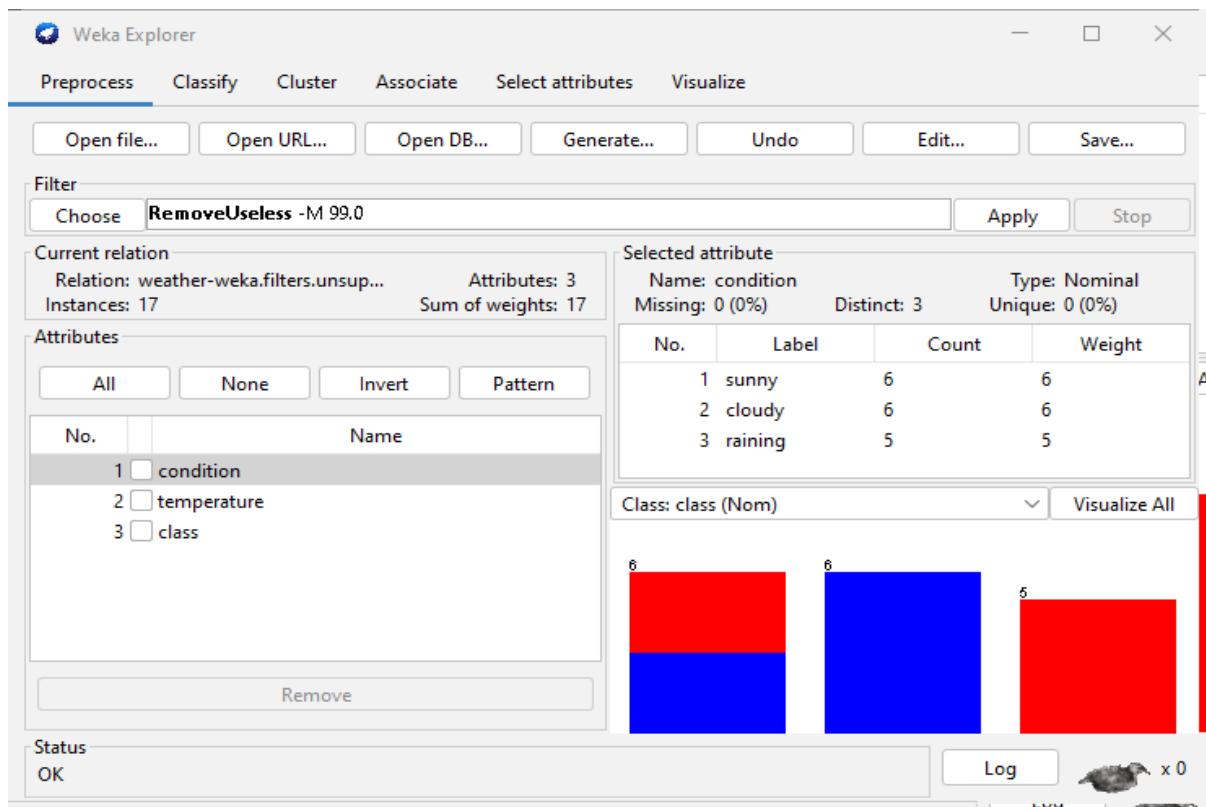
No.	1: condition Nominal	2: temperature Numeric	3: class Nominal
1	sunny	85.0	no
2	sunny	75.0	no
3	cloudy	40.0	yes
4	cloudy	53.0	yes
5	raining	35.0	no
6	sunny	36.0	yes
7	raining	45.0	no
8	cloudy	50.0	yes
9	sunny	75.0	no
10	raining	20.0	no
11	cloudy	35.0	yes
12	raining	27.0	no
13	raining	37.0	no
14	cloudy	22.0	yes
15	sunny	43.0	yes
16	cloudy	65.0	yes
17	sunny	38.0	yes

Add instance   Undo   OK

**STEP-2:** Apply any preprocessing methods by clicking on “CHOOSE” option.

Here we will apply any two preprocessing techniques.

By applying “RemoveUseless” in unsupervised attributes.



**Weka Explorer**

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter Choose RemoveUseless -M 99.0 Apply Stop

Current relation  
Relation: weather-weka.filters.unsup... Attributes: 3  
Instances: 17 Sum of weights: 17

Attributes

All None Invert Pattern

No.	Name
1	<input type="checkbox"/> condition
2	<input type="checkbox"/> temperature
3	<input type="checkbox"/> class

Remove

Status OK

Selected attribute  
Name: class Type: Nominal  
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count	Weight
1	yes	9	9
2	no	8	8

Class: class (Nom) Visualize All

9 8

Log Log x 0

Viewer

Relation: weather-weka.filters.unsupervised.attribute... [edit]

No.	1: condition	2: temperature	3: class
	Nominal	Numeric	Nominal
1	sunny	85.0	no
2	sunny	75.0	no
3	cloudy	40.0	yes
4	cloudy	53.0	yes
5	raining	35.0	no
6	sunny	36.0	yes
7	raining	45.0	no
8	cloudy	50.0	yes
9	sunny	75.0	no
10	raining	20.0	no
11	cloudy	35.0	yes
12	raining	27.0	no
13	raining	37.0	no
14	cloudy	22.0	yes
15	sunny	43.0	yes
16	cloudy	65.0	yes
17	sunny	38.0	yes

Add instance Undo OK

Add instance Undo OK

After applying “RemoveUseless” no change occurred

By applying “RemoveDuplicates” in unsupervised instance.

**Weka Explorer**

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter Choose RemoveDuplicates Apply Stop

Current relation  
Relation: weather-weka.filters.unsup... Attributes: 3  
Instances: 16 Sum of weights: 16

Selected attribute  
Name: condition Type: Nominal  
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count	Weight
1	sunny	5	5
2	cloudy	6	6
3	raining	5	5

Attributes  
All None Invert Pattern

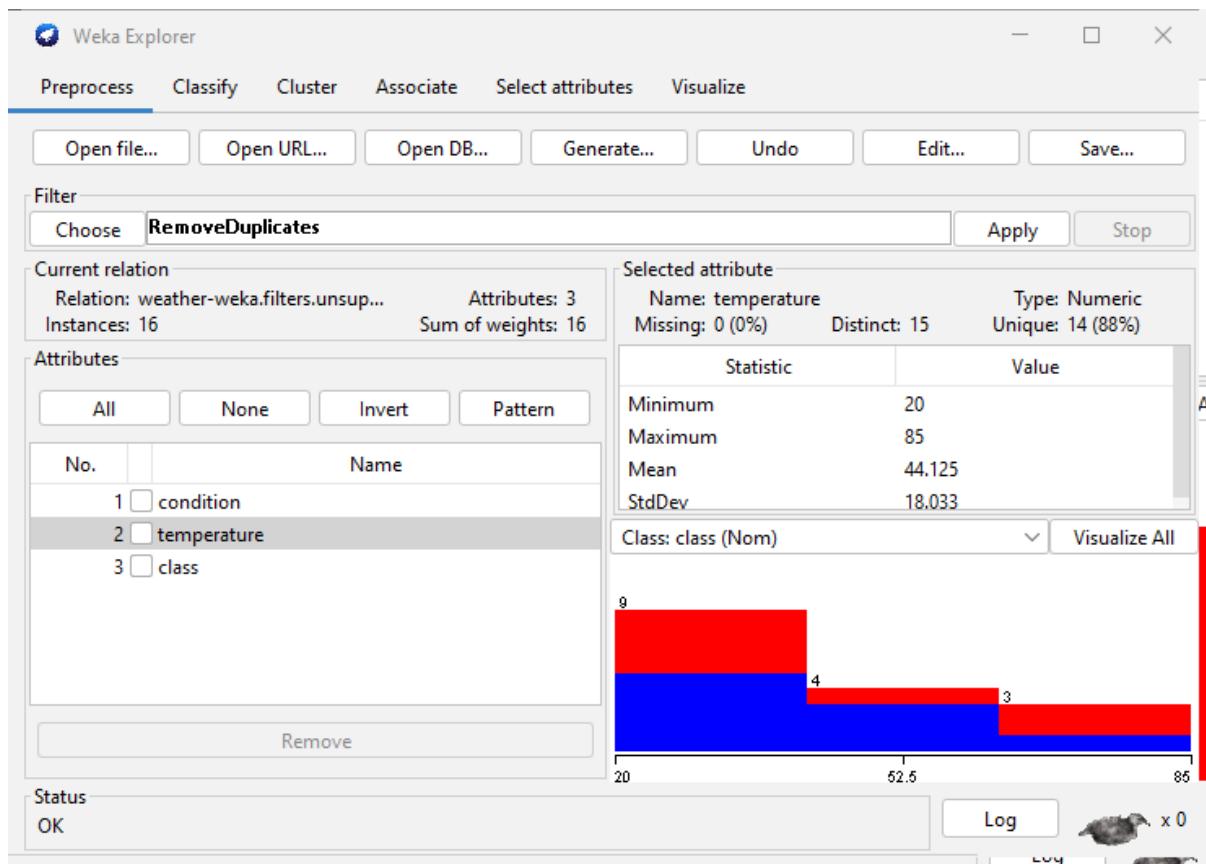
No.	Name
1	<input checked="" type="checkbox"/> condition
2	<input type="checkbox"/> temperature
3	<input type="checkbox"/> class

Remove

Status OK Log

Class: class (Nom) Visualize All

A bar chart titled "Class: class (Nom)" showing the distribution of the "condition" attribute. The x-axis has three categories: "sunny", "cloudy", and "raining". Each category has a bar divided into two segments: a blue segment at the bottom and a red segment at the top. The total height of each bar corresponds to the count listed next to it: 5 for "sunny", 6 for "cloudy", and 5 for "raining".



**Weka Explorer**

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter Choose RemoveDuplicates Apply Stop

Current relation  
Relation: weather-weka.filters.unsup... Attributes: 3  
Instances: 16 Sum of weights: 16

Selected attribute  
Name: class Type: Nominal  
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count	Weight
1	yes	9	9
2	no	7	7

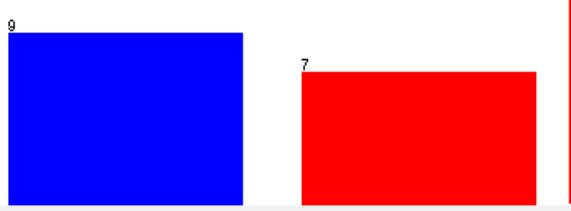
Attributes  
All None Invert Pattern

No.	Name
1	<input type="checkbox"/> condition
2	<input type="checkbox"/> temperature
3	<input type="checkbox"/> class

Remove

Status OK Log 

Class: class (Nom) Visualize All



A bar chart titled "Visualize All" showing the distribution of the "class" attribute. The x-axis categories are "yes" and "no". The "yes" bar is blue and reaches a height of 9, labeled "9" at the top. The "no" bar is red and reaches a height of 7, labeled "7" at the top. The chart has a light gray background with a vertical red bar on the right side.

Viewer

Relation: weather-weka.filters.unsupervised.attr...

No.	1: condition Nominal	2: temperature Numeric	3: class Nominal
1	sunny	85.0	no
2	sunny	75.0	no
3	cloudy	40.0	yes
4	cloudy	53.0	yes
5	raining	35.0	no
6	sunny	36.0	yes
7	raining	45.0	no
8	cloudy	50.0	yes
9	raining	20.0	no
10	cloudy	35.0	yes
11	raining	27.0	no
12	raining	37.0	no
13	cloudy	22.0	yes
14	sunny	43.0	yes
15	cloudy	65.0	yes
16	sunny	38.0	yes

Add instance   Undo   OK

After applying “RemoveDuplicates” one duplicate found and removed.

i.e.; “sunny 75.0 no”

So our data entries reduced to 16 from 17.

## RESULT:

Thus, the dataset in .arff file format is created and Demonstration of preprocessing using weka dataset is shown.

## **12. Demonstration of Association rule process on any supermarket dataset using Apriori algorithm using R.**

### **AIM:**

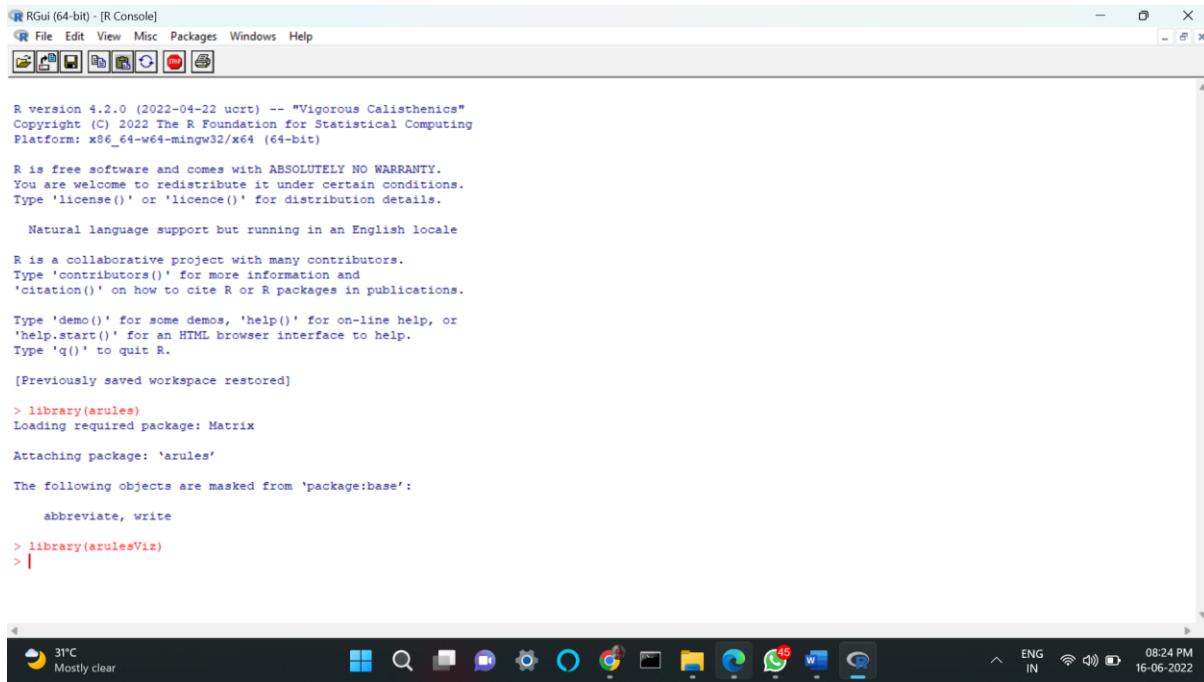
To demonstration of Association rule process on any supermarket dataset using Apriori algorithm using R.

### **PROCEDURE:**

- **Step 1:** The first part of any analysis is to bring in the dataset. We will be using an inbuilt dataset “Groceries” from the ‘arules’ package to simplify our analysis.
- **Step 2:** All stores and retailers store their information of transactions in a specific type of dataset called the “Transaction” type dataset.
- **Step 3:** The ‘pacman’ package is an assistor to help load and install the packages. we will be using pacman to load the arules package.
- **Step 4:** The p\_load() function from “pacman” takes names of packages as arguments.
- **Step 5:** If your system has those packages, it will load them and if not, it will install and load them.

### **CODE IN R:**

```
library(arules)
library(arulesViz)
```



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
[Icons]

R version 4.2.0 (2022-04-22 ucrt) -- "Vigorous Calisthenics"
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.

Natural language support but running in an English locale

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.

[Previously saved workspace restored]

> library(arules)
Loading required package: Matrix

Attaching package: 'arules'

The following objects are masked from 'package:base':
  abbreviate, write

> library(arulesViz)
> |
```

The screenshot shows the RGui (64-bit) - R Console window. The title bar reads "RGui (64-bit) - [R Console]". The menu bar includes File, Edit, View, Misc, Packages, Windows, and Help. Below the menu is a toolbar with various icons. The main console area displays the R startup message, followed by the command to load the "arules" package, which successfully loads the "Matrix" package. Then, the command to load "arulesViz" is entered. At the bottom of the window, there is a status bar showing the date and time (16-06-2022, 08:24 PM), battery level (45%), signal strength, and network connection. The taskbar at the bottom of the screen shows other open applications like File Explorer, Google Chrome, and Microsoft Edge.

```
data(Groceries)
```

```
data <- list(c("a","b","c"),
```

```
c("a","b"),
```

```
c("a","b","d"),
```

```
c("b","e"),
```

```
c("b","c","e"),
```

```
c("a","d","e"),
```

```
c("a","c"),
```

```
c("a","b","d"),
```

```
c("c","e"),
```

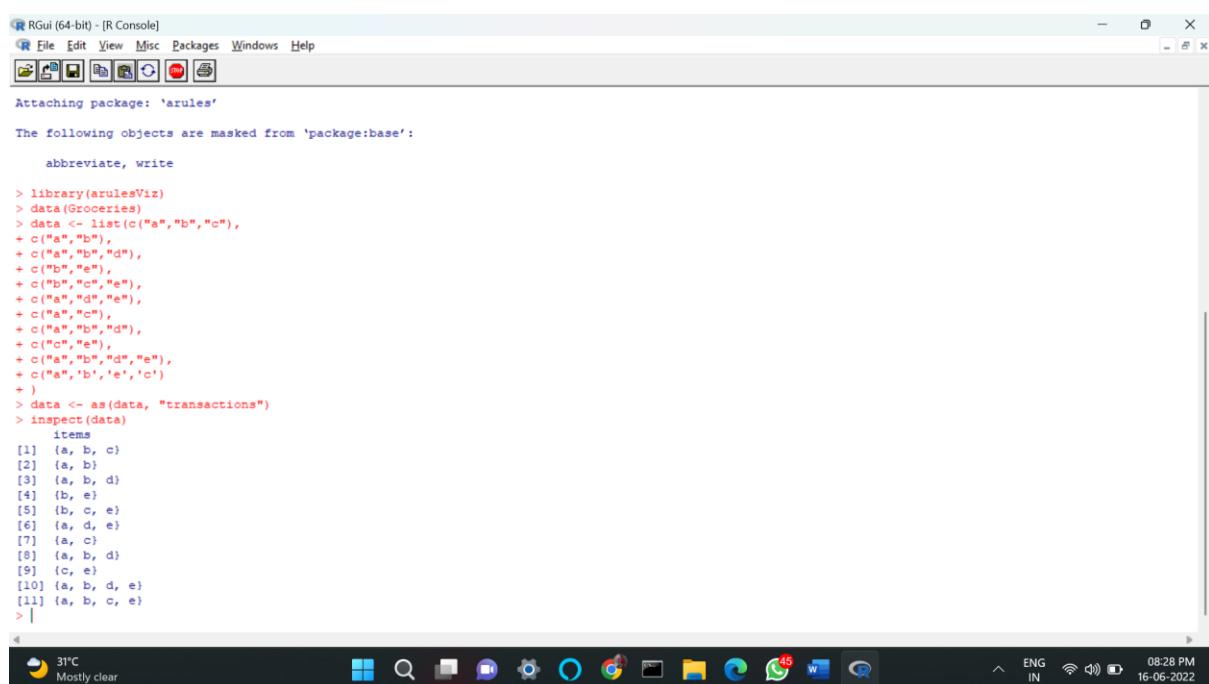
```
c("a","b","d","e"),
```

```
c("a",'b','e','c')
```

```
)
```

```
data <- as(data, "transactions")
```

```
inspect(data)
```



The screenshot shows the RGui interface with the title bar "RGui (64-bit) - [R Console]". The menu bar includes File, Edit, View, Misc, Packages, Windows, Help, and a toolbar with various icons. The main window displays the R session history:

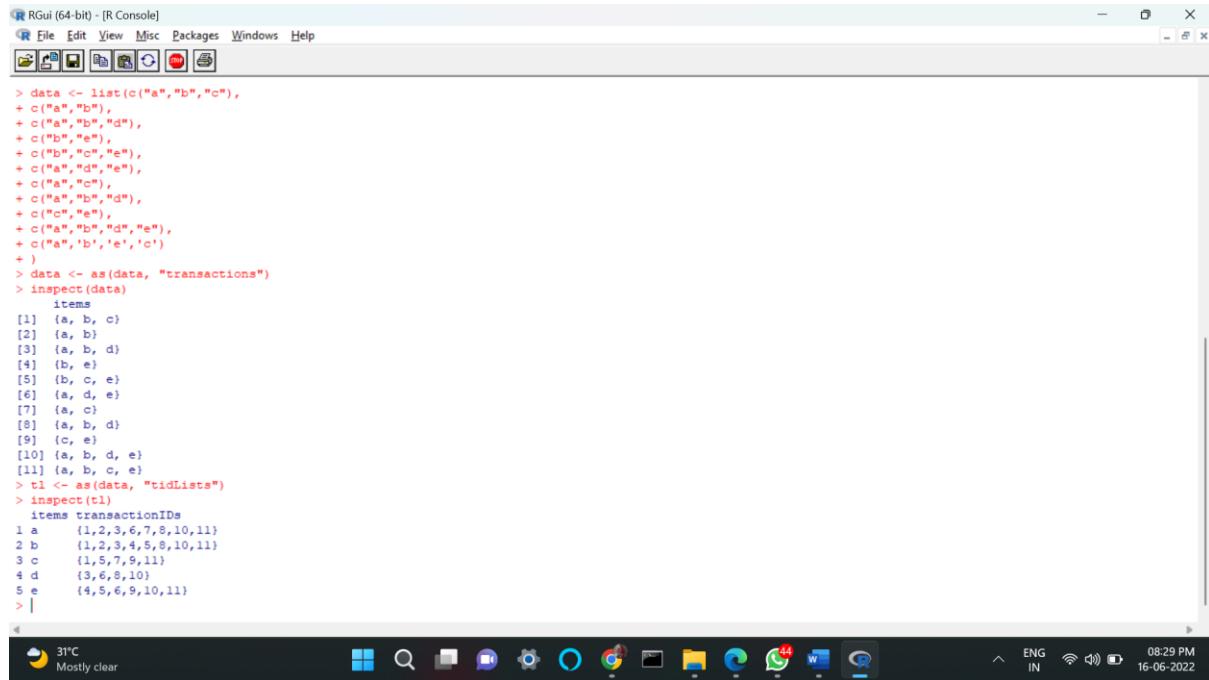
```
Attaching package: 'arules'
The following objects are masked from 'package:base':
  abbreviate, write

