Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

RECORD NOTEBOOK

DATA ANALYTICS LAB USING
MACHINE LEARNING ALGORITHMS–(EBCS22L09)

2025-2026(ODD SEMESTER)

DEPARTMENT

Of

ARTIFICIAL INTELLIGENCE

NAME	**:**	**NUNE SOMA SEKHARA GUPTA**
REGISTER NO	**:**	**221211101107**
COURSE	**:**	**B.TECH AI**
YEAR/SEM/SEC	**:**	**IV / VII / B**

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

BONAFIDE CERTIFICATE

Register No : **221211101107**

Name of Lab : **DATA ANALYTICS LAB USING MACHINE LEARNING ALGORITHMS–(EBCS22L09)**

Department : **ARTIFICIAL INTELLIGENCE**

Certified that this is the bonafide record of work done by **NUNE SOMA SEKHARA GUPTA 221211101107** of IV Year B.Tech(AI), Sec-'B' in the **DATA ANALYTICS LAB USING MACHINE LEARNING ALGORITHMS** during the year 2025-2026.

Signature of Lab-in-Charge **Signature of Head of Dept**

Submitted for the Practical Examination held on ----------

Internal Examiner **External Examiner**

INDEX

EXP NO:01 **DATE:**

EXPLORATION OF INSTALLTION OF HADOOP

AIM: To install and setup Hadoop environment in windows

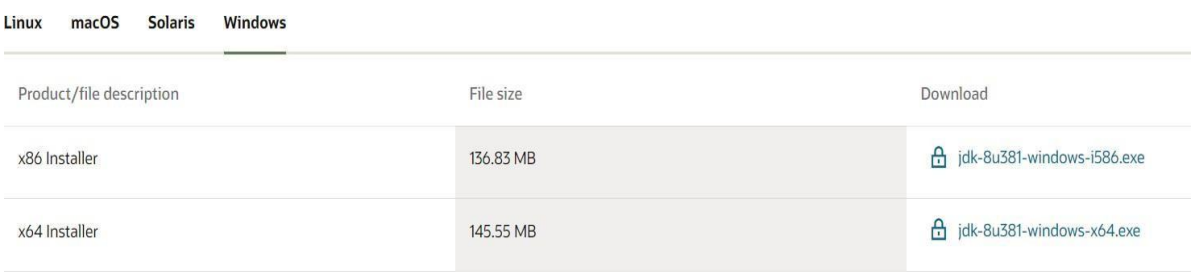
ALGORITHM:

STEP 1: To install Hadoop the primary task is to setup and install java environment

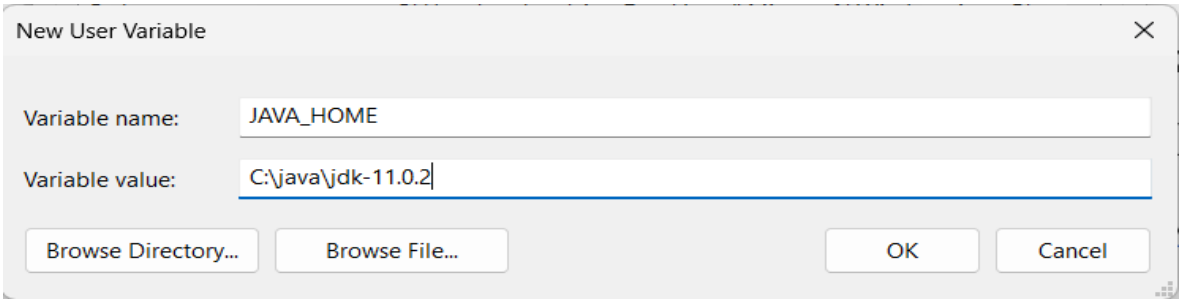
STEP 2: The java version that needed to be installed depends on the Hadoop's version. Here we are installing the latest version of Hadoop which is 3.3.0 which supports java version varying from 8-11(runtime only).

STEP 3: Use the following link to install java

https://www.oracle.com/java/technologies/downloads/#java8-windows



STEP 4: After installing java setup, the java environment in environmental variables directing the bin folder inside the java folder (**C:\java\jdk-11.0.2\bin**) copy the path till bin folder and paste it in the environmental variable define the new path and add the bin folder location as **JAVA_HOME=" C:\java\jdk11.0.2\bin"** and apply the changes



STEP 5: Now after setting up the java environment check the setup has been successfully set by using **java -version** command in your command prompt and it should display the version of java you have installed.

```
C:\Windows\System32>java -version
java version "1.8.0_381"
Java(TM) SE Runtime Environment (build 1.8.0_381-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.381-b09, mixed mode)
```

STEP 7: Hadoop is Unix distribution-based file with tar.gz extension we have to extract the file using the 7-zip manager which supports multiple formats follow this link to install 7-zip

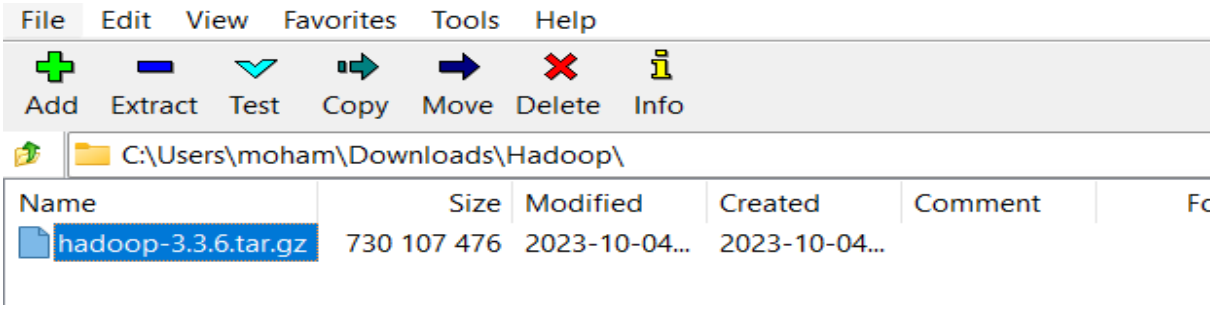
https://7 - zip.org/

STEP 8: Now install the Notepad++ text editor which is further used to modify or edit the configuration file within Hadoop as per our requirement

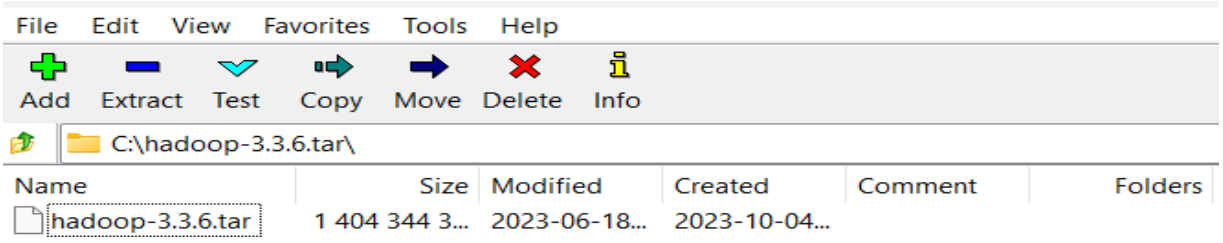
STEP 9: After installing and setting up all the required application install Hadoop from the official Apache Hadoop website https://hadoop.apache.org/releases.html download the binary download which can run directly without any need for compilation.

Version	Release date	Source download	Binary download	Release notes
3.3.6	2023 Jun 23	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)	Announcement
3.2.4	2022 Jul 22	source (checksum signature)	binary (checksum signature)	Announcement
2.10.2	2022 May 31	source (checksum signature)	binary (checksum signature)	Announcement

STEP 10: Run 7-zip manager as administrator and navigate to the path where Hadoop is located for extract the compiled binary download of Hadoop



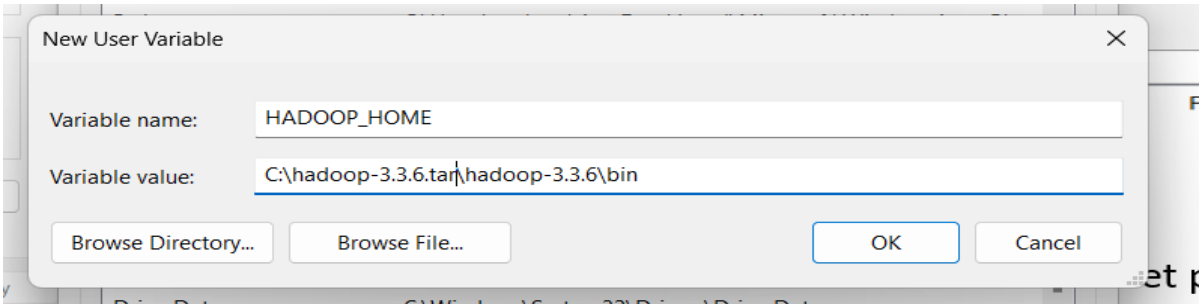
STEP 11: After doing the extraction process there is another compressed file with in the extracted file extract that as well.



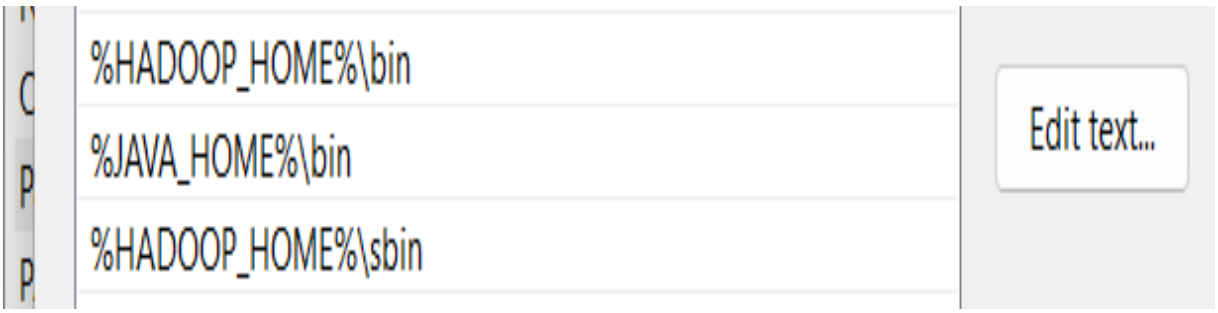
STEP 12: From the extracted folder replace the bin file with the reliable windows supported configured file here is the drive link to download the bin file

https://drive.google.com/file/d/1kVhX9snOZ3oLUxDjh3AVI8fcRnEWAAE4/view

STEP 13: Setup the Hadoop environment in environment variable and set path location as **HADOOP_HOME= "C:\hadoop-3.3.6.tar\hadoop-3.3.6\bin"**



STEP 14: Add the Hadoop bin and sbin path location by editing the path. And add the bin, sbin location there