> library(arulesViz)
> data(Groceries)
> data <- list(c("a","b","c"),
+ c("a","b"),
+ c("a","b","d"),
+ c("b","e"),
+ c("a","c"),
+ c("a","d","e"),
+ c("b","c"),
+ c("a","b","d"),
+ c("c","e"),
+ c("a","b","d","e"),
+ c("a",'b','e','c')
+ )
> data <- as(data, "transactions")
> inspect(data)
  items
[1] {a, b, c}
[2] {a, b}
[3] {a, b, d}
[4] {b, e}
[5] {b, c, e}
[6] {a, d, e}
[7] {a, c}
[8] {a, b, d}
[9] {c, e}
[10] {a, b, d, e}
[11] {a, b, c, e}
> |
```

The system tray at the bottom shows the date and time (16-06-2022, 08:28 PM), battery level (45%), signal strength, and language (ENG IN).

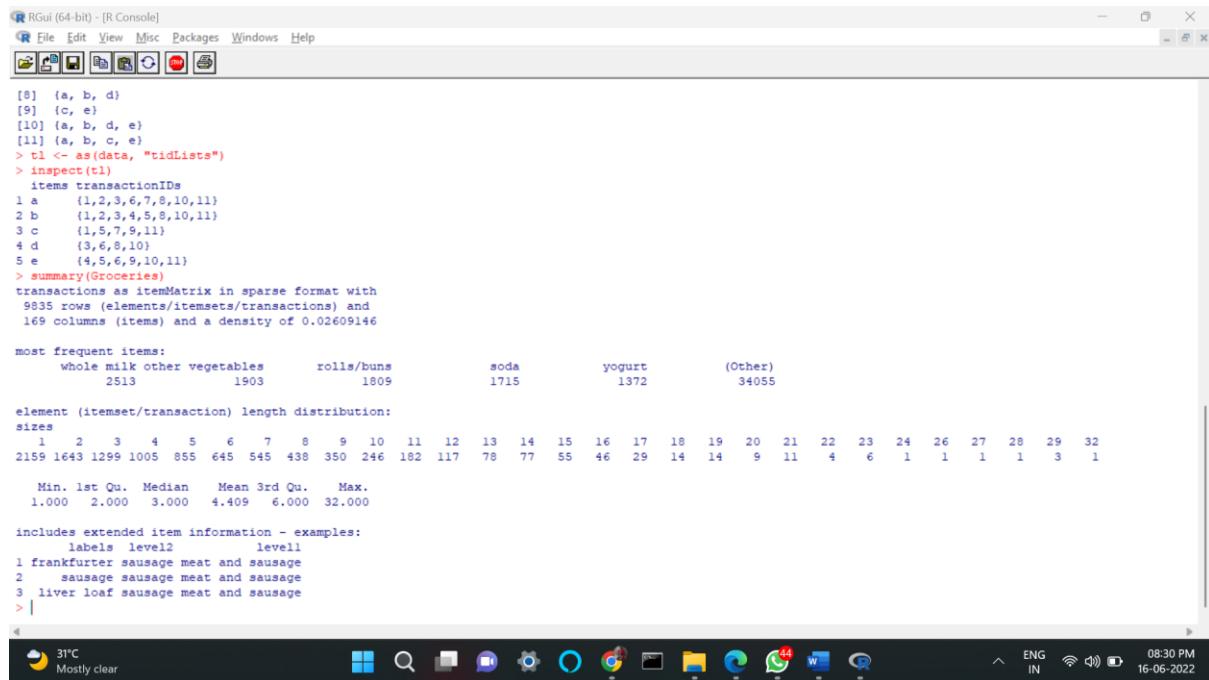
```
tl <- as(data, "tidLists")
```

```
inspect(tl)
```



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
[1] > data <- list(c("a","b","c"),
+ c("a","b"),
+ c("a","b","d"),
+ c("b","c","e"),
+ c("b","c","e"),
+ c("a","c"),
+ c("a","c"),
+ c("a","b","d"),
+ c("c","e"),
+ c("a","b","d"),
+ c("a","b","e"),
+ c("a","b","c"))
+ )
> data <- as(data, "transactions")
> inspect(data)
  items
[1] {a, b, c}
[2] {a, b}
[3] {a, b, d}
[4] {b, e}
[5] {b, c, e}
[6] {a, d, e}
[7] {a, c}
[8] {a, b, d}
[9] {c, e}
[10] {a, b, d, e}
[11] {a, b, c, e}
> tl <- as(data, "tidLists")
> inspect(tl)
  items transactionIDs
1 a      {1,2,3,6,7,8,10,11}
2 b      {1,2,3,4,5,8,10,11}
3 c      {1,5,7,9,11}
4 d      {3,6,8,10}
5 e      {4,5,6,9,10,11}
> |
```

```
summary(Groceries)
```



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
[8] {a, b, d}
[9] {c, e}
[10] {a, b, d, e}
[11] {a, b, c, e}
> tl <- as(data, "tidLists")
> inspect(tl)
  items transactionIDs
1 a      {1,2,3,6,7,8,10,11}
2 b      {1,2,3,4,5,8,10,11}
3 c      {1,5,7,9,11}
4 d      {3,6,8,10}
5 e      {4,5,6,9,10,11}
> summary(Groceries)
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146

most frequent items:
      Whole milk other vegetables      rolls/buns      soda      yogurt      (Other)
      2513     1903        1809       1715       1372       34055

element (itemset/transaction) length distribution:
sizes
  1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   26   27   28   29   32
2159 1643 1299 1005 855 645 545 438 350 246 182 117 78 77 55 46 29 14 14 9 11 4 6 1 1 1 3 1

Min. 1st Qu. Median Mean 3rd Qu. Max.
1.000 2.000 3.000 4.409 6.000 32.000

includes extended item information - examples:
  labels level2 level1
1 frankfurter sausage meat and sausage
2 sausage sausage meat and sausage
3 liver loaf sausage meat and sausage
> |
```

```
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.80))
```

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
[Icons]

element (itemset/transaction) length distribution:
sizes
  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   31
2159 1643 1299 1005 855 645 545 438 350 246 182 117 78 77 55 46 29 14 14 9 11 4 26 6 1 1 27 28 29 30 1

Min. 1st Qu. Median Mean 3rd Qu. Max.
1.000 2.000 3.000 4.409 6.000 32.000

includes extended item information - examples:
  labels level1 level2
1 frankfurter sausage meat and sausage
2 sausage sausage meat and sausage
3 liver loaf sausage meat and sausage
> rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.80))
Apriori

Parameter specification:
confidence minval max arcm aval originalSupport maxtime support minlen maxlen target ext
0.8      0.1    1 none FALSE           TRUE      5 0.001    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: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
Checking subsets of size 1 2 3 4 5 6 done [0.03s].
writing ... [410 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
> |
```

The screenshot shows the RGui interface with the command `rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.80))` entered in the console. The output displays the length distribution of itemsets, the inclusion of extended item information, and the parameters used for the Apriori algorithm. The system tray at the bottom indicates it's 16-06-2022 at 08:32 PM.

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

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
[Icons]

2 sausage sausage meat and sausage
3 liver loaf sausage meat and sausage
> rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.80))
Apriori

Parameter specification:
confidence minval max arcm aval originalSupport maxtime support minlen maxlen target ext
0.8      0.1    1 none FALSE           TRUE      5 0.001    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: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
Checking subsets of size 1 2 3 4 5 6 done [0.03s].
writing ... [410 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
> inspect(rules[1:10])
      lhs                      rhs          support  confidence coverage lift       count
[1] (liquor, red/blush wine) => (bottled beer) 0.001931876 0.9047619 0.002135231 11.235269 19
[2] (curd, cereals)          => (whole milk)  0.001016777 0.9090909 0.001118454 3.557863 10
[3] (yogurt, cereals)        => (whole milk)  0.001728521 0.8095238 0.002135231 3.168192 17
[4] (butter, jam)            => (whole milk)  0.001016777 0.8333333 0.001220132 3.261374 10
[5] ( soups, bottled beer)  => (whole milk)  0.001118454 0.9166667 0.001220132 3.587512 11
[6] (napkins, house keeping products) => (whole milk) 0.001321810 0.8125000 0.001626843 3.179840 13
[7] (whipped/sour cream, house keeping products) => (whole milk) 0.001220132 0.9230769 0.001321810 3.612599 12
[8] (pastry, sweet spreads)   => (whole milk)  0.001016777 0.9090909 0.001118454 3.557863 10
[9] (turkey, curd)           => (other vegetables) 0.001220132 0.8000000 0.001525165 4.134524 12
[10] (rice, sugar)           => (whole milk)  0.001220132 1.0000000 0.001220132 3.913649 12
> |
```

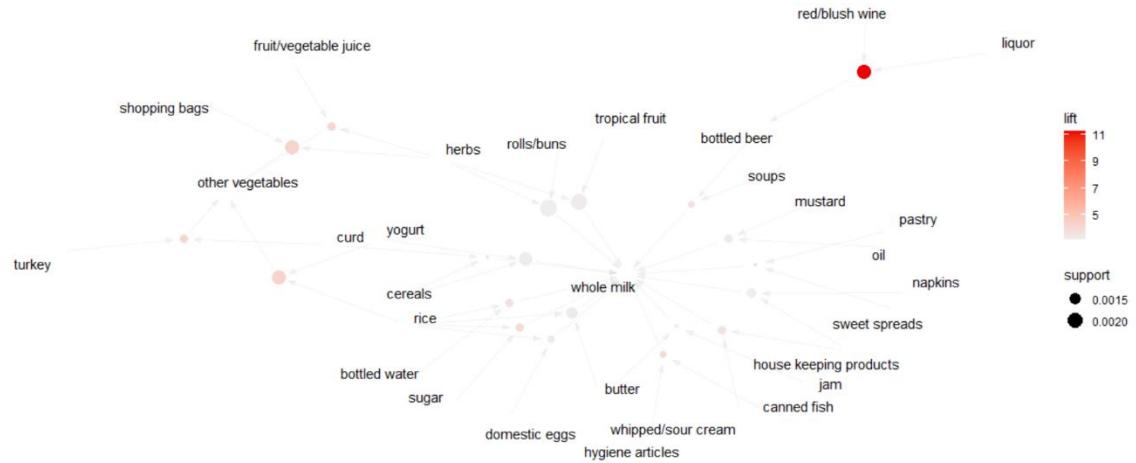
The screenshot shows the RGui interface with the command `inspect(rules[1:10])` entered in the console. The output displays the first 10 rules generated by the Apriori algorithm, showing the left-hand side (lhs), right-hand side (rhs), support, confidence, coverage, lift, and count for each rule. The system tray at the bottom indicates it's 16-06-2022 at 08:32 PM.

```
plot(rules[1:20], method = "graph", control = list(type = "items"))
```

```
RGUI (64-bit) - [R Console]
File Edit View Mis Packages Windows Help

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.03s].
writing ... [410 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
> inspect(rules[1:10])
      lhs                                rhs          support    confidence coverage     lift    count
[1] (liquor, red/blush wine)           => (bottled beer) 0.001931876 0.9047619 0.002135231 11.235269 19
[2] (curd, cereals)                  => (Whole milk)   0.001016777 0.9090909 0.001118454 3.557863 10
[3] (yogurt, cereals)                => (Whole milk)   0.001728521 0.8995238 0.002135231 3.168192 17
[4] (butter, jam)                   => (Whole milk)   0.001016777 0.8333333 0.001220132 3.261374 10
[5] (soups, bottled beer)            => (Whole milk)   0.001118454 0.9166667 0.001220132 3.587512 11
[6] (napkins, house keeping products) => (Whole milk)   0.001321810 0.8125000 0.001626843 3.179840 13
[7] (whipped/sour cream, house keeping products) => (Whole milk) 0.001220132 0.9230769 0.001321810 3.612599 12
[8] (pastry, sweet spreads)          => (Whole milk)   0.001016777 0.9090909 0.001118454 3.557863 10
[9] (turkey, curd)                  => (other vegetables) 0.001220132 0.8000000 0.001525165 4.134524 12
[10] (rice, sugar)                 => (whole milk)   0.001220132 1.0000000 0.001220132 3.913649 12

> plot(rules[1:20],
+ method = "graph",
+ control = list(type = "items"))
Warning: Unknown control parameters: type
Available control parameters (with default values):
layout      = stress
circular    = FALSE
ggraphdots  = NULL
edges       = <environment>
nodes       = <environment>
nodetext    = <environment>
colors      = c("#E60000FF", "#EEEEEFFF")
engine      = ggraph2
max        = 100
verbose    = FALSE
> |
```



## RESULT:

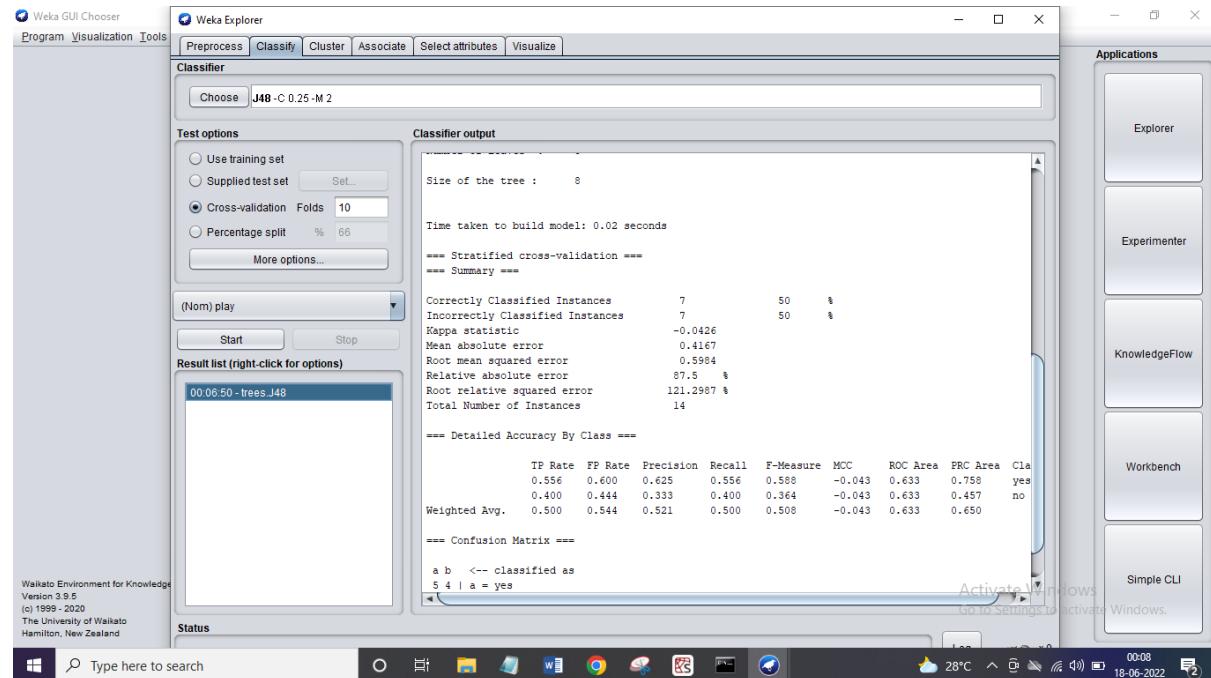
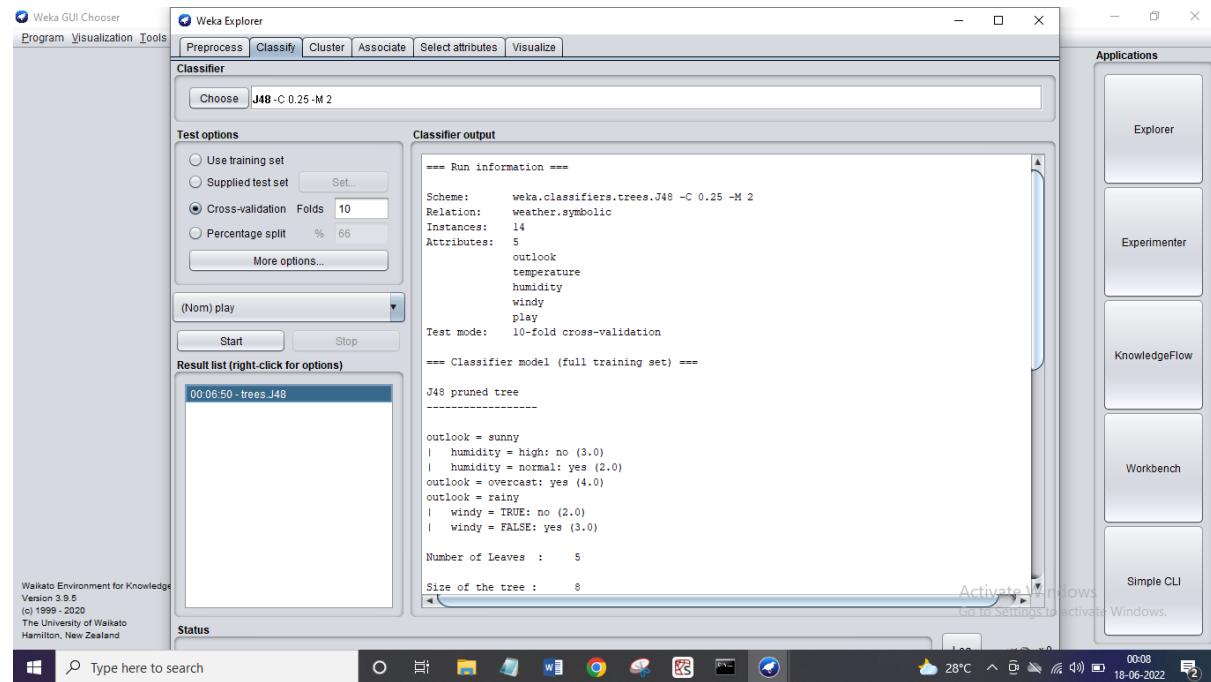
Thus, the implementation of Apriori Algorithm is executed in R language is shown.

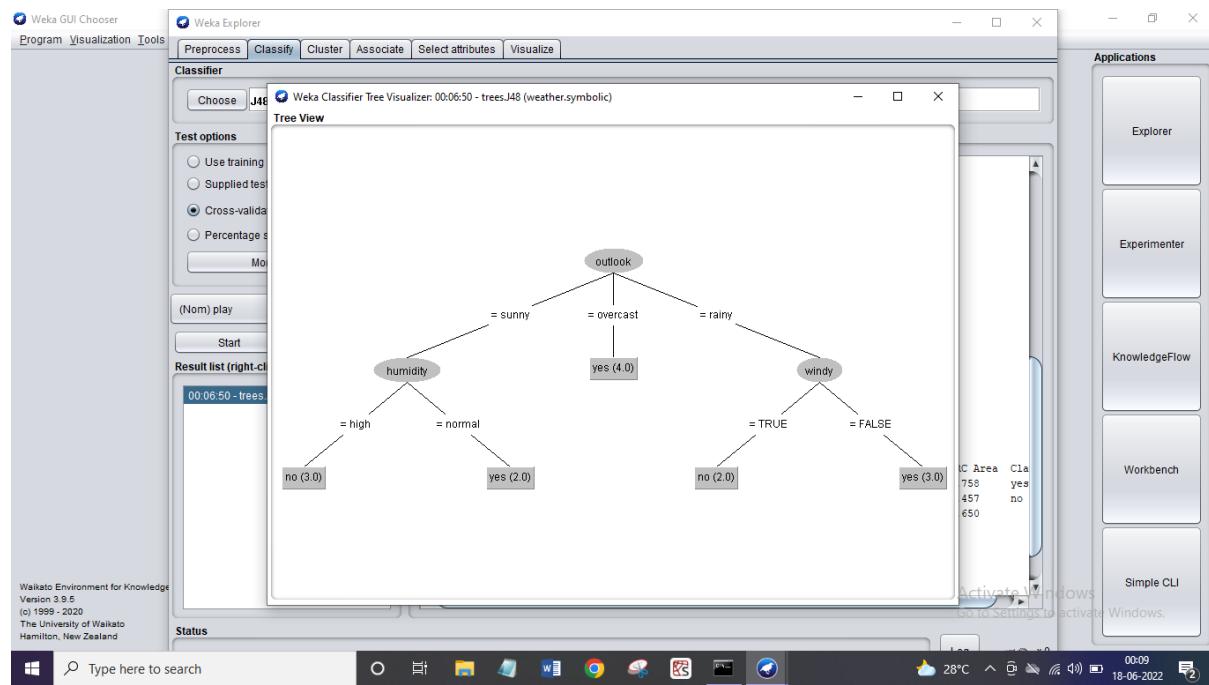
## 13. Demonstration of classification ruleprocess on WEKA dataset using j48 algorithm

### AIM:

To demonstrate the classification ruleprocess on WEKA dataset using j48 algorithm in WEKA

### PROGRAM AND OUTPUT Screenshot:





## RESULT:

Thus, the demonstrate the classification ruleprocess on WEKA dataset using j48 algorithm in WEKA is shown.

#### **14. Demonstration of classification rule process on WEKA dataset using naïve bayes algorithm. Compute the accuracy of the classifier, precision, and recall.**

**Aim:** This experiment illustrates the use of naïve bayes classifier in WEKA. The sample data set used in this experiment is “iris” data available at arff format. This document assumes that appropriate data pre-processing has been performed.

#### **PROCEDURE:**

**Step-1:** We begin the experiment by loading the data (iris.arff) into WEKA.

**Step-2:** Next we select the “classify” tab and click “choose” button to select the “naïve bayes” classifier.

**Step-3:** Now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values his default version does perform some pruning but does not perform error pruning.

**Step-4:** Under the “text “options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don’t have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

**Step-5:** We now click ”start” to generate the model. The ASCII version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

**Step-6:** Note that the classification accuracy of model is about 96%. This indicates that we may find more work. (Either in pre-processing or in selecting current parameters for the classification)

**Step-7:** Now WEKA also lets us a view a graphical version of the classification. This can be done by right clicking the last result set and selecting “visualize” from the pop-up menu.

**Step-8:** We will use our model to classify the new instances.

**Step-9:** In the main panel under “text “options click the “supplied test set” radio button and then click the “set” button. This will show pop-up window which will allow you to open the file containing test instances.

The following screenshot shows the classification rules that were generated when naive bayes algorithm is applied on the given dataset.

## Dataset – Iris Dataset in WEKA

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. Under 'Classifier', 'NaiveBayes' is chosen. In the 'Test options' section, 'Supplied test set' is selected with 'Set...' button. The 'Result list (right-click for options)' pane shows three entries: '14:47:57 - bayes.NaiveBayes', '15:45:28 - bayes.NaiveBayes', and '15:56:37 - bayes.NaiveBayes'. The main window displays the 'Classifier output' and 'Relation: iris' tables. The 'Classifier output' table includes columns for mean, std. dev., weight sum, and precision. The 'Relation: iris' table lists 50 instances with columns for No., sepal length, sepal width, petal length, petal width, and class. A context menu is open over the table, with the option 'Right click (or left+alt) for context menu' highlighted.

## Using Naïve Bayes Classifier

No of instances = 150

## Output:

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier Choose NaiveBayes

Test options

- Use training set
- Supplied test set Set... (Folds: 10)
- Cross-validation Folds: 10
- Percentage split %: 66
- More options...

(Nom) class

Start Stop

Result list (right-click for options)

14:47:57 - bayes.NaiveBayes  
15:45:28 - bayes.NaiveBayes

Classifier output

```
==== Run information ====
Scheme: weka.classifiers.bayes.NaiveBayes
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Test mode: 10-fold cross-validation
==== Classifier model (full training set) ====
Naive Bayes Classifier
```

Attribute	Iris-setosa	Iris-versicolor	Iris-virginica
Class	(0.33)	(0.33)	(0.33)
sepallength			
mean	4.9913	5.9379	6.5795
std. dev.	0.355	0.5042	0.6353
weight sum	50	50	50
precision	0.1059	0.1059	0.1059
sepalwidth			
mean	3.4015	2.7687	2.9625
std. dev.	0.3925	0.3038	0.3008
weight sum	50	50	50
precision	0.1091	0.1091	0.1091
petallength			
mean	1.4694	4.2452	5.5516
std. dev.	0.1782	0.4712	0.5529
weight sum	50	50	50
precision	0.1143	0.1143	0.1143

Status OK Log x 0

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier Choose NaiveBayes

Test options

- Use training set
- Supplied test set Set... (Folds: 10)
- Cross-validation Folds: 10
- Percentage split %: 66
- More options...

(Nom) class

Start Stop

Result list (right-click for options)

14:47:57 - bayes.NaiveBayes  
15:45:28 - bayes.NaiveBayes

Classifier output

```
precision 0.1405 0.1405 0.1405
petalwidth
mean 0.2743 1.3097 2.0343
std. dev. 0.1096 0.1915 0.2646
weight sum 50 50 50
precision 0.1143 0.1143 0.1143
```

Time taken to build model: 0 seconds

==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances 144 96 %
Incorrectly Classified Instances 6 4 %
Kappa statistic 0.94
Mean absolute error 0.0342
Root mean squared error 0.155
Relative absolute error 7.6997 %
Root relative squared error 32.9794 %
Total Number of Instances 150

==== Detailed Accuracy By Class ====

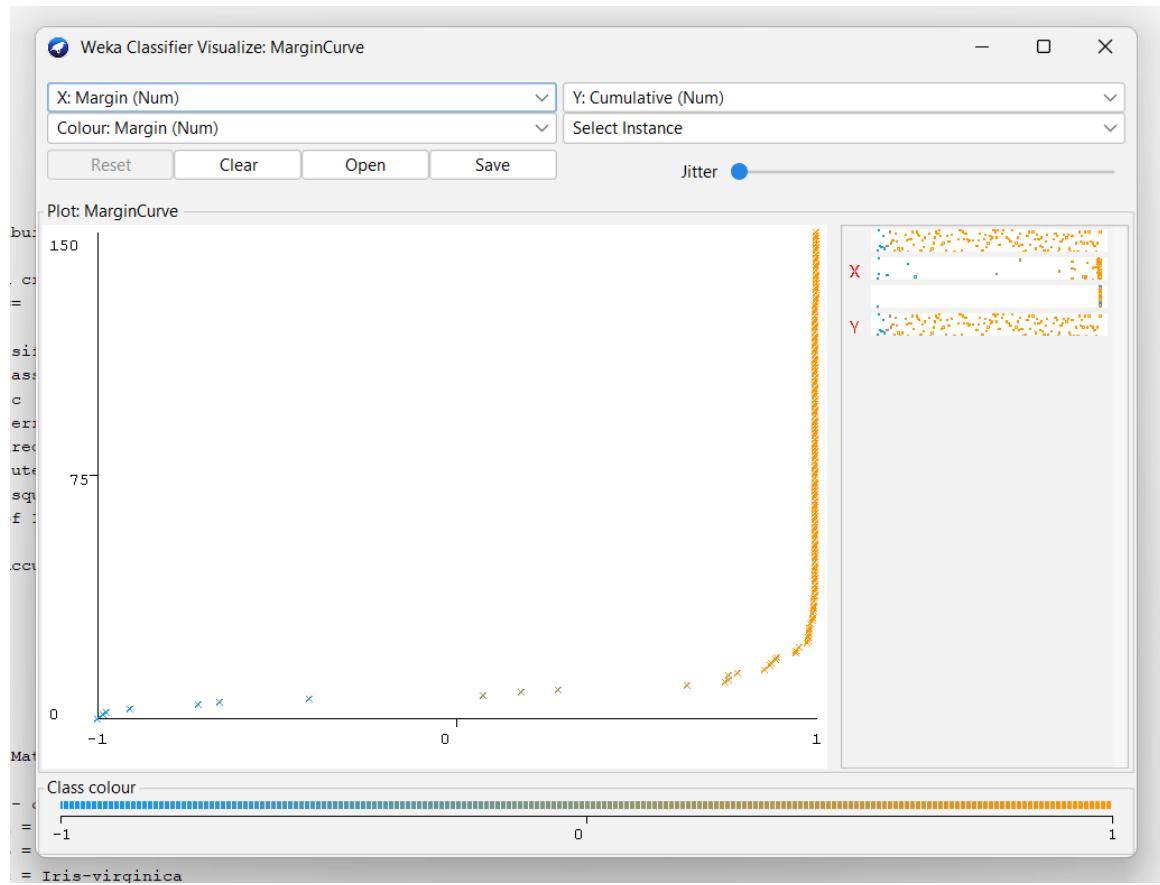
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-setosa
0.960	0.040	0.923	0.960	0.941	0.911	0.952	0.983	0.986	Iris-versicolor
0.920	0.020	0.958	0.920	0.939	0.910	0.992	0.986	0.986	Iris-virginica
Weighted Avg.	0.960	0.020	0.960	0.960	0.940	0.994	0.989	0.989	

==== Confusion Matrix ====

a b c	---	classified as
50 0 0		a = Iris-setosa
0 48 2		b = Iris-versicolor
0 0 46		c = Iris-virginica

Status

## Margin Curve Output:



## RESULT:

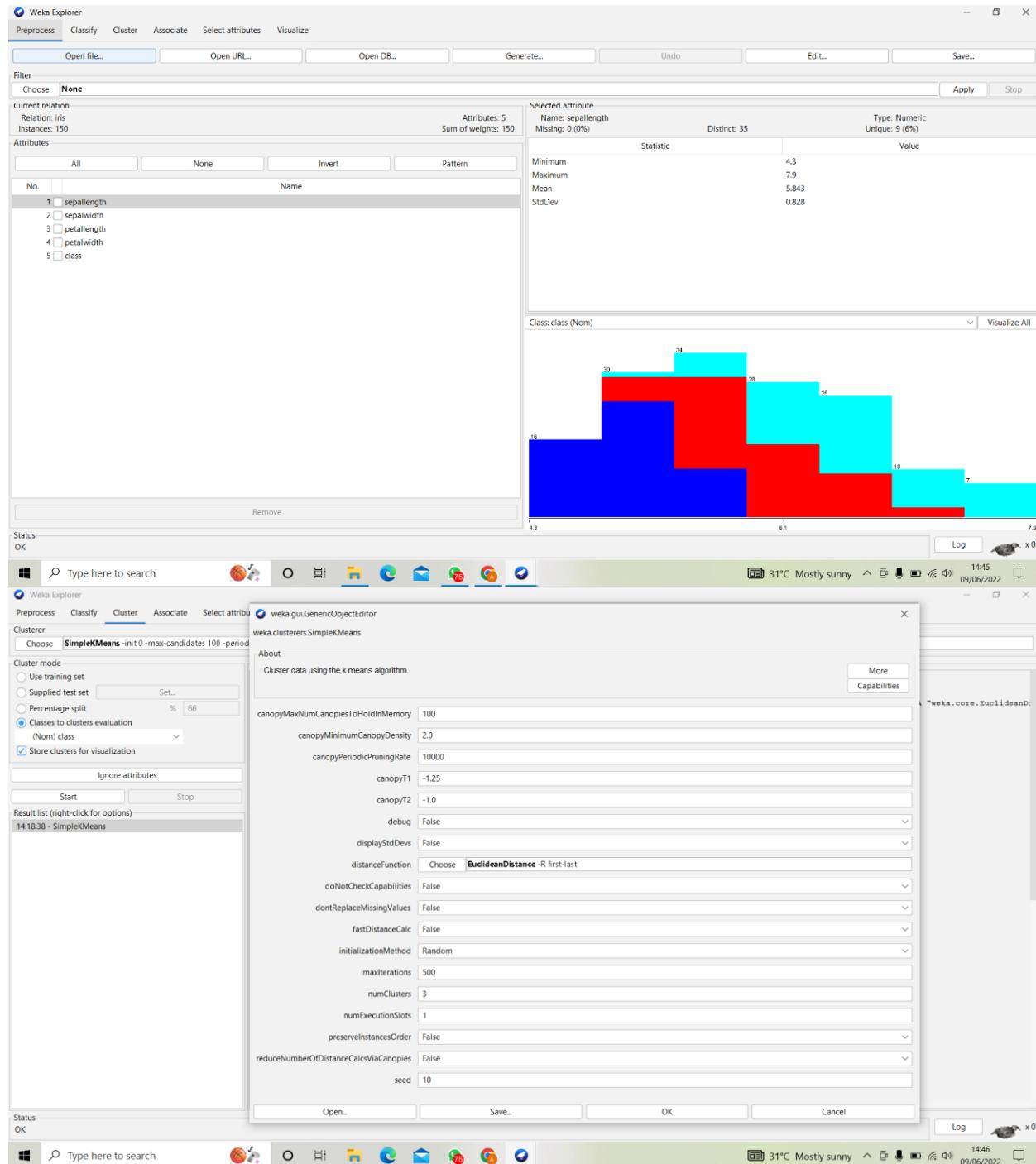
Thus, the Demonstration of classification rule process on WEKA dataset using naïve bayes algorithm and Compute the accuracy of the classifier, precision, and recall shown.

## 15. Demonstration of clustering rule process on data set iris.arff using simple K.Means

### AIM:

To find demonstration of clustering rule process on data set iris.arff using simple K-Means in WEKA

### PROCEDURE AND OUTPUT SCREENSHOT:



**Weka Explorer**

- Preprocess Classify Cluster Associate Select attributes

**Clusterer**

Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-pruning 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance" -R first-last -l 500 -num-slots 1 -S 10

**Cluster mode**

- Use training set
- Supplied test set Set...
- Percentage split % 66
- Classes to clusters evaluation (Nom) class
- Store clusters for visualization

**Ignore attributes**

Start Stop

**Result list (right-click for options)**

14:18:38 - SimpleKMeans

weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

canopyMaxNumCanopiesToHoldInMemory 100  
 canopyMinimumCanopyDensity 2.0  
 canopyPeriodicPruningRate 10000  
 canopyT1 -1.25  
 canopyT2 -1.0  
 debug False  
 displayStdDevs False  
 distanceFunction Choose EuclideanDistance -R first-last  
 doNotCheckCapabilities False  
 dontReplaceMissingValues False  
 fastDistanceCalc False  
 initializationMethod Random  
 maxIterations 500  
 numClusters 3  
 numExe set number of clusters  
 preserveInstancesOrder False  
 reduceNumberOfDistanceCalcsViaCanopies False  
 seed 10

**Status**

OK

Type here to search

Log x0

31°C Mostly sunny 14:46 09/06/2022

---

**Weka Explorer**

- Preprocess Classify Cluster Associate Select attributes Visualize

**Clusterer**

Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance" -R first-last -l 500 -num-slots 1 -S 10

**Cluster mode**

- Use training set
- Supplied test set Set...
- Percentage split % 66
- Classes to clusters evaluation (Nom) class
- Store clusters for visualization

**Ignore attributes**

Start Stop

**Result list (right-click for options)**

14:18:38 - SimpleKMeans

**Cluster output**

```
==== Run information ====
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance"
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
Ignored:
class
Test mode: Classes to clusters evaluation on training data
==== Clustering model (full training set) ====
KMeans
=====
Number of iterations: 6
Within cluster sum of squared errors: 6.998114004026762
Initial starting points (random):
Cluster 0: 6.1,2.9,4.7,1.4
Cluster 1: 6.2,2.9,4.9,1.3
Cluster 2: 6.9,3.1,5.1,2.3
Missing values globally replaced with mean/mode
Final cluster centroids:
Attribute Full Data Cluster#
(150.0) (61.0) (50.0) (39.0)
sepallength 5.8433 5.8885 5.006 6.8462
```

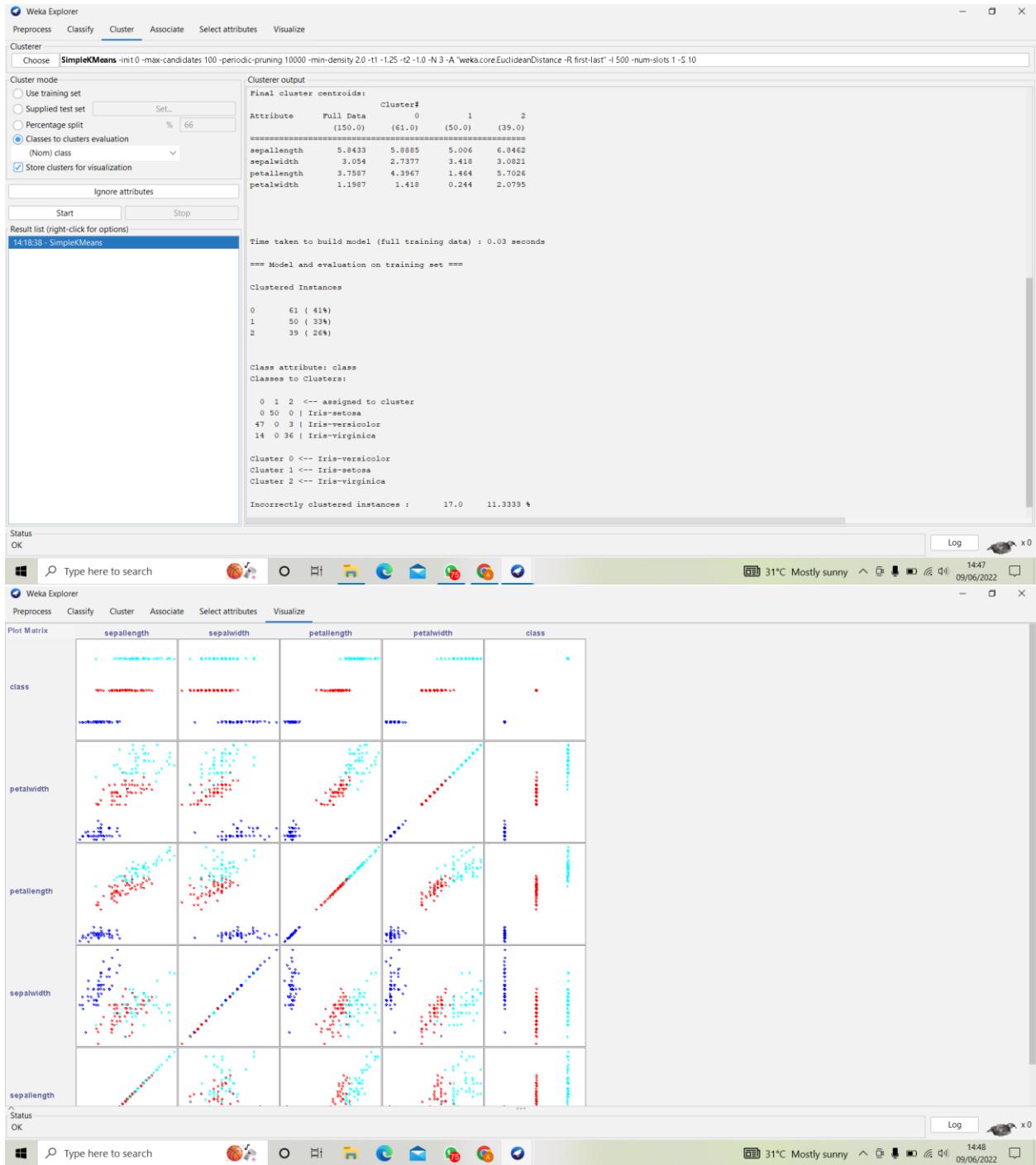
**Status**

OK

Type here to search

Log x0

31°C Mostly sunny 14:46 09/06/2022



## RESULT:

Thus, the demonstration of clustering rule process on data set iris.arff using simple K-Means in WEKA is shown.

## 16. Demonstration of classification rule a process on student dataset using KNN algorithm in Python.

### AIM:

To find Demonstration of classification rule a process on student dataset using KNN algorithm in Python.

### PROGRAM CODE AND OUTPUT SCREENSHOT:

```
In [81]: #Load the necessary python libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

```
In [82]: #Load the dataset
df = pd.read_csv('Data.csv')

#print the first 5 rows of the dataframe.
df.head()
```

Out[82]:

	gender	Nationality	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView	Discussion
0	M	KW	KuwalT	lowerlevel	G-04	A	IT	F	Father	15	16	2	20
1	M	KW	KuwalT	lowerlevel	G-04	A	IT	F	Father	20	20	3	25
2	M	KW	KuwalT	lowerlevel	G-04	A	IT	F	Father	10	7	0	30
3	M	KW	KuwalT	lowerlevel	G-04	A	IT	F	Father	30	25	5	35
4	M	KW	KuwalT	lowerlevel	G-04	A	IT	F	Father	40	50	12	50

```
In [83]: #Let's observe the shape of the dataframe.
df.shape
```

Out[83]: (480, 17)

```
In [84]: x = df[['Discussion','raisedhands']]  
y = df['VisITEDResources']
```

```
In [85]: x
```

```
Out[85]:
```

	Discussion	raisedhands
0	20	15
1	25	20
2	30	10
3	35	30
4	50	40
...	...	...
475	8	5
476	28	50
477	29	55
478	57	30
479	62	35

480 rows × 2 columns

```
In [86]: y
```

```
Out[86]: 0      16  
1      20  
2       7  
3      25  
4      50  
..  
475     4  
476    77  
477    74  
478    17  
479    14  
Name: VISITEDResources, Length: 480, dtype: int64
```

```
In [87]: #importing train_test_split  
from sklearn.model_selection import train_test_split
```

```
In [88]: X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=10)
```

```
In [89]: len(X_train)
```

```
Out[89]: 384
```

```
In [90]: len(y_test)
```

```
Out[90]: 96
```

```
In [91]: #import KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier

#Setup arrays to store training and test accuracies
neighbors = np.arange(1,9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

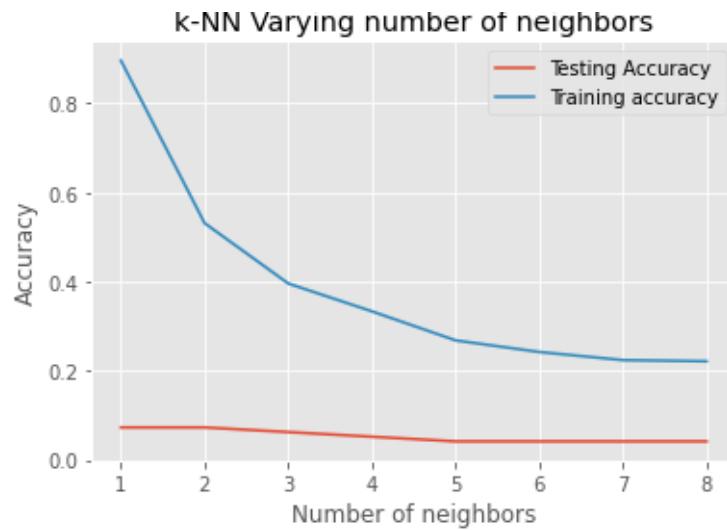
for i,k in enumerate(neighbors):
    #Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)

    #Fit the model
    knn.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, y_test)
```

```
In [92]: #Generate plot
plt.title('k-NN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
```



```
In [93]: knn = KNeighborsClassifier(n_neighbors=7)
```

```
In [94]: knn.fit(X_train,y_train)
```

```
Out[94]: KNeighborsClassifier(n_neighbors=7)
```

```
In [95]: knn.score(X_test,y_test)
```

```
Out[95]: 0.041666666666666664
```

```
In [96]: from sklearn.metrics import confusion_matrix  
  
In [97]: y_pred = knn.predict(X_test)  
  
In [80]: confusion_matrix(y_test,y_pred)  
  
Out[80]: array([[0, 0, 0, ..., 0, 0, 0],  
                 [0, 0, 0, ..., 0, 0, 0],  
                 [0, 1, 0, ..., 0, 0, 0],  
                 ...,  
                 [0, 1, 0, ..., 0, 0, 0],  
                 [1, 0, 1, ..., 0, 0, 0],  
                 [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

## RESULT:

Thus, the demonstration of classification rule a process on student dataset using KNN algorithm in Python is shown.

## 17. Demonstrate Decision tree ID3 algorithm for any dataset using PYTHON

### AIM:

To implement the Navie Bayes algorithm using python

### PROCEDURE:

**Step 1:** Data Pre-processing step

**Step 2:** Fitting Naive Bayes to the Training Set

**Step 3:** Prediction of the test set result

**Step 4:** Creating Confusion Matrix

### PROGRAM CODE AND OUTPUT SCREENSHOT:

```
In [1]: import numpy as np

In [2]: def fit(X_train,Y_train):
    result = {}
    class_values = set(Y_train)
    for curr_value in class_values:
        result[curr_value] = {}
        result["total_data"] = len(Y_train)
        curr_class_rows = (Y_train == curr_value)
        X_train_curr = X_train[curr_class_rows]
        Y_train_curr = Y_train[curr_class_rows]
        num_features = X_train.shape[1]
        result[curr_value]["total_count"] = len(Y_train_curr)
        for j in range(1,num_features+1):
            result[curr_value][j] = {}
            all_possible_values = set(X_train[:,j-1])
            for this_value in all_possible_values:
                result[curr_value][j][this_value] = (X_train_curr[:,j-1]==this_value).sum()
    return result

In [3]: def probability(dictionary,x,current_class):
    output= np.log(dictionary[current_class]["total_count"])-np.log(dictionary["total_data"])
    num_features = len(dictionary[current_class].keys())-1;
    for j in range(1,num_features+1):
        xj = x[j-1]
        count_current_class_with_value_xj = dictionary[current_class][j][xj] + 1
        count_current_class = dictionary[current_class]["total_count"] + len(dictionary[current_class][j].keys())
        current_xj_prob = np.log(count_current_class_with_value_xj) -np.log(count_current_class)
        output = output + current_xj_prob
    return output
```

```
In [4]: def predictSinglePoint(dictionary,x):
    classes = dictionary.keys()
    best_p = -1000
    best_class = -1
    first_run = True
    for current_class in classes:
        if(current_class == "total_data"):
            continue
        p_curr_class = probability(dictionary,x,current_class)
        if(first_run or p_curr_class > best_p):
            best_p = p_curr_class
            best_class = current_class
        first_run = False
    return best_class
```

```
In [5]: def predict(dictionary,X_test):
    Y_pred = []
    for x in X_test:
        x_class = predictSinglePoint(dictionary,x)
        Y_pred.append(x_class)
    return Y_pred
```

```
In [6]: def makelabelled(column):
    second_limit = column.mean()
    first_limit = 0.5 * second_limit
    third_limit = 1.5 * second_limit
    for i in range(0,len(column)):
        if(column[i]<first_limit):
            column[i] = 0
        elif(column[i] < second_limit):
            column[i] = 1
        elif(column[i]<third_limit):
            column[i] = 2
        else:
            column[i] = 3
    return column
```

```
In [7]: from sklearn import datasets
iris = datasets.load_iris()
x = iris.data
y = iris.target
```

```
In [8]: for i in range(0,x.shape[-1]):
    x[:,i] = makelabelled(x[:,i])
```

```
In [9]: from sklearn import model_selection
X_train,X_test,Y_train,Y_test = model_selection.train_test_split(x,y,test_size=0.25,random_state=0)
```

```
In [10]: dictionary = fit(X_train,Y_train)
```

```
In [10]: dictionary = fit(X_train,Y_train)

In [11]: Y_pred = predict(dictionary,X_test)

In [12]: from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(Y_test,Y_pred))
print(confusion_matrix(Y_test,Y_pred))

precision    recall  f1-score   support

          0       1.00      1.00      1.00      13
          1       0.94      1.00      0.97      16
          2       1.00      0.89      0.94       9

   accuracy                           0.97      38
  macro avg       0.98      0.96      0.97      38
weighted avg       0.98      0.97      0.97      38

[[13  0  0]
 [ 0 16  0]
 [ 0  1  8]]
```

## RESULT:

Thus, the implement the Navie Bayes algorithm using python

**18. Write a Program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/ Python ML library classes can be used for this problem.**

**AIM:**

To implement k-Nearest Neighbour algorithm to classify the iris data set and print both correct and wrong predictions.

**PROCEDURE:**

**STEP1:** The data is imported and then it goes through splitting process.

**STEP2:** Then by using the KNN classifier the model is trained and value of k is taken as 3.

**STEP3:** The distance is calculated with all the training points to test the point and then select the three lowest distance points.

**STEP4:** After that the prediction process is done for getting the result.

**PROGRAM CODING IN PYTHON:**

```
import sklearn  
import pandas as pd  
from sklearn.datasets import load_iris  
iris=load_iris()  
iris.keys()  
df=pd.DataFrame(iris['data'])  
print(df)  
print(iris['target_names'])  
iris['feature_names']  
X=df  
y=iris['target']  
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train,y_train)

import numpy as np
x_new=np.array([[5,2.9,1,0.2]])

prediction=knn.predict(x_new)
iris['target_names'][prediction]

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
y_pred=knn.predict(X_test)
cm=confusion_matrix(y_test,y_pred)
print(cm)

print(" correct prediction",accuracy_score(y_test,y_pred))
print(" wrong prediction",(1-accuracy_score(y_test,y_pred)))
```

## OUTPUT SCREENSHOT:

The screenshot shows a Jupyter Notebook interface. The title bar says "Jupyter knn algo Last Checkpoint: 3 hours ago (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3 (ipykernel). The toolbar has icons for file operations like Open, Save, Run, and Cell. The code cell (In [9]) contains Python code for importing sklearn.metrics, predicting on the Iris dataset, and calculating a confusion matrix and accuracy score. The output cell (out[7]) shows the resulting confusion matrix and accuracy score.

```
iris['target_names'][prediction]
out[7]: array(['setosa'], dtype='<U10')

In [9]: from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import classification_report
        y_pred=knn.predict(X_test)
        cm=confusion_matrix(y_test,y_pred)
        print(cm)
        print(" correct prediciton",accuracy_score(y_test,y_pred))
        print(" wrong prediction",(1-accuracy_score(y_test,y_pred)))
[[19  0  0]
 [ 0 15  0]
 [ 0  1 15]]
correct prediciton 0.98
wrong prediction 0.02000000000000018
```

## RESULT:

Thus, the implement k-Nearest Neighbour algorithm to classify the iris data set and print both correct and wrong predictions.

## **INTERNAL - 2**

### **19. KMEANS CLUSTERING FOR IRIS DATASET USING R**

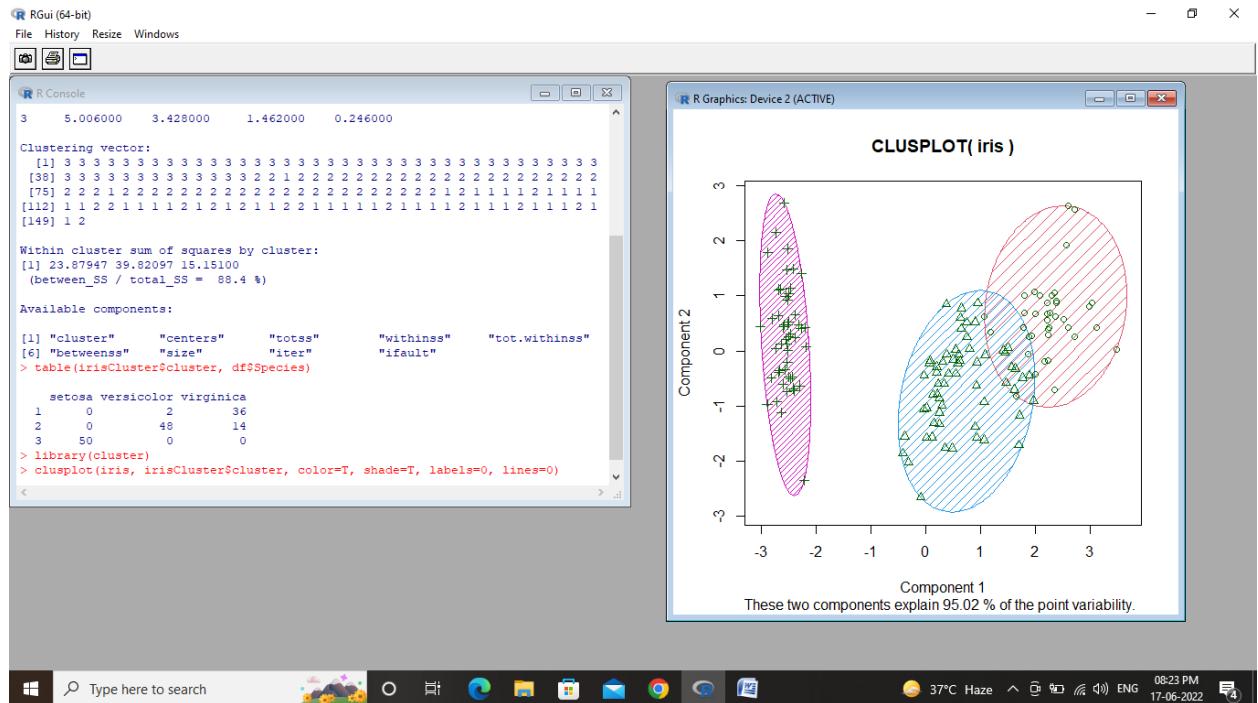
#### **AIM:**

To find KMEANS Clustering for IRIS dataset using R

#### **PROCEDURE:**

```
library(ggplot2)
df <- iris
head(iris)
ggplot(df, aes(Petal.Length, Petal.Width)) + geom_point(aes(col=Species), size=4)
set.seed(101)
irisCluster <- kmeans(df[,1:4], center=3, nstart=20)
irisCluster
table(irisCluster$cluster, df$Species)
library(cluster)
clusplot(iris, irisCluster$cluster, color=T, shade=T, labels=0, lines=0)
```

## **OUTPUT SCREENSHOT:**



## **RESULT:**

Thus, the implementation of KMEANS Clustering for IRIS dataset is shown.

## 20. Outliner Detection using K Means Classification algorithm on IRIS dataset using R

### AIM:

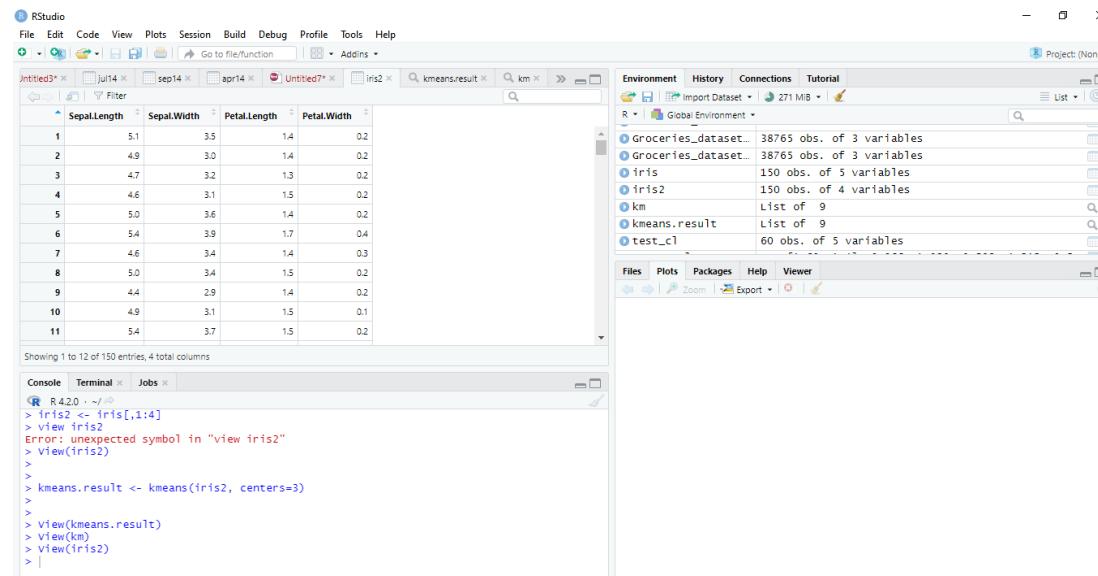
To apply Outliner Detection using K Means Classification algorithm on IRIS dataset using R

### Procedure:

```
iris2 <- iris[,1:4]
```

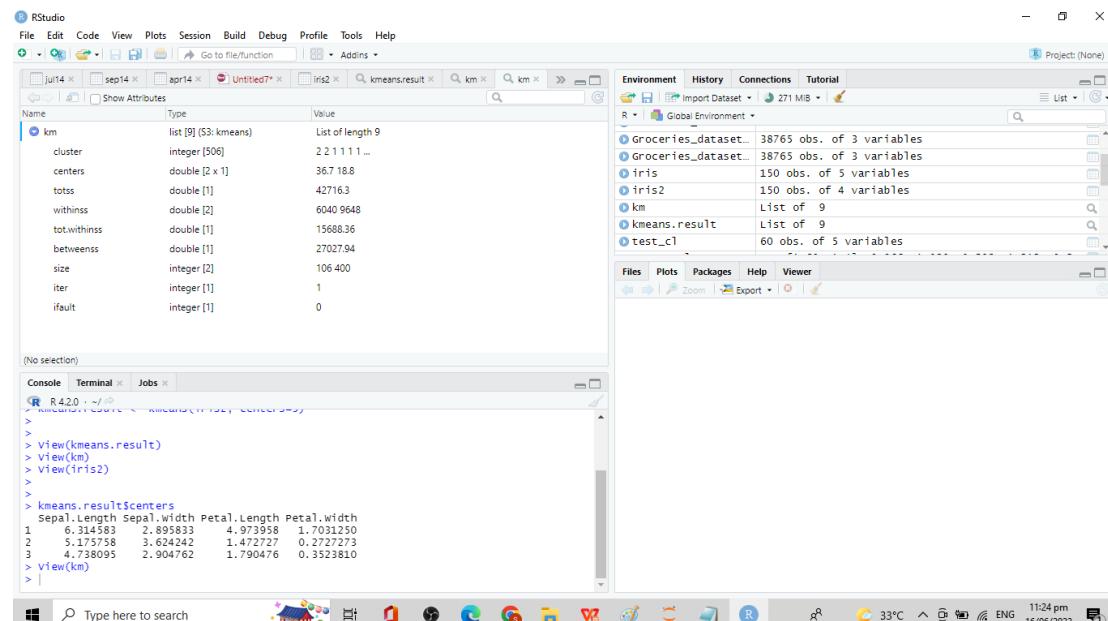
```
kmeans.result <- kmeans(iris2, centers=3)
```

```
kmeans.result$centers
```



The screenshot shows the RStudio interface. The Global Environment tab is selected, displaying various objects: Groceries\_dataset, Groceries\_dataset, iris, iris2, km, kmeans.result, and test\_c1. The 'iris2' dataset is a data frame with 150 rows and 4 columns. The 'kmeans.result' object is a list of length 9. The Console tab shows the R session code used to load the data and run the k-means algorithm.

```
R 4.2.0 --> 
> iris2 <- iris[,1:4]
> view iris2
Error: unexpected symbol in "view iris2"
> View(iris2)
>
> kmeans.result <- kmeans(iris2, centers=3)
>
> View(kmeans.result)
> View(km)
> View(iris2)
> |
```



The screenshot shows the RStudio interface. The Global Environment tab is selected, displaying various objects. The 'km' object is expanded, showing its components: cluster, centers, totss, withinss, tot.withinss, betweenss, size, iter, and ifault. The 'kmeans.result' object is a list of length 9. The Console tab shows the R session code used to run the k-means algorithm and print the centers.

```
R 4.2.0 --> 
> kmeans(result)
>
> View(kmeans.result)
> View(km)
> View(iris2)
>
> kmeans.result$centers
Sepal.Length Sepal.Width Petal.Length Petal.Width
1   6.314583   2.895833   4.973958   1.7031250
2   5.175758   3.624242   1.472727   0.2727273
3   4.738095   2.904762   1.790476   0.3523810
> View(km)
> |
```

kmeans.result\$cluster

The screenshot shows the RStudio interface with the following components:

- File Explorer**: Shows files like `sep14.R`, `april4.R`, `Untitled.R`, `iris2.R`, `kmeans.result`, and `km`.
- Environment View**: Displays the `kmeans.result` object as a list with elements: `cluster`, `centers`, `totss`, `withinss`, `totwithnss`, `betweenss`, `size`, `iter`, and `ifault`. It also lists other objects: `Groceries_dataset..`, `iris`, `iris2`, `km`, `kmeans.result`, and `test_c1`.
- Console View**: Shows the R command history, including the loading of the `iris` dataset and running the `kmeans` function.
- Plots**: A small preview of a scatter plot is visible in the bottom right corner.

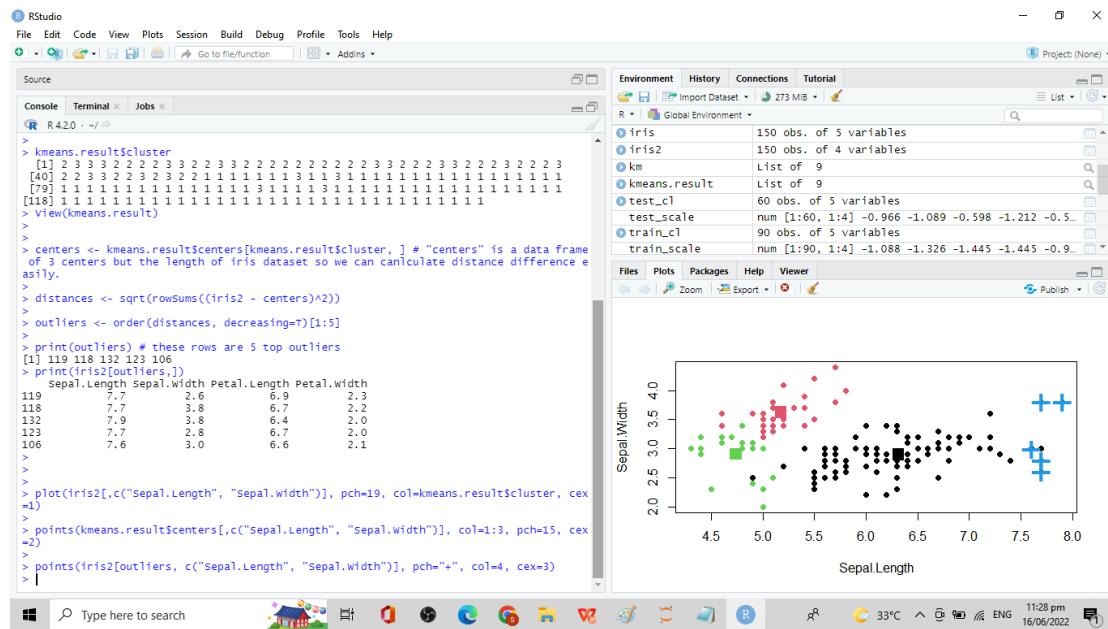
The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Source Editor:** Displays R code for k-means clustering on the Iris dataset. The code includes loading the dataset, performing k-means, and calculating distances and outliers.
- Environment View:** Shows the global environment with objects like iris, iris2, km, kmeans.result, test\_c1, train\_c1, and various scale matrices.
- Console Output:** Shows the R session output, including the k-means results and the top 5 outliers.
- Status Bar:** Shows the date (16/06/2022), time (11:27 pm), temperature (33°C), and battery level (ENG).