STEP 15: Now open etc folder inside the Hadoop folder and locate the file Hadoop-env.cmd and set the java home location

```
@rem remote nodes.

@rem The java implementation to use.  Required.
set JAVA_HOME=%JAVA_HOME%
set JAVA_HOME=C:\java\java8

@rem The jsvc implementation to use. Jsvc is required to run secure datanodes.
@rem set JSVC_HOME=%JSVC_HOME%

@rem set HADOOP_CONF_DIR=
```

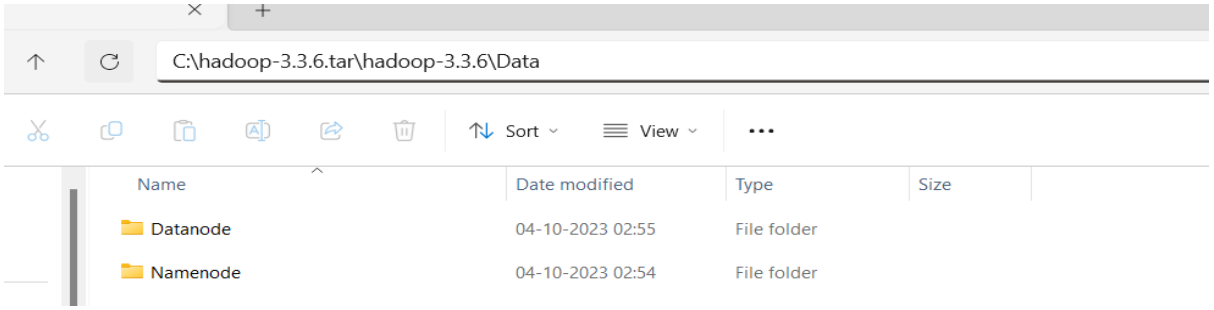
STEP 16: Edit the following configuration XML files core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml are used to configure the behaviour of your Hadoop Cluster and save them.

STEP 17: Starting from core-site.xml edit it using notepad++ and following the program to configure.

PROGRAM (CORE-SITE.XML)

```
<configuration>
 <property>
 <name>fs.default.name</name>
 <value>hdfs://localhost:9000</value>
</property>
</configuration>
```

STEP 18: To edit the hdfs-site.xml create Date folder and then within the Data folder create Namenode, Datenode which are used to manage and cluster flow date and log files.



STEP 19: Now open the hdfs-site.xml in notepad++ and add the following program

```
<configuration>
 <property>
<name>dfs.replication</name>
 <value>1</value>
 </property>
 <property>
 <name>dfs.namenode.name.dir</name>
 <value> C:/hadoop-3.3.6.tar/hadoop-3.3.6/Data/Datanode </value>
 </property>
<property>
 <name>dfs.datanode.data.dir</name>
<value>> C:/hadoop-3.3.6.tar/hadoop-3.3.6/Data/Datanode </value>
</property>
</configuration>
```

STEP 20: Edit mapred-site.xml file using notepad++ add this following program

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

STEP 21: Edit the yarn-site.xml with following program and save it.

```
<configuration>
<property>
 <name>yarn.nodemanager.aux-services</name>
 <value>mapreduce_shuffle</value>
 </property>
 <property>
 <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>
</configuration>
```

STEP 22: Now save them and open command prompt as administrator and run the following command to **hdfs namenode -format** to format the contents of namenode

```
\current\fsimage.ckpt_0000000000000000000 using no compression
2023-11-06 11:45:46,760 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.0\data\namenode1290\curren
t\fsimage.ckpt_0000000000000000000 of size 400 bytes saved in 0 seconds .
2023-11-06 11:45:46,783 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2023-11-06 11:45:46,794 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-11-06 11:45:46,795 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at Azhar/127.0.0.1
************************************************************/

C:\Windows\system32>
```

STEP 23: To check the daemons configured correctly open command prompt as administrator and run the following command's

hdfs namenode, hdfs Datanode, yarn nodemanager, yarn resourcemanager

hdfs namenode

```
C:\hadoop-3.3.0\sbin>hdfs namenode
2023-11-06 11:47:10,614 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = Azhar/127.0.0.1
STARTUP_MSG:   args = []
STARTUP_MSG:   version = 3.3.0
STARTUP_MSG:   classpath = C:\hadoop-3.3.0\etc\hadoop;C:\hadoop-3.3.0\share\hadoop\common;C:\hadoop-3.3.0\share\
hadoop\common\lib\accessors-smart-1.2.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\animal-sniffer-annotations-1.1
```

hdfs datanode

```
C:\Windows\system32>hdfs datanode
2023-11-06 11:51:47,958 INFO datanode.DataNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting DataNode
STARTUP_MSG:   host = Azhar/127.0.0.1
STARTUP_MSG:   args = []
STARTUP_MSG:   version = 3.3.0
STARTUP_MSG:   classpath = C:\hadoop-3.3.0\etc\hadoop;C:\hadoop-3.3.0\share\hadoop\common;C:\hadoop-3.
3.0\share\hadoop\common\lib\accessors-smart-1.2.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\animal-sni
ffer-annotations-1.17.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\asm-5.0.4.jar;C:\hadoop-3.3.0\share\
```

yarn nodemanager

```
C:\Windows\system32>yarn nodemanager
2023-11-06 11:52:28,180 INFO nodemanager.NodeManager: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NodeManager
STARTUP_MSG:   host = Azhar/127.0.0.1
STARTUP_MSG:   args = []
STARTUP_MSG:   version = 3.3.0
STARTUP_MSG:   classpath = C:\hadoop-3.3.0\etc\hadoop;C:\hadoop-3.3.0\etc\hadoop;C:\hadoop-3.3.0\etc\h
adoop;C:\hadoop-3.3.0\share\hadoop\common;C:\hadoop-3.3.0\share\hadoop\common\lib\accessors-smart-1.2.
jar;C:\hadoop-3.3.0\share\hadoop\common\lib\animal-sniffer-annotations-1.17.jar;C:\hadoop-3.3.0\share\
```

yarn resourcemanager

```
C:\Windows\system32>yarn resourcemanager
2023-11-06 11:53:32,487 INFO resourcemanager.ResourceManager: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting ResourceManager
STARTUP_MSG:   host = Azhar/127.0.0.1
STARTUP_MSG:   args = []
STARTUP_MSG:   version = 3.3.0
STARTUP_MSG:   classpath = C:\hadoop-3.3.0\etc\hadoop;C:\hadoop-3.3.0\etc\hadoop;C:\hadoop-3.3.0\etc\h
adoop;C:\hadoop-3.3.0\share\hadoop\common;C:\hadoop-3.3.0\share\hadoop\common\lib\accessors-smart-1.2.
jar;C:\hadoop-3.3.0\share\hadoop\common\lib\animal-sniffer-annotations-1.17.jar;C:\hadoop-3.3.0\share\
```

STEP 24: To check the check the daemons that are running in background we can use Java Virtual Machine Process Status which is used to list the java virtual machines that are currently running on a system it is used to display the process ID (PID) of each JVM

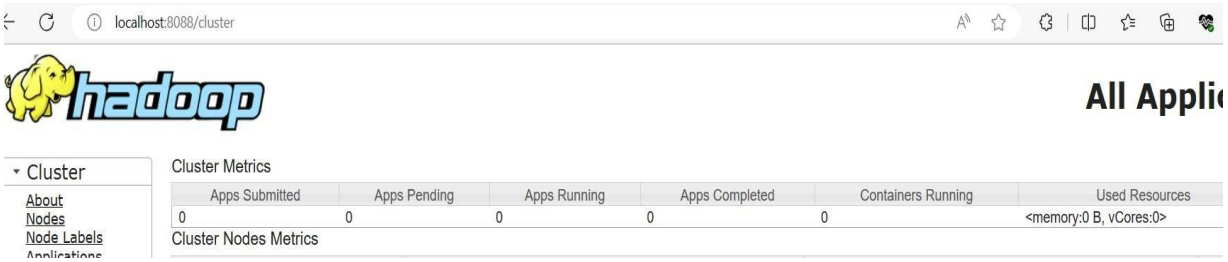
```
C:\Windows\System32>jps
9712 NodeManager
8212 Jps
16056 NameNode
1800 ResourceManager
18200 DataNode
```

STEP 25: Now we can access the Namenode and Datanode as web user interface (web-UI) by using the following localhost address

localhost:9870

STEP 26: To access the Datanode use the following localhost address

localhost:8088



RESULT: Thus, the program to install and setup Hadoop has been completed and Output is verified successfully completed.

EXP.NO:02 **DATE:**

IMPLEMENTATION OF FILE MANAGEMENT TASKS

AIM: To implement file management tasks in Hadoop file system (HDFS).

ALGORITHM:

1. **Creating a directory in HDFS**

SYNTAX:

hadoop fs -mkdir <paths>

EXAMPLE:

hadoop fs -mkdir /user hadoop

fs -mkdir /user/dir1

```
C:\Windows\System32>hadoop  fs -mkdir /user

C:\Windows\System32>hadoop fs -mkdir /user/dir1
```

2. **Listing the contents of a directory**

SYNTAX:

hadoop fs -ls <directory name>

EXAMPLE:

hadoop fs -ls /user hadoop

fs -ls /user/dir1

```
C:\Windows\System32>hadoop fs -ls /user
Found 1 items
drwxr-xr-x   - moham supergroup          0 2023-11-05 08:47 /user/dir1

C:\Windows\System32>hadoop fs -ls /user/dir1
```

3. Uploading and downloading a file in HDFS

SYNTAX: (UPLOAD)

hadoop fs -put <local file system path> <hdfs destination path>

EXAMPLE:

hadoop fs -put C:\Home\samplefile.txt.txt /user/dir1/

hadoop fs -ls /user/dir1

```
C:\Windows\System32>hadoop fs -put C:\Home\samplefile.txt /user/dir1

C:\Windows\System32>hadoop fs -ls /user/dir1
Found 1 items
-rw-r--r--   1 moham supergroup         13 2023-11-05 14:46 /user/dir1/samplefile.txt
```

SYNTAX: (DOWNLOAD)

hadoop fs -get <hdfs src> <local dst>

EXAMPLE:

hadoop fs -get /user/dir1/samplefile.txt C:\Home\Hadoopfiles

```
C:\Windows\System32>hadoop fs -get /user/dir1/samplefile.txt C:\Home\Hadoopfiles
```

4. See the contents of a file

SYNTAX:

hadoop fs -cat <path[filename]>

EXAMPLE:

hadoop fs -cat /user/dir1/samplefile.txt

```
C:\Windows\System32>hadoop fs -cat /user/dir1/samplefile.txt
1
2
3
4
5
```

5. Copy a file from source to destination

SYNTAX:

hadoop fs -cp <src> <dst>

EXAMPLE:

hadoop fs /user/dir1/samplefile.txt /user/dir2

```
C:\Windows\System32>hadoop fs -cp /user/dir1/samplefile.txt /user/dir2

C:\Windows\System32>hadoop fs -ls /user/dir2
Found 1 items
-rw-r--r--   1 moham supergroup         13 2023-11-05 14:57 /user/dir2/samplefile.txt
```