```
plot(iris2[,c("Sepal.Length", "Sepal.Width")], pch=19, col=kmeans.result$cluster, cex=1)
```

```
points(kmeans.result$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3, pch=15, cex=2)
```

```
points(iris2[outliers, c("Sepal.Length", "Sepal.Width")], pch= "+", col=4, cex=3)
```



## RESULT:

Thus the implementation of KMeans Classification algorithm for IRIS dataset using R

## 21. KNN Algorithm in BodyFat dataset using R.

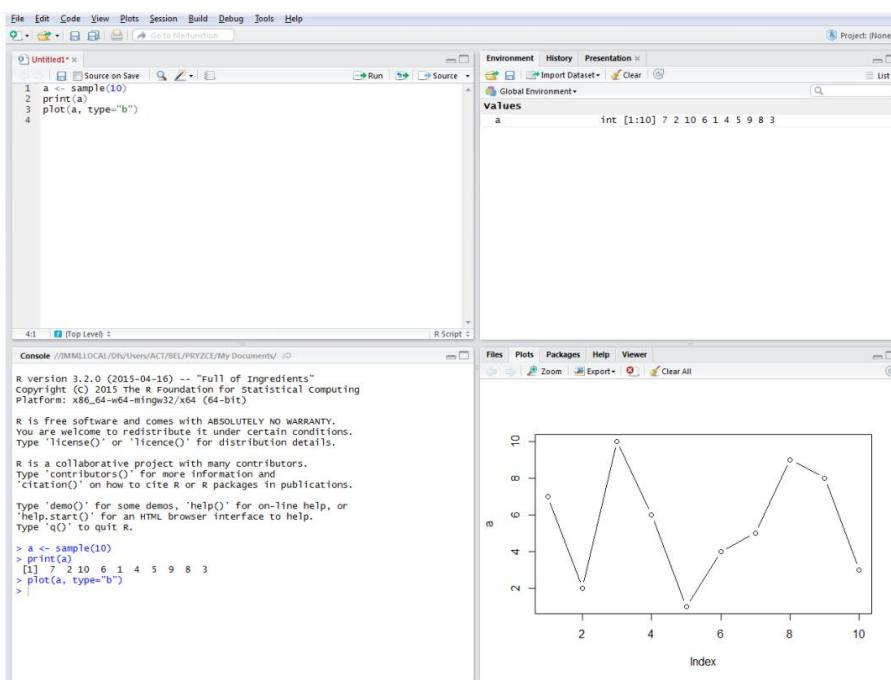
**Solution:**

### Introduction to RStudio

RStudio 10 is an integrated development environment (IDE) for R and can run on various operating systems like Windows, Mac OS X and Linux. It is a very useful and powerful tool for R programming, and therefore, readers are suggested to use RStudio when learning from this book or doing their projects, although all the provided code can run without it. What you normally need is RStudio Desktop open-source edition, which is free of charge.

There are four panels:

- Source panel (top left), which shows your R source code. If you cannot see the source panel, you can find it by clicking menu “File”, “New File” and then “R Script”. You can run a line or a selection of R code by clicking the “Run” bottom on top of source panel, or pressing “Ctrl + Enter”.
- Console panel (bottom left), which shows outputs and system messages displayed in a normal R console;
- Environment/History/Presentation panel (top right), whose three tabs show respectively all objects and function loaded in R, a history of submitted R code, and Presentations generated with R;



It is always a good practice to begin R programming with an RStudio project, which is a folder where to put your R code, data files and figures. To create a new project, click the

“Project” button at the top-right corner and then choose “New Project”. After that, select “create project from new directory” and then “Empty Project”. After typing a directory name, which will also be your project name, click “Create Project” to create your project folder and files. If you open an existing project, RStudio will automatically set the working directory to the project directory, which is very convenient. After that, create three folders as below:

- code, where to put your R source code;
- data, where to put your datasets; and
- figures, where to put produced diagrams.

In addition to above three folders which are useful to most projects, depending on your project and preference, you may create additional folders below:

- rawdata, where to put all raw data,
- models, where to put all produced analytics models, and
- reports, where to put your analysis reports.

## Dataset

The BodyFat data is used in this practical for demonstration of KNN Algorithm with R.

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
1	1.0708	12.3	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
2	1.0653	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
3	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
4	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	16.2
5	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7
6	1.0502	20.9	24	210.25	74.75	39.0	104.5	94.4	107.8	66.0	42.0	25.6	35.7	30.6	18.8
7	1.0549	19.2	26	181.00	69.75	36.4	105.1	90.7	100.3	58.4	38.3	22.9	31.9	27.8	17.7
8	1.0704	12.4	25	176.00	72.50	37.8	99.6	88.5	97.1	60.0	39.4	23.2	30.5	29.0	18.8
9	1.0900	4.1	25	191.00	74.00	38.1	100.9	82.5	99.9	62.9	38.3	23.8	35.9	31.1	18.2
10	1.0722	11.7	23	198.25	73.50	42.1	99.6	88.6	104.1	63.1	41.7	25.0	35.6	30.0	19.2
11	1.0830	7.1	26	186.25	74.50	38.5	101.5	83.6	98.2	59.7	39.7	25.2	32.8	29.4	18.5
12	1.0812	7.8	27	216.00	76.00	39.4	103.6	90.9	107.7	66.2	39.2	25.9	37.2	30.2	19.0
13	1.0513	20.8	32	180.50	69.50	38.4	102.0	91.6	103.9	63.4	38.3	21.5	32.5	28.6	17.7
14	1.0505	21.2	30	205.25	71.25	39.4	104.1	101.8	108.6	66.0	41.5	23.7	36.9	31.6	18.8
15	1.0484	22.1	35	187.75	69.50	40.5	101.3	96.4	100.1	69.0	39.0	23.1	36.1	30.5	18.2
16	1.0512	20.9	35	162.75	66.00	36.4	99.1	92.8	99.2	63.1	38.7	21.7	31.1	26.4	16.9
17	1.0333	29.0	34	195.75	71.00	38.9	101.9	96.4	105.2	64.8	40.8	23.1	36.2	30.8	17.3
18	1.0468	22.9	32	209.25	71.00	42.1	107.6	97.5	107.0	66.9	40.0	24.4	38.2	31.6	19.3
19	1.0622	16.0	28	183.75	67.75	38.0	106.8	89.6	102.4	64.2	38.7	22.9	37.2	30.5	18.5
20	1.0610	16.5	33	211.75	73.50	40.0	106.2	100.5	109.0	65.8	40.6	24.0	37.1	30.1	18.2
21	1.0551	19.1	28	179.00	68.00	39.1	103.3	95.9	104.9	63.5	38.0	22.1	32.5	30.3	18.4
22	1.0640	15.2	28	200.50	69.75	41.3	111.4	98.8	104.8	63.4	40.6	24.6	33.0	32.8	19.9
23	1.0631	15.6	31	140.25	68.25	33.9	86.0	76.4	94.6	57.4	35.3	22.2	27.9	25.9	16.7
24	1.0584	17.7	32	148.75	70.00	35.5	86.7	80.0	93.4	54.9	36.2	22.1	29.8	26.7	17.1
25	1.0668	14.0	28	151.25	67.75	34.5	90.2	76.3	95.8	58.4	35.5	22.9	31.1	28.0	17.6
26	1.0911	3.7	27	159.25	71.50	35.7	89.6	79.7	96.5	55.0	36.7	22.5	29.9	28.2	17.7
27	1.0811	7.9	34	131.50	67.50	36.2	88.6	74.6	85.3	51.7	34.7	21.4	28.7	27.0	16.5
28	1.0468	22.9	31	148.00	67.50	38.8	97.4	88.7	94.7	57.5	36.0	21.0	29.2	26.6	17.0
29	1.0910	3.7	27	133.25	64.75	36.4	93.5	73.9	88.5	50.1	34.5	21.3	30.5	27.9	17.2
30	1.0790	6.6	29	160.75	69.00	36.7	97.4	83.5	98.7	58.9	35.3	22.6	30.1	26.7	17.6
31	1.0716	11.9	32	182.00	73.75	38.7	100.5	88.7	99.8	57.5	38.7	33.9	32.5	27.7	18.4
32	1.0862	5.7	29	160.25	71.25	37.3	93.5	84.5	100.6	58.5	38.8	21.5	30.1	26.4	17.9
33	1.0719	11.8	27	168.00	71.25	38.1	93.0	79.1	94.5	57.3	36.2	24.5	29.0	30.0	18.8

Showing 1 to 34 of 252 entries, 15 total columns

## PROGRAM & OUTPUT SCREENSHOT

The screenshot shows the RStudio interface with the code editor and console panes.

**Code Editor:**

```
1 library(caret)
2 library(e1071)
3 library(catools)
4 library(class)
5 data <- read.csv("C:/Users/sunil/Desktop/DM Lab/bodyfat.csv")
6 split <- sample.split(data, splitRatio = 0.7)
7 train_cl <- subset(data, split == "TRUE")
8 test_cl <- subset(data, split == "FALSE")
9 train_scale <- scale(train_cl[, 1:4])
10 test_scale <- scale(test_cl[, 1:4])
11 classifier_knn <- knn(train = train_scale,
12                           test = test_scale,
13                           cl = train_cl$BodyFat,
14                           k = 1)
15 classifier_knn
16 cm <- table(test_cl$BodyFat, classifier_knn)
17 cm
18 classifier_knn <- knn(train = train_scale,
19                           test = test_scale,
20                           cl = train_cl$BodyFat,
21                           k = 7)
22 accuracy <- mean(classifier_knn != test_cl$BodyFat)
23 print(paste('Accuracy = ', accuracy))
24
25
26
27
28
29
30
31
```

**Console Output:**

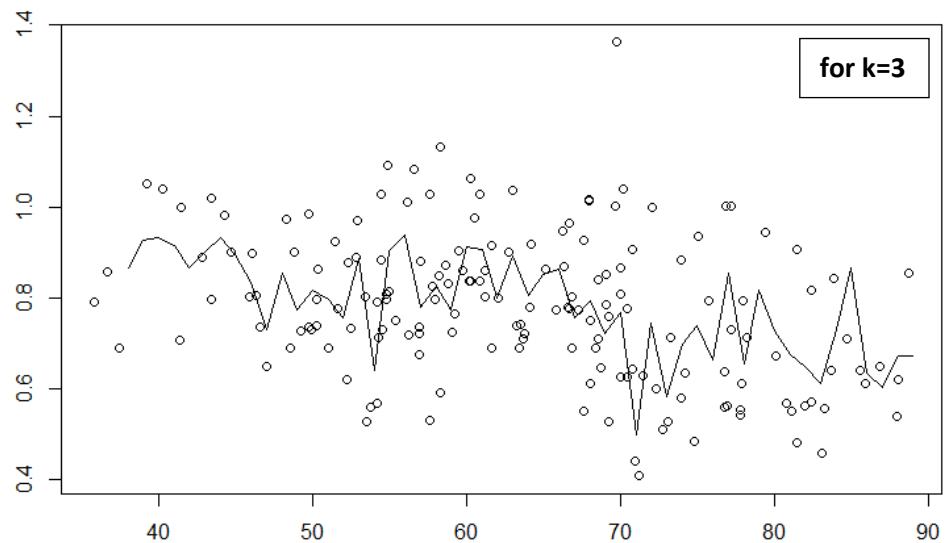
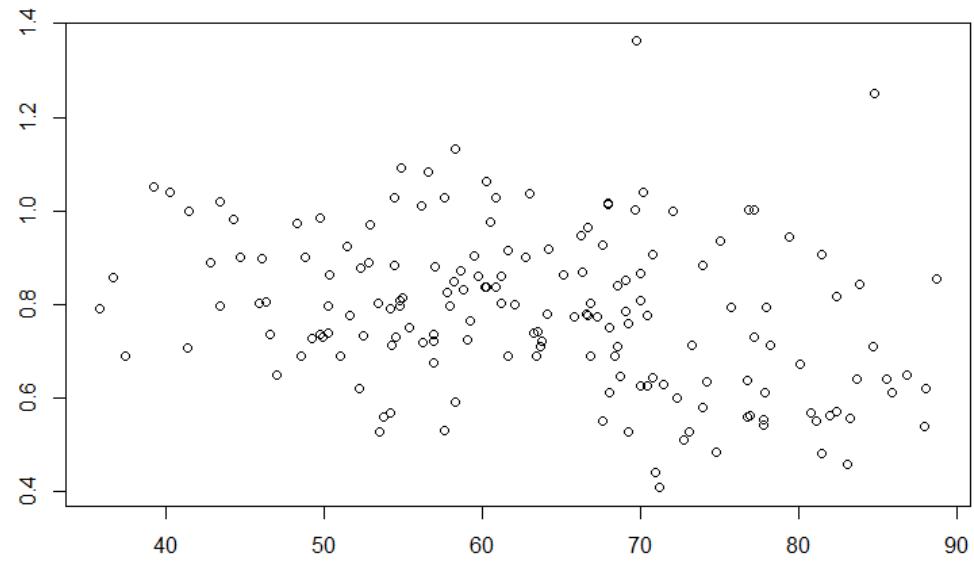
```
> data <- read.csv("C:/Users/sunil/Desktop/DM Lab/bodyfat.csv")
> split <- sample.split(data, splitRatio = 0.7)
> train_cl <- subset(data, split == "TRUE")
> test_cl <- subset(data, split == "FALSE")
> train_scale <- scale(train_cl[, 1:4])
> test_scale <- scale(test_cl[, 1:4])
> classifier_knn <- knn(train = train_scale,
+                           test = test_scale,
+                           cl = train_cl$BodyFat,
+                           k = 1)
> classifier_knn
[1] 21.8 19.1 10.4 20.5 21 19.2 14.6 12.5 20.9 3 8.8 20.5 32.3 27.2 32.9 9.9 7.7 4 29 32.3 30 23.3 12.4 8.8 12.5
[26] 18.8 26.6 27 24.3 6 7.5 6 21.2 19.7 21.2 18.7 27.1 25.4 10.9 25.3 12.1 25.4 13.9 21.2 23.3 27.1 17.5 10.4 25.2 3.7
[51] 9.4 29 20.5 11.9 15.6 19.2 0 26.6 32.3 20.8 10.8 21.3 25.3 17.3 21.3 18 22 32.9 32.9 7.5 18.1 12.4 13.8 8.8 6.3
[76] 22.6 15.4 24.3 26.1 11.8 32.6 31.9 26
132 Levels: 0 3 3.7 4 4.1 5.7 6 6.1 6.3 6.6 7.1 7.5 7.7 7.8 7.9 8 8.6 8.8 9 9.4 9.9 10 10.2 10.4 10.6 10.8 10.9 11 11.3 ... 47.5
>
```

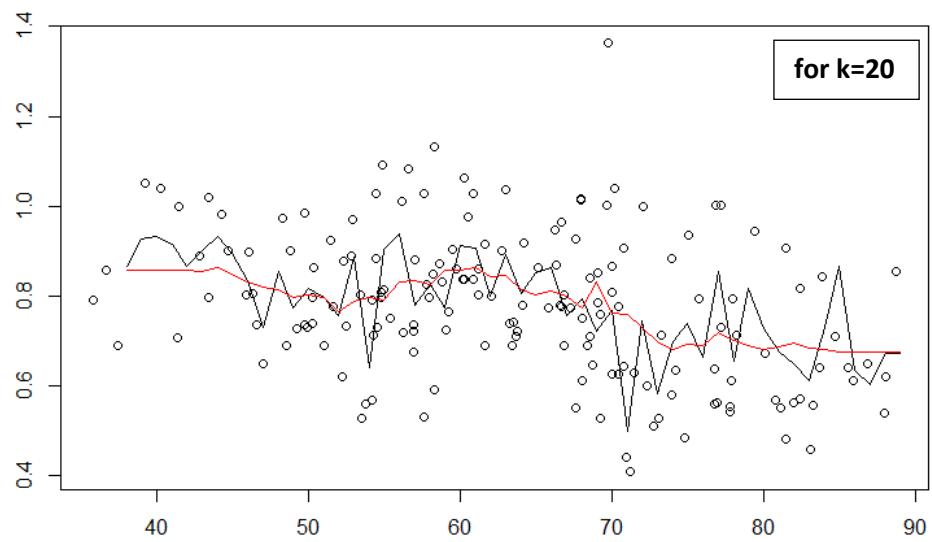
```

classifier_knn
 0 3 3.7 4 4.1 5.7 6 6.1 6.3 6.6 7.1 7.5 7.7 7.8 7.9 8 8.6 8.8 9 9.4 9.9 10 10.2 10.4 10.6 10.8 10.9 11 11.3 11.4 11.8 11.9
0.7 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.7 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.9 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.3 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  classifier_knn
 12.1 12.2 12.3 12.4 12.5 13.1 13.6 13.8 13.9 14.1 14.2 14.6 14.8 14.9 15.2 15.4 15.6 16 16.5 16.6 16.7 16.9 17 17.3 17.4
0.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  classifier_knn
 17.5 17.7 17.8 18 18.1 18.2 18.4 18.5 18.6 18.7 18.8 19.1 19.2 19.3 19.5 19.6 19.7 20.1 20.4 20.5 20.8 20.9 21 21.2 21.3
0.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  classifier_knn
 21.4 21.5 21.8 22 22.1 22.4 22.6 22.7 22.8 23.3 23.6 24.3 24.4 24.5 24.6 24.8 24.9 25.2 25.3 25.4 25.5 25.8 26 26.1 26.6
0.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  classifier_knn
 26.7 27 27.1 27.2 27.3 27.9 28.4 28.7 29 29.8 29.9 30 31.4 31.9 32.3 32.6 32.9 33.6 34.3 34.5 35 35.2 38.1 40.1 47.5
0.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[ reached getOption("max.print") -- omitted 65 rows ]
| [ reached getOption("max.print") -- omitted 65 rows ]
| [ reached getOption("max.print") -- omitted 65 rows ]
| classifier_knn
+ classifier_knn <- knn(train = train_scale,
+
+   test = test_scale,
+
+   cl = train_cl$bodyFat,
+
+   k = 7)
> accuracy<- mean(classifier_knn != test_cl$bodyFat)
> print(paste('Accuracy = ', accuracy))
[1] "Accuracy = 0.987951807228916"
>

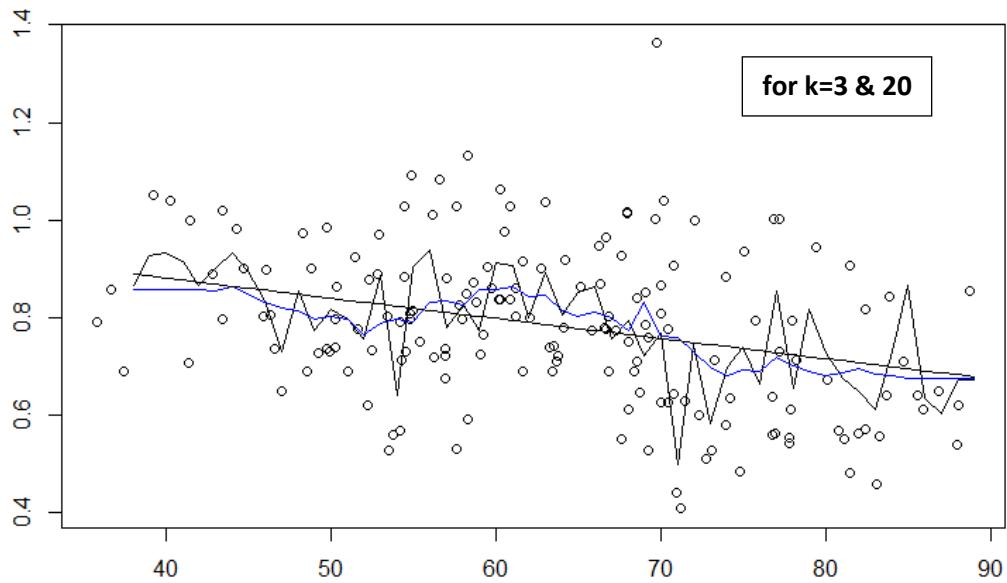
```

**Graph plot:** for  $k = 3$  & 20





### Linear Prediction:



### RESULT:

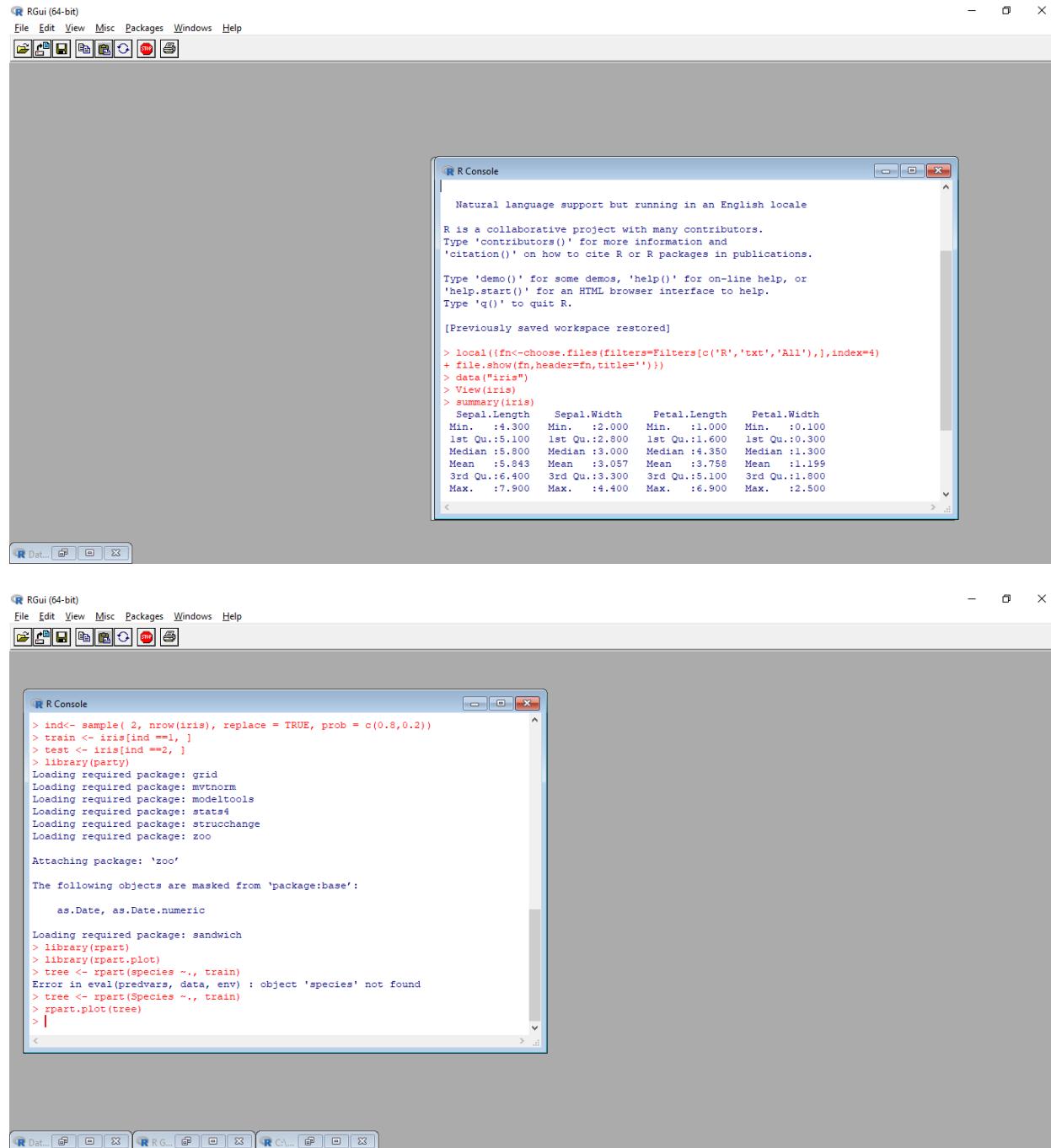
Thus, the implementation of KNN algorithm for bodyfat dataset using R is shown.

## 22. Generate a decision tree using ID3 algorithm for iris dataset using R studio

### AIM:

To find a decision tree using ID3 algorithm for iris dataset using R studio

### PROGRAM CODE AND OUTPUT Screenshot:



The screenshot shows the RStudio interface with two main windows: the R Console and the Data View.

**R Console Output:**

```
Natural language support but running in an English locale
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.

[Previously saved workspace restored]

> local((fn<-choose.files(filters=c('R','txt','All'),],index=4)
+ file.show(fn,header=fn$title==''))
> data("iris")
> View(iris)
> summary(iris)
   Sepal.Length   Sepal.Width    Petal.Length   Petal.Width
Min. :4.300   Min. :2.000   Min. :1.000   Min. :0.100
1st Qu.:5.100  1st Qu.:2.800  1st Qu.:1.600  1st Qu.:0.300
Median :5.800  Median :3.000  Median :4.350  Median :1.300
Mean   :5.843  Mean   :3.057  Mean   :1.758  Mean   :1.199
3rd Qu.:6.400  3rd Qu.:3.300  3rd Qu.:5.100  3rd Qu.:1.800
Max.   :7.900  Max.   :4.400  Max.   :4.900  Max.   :2.500
```

**Data View Output:**

```
ind<- sample( 2, nrow(iris), replace = TRUE, prob = c(0.8,0.2))
train <- iris[ind ==1, ]
test <- iris[ind ==2, ]
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
> library(sparty)
> library(sparty.plot)
> tree <- sparty(species ~., train)
Error in eval(predvars, data, env) : object 'species' not found
> tree <- sparty(Species ~., train)
> sparty.plot(tree)
> |
```

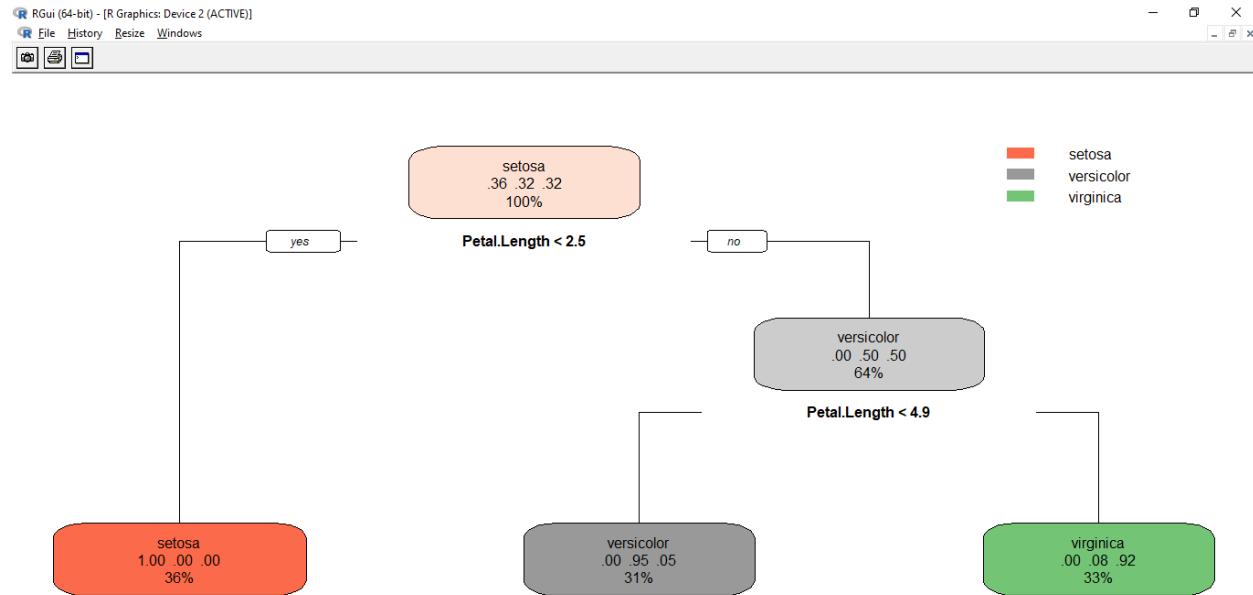
## Iris Dataset:

RGui (64-bit) - [Data: iris]

File

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.5	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	5.1	3.7	1.5	0.4	setosa
23	4.6	3.6	1.0	0.2	setosa
24	5.1	3.3	1.7	0.5	setosa
25	4.8	3.4	1.9	0.2	setosa
26	5.0	3.0	1.6	0.2	setosa
27	5.0	3.4	1.6	0.4	setosa
28	5.2	3.5	1.5	0.2	setosa
29	5.2	3.4	1.4	0.2	setosa
30	4.7	3.2	1.6	0.2	setosa
31	4.8	3.1	1.6	0.2	setosa

## Graphical representation of decision tree model:



## RESULT:

Thus, the implementation of decision tree using ID3 algorithm for iris dataset using R is shown.

## 23. Apply random forest algorithm for bodyfat dataset using R

### AIM:

To find random forest algorithm for bodyfat dataset using R studio

### PROCEDURE AND OUTPUT SCREENSHOT:

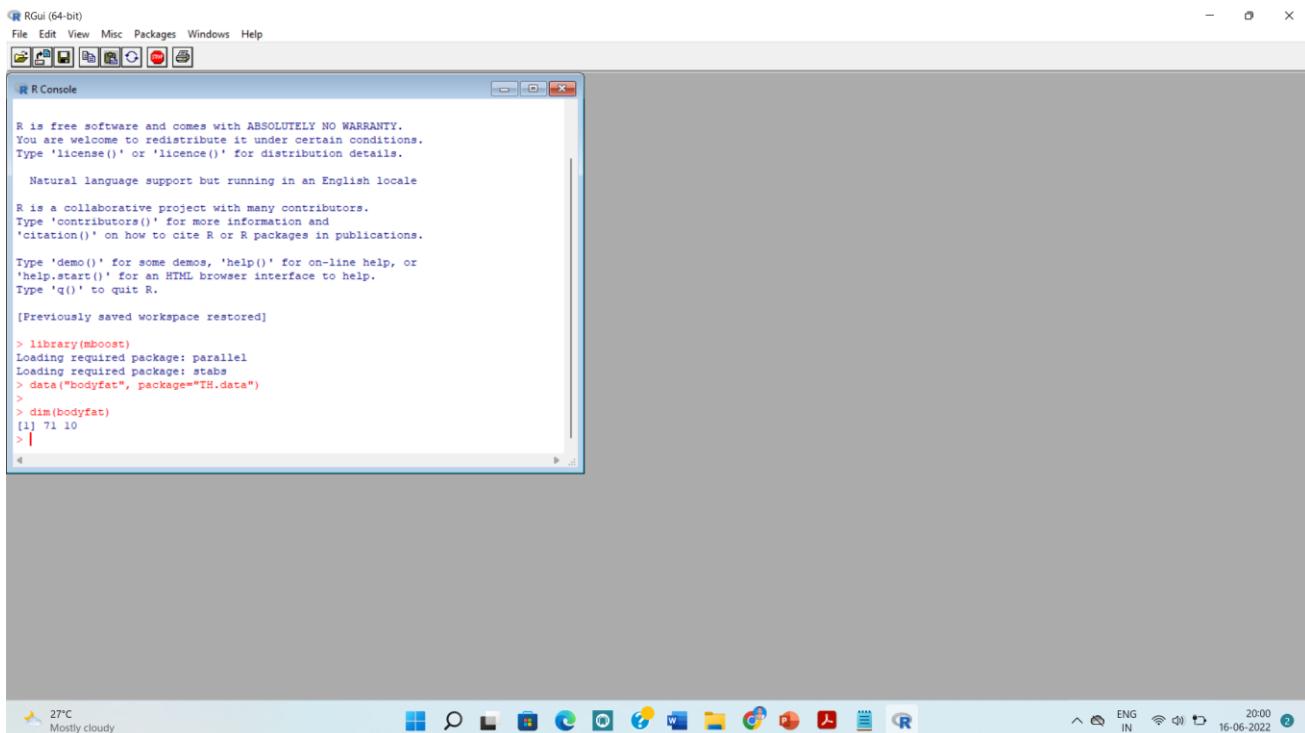
**STEP 1:** Open R and take new R Console

**STEP 2:** Type the following steps in R Console

```
library(mboost)
```

```
data("bodyfat", package="TH.data")
```

```
dim(bodyfat)
```



```
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.  
  
Natural language support but running in an English locale  
  
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.  
  
[Previously saved workspace restored]  
  
> library(mboost)  
Loading required package: parallel  
Loading required package: stats  
> data("bodyfat", package="TH.data")  
>  
> dim(bodyfat)  
[1] 71 10  
>
```

```
attributes(bodyfat)
```

```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> library(mbost)
Loading required package: parallel
Loading required package: stabs
> data("bodyfat", package="TH.data")
>
> dim(bodyfat)
[1] 71 10
> attributes(bodyfat)
$names
[1] "age"      "DEXfat"    "waistcirc"  "hipcirc"    "elbowbreadth"
[6] "kneebreadth" "anthro3a"   "anthro3b"   "anthro3c"   "anthro4"
$rownames
[1] "47"   "48"   "49"   "50"   "51"   "52"   "53"   "54"   "55"   "56"   "57"   "58"
[13] "59"   "60"   "61"   "62"   "63"   "64"   "65"   "66"   "67"   "68"   "69"   "70"
[25] "71"   "72"   "73"   "74"   "75"   "76"   "77"   "78"   "79"   "80"   "81"   "82"
[37] "83"   "84"   "85"   "86"   "87"   "88"   "89"   "90"   "91"   "92"   "93"   "94"
[49] "95"   "96"   "97"   "98"   "99"   "100"  "101"  "102"  "103"  "104"  "105"  "106"
[61] "107"  "108"  "109"  "110"  "111"  "112"  "113"  "114"  "115"  "116"  "117"
$class
[1] "data.frame"
> |
```

bodyfat[1:5,]

```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
$rownames
[1] "47"   "48"   "49"   "50"   "51"   "52"   "53"   "54"   "55"   "56"   "57"   "58"
[13] "59"   "60"   "61"   "62"   "63"   "64"   "65"   "66"   "67"   "68"   "69"   "70"
[25] "71"   "72"   "73"   "74"   "75"   "76"   "77"   "78"   "79"   "80"   "81"   "82"
[37] "83"   "84"   "85"   "86"   "87"   "88"   "89"   "90"   "91"   "92"   "93"   "94"
[49] "95"   "96"   "97"   "98"   "99"   "100"  "101"  "102"  "103"  "104"  "105"  "106"
[61] "107"  "108"  "109"  "110"  "111"  "112"  "113"  "114"  "115"  "116"  "117"
$class
[1] "data.frame"
> bodyfat[1:5,]
  age DEXfat waistcirc hipcirc elbowbreadth kneebreadth anthro3a anthro3b
47  57  41.68    100.0   112.0      7.1     9.4     4.42    4.95
48  65  43.29    99.5    116.5      6.5     8.9     4.63    5.01
49  59  35.41    96.0    108.5      6.2     8.9     4.12    4.74
50  58  22.79    72.0     96.5      6.1     9.2     4.03    4.48
51  60  36.42    89.5    100.5      7.1    10.0     4.24    4.68
  anthro3c anthro4
47     4.50    6.13
48     4.48    6.37
49     4.60    5.82
50     3.91    5.66
51     4.15    5.91
> |
```

set.seed(1234)

ind <- sample(2, nrow(bodyfat), replace=TRUE, prob=c(0.7, 0.3))

```

bodyfat.train <- bodyfat[ind==1,]

bodyfat.test <- bodyfat[ind==2,]

# train a decision tree

library(rpart)

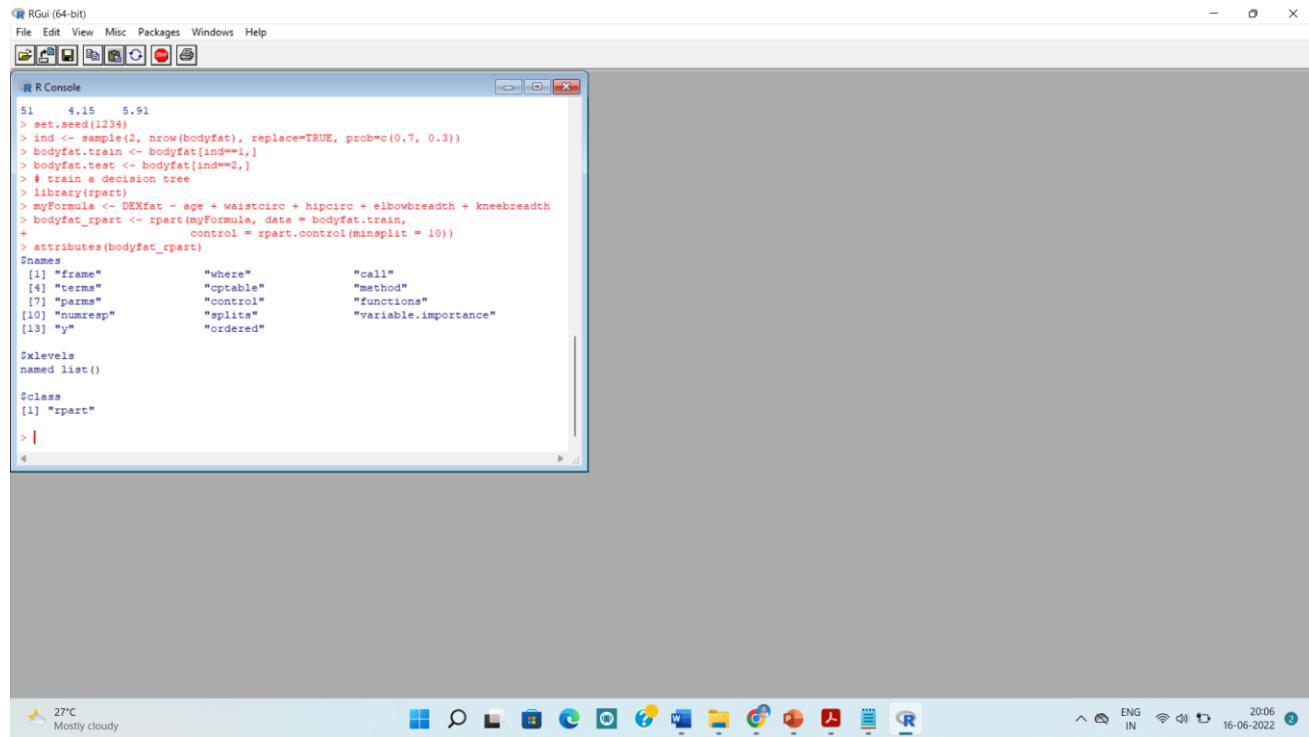
myFormula <- DEXfat ~ age + waistcirc + hipcirc + elbowbreadth + kneebreadth

bodyfat_rpart <- rpart(myFormula, data = bodyfat.train,

control = rpart.control(minsplit = 10))

attributes(bodyfat_rpart)

```



```
print(bodyfat_rpart$cptable)
```

```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> attributes(bodyfat_rpart)
@name
[1] "frame"          "where"           "call"
[4] "terms"          "optable"         "method"
[7] "parms"          "control"         "functions"
[10] "numresp"        "splits"          "variable.importance"
[13] "y"               "ordered"

$levels
named list()

$class
[1] "rpart"

> print(bodyfat_rpart)
    CP nsplit rel error   xerror      xstd
1 0.67272638 0 1.00000000 1.0427457 0.19016187
2 0.09390665 1 0.32727362 0.5081173 0.11702581
3 0.06037503 2 0.23336696 0.4522296 0.09801847
4 0.03420446 3 0.17299193 0.3967005 0.09676249
5 0.01708278 4 0.13878747 0.3015476 0.07385485
6 0.01695763 5 0.12170469 0.2929969 0.06850104
7 0.01007079 6 0.10474706 0.2713231 0.06650466
8 0.01000000 7 0.09467627 0.2713231 0.06650466
>

```

print(bodyfat\_rpart)

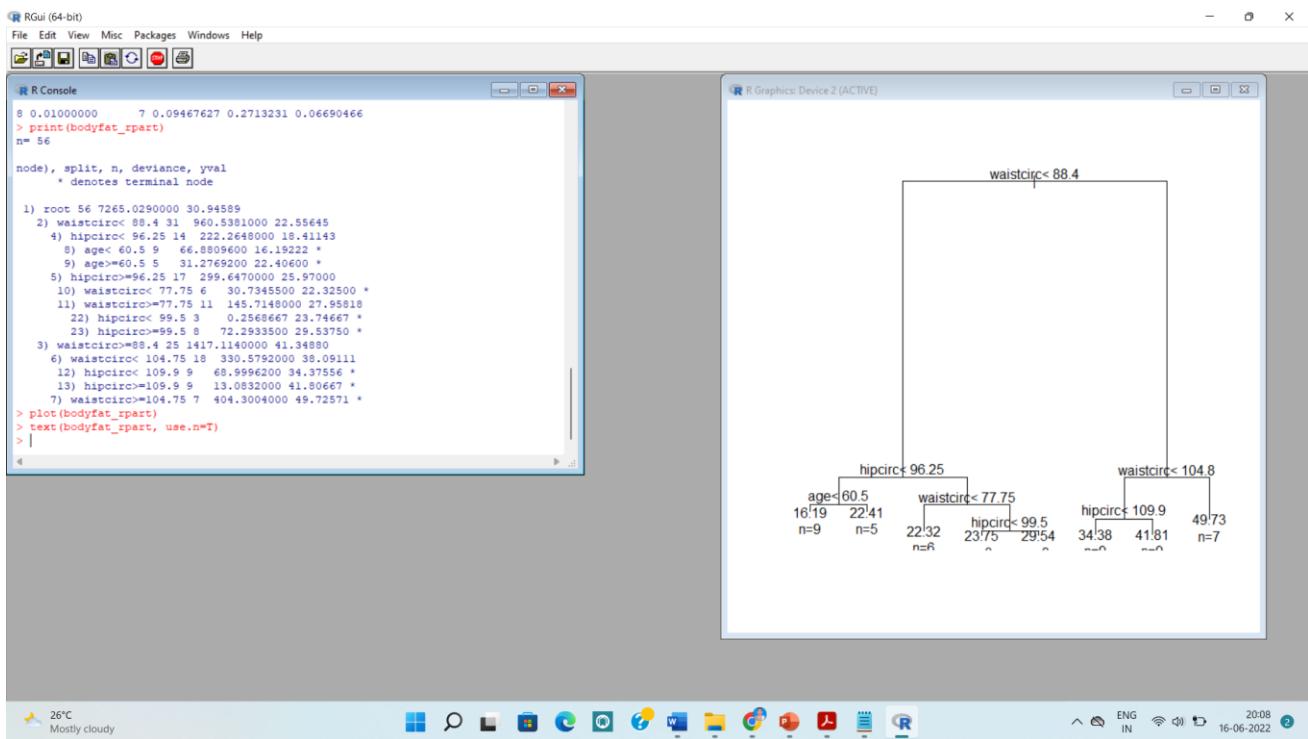
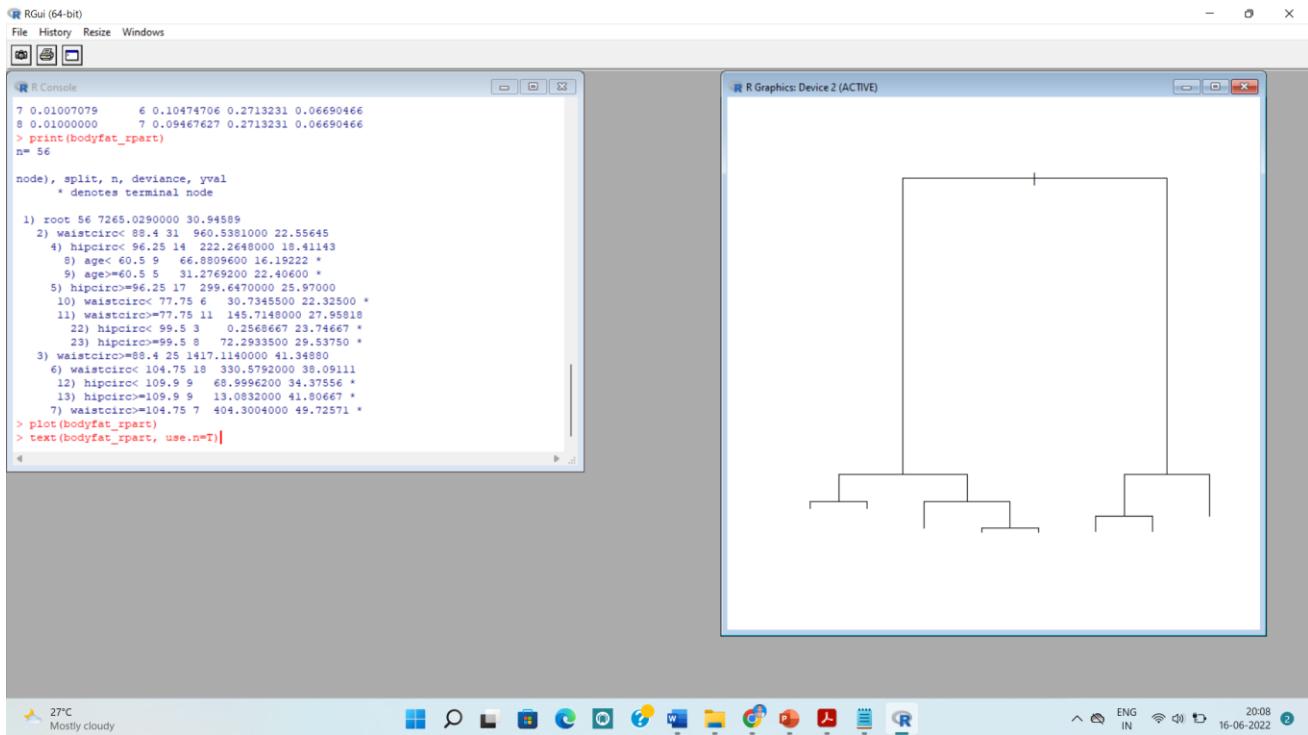
```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
6 0.01695763      5 0.12170469 0.2929969 0.06850104
7 0.01007079      6 0.10474706 0.2713231 0.06650466
8 0.01000000      7 0.09467627 0.2713231 0.06650466
> print(bodyfat_rpart)
n= 56

node), split, n, deviance, yval
 * denotes terminal node

1) root 56 7265.0290000 30.94588
  2) waistcirc< 88.4 31  960.5381000 22.55645
    4) hipcirc< 56.25 14  222.2648000 18.41143
      8) age<=60.5 9   66.8808600 16.19222 *
      9) age>=60.5 5   31.2769200 22.40600 *
    5) hipcirc>=56.25 17  289.6470000 25.97000
    10) waistcirc< 77.75 6   30.7345500 22.32500 *
     11) waistcirc<=77.75 11  145.7148000 27.95818
      22) hipcirc< 99.5 3   0.2565667 23.74667 *
      23) hipcirc>=99.5 8   72.2933500 29.53750 *
    3) waistcirc<=88.4 25 1417.1140000 41.34880
    6) waistcirc< 104.75 18  330.5752000 38.09111
    12) hipcirc< 105.9 9   68.8996200 34.37556 *
    13) hipcirc>=105.9 9   13.0832000 41.80667 *
    7) waistcirc>=104.75 7   404.3004000 45.72571 *
>
```

plot(bodyfat\_rpart)

text(bodyfat\_rpart, use.n=T)