6. Copy a file from and to local file system to hdfs

SYNTAX: (FROM)

hadoop fs -copyFromLocal <local file system file path> <hdfs dst>

EXAMPLE:

hadoop fs -copyFromLocal C:\Home\test.txt /user/dir1

```
C:\Windows\System32>hadoop fs -copyFromLocal C:\Home\test.txt /user/dir1

C:\Windows\System32>hadoop fs -ls /user/dir1
Found 2 items
-rw-r--r--   1 moham supergroup         13 2023-11-05 14:46 /user/dir1/samplefile.txt
-rw-r--r--   1 moham supergroup          0 2023-11-05 15:00 /user/dir1/test.txt
```

SYNTAX: (TO)

hadoop fs -copyToLocal <hdfs src> <local dst>

EXAMPLE:

hadoop fs -copyToLocal /user/dir1/samplefile.txt C:\Home\copy

```
C:\Windows\System32>hadoop fs -copyToLocal /user/dir1/samplefile.txt C:\Home\copy
```

7. Move file from source to destination

SYNTAX:

hadoop fs -mv <src> <dst>

EXAMPLE:

hadoop fs -mv /user/dir1/test.txt /user/dir2

```
C:\Windows\System32>hadoop fs -ls /user/dir2
Found 2 items
-rw-r--r--   1 moham supergroup         13 2023-11-05 14:57 /user/dir2/samplefile.txt
-rw-r--r--   1 moham supergroup          0 2023-11-05 15:00 /user/dir2/test.txt
```

8. Remove a file or directory in hdfs

SYNTAX:

hadoop fs -rm <arg>

EXAMPLE:

hadoop fs -rm /user/dir1/samplefile.txt

```
C:\Windows\System32>hadoop fs -rm /user/dir1/samplefile.txt
Deleted /user/dir1/samplefile.txt
```

SYNTAX: (Recursive method for deleting directories)

hadoop fs -rm -r <arg>

EXAMPLE:

hadoop fs -rm -r /user/dir1

```
C:\Windows\System32>hadoop fs -rm -r /user/dir1
Deleted /user/dir1

C:\Windows\System32>hadoop fs -ls  /user
Found 1 items
drwxr-xr-x   - moham supergroup          0 2023-11-05 15:05 /user/dir2
```

9. Display few lines of a file

SYNTAX:

hadoop fs -tail <path[filename]>

EXAMPLE:

hadoop fs -tail /user/dir2/samplefile.txt

```
C:\Windows\System32>hadoop fs -tail /user/dir2/samplefile.txt
1
2
3
4
5
```

10. Display the aggregate length of a file

SYNTAX:

hadoop fs -du <path>

EXAMPLE:

hadoop fs -du /user/dir2/samplefile.txt

```
C:\Windows\System32>hadoop fs -du /user/dir2/samplefile.txt
17  17  /user/dir2/samplefile.txt
```

RESULT: Thus, the implementation of file management tasks in hadoop file system was implemented and Output is verified successfully.

EXP.NO:03 **DATE:**

IMPLEMENTATION OF MATRIX MULTIPLICATION WITH HADOOP MAP REDUCE

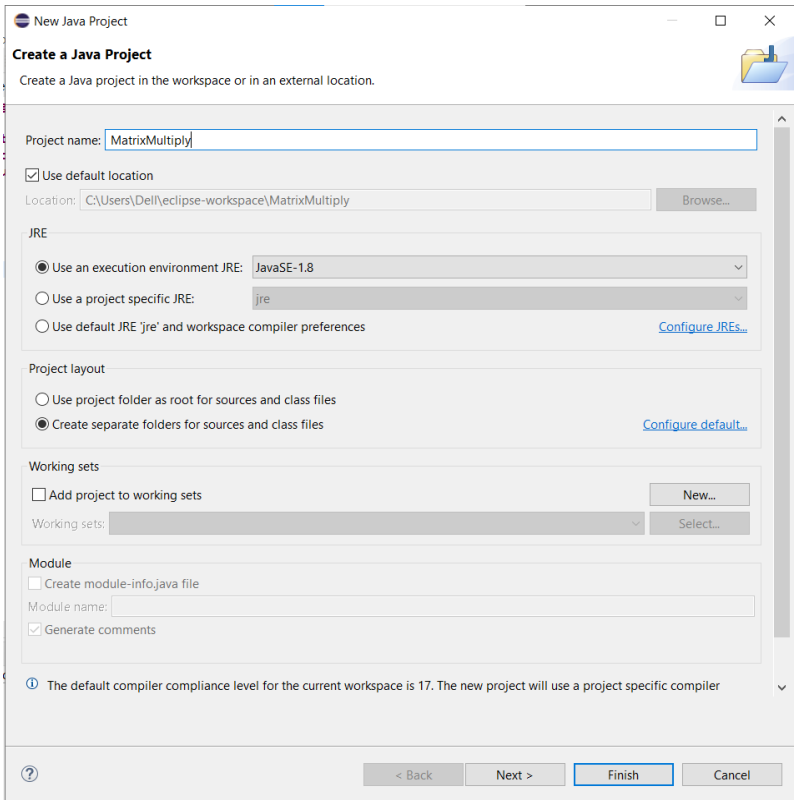
AIM: To implement of Matrix Multiplication with Hadoop Map Reduce.

ALGORITHM:

STEP 1: Run Eclipse for Java Developers

STEP 2: Create a new Java Project with name "MatrixMultiply "

STEP 3: Set the Java Environment Version to your current version of Java (JRE : 1.8)



STEP 4 : Add a Package with name "com.mapreduce.java" and Create three Classes in it.

STEP 5 : Create a New Class With name Map.java.

STEP 6: Now write the below program in the "Map.java" Class

PROGRAM:

```
package com.MapReduce.wc;
import
org.apache.hadoop.conf.*;
```

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
//import
org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
public class Map extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text,
Text>

{

        @Override

public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
                Configuration conf =
                context.getConfiguration(); int m =
                Integer.parseInt(conf.get("m"));
                int p = Integer.parseInt(conf.get("p"));
String line = value.toString();
// (M, i, j, Mij);

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

if
(indicesAndValue[0].equals("M")) {
for (int k = 0; k < p; k++) {
outputKey.set(indicesAndValue[1] + "," + k);

// outputKey.set(i,k);

outputValue.set(indicesAndValue[0] + "," + indicesAndValue[2]

+ "," + indicesAndValue[3]);

// outputValue.set(M,j,Mij);
context.write(outputKey,
outputValue);
}
```

```
} else {

        // (N, j, k, Njk);

        for (int i = 0; i < m; i++) {

        outputKey.set(i + "," + indicesAndValue[2]); outputValue.set("N," + indicesAndValue[1] +
        ","+ indicesAndValue[3]); context.write(outputKey, outputValue);
        }
        }
        }
}
```

STEP 7: Now Create another class with name “Reduce.java” and paste the below program in it.

PROGRAM:

```
package com.MapReduce.wc;
import
org.apache.hadoop.io.Text;
// import
org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
import java.util.HashMap;
public class Reduce
extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> {
@Override public void reduce(Text key, Iterable<Text> values, Context context)
throws IOException, InterruptedException {
String[] value;
//key=(i,k),

//Values = [(M/N,j,V/W),..]

HashMap<Integer, Float> hashA = new HashMap<Integer, Float>(); HashMap<Integer, Float>
hashB
= new HashMap<Integer, Float>(); for (Text val : values)
{ value = val.toString().split(",");
if (value[0].equals("M")) {

hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2])); } else {
hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
```

```
}
}
int n =
Integer.parseInt(context.getConfiguration().get("n"));
float result = 0.0f;
float m_ij;
float n_jk;
for (int j = 0; j < n; j++) {
m_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f; n_jk = hashB.containsKey(j) ? hashB.get(j)
: 0.0f; result += m_ij * n_jk;

}
if (result != 0.0f) {
context.write(null,
new Text(key.toString() + "," + Float.toString(result)));
}
}
}
```

STEP 8: Now, Create another class with name "MatrixMultiply.java" and paste the below program in it.

PROGRAM:

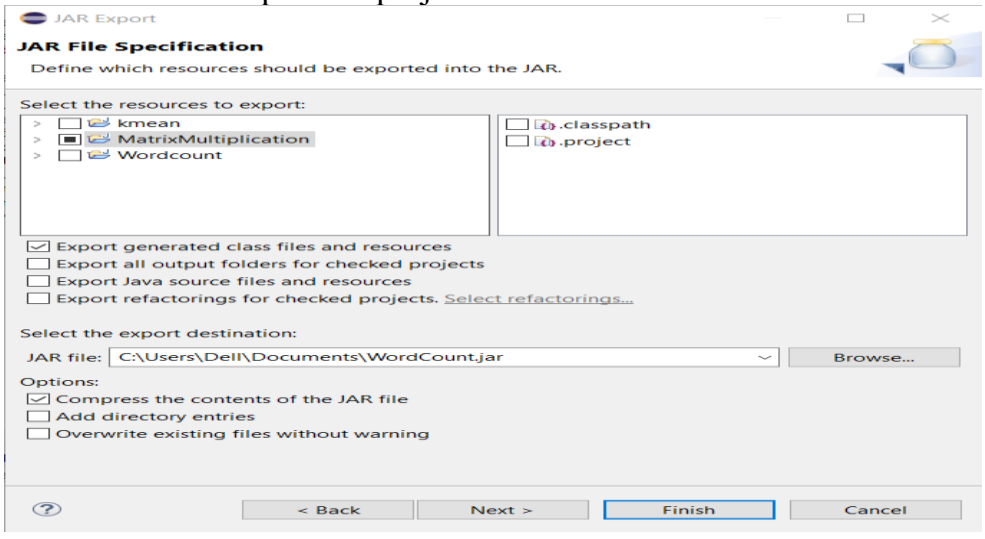
```
package com.MapReduce.wc;
import
org.apache.hadoop.conf.*;
import
org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

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 MatrixMultiply {
public static void main(String[] args) throws Exception { if (args.length
!= 2) { System.err.println("Usage: MatrixMultiply <in_dir> <out_dir>");
System.exit(2);
```