```
opt <- which.min(bodyfat_rpart$cptable[, "xerror"])
```

```
cp <- bodyfat_rpart$cptable[opt, "CP"]
```

```
bodyfat_prune <- prune(bodyfat_rpart, cp = cp)
```

```
print(bodyfat_prune)
```

RGui (64-bit)  
File Edit View Misc Packages Windows Help

R Console

```
> plot(bodyfat_rpart)
> text(bodyfat_rpart, use.n=T)
> opt <- which.min(bodyfat_rpart$cpstable[, "xerror"])
> cp <- bodyfat_rpart$cpstable[opt, "CP"]
> bodyfat_prune <- prune(bodyfat_rpart, cp = cp)
> print(bodyfat_prune)
n= 56

node), split, n, deviance, yval
 * denotes terminal node

1) root 56 7265.02900 30.94589
  2) waistcirc< 88.4 31  960.53810 22.55645
    4) hipcirc< 96.25 14  222.26480 18.41143
      8) age< 60.5 9   66.88096 16.19222 *
      9) age>=60.5 5   31.27692 22.40600 *
    5) hipcirc>=96.25 17  259.64700 25.97000
    10) waistcirc< 77.75 6   30.73455 22.32500 *
    11) waistcirc>=77.75 11  145.71480 27.95818 *
  3) waistcirc>=88.4 25 1417.11400 41.34880
  6) waistcirc< 104.75 18  330.57920 38.09111
  12) hipcirc< 109.9 9   68.89962 34.37556 *
  13) hipcirc>=109.9 9   13.08320 41.80667 *
  ?) waistcirc>=104.75 7   404.30040 49.72571 *
```

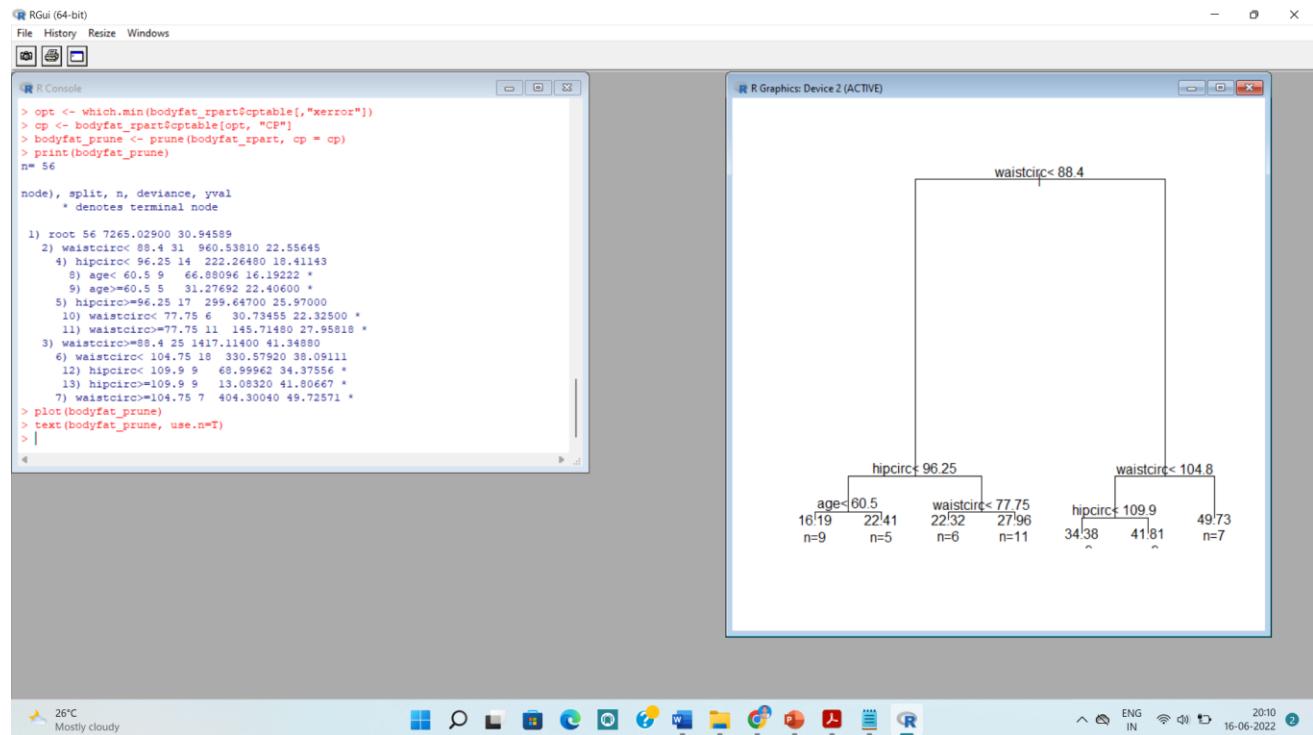
R Graphics...

26°C  
Mostly cloudy

ENG IN 20:09 16-06-2022

```
plot(bodyfat_prune)
```

```
text(bodyfat_prune, use.n=T)
```



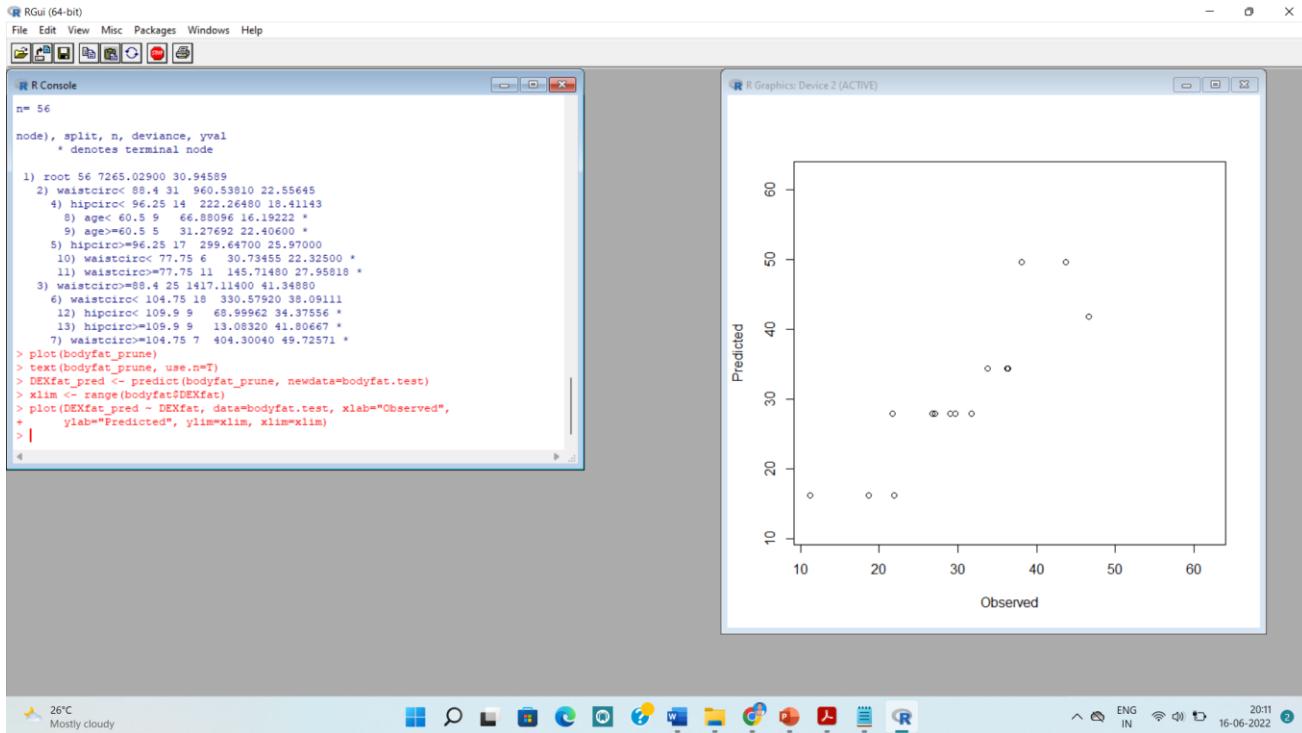
```

DEXfat_pred <- predict(bodyfat_prune, newdata=bodyfat.test)

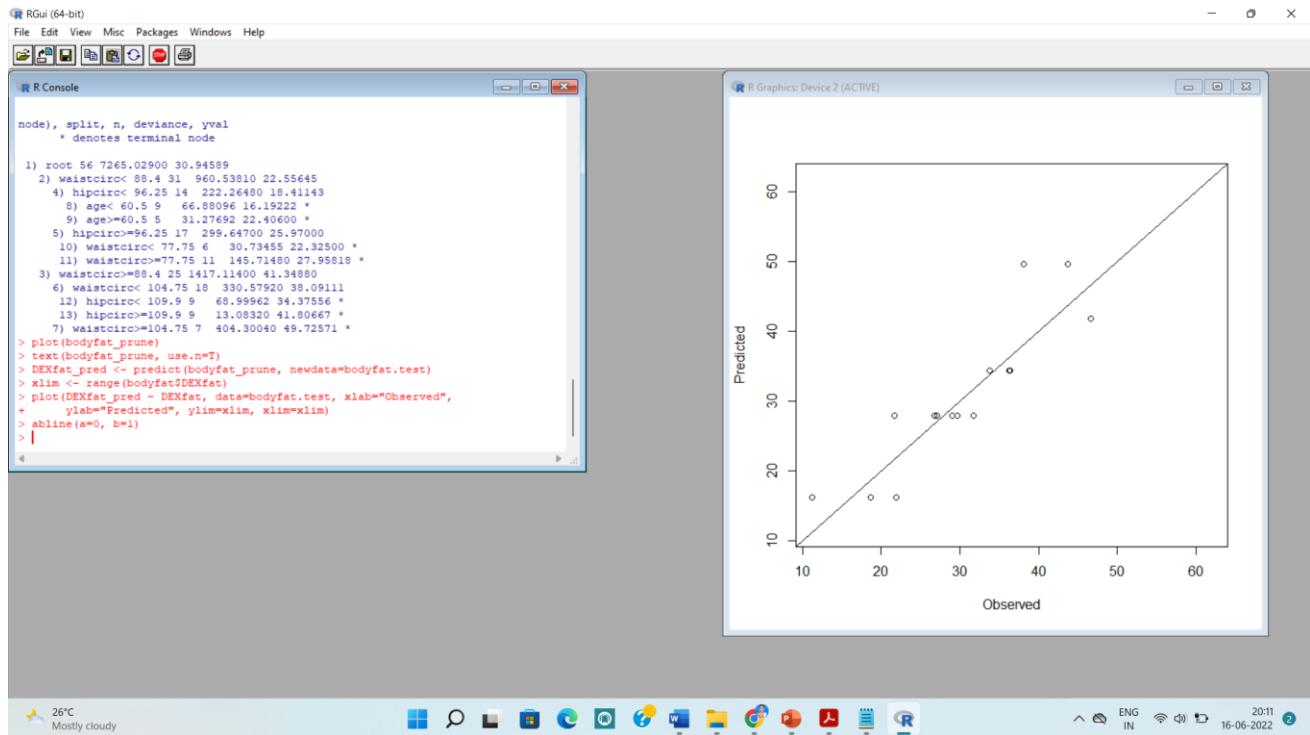
xlim <- range(bodyfat$DEXfat)

plot(DEXfat_pred ~ DEXfat, data=bodyfat.test, xlab="Observed",
     ylab="Predicted", ylim=xlim, xlim=xlim)

```



```
abline(a=0, b=1)
```



## RESULT:

Thus, the implementation of random forest algorithm for bodyfat dataset using R

## 24. Multiple Linear Regression using bodyfat dataset

### AIM:

To apply the Multiple Linear Regression algorithm on Bodyfat dataset using R studio.

### PROGRAM AND OUTPUT SCREENSHOT:

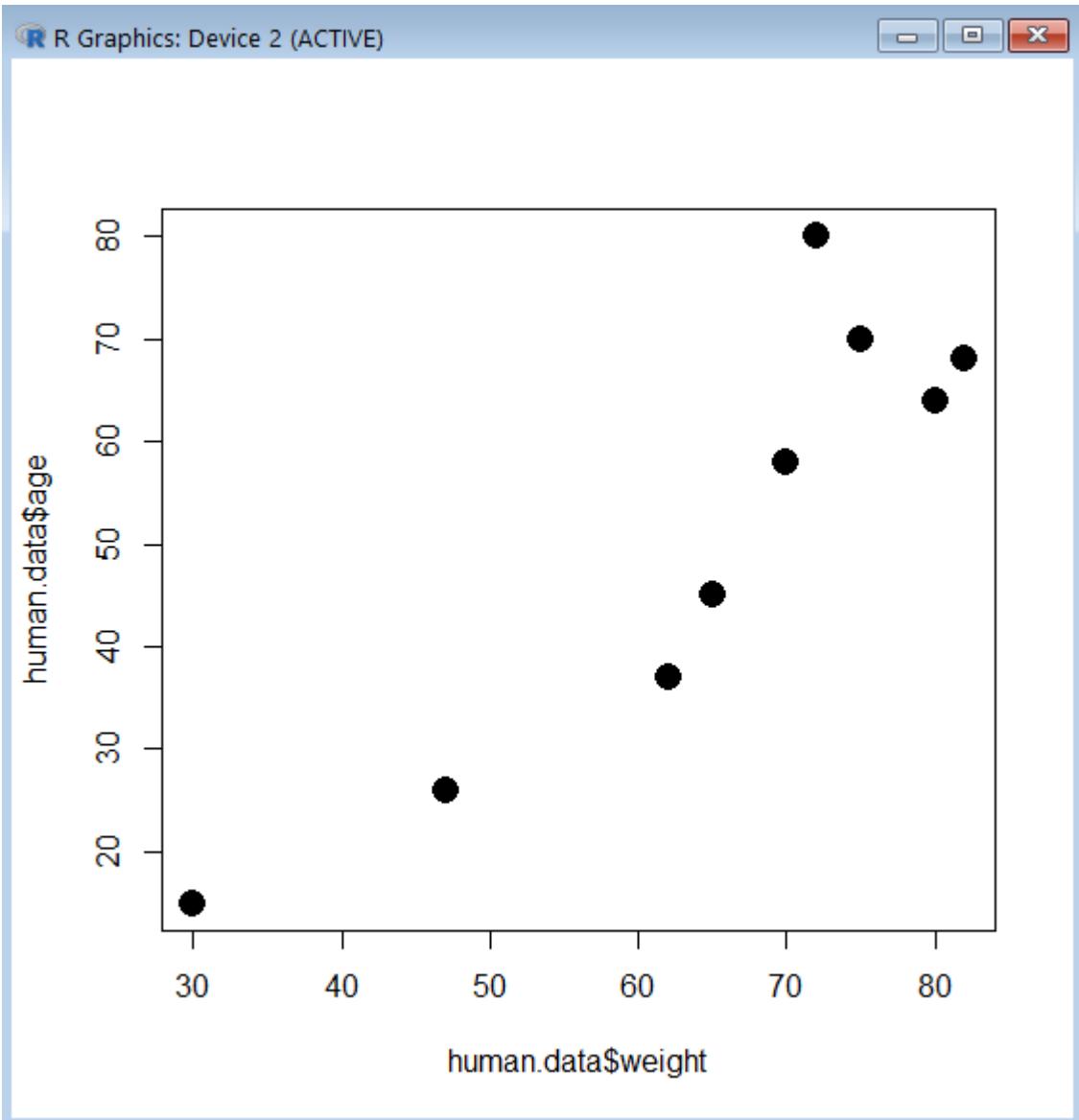
The screenshot shows the R Console window with the following text:

```
R Console

'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> human.data <- data.frame(
+   age = c(21,25,30,34,40,43,47,50,60),
+   weight = c(53,,56,62,65,70,80,82,75,72),
+   bodyfat = c(14.8,15.6,16.5,17.5,18.2,19.2,20.6,20.8,22.1))
Error in c(53, , 56, 62, 65, 70, 80, 82, 75, 72) : argument 2 is empty
> human.data
  age weight bodyfat
1   15     30    0.7
2   26     47    1.3
3   37     62    0.7
4   45     65    2.0
5   58     70    3.6
6   64     80    3.0
7   68     82    2.9
8   70     75    3.9
9   80     72    4.0
> plot(human.data$weight, human.data$age, pch=16, cex=2)
> |
```



R Console

```
> plot(human.data$weight, human.data$age, pch=16, cex=2)
> simple.regression <- lm(age ~ weight, data=human.data)
> summary(simple.regression)

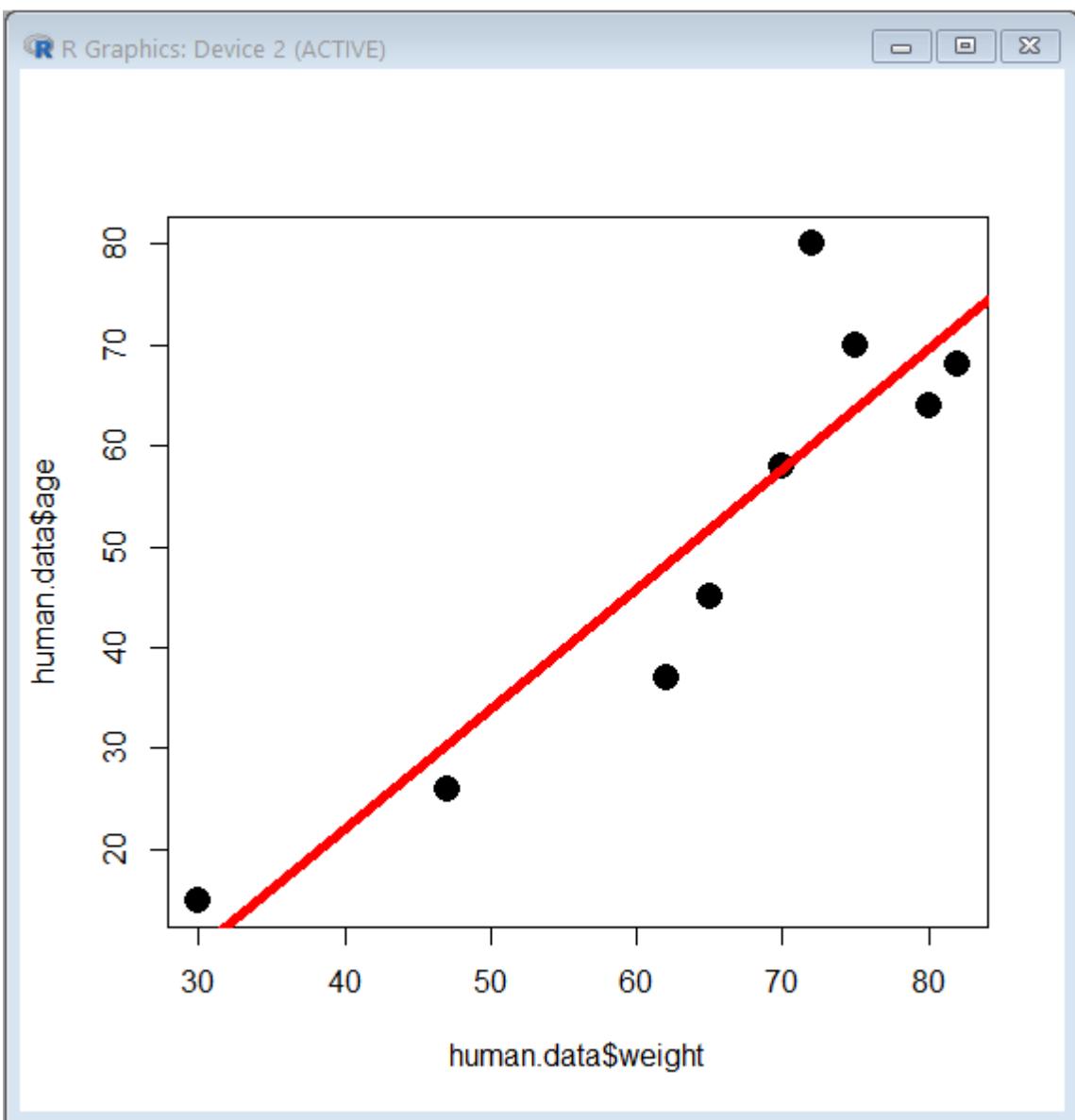
Call:
lm(formula = age ~ weight, data = human.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-11.148 -5.509 -3.883  4.828 19.985 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -25.4299    14.0472  -1.810 0.113162    
weight       1.1867     0.2107   5.633 0.000788 ***  
---
Signif. codes:  0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 1 