```
}
Configuration conf = new Configuration();
conf.set("m", "1000");

conf.set("n", "100");

conf.set("p", "1000");
@SuppressWarnings("deprecation")
Job job = new Job(conf, "MatrixMultiply");
job.setJarByClass(MatrixMultiply.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);
}
}
```

STEP 9: To resolve the errors in the programs we should add two External jar files to it.

- Hadoop_common :2.7.3.jar
- Hadoop_mapreduce:client:core:2.7.1.jar

STEP 10: Now export the project into a Jar file and name it as “MatrixMultiply.jar”



STEP 11: Now create a Text file in Notepad and name it as “1.txt” and “2.txt. write some content inside the text file and save it.

1 - Notepad

File Edit Format View Help

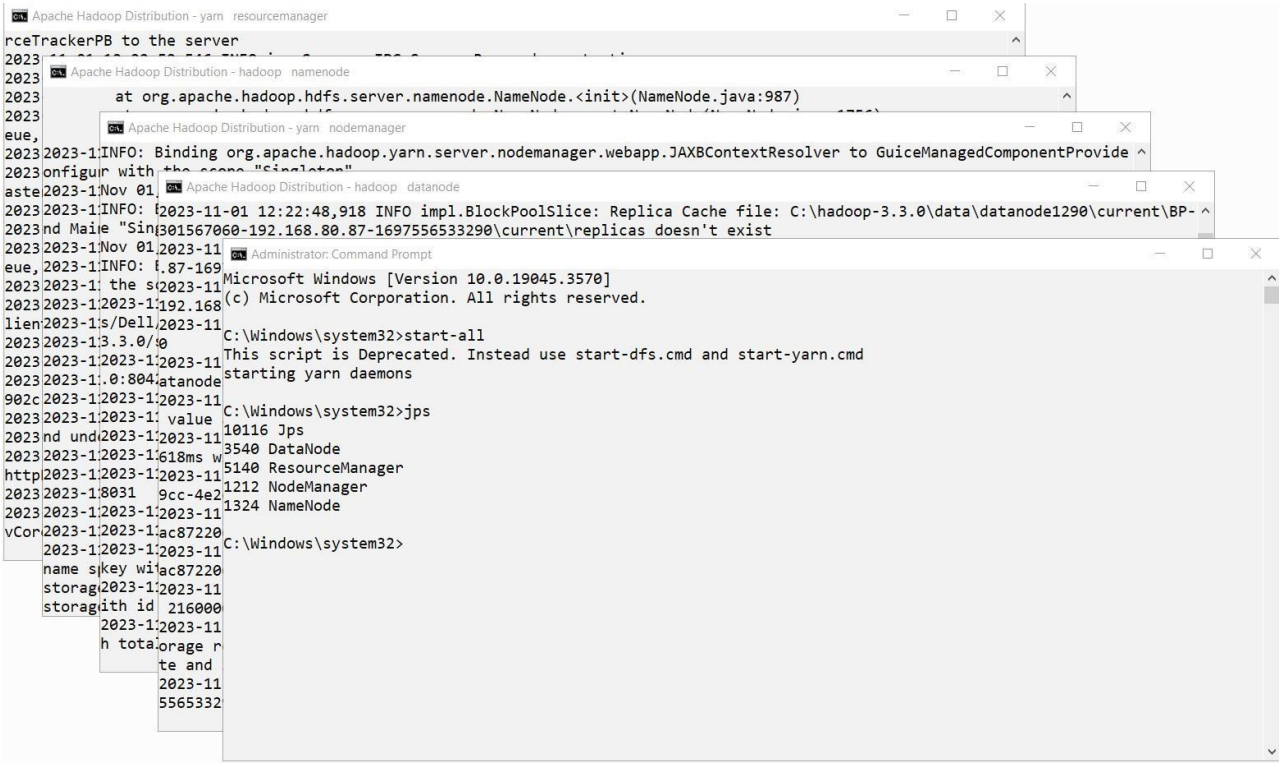
```
M,0,0,1
M,0,1,2
M,1,0,3
M,1,1,4
```

2 - Notepad

File Edit Format View Help

```
N,0,0,1
N,0,1,2
N,1,0,3
N,1,1,4
```

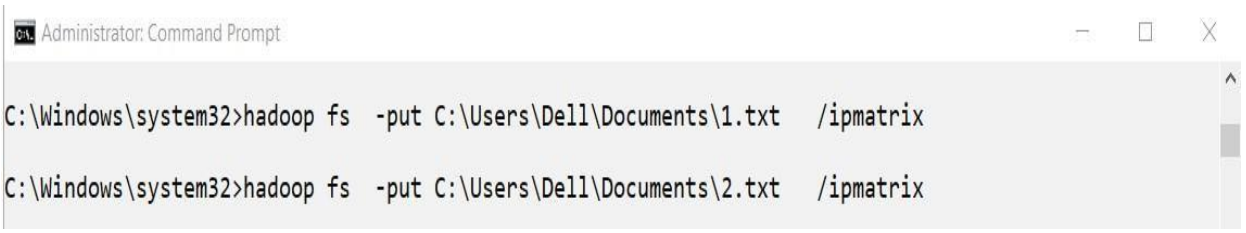
STEP 12: Now run all the deamons in Hadoop



STEP 13: Create a new input directory named as "ipmatrix".
By using the command: **hadoop fs -mkdir /ipmatrix**

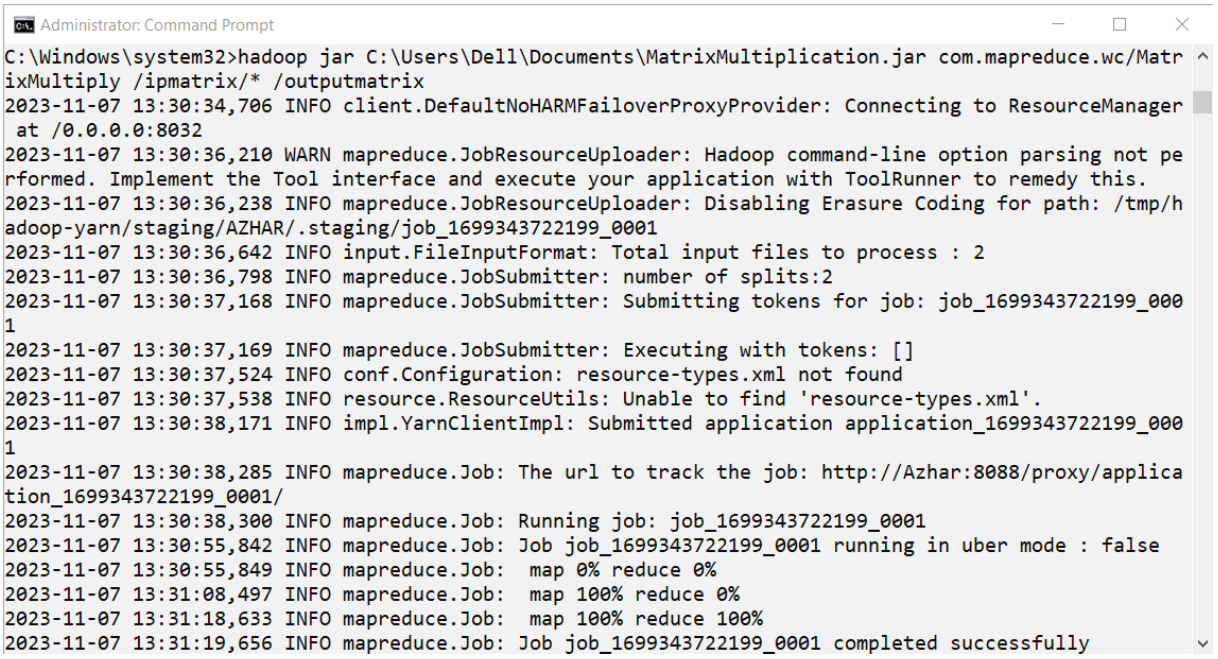
STEP 14: Now put the "1.txt" and 2.txt file to the ipmatrix directory.
By using these commands: **hadoop fs -put C:\Users\Dell\Documents\1.txt /ipmatrix**

hadoop fs -put C:\ Users\Dell\Documents\2.txt /ipmatrix



STEP 15: Run the Jar file created from the project

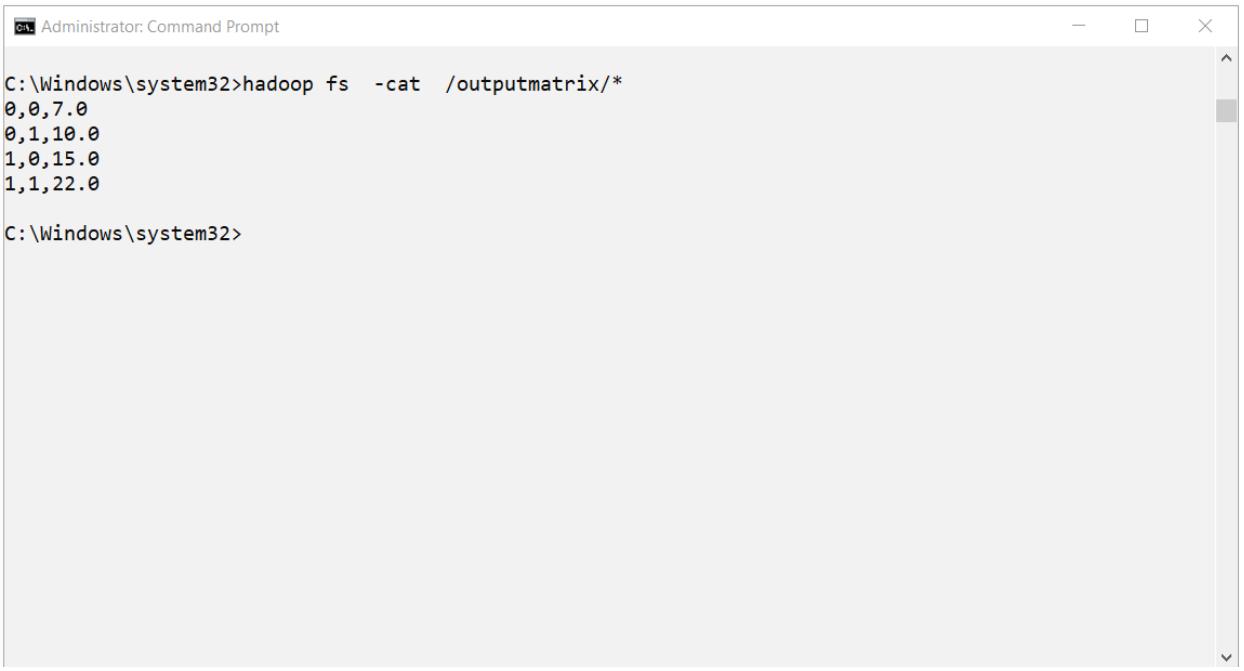
Using the command: **hadoop jar C:\ Users\Dell\Documents \matrixmultiplication.jar com.mapreduce.wc/MatrixMultiply /ipmatrix/ * /outputmatrix**



STEP 16: At last Print your output for the MatrixMultiply text file.

Using the Command : **hadoop fs -cat /outputmatrix/***

OUTPUT :



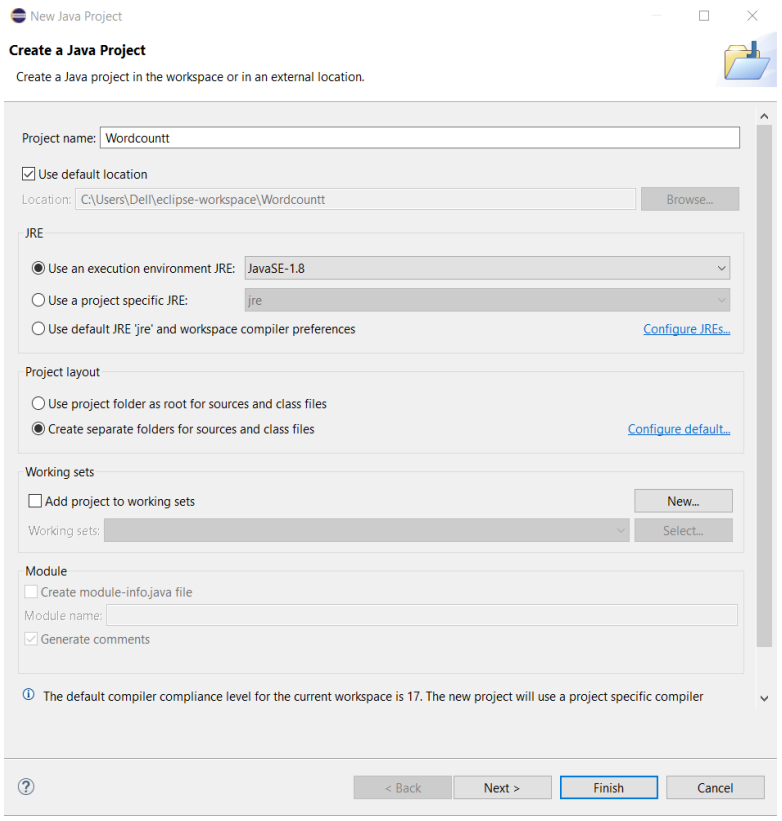
RESULT : Thus the program to run a basic wordcount mapreduce program to understand mapreduce is excecuted and output is verified successfully.

EXP.NO:04 **DATE:**

RUN A BASIC WORD COUNT MAP REDUCE PROGRAM TO UNDERSTAND MAP REDUCE PARADIGM

AIM: To Run a Basic Word Count Map Reduce program to understand Map Reduce Paradigm.

ALGORITHM:
STEP 1: Run Eclipse for Java Developers
STEP 2: Create a new Java Project with name "WordCount "
STEP 3: Set the Java Environment Version to your current version of Java (JRE : 1.8)



STEP 4 : Add a Package with name "com.mapreduce.java" and Create three Classes in it.
STEP 5 : Create a New Class With name WC_Mapper.java.
STEP 6: Now write the below program in the "WC_Mapper.java" Class

PROGRAM:

```
package com.mapreduce.java;
import java.io.IOException;
import
```

```
java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import
org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>{

   private final static IntWritable one = new
   IntWritable(1); private Text word = new Text();
   public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable>
        output, Reporter reporter) throws IOException{
      String line = value.toString();

      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()){
         word.set(tokenizer.nextToken());
         output.collect(word, one);
      } }
```

STEP 7: Now Create another class with name "WC_Reducer.java" and paste the below program in it.

PROGRAM:

```
package com.mapreduce.java;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import
org.apache.hadoop.mapred.MapReduceBase;
import
org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

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

```
public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable>
 output, Reporter reporter) throws IOException {
int sum=0;

while (values.hasNext()) {
sum+=values.next().get();
}
output.collect(key,new IntWritable(sum));
}
}
```

STEP 8: Now, Create another class with name "WC_runner.java" and paste the below program in it.

PROGRAM:

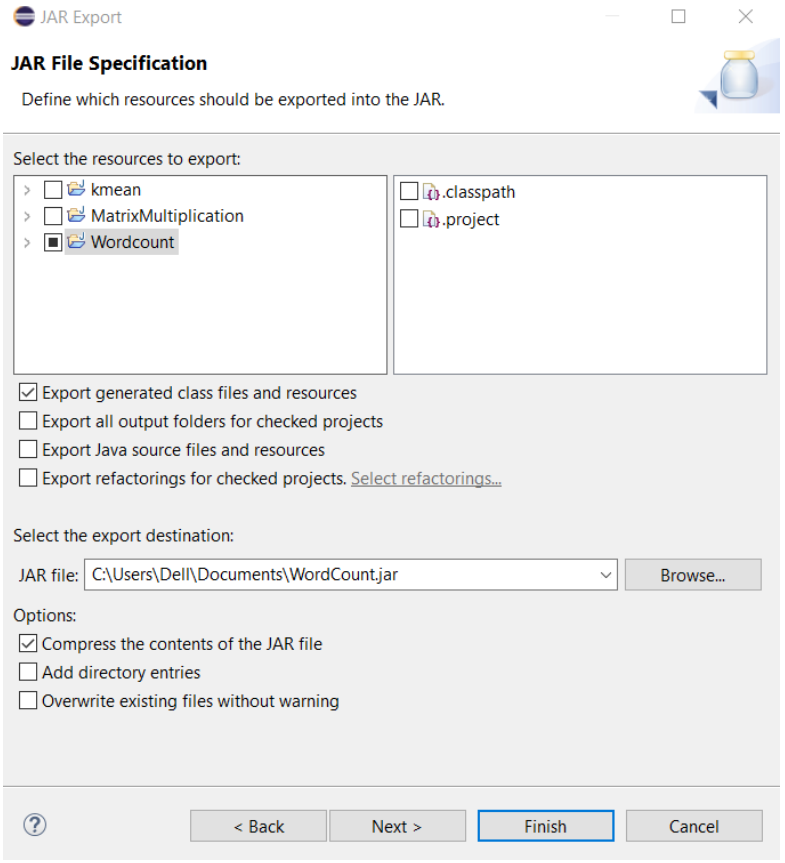
```
package com.mapreduce.java;
import java.io.IOException;
import
org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.TextInputFormat;
import
org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
   public static void main(String[] args) throws
      IOException{ JobConf conf = new
      JobConf(WC_Runner.class);
      conf.setJobName("WordCount");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      conf.setMapperClass(WC_Mapper.class);
      conf.setCombinerClass(WC_Reducer.class);
      conf.setReducerClass(WC_Reducer.class);
      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);
      FileInputFormat.setInputPaths(conf,new Path(args[0]));
      FileOutputFormat.setOutputPath(conf,new Path(args[1]));
      JobClient.runJob(conf);
```

```
    }
}
```

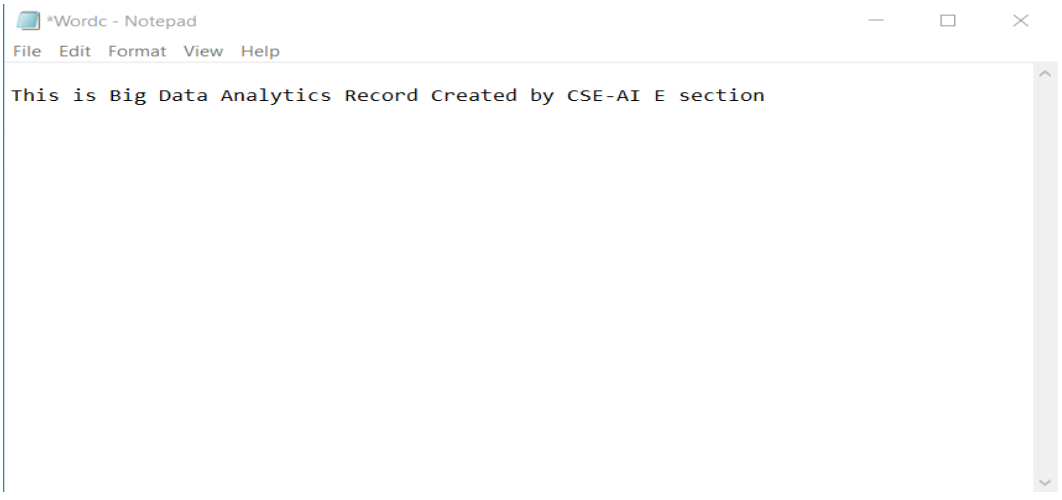
STEP 9: To resolve the errors in the programs we should add two External jar files to it.

- Hadoop_common :2.7.3.jar
- Hadoop_mapreduce:client:core:2.7.1.jar

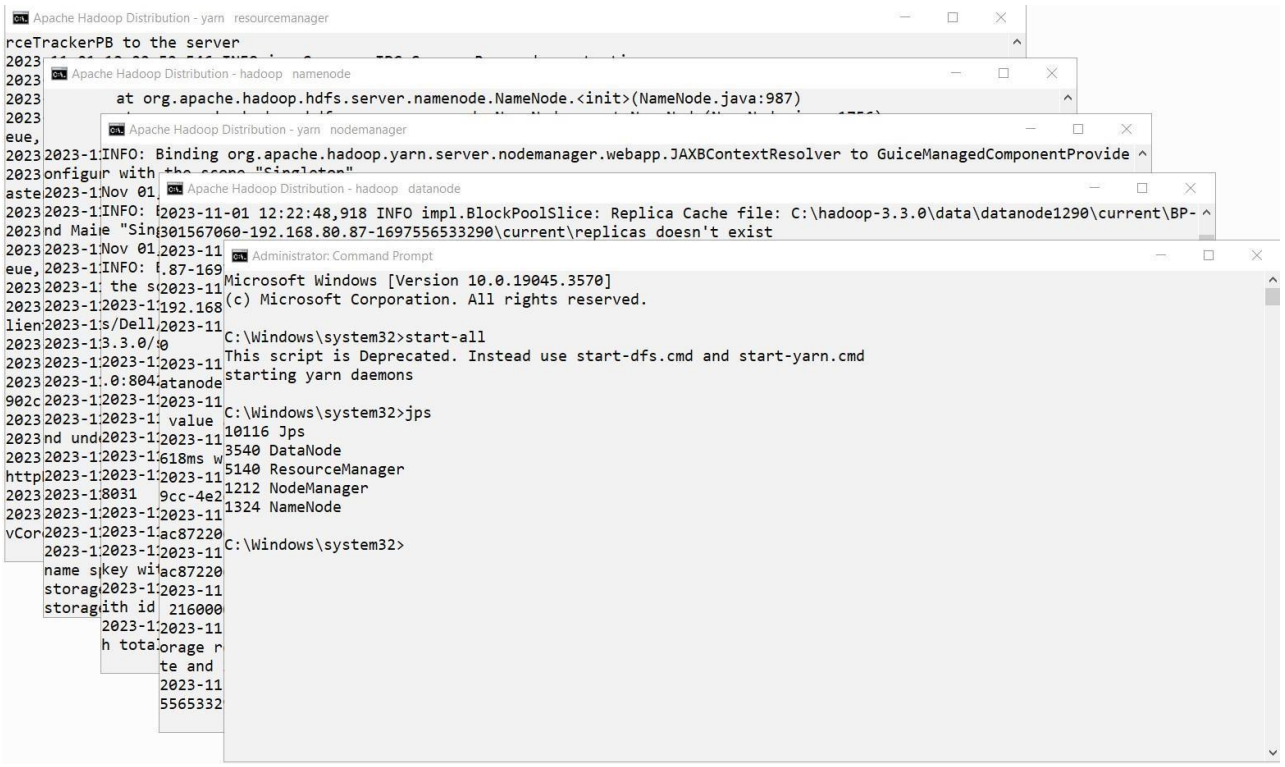
STEP 10: Now export the project into a Jar file and name it as “WordCount.jar”



STEP 11: Now create a Text file in Notepad and name it as “wordc.txt” and write some content inside the text file and save it.