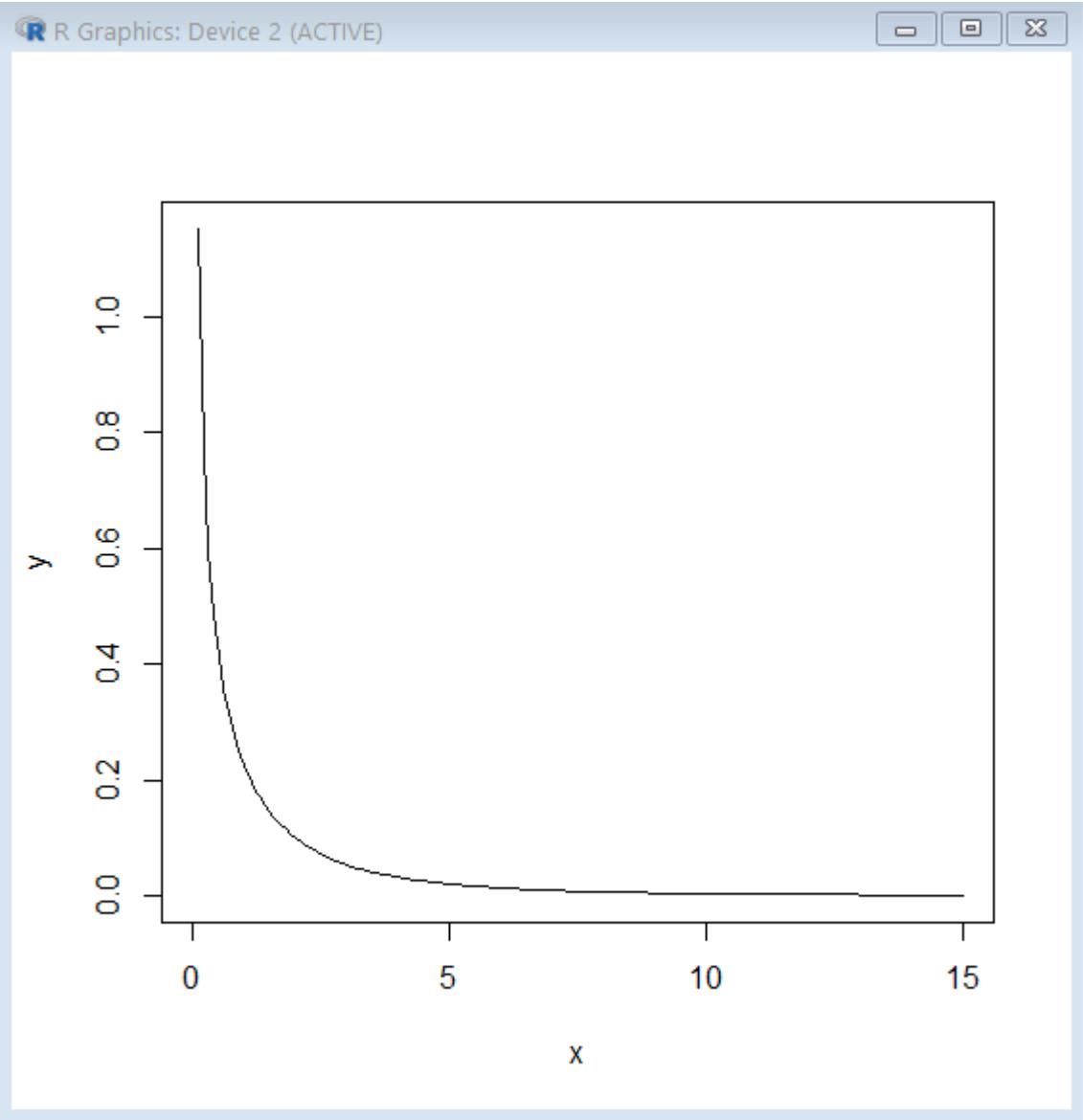
Residual standard error: 9.984 on 7 degrees of freedom
Multiple R-squared:  0.8193,    Adjusted R-squared:  0.7934 
F-statistic: 31.73 on 1 and 7 DF,  p-value: 0.0007884

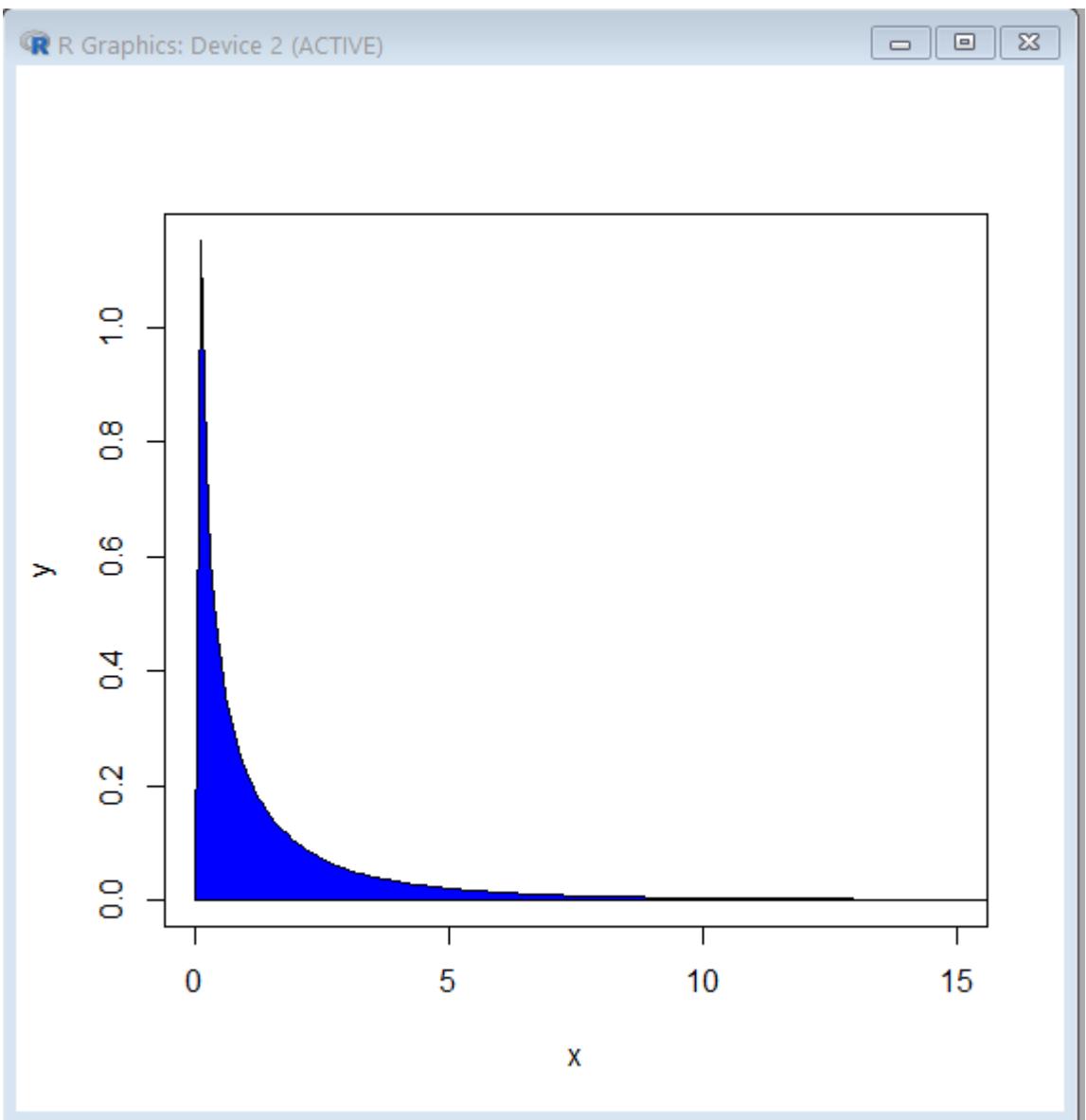
> |
```



R Console

```
Estimate Std. Error t value Pr(>|t|)  
(Intercept) -25.4299    14.0472  -1.810 0.113162  
weight       1.1867     0.2107   5.633 0.000788 ***  
---  
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 9.984 on 7 degrees of freedom  
Multiple R-squared:  0.8193,    Adjusted R-squared:  0.7934  
F-statistic: 31.73 on 1 and 7 DF,  p-value: 0.0007884  
  
> abline(simple.regression, lwd=5, col="red")  
> ss.mean <- sum(human.data$age - mean(human.data$age))^2  
> ss.simple <- sum(simple.regression$residuals^2)  
> (ss.mean - ss.simple) / ss.mean  
[1] 0.8192607  
> f.simple <- ((ss.mean - ss.simple) / (2 - 1)) /  
+   (ss.simple / (nrow(human.data) - 2))  
> f.simple  
[1] 31.72981  
> x <- seq(from=0, to=15, by=0.1)  
> y <- df(x, df1=1, df2=7)  
> plot(x, y, type="l")  
> |
```





R Console

```
[1] 0.8192607
> f.simple <- ((ss.mean - ss.simple) / (2 - 1)) /
+   (ss.simple / (nrow(human.data) - 2))
> f.simple
[1] 31.72981
> x <- seq(from=0, to=15, by=0.1)
> y <- df(x, df1=1, df2=7)
> plot(x, y, type="l")
> abline(v=f.simple, col="red")
> x.zero.to.line <- seq(from=0, to=f.simple, by=0.1)
> y.zero.to.line <- df(x.zero.to.line, df1=1, df2=7)
> polygon(x=c(x.zero.to.line, 0), y=c(y.zero.to.line, 0), col="blue")
> x.line.to.100 <- seq(from=f.simple, to=100, by=0.1)
> y.line.to.100 <- df(x.line.to.100, df1=1, df2=7)
> polygon(x=c(x.line.to.100, f.simple), y=c(y.line.to.100, 0), col="red")
> pf(f.simple, df1=1, df2=7)
[1] 0.9992116
> 1-pf(f.simple, df1=1, df2=7)
[1] 0.0007883748
> summary(simple.regression)

Call:
lm(formula = age ~ weight, data = human.data)
```

R Console

```
> summary(simple.regression)

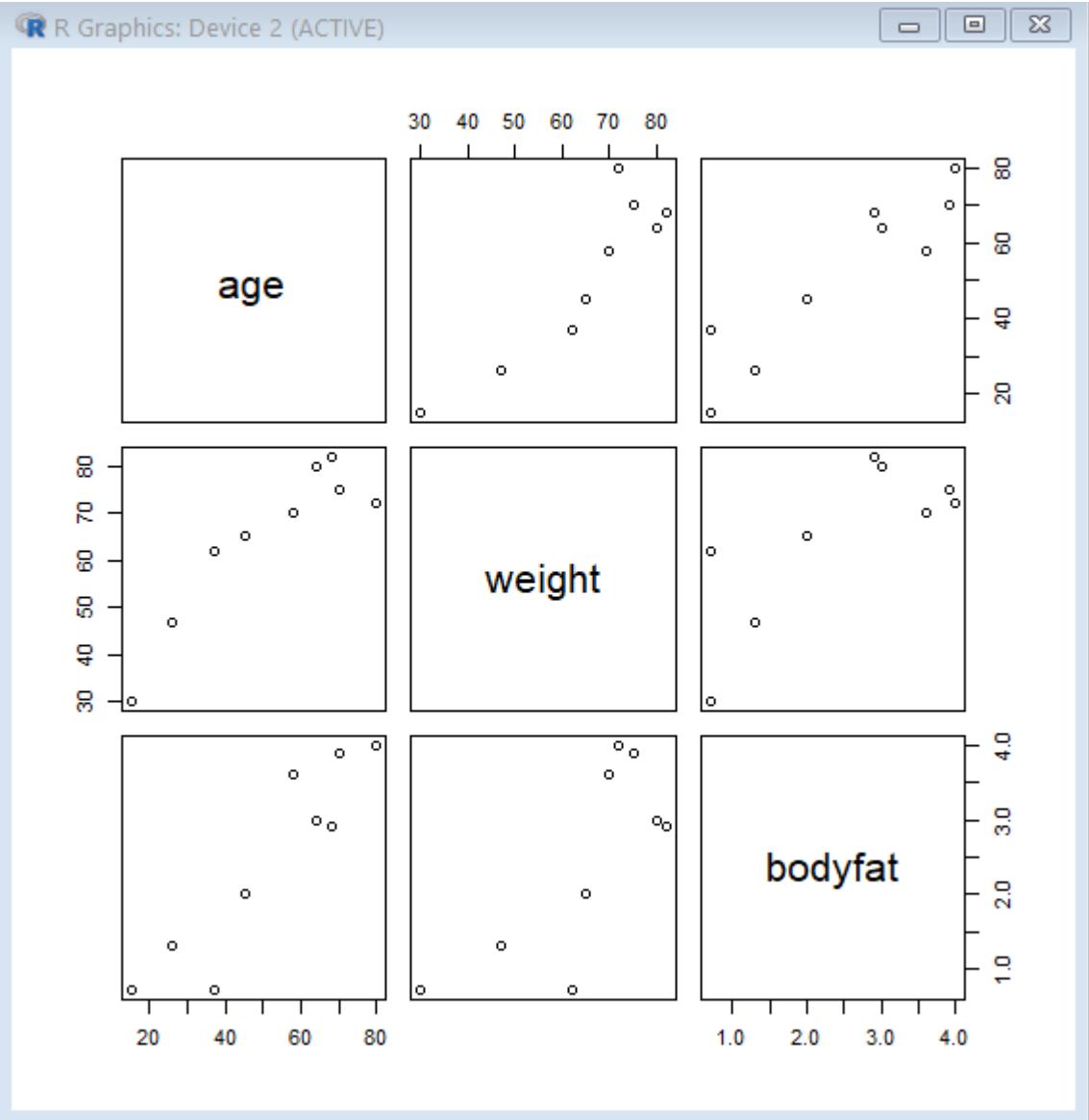
Call:
lm(formula = age ~ weight, data = human.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-11.148  -5.509  -3.883   4.828  19.985 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -25.4299    14.0472  -1.810 0.113162  
weight       1.1867     0.2107   5.633 0.000788 *** 
---
Signif. codes:  0 '****' 0.001 '***' 0.01 '**' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.984 on 7 degrees of freedom
Multiple R-squared:  0.8193,    Adjusted R-squared:  0.7934 
F-statistic: 31.73 on 1 and 7 DF,  p-value: 0.0007884

>
> plot(human.data)
> |
```

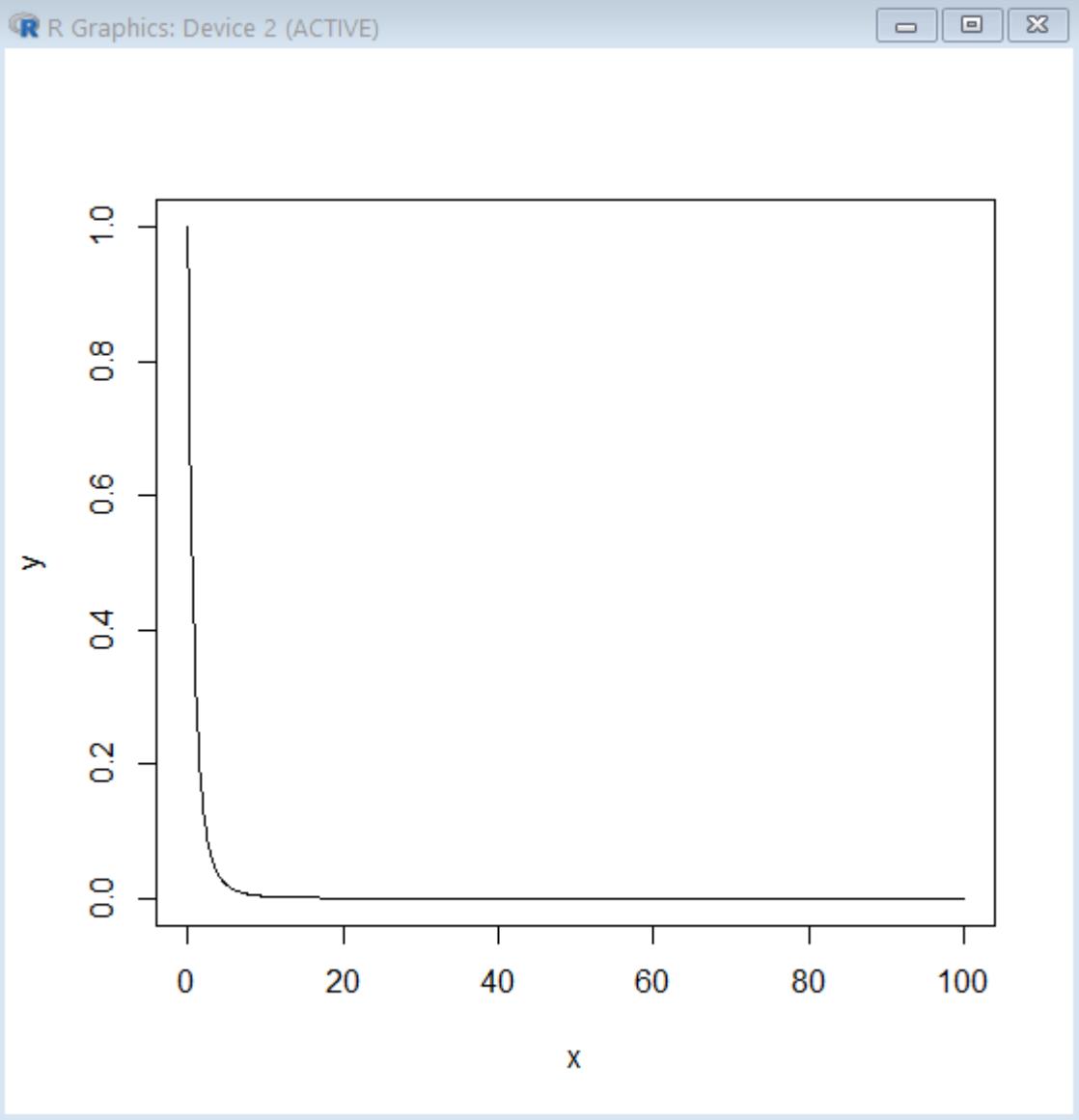


R Console

```
(Intercept) 0.58877 0.88709 0.664 0.5315
weight      -0.03379 0.02583 -1.308 0.2386
age         0.07884 0.01970 4.002 0.0071 **
---
Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5203 on 6 degrees of freedom
Multiple R-squared: 0.8838, Adjusted R-squared: 0.8451
F-statistic: 22.82 on 2 and 6 DF, p-value: 0.001568

> ss.multiple <- sum(multiple.regression$residuals^2)
> (ss.mean - ss.multiple) / ss.mean
[1] 0.9995792
> f.multiple <- ((ss.mean - ss.multiple) / (3 - 1)) /
+   (ss.multiple / (nrow(human.data) - 3))
>
> f.multiple
[1] 7126.215
>
> x <- seq(from=0, to=100, by=0.1)
> y <- df(x, df1=2, df2=6)
> plot(x, y, type="l")
> |
```



R Console

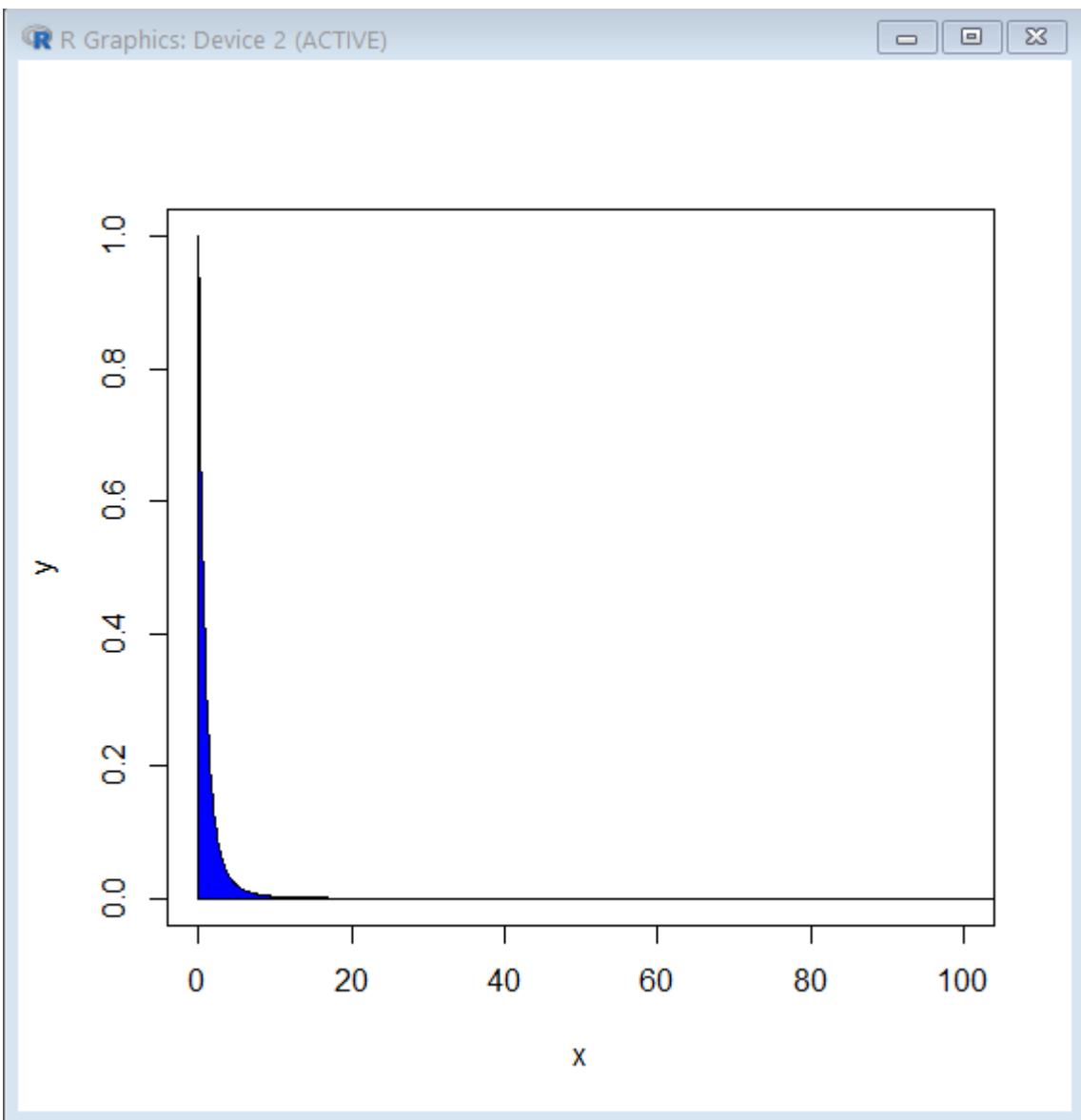
```
> 1-pf(f.simple.v.multiple, df1=1, df2=6)
[1] 3.947392e-09
> summary(multiple.regression)

Call:
lm(formula = bodyfat ~ weight + age, data = human.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.71066 -0.27878  0.06002  0.24967  0.80409 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 0.58877   0.88709   0.664   0.5315    
weight      -0.03379   0.02583  -1.308   0.2386    
age         0.07884   0.01970   4.002   0.0071 **  
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5203 on 6 degrees of freedom
Multiple R-squared:  0.8838,    Adjusted R-squared:  0.8451 
F-statistic: 22.82 on 2 and 6 DF,  p-value: 0.001568
```



**RESULT:**

Thus, the implementation of Multiple Linear Regression algorithm for bodyfat dataset using R is shown.