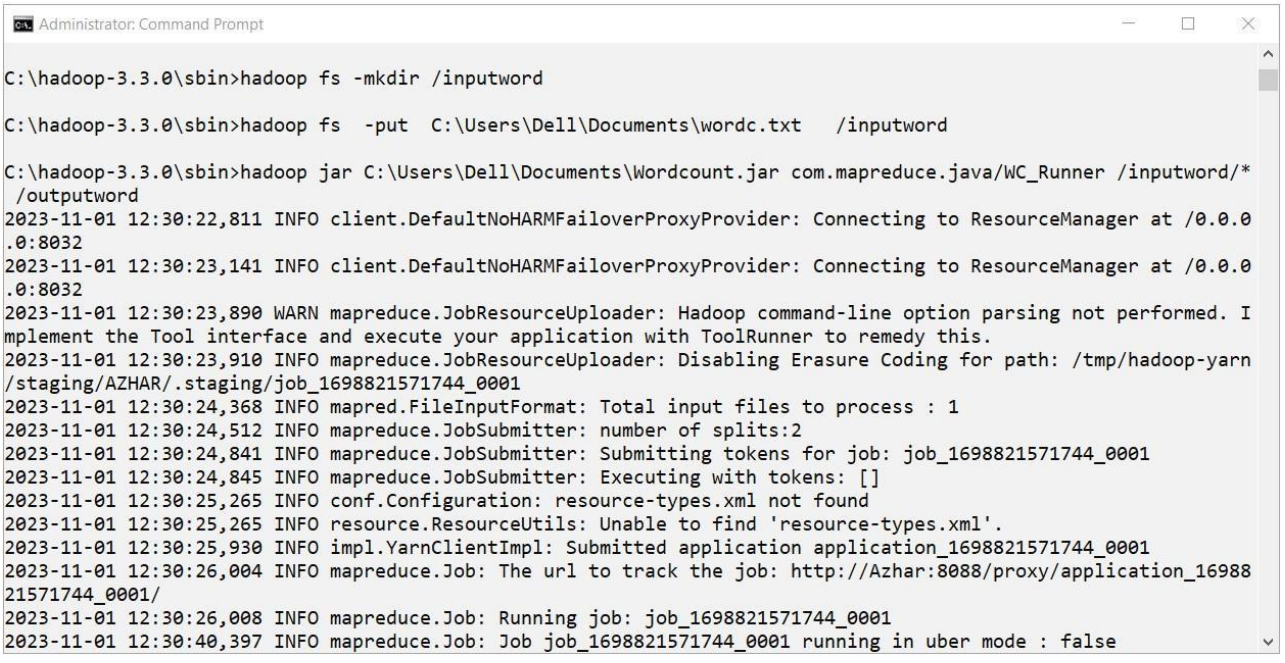
STEP 12: Now run all the demons in Hadoop.



STEP 13: Create a new input directory named as “inputword”.

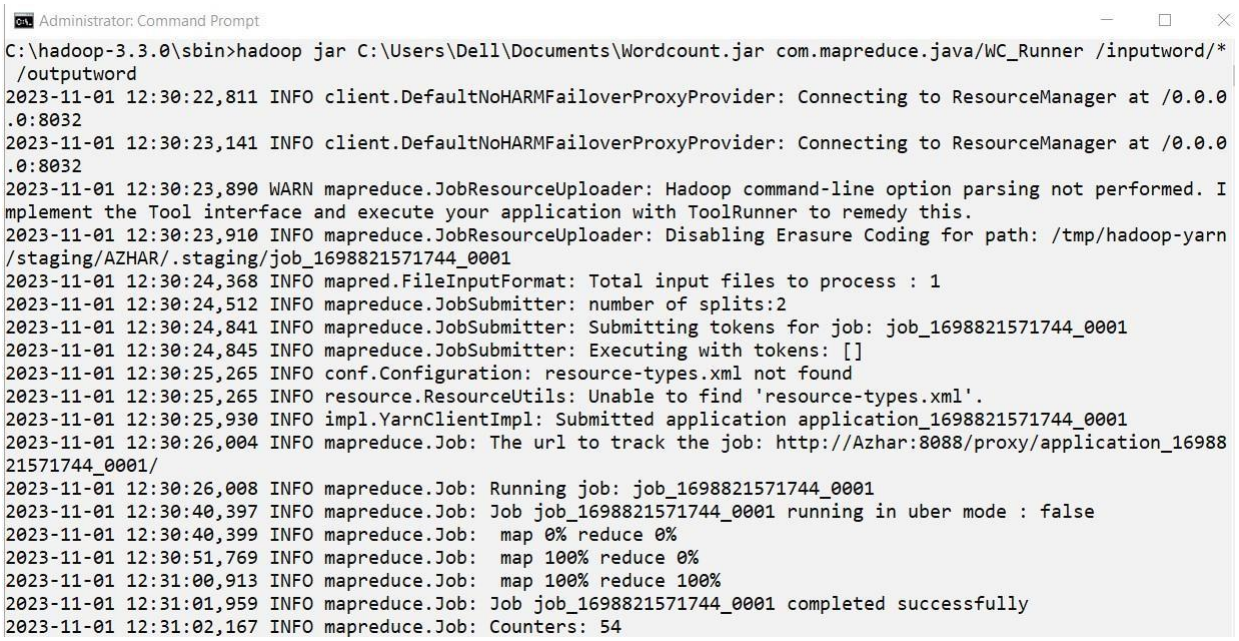
By using the command: hadoop fs -mkdir /inputword

STEP 14: Now put the “wordc.txt” file to the inputword directory.By using the command: hadoop fs -put C:\Users\Dell\Documents\wordc.txt /inputword



STEP 15: Run the Jar file created from the project

Using the command: hadoop jar C:\Users\Dell\Documents\Wordcount.jar com.mapreduce.java/WC_Runner /inputword/* /outputword



STEP 16: At last Print your output for the WordCount text file.

Using the Command : hadoop fs -cat /outputword/*

OUTPUT:

RESULT: Thus the program to run a basic wordcount mapreduce program to understand mapreduce is excecuted and output is verified successfully.

EXP.NO : 05 **DATE :**

IMPLEMENTATION OF K-MEANS CLUSTERING USING MAPREDUCE

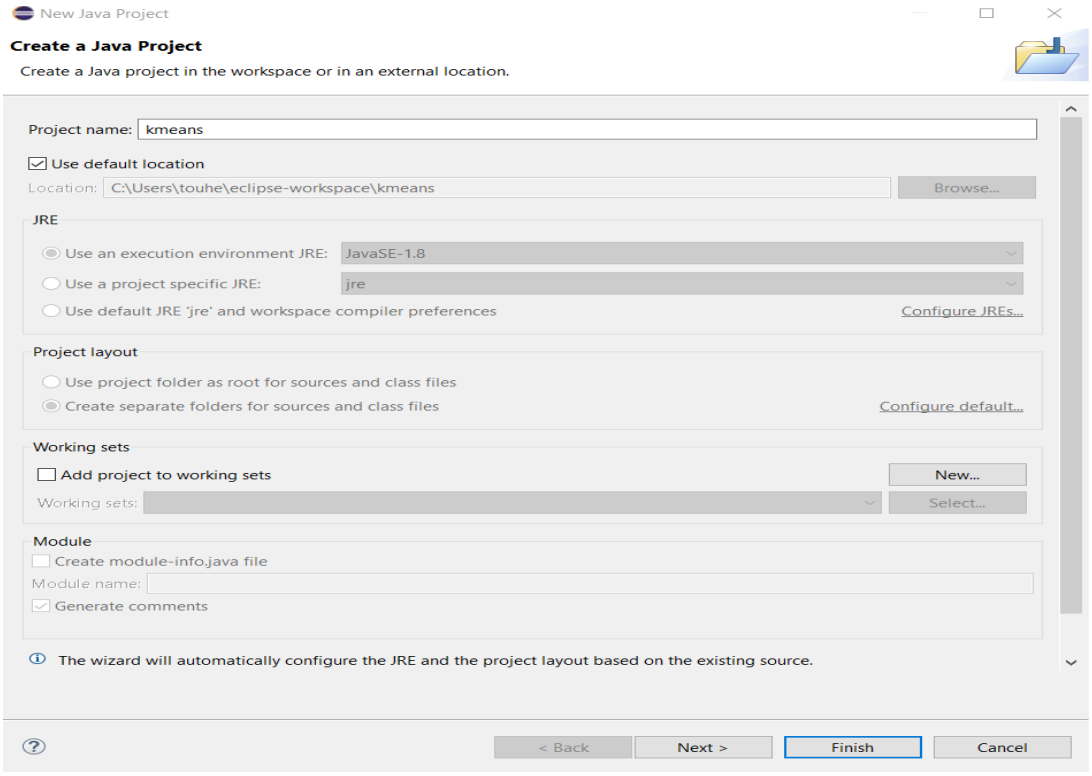
AIM : To implement K-means clustering using mapreduce.

ALGORITHM:

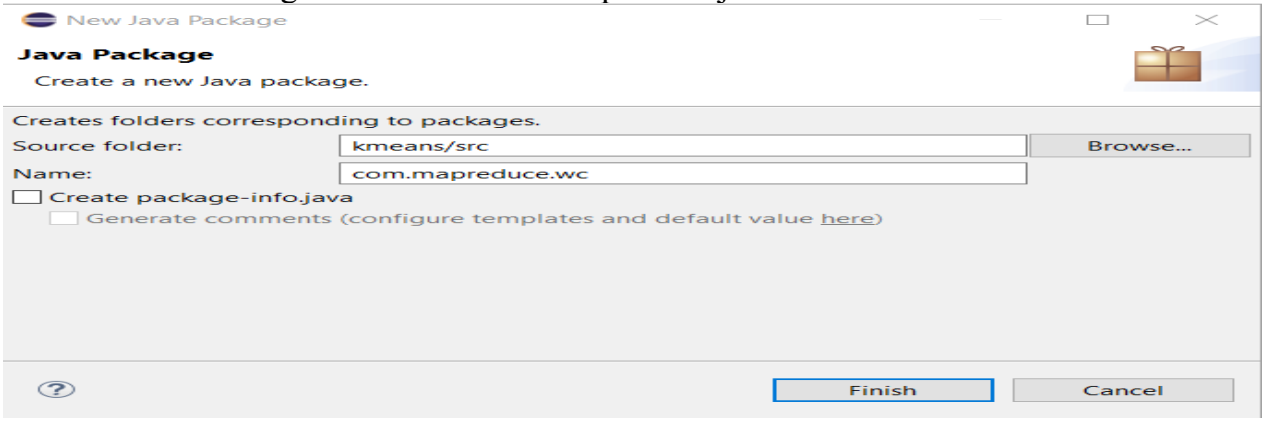
STEP 1: Run Eclipse for Java Developers

STEP 2: Create a new Java Project with name "Kmeans ".

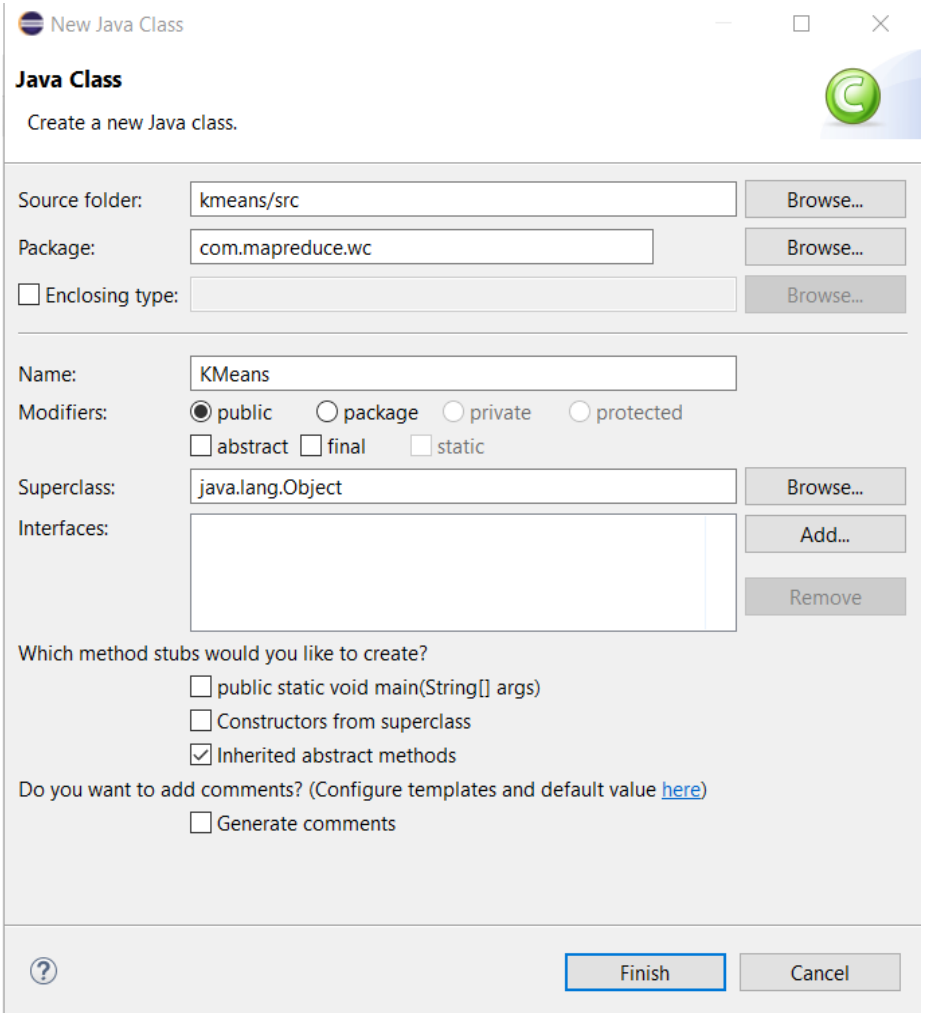
STEP 3: Set the Java Environment Version to your current version of Java (JRE - 1.8)



STEP 4: Add a Package with name "com.mapreduce.java" and Create three Classes in it.



STEP 5 : Create a New Class With name “KMeans.java”



STEP 6: Now write the below program in the “KMeans.java” Class

PROGRAM :

```
package com.mapreduce.wc;
        import
        java.util.ArrayList;
        import
        java.util.HashMap;
        import java.util.Iterator;
        import java.util.List;
        import java.util.Map;
        public class KMeans {
            List<Record> data = new ArrayList<Record>();
```

```
        List<Cluster> clusters = new
        ArrayList<Cluster>();
Map<Cluster, List<Record>> clusterRecords = new
        HashMap<Cluster,List<Record>>(); public static void main(String[] args) {
            int clusterNumber = 2;
            KMeans demo = new KMeans();
            demo.genereateRecord();
         demo.initiateClusterAndCentroid(clusterNumber);
            demo.printRecordInformation();
            demo.printClusterInformation();
        }
           private void genereateRecord() {
            Record record = new Record(1, 19, 15, 39);
            data.add(record);
            record = new Record(2, 21, 15, 81);
            data.add(record);
        record = new Record(3, 20, 16, 6);
        data.add(record);
        record = new Record(4, 23, 16, 77);
        data.add(record);
        record = new Record(5, 31, 17, 40);
        data.add(record);
        record = new Record(6, 22, 17, 76);
        data.add(record);
    }

private void initiateClusterAndCentroid(int clusterNumber) {
        int counter = 1;
        Iterator<Record> iterator =
        data.iterator(); Record record = null;
        while(iterator.hasNext()) {
            record = iterator.next();
```

```
                if(counter <=
                clusterNumber) {
                        record.setClusterNumber(counter);
                        initializeCluster(counter, record):
                    counter++;
                }else {
                        System.out.println(record);
                        System.out.println("** Cluster Information
                        **"); for(Cluster cluster : clusters) {
                                System.out.println(cluster);
                        }
                        System.out.println("**********************");
        double minDistance = Integer.MAX_VALUE;
        Cluster whichCluster = null;

        for(Cluster cluster : clusters) {
            double distance = cluster.calculateDistance(record);
            System.out.println(distance);
            if(minDistance > distance) {
                    minDistance =
                    distance; whichCluster
                    = cluster;
            }
        }

        record.setClusterNumber(whichCluster.getClusterNumber());
                        whichCluster.updateCentroid(record);
                        clusterRecords.get(whichCluster).add(record);
                }
                System.out.println("** Cluster Information **");

 for(Cluster cluster : clusters) {
```

```
                    System.out.println(cluster);
                }
                System.out.println("**********************");
            }
        }
        private void initializeCluster(int clusterNumber, Record record) {Cluster
            System.out.println("****** Each Record INFORMATIN *********");
            for(Record record : data) {
                System.out.println(record);
          }clusterCluster(clusterNumber,record.getAge(),record.getIncome(),record.getScore());
            clusters.add(cluster);
            List<Record> clusterRecord = new ArrayList<Record>();
            clusterRecord.add(record);
            clusterRecords.put(cluster, clusterRecord);
        }
        private void printRecordInformation() {
          }
        private void printClusterInformation() {
          System.out.println("****** FINAL CLUSTER INFORMATIN
          *********"); for (Map.Entry<Cluster, List<Record>> entry :
          clusterRecords.entrySet())  {
      System.out.println("Key = " +
            entry.getKey() + ", Value = " +
            entry.getValue());
       }
    }
    }
```

STEP 7: Now Create another class with name “Cluster.java” and write the below program in it.

PROGRAM :

```
package
   com.mapreduce.wc;
   public class Cluster  {
```

```
private int ageCentroid;
private int
incomeCentroid; private
int scoreCentroid; private
int clusterNumber;
public Cluster(int clusterNumber, int ageCentroid, int incomeCentroid, int scoreCentroid)
    { super();
    this.clusterNumber =
    clusterNumber; this.ageCentroid =
    ageCentroid; this.incomeCentroid =
    incomeCentroid; this.scoreCentroid
    = scoreCentroid;
}
public int getAgeCentroid()
    { return ageCentroid;
}
public void setAgeCentroid(int ageCentroid) {
    this.ageCentroid = ageCentroid;
}
public int getIncomeCentroid() {
    return incomeCentroid;
}
public void setIncomeCentroid(int incomeCentroid) {
    this.incomeCentroid = incomeCentroid;
}
    public int getScoreCentroid()
        { return
        scoreCentroid;
    }
    public void setScoreCentroid(int scoreCentroid) {
        this.scoreCentroid = scoreCentroid;
    }
```

```
	public int getClusterNumber() {
		return clusterNumber;
  }
	public void setClusterNumber(int clusterNumber) {
		this.clusterNumber = clusterNumber;
	}
	@Override
	public String toString() {
		return "Cluster [ageCentroid=" + ageCentroid + ", incomeCentroid=" + incomeCentroid
+ ", scoreCentroid="+ scoreCentroid + ", clusterNumber=" + clusterNumber + "]";
	}
	// Euclidean distance calculation
	public double calculateDistance(Record record) {
		return Math.sqrt(Math.pow((getAgeCentroid() - record.getAge()), 2) +
Math.pow((getIncomeCentroid() - record.getIncome()),2) + Math.pow((getScoreCentroid() -
record.getScore()), 2));
  }
	// Binod Suman Academy YouTube Video on K-Mean
	Algorithm public void updateCentroid(Record record) {
		setAgeCentroid((getAgeCentroid()+record.getAge())/2);
		setIncomeCentroid((getIncomeCentroid()+record.getIncome())/2);
		setScoreCentroid((getScoreCentroid()+record.getScore())/2);
	}
}
```

STEP 8: Now Create another class with name “Record.java” and write the below program in it.

PROGRAM :

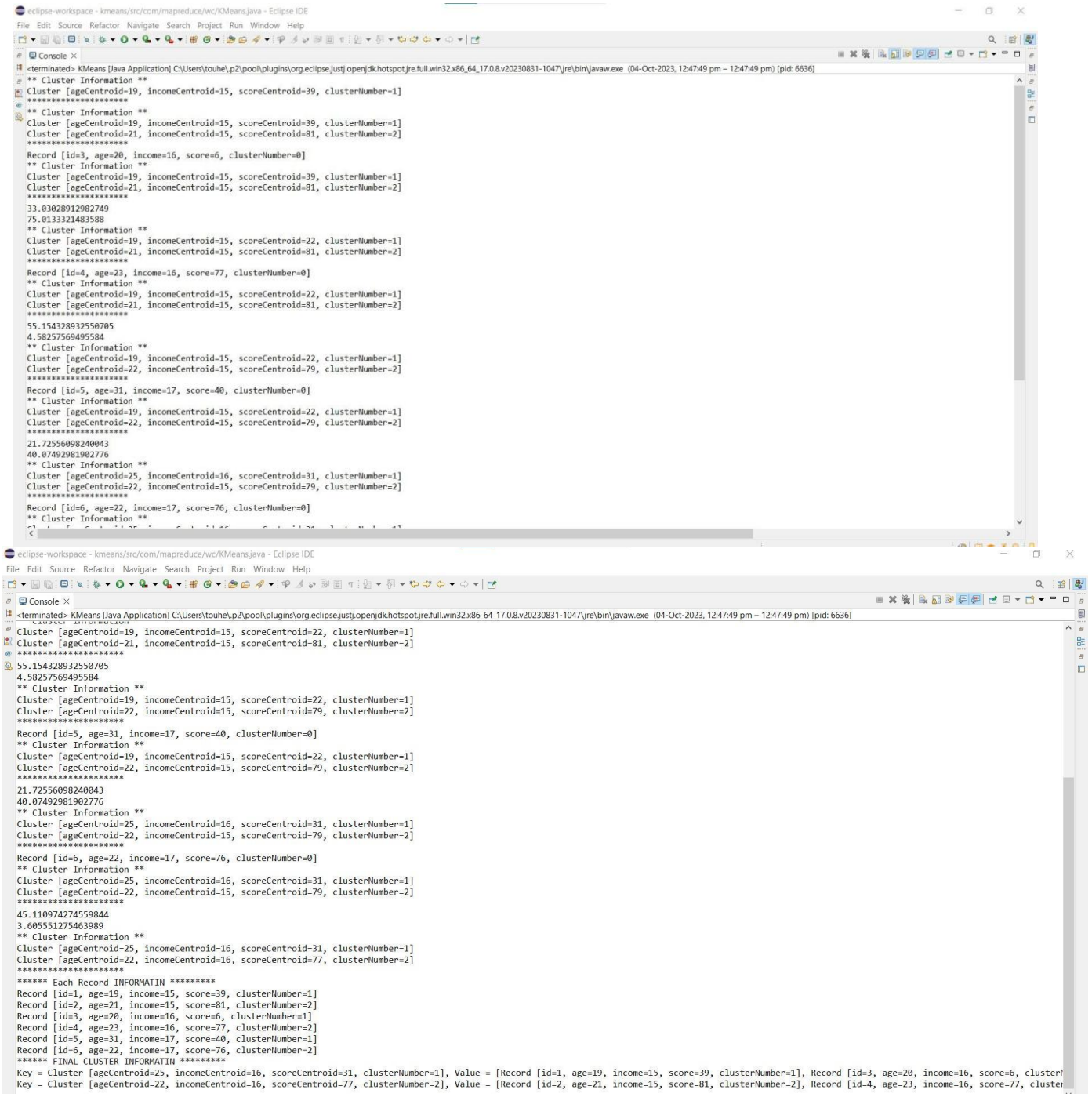
```
package
com.mapreduce.wc;
public class Record {
```

```
private int id;
private int age;
private int income;
private int score;
private int clusterNumber
public Record(int id, int age, int income, int score) {
        super();
        this.id = id;
        this.age = age;
        this.income = income;
        this.score = score;
}
public int getId() {
        return id;
}
public void setId(int id)
        { this.id = id;
}
public int getAge() {
        return age;
}
public void setAge(int age) {
        this.age = age;
}
public int getIncome() {
        return income;
}
public void setIncome(int income) {
        this.income = income;
}
public int getScore() {
        return score;
```

```
	}
	public void setScore(int score) {
		this.score = score;
	}
	public int getClusterNumber() {
		return clusterNumber;
	}
	public void setClusterNumber(int clusterNumber) {
		this.clusterNumber = clusterNumber;
}
	@Override
	public String toString() {
	return "Record [id=" + id + ", age=" + age + ", income=" + income + ", score=" + score + ",
      clusterNumber="+ clusterNumber + "]";
	}
}
```

STEP 9: Run the “KMeans.java” class from the project to get the output.

OUTPUT:



RESULT: Thus the program to implement K-means clustering using mapreduce is executed the output is verified successfully.

EXP NO:06 **DATE:**

FIND-S ALGORITHM

AIM: To implement the Find-S algorithm for finding the most specific hypothesis that is consistent with a given set of training examples.

ALGORITHM:

STEP 1: Initialize the hypothesis h to the most specific hypothesis (all values = 0).

STEP 2: For each training example:

- If the example is **positive**:
  - For each attribute:
    - If h[j] is 0, set h[j] = value.
    - Else if h[j] != value, set h[j] = ?.
- If the example is **negative**, ignore it.

STEP 3: After processing all examples, output the final hypothesis h.

PROGRAM:

```
import pandas as pd
def find_s_algorithm(filename):
    # Load CSV
    data = pd.read_csv("D:\\bharathcode\\deeplearning\\data.csv")

    # Separate features and target
    attributes = data.iloc[:, :-1].values   # all columns except last
    target = data.iloc[:, -1].values        # last column

    # Step 1: Initialize most specific hypothesis
    hypothesis = ['0'] * (attributes.shape[1])

    # Step 2: Iterate through training examples
    for i, val in enumerate(attributes):
        if target[i].lower() == "yes":  # positive example
            for j in range(len(hypothesis)):
                if hypothesis[j] == '0':
                    hypothesis[j] = val[j]
                elif hypothesis[j] != val[j]:
                    hypothesis[j] = '?'
        return hypothesis
if __name__ == "__main__":
    final_hypothesis = find_s_algorithm("training_data.csv")
print("Final Hypothesis:", final_hypothesis)
```

OUTPUT:

Final Hypothesis: `['Sunny', 'Warm', '?', 'Strong', '?', '?']`

RESULT: Thus the program to implement the Find-S algorithm for finding the most specific hypothesis that is consistent with a given set of training examples was executed and output verified successfully

EXP NO:07 **DATE:**

CANDIDATE ELIMINATION ALGORITHM

AIM: To implement the Candidate Elimination Algorithm for determining the version space bounded by the most specific hypothesis S and the most general hypothesis G.

ALGORITHM:

STEP 1: Initialize:

- S = most specific hypothesis (['0', '0', ..., '0'])
- G = most general hypothesis (['?', '?', ..., '?'])

STEP 2: For each training example:

- If the example is **positive**:
  - Remove inconsistent hypotheses from G.
  - Generalize S minimally to include the example.
- If the example is **negative**:
  - Remove inconsistent hypotheses from S.
  - Specialize G minimally to exclude the example.

STEP 3: Continue until all examples are processed.

STEP 4: Output the final boundaries of the version space (S and G).

PROGRAM:

```
import pandas as pd
def candidate_elimination(filename):
    # Load dataset
    data = pd.read_csv("D:\\bharathcode\\deeplearning\\data.csv")
    concepts = data.iloc[:, :-1].values
    target = data.iloc[:, -1].values
    # Step 1: Initialize S and G
    S = [['0'] * len(concepts[0])]
    G = [['?'] * len(concepts[0])]
    print("\nInitial S:", S)
    print("Initial G:", G)
    # Step 2: Process each training example
    for i, h in enumerate(concepts):
        if target[i].lower() == "yes":  # positive example
            # Remove from G inconsistent hypotheses
            G = [g for g in G if all(g[j] in ['?', h[j]] for j in range(len(h)))]
            # Generalize S
            for j in range(len(S[0])):
                if S[0][j] == '0':
                    S[0][j] = h[j]
                elif S[0][j] != h[j]:
                    S[0][j] = '?'
        else:  # negative example
            # Remove from S inconsistent hypotheses
            if all(S[0][j] in ['?', h[j]] for j in range(len(h))):
                # Specialize G
                new_G = []
                for j in range(len(S[0])):
                    if S[0][j] == '?':
                        new_hypothesis = S[0].copy()
                        new_hypothesis[j] = h[j] + "_not"  # mark specialization
                        new_G.append(new_hypothesis)
                G.extend(new_G)
        print(f"\nAfter example {i+1} ({h}, {target[i]}):")
        print("S:", S)
        print("G:", G)
    return S, G
if __name__ == "__main__":
    S_final, G_final = candidate_elimination("training_data.csv")
    print("\nFinal Specific Boundary (S):", S_final)
    print("Final General Boundary (G):", G_final)
```

OUTPUT:

```
Initialization of specific_h and general_h
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 1
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 2
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 3
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', 'Same']]

Steps of Candidate Elimination Algorithm 4
['Sunny' 'Warm' '?' 'Strong' '?' '?']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]

Final Specific_h:
['Sunny' 'Warm' '?' 'Strong' '?' '?']

Final General_h:
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
```

RESULT: Thus the program To implement the Candidate Elimination Algorithm for determining the version space bounded by the most specific hypothesis S and the most general hypothesis G.

EXP NO:08 **DATE:**

ID3 ALGORITHM

AIM: To implement the **ID3 algorithm** for decision tree learning using the concept of entropy and information gain.

ALGORITHM:

STEP 1: If all training examples belong to the same class, return that class (leaf node).

STEP 2: If no features remain, return the majority class.

STEP 3: Otherwise:

- Calculate **entropy** of the dataset.
- For each feature, compute **information gain**.
- Select the feature with the highest information gain as the root node.

STEP 4: For each possible value of the chosen feature:

- Split the dataset into subsets.
- Recursively apply the ID3 algorithm to build subtrees.

STEP 5: Continue until the tree is complete.

STEP 6: Use the decision tree to classify new examples.

PROGRAM:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
import math

def entropy(y):
    counter = Counter(y)
    total = len(y)
    return -sum((count/total) * math.log2(count/total) for count in counter.values())


def info_gain(data, feature, target):
    total_entropy = entropy(data[target])
    values = data[feature].unique()
    weighted_entropy = 0

    for v in values:
        subset = data[data[feature] == v]
        weighted_entropy += (len(subset)/len(data)) * entropy(subset[target])
    return total_entropy - weighted_entropy

def id3(data, features, target):
    # If all examples have the same label, return it
    if len(set(data[target])) == 1:
        return list(data[target])[0]

    # If no features left, return majority class
    if len(features) == 0:
        return Counter(data[target]).most_common(1)[0][0]

    # Choose best feature
    gains = [info_gain(data, f, target) for f in features]
    best_feature = features[np.argmax(gains)]

    tree = {best_feature: {}}
    remaining_features = [f for f in features if f != best_feature]

    for value in data[best_feature].unique():
        subset = data[data[best_feature] == value]
        subtree = id3(subset, remaining_features, target)
```

```
            tree[best_feature][value] = subtree

        return tree

    def predict(tree, sample):
        if not isinstance(tree, dict):  # Leaf node
            return tree
        root = next(iter(tree))
        value = sample.get(root)
        if value in tree[root]:
            return predict(tree[root][value], sample)
        else:
            return None

    def plot_tree(tree, depth=0, indent="  "):
        if not isinstance(tree, dict):
            print(indent * depth + f"--> {tree}")
            return
        for key, value in tree.items():
            print(indent * depth + str(key))
            for k in value:
                print(indent * (depth+1) + f"[{k}]")
                plot_tree(value[k], depth+2, indent)

    if __name__ == "__main__":
        # Example dataset (Play Tennis)
        data = pd.read_csv("D:\\bharathcode\\deeplearning\\tennis.csv")
        target = 'PlayTennis'
        features = list(data.columns[:-1])

        # Build Decision Tree
        decision_tree = id3(data, features, target)

        print("\nDecision Tree:")
        plot_tree(decision_tree)

        # Classify a new sample
        new_sample = {'Outlook': 'Sunny', 'Temperature': 'Cool', 'Humidity': 'High', 'Wind': 'Strong'}
        prediction = predict(decision_tree, new_sample)
        print("\nNew Sample:", new_sample)
        print("Prediction:", prediction)
```

OUTPUT:

```
Outlook
    Overcast
        b'Yes'
    Rainy
        Windy
            b'False'
                b'Yes'
            b'True'
                b'No'
    Sunny
        Humidity
            b'High'
                b'No'
            b'Normal'
                b'Yes'
```

RESULT : Thus the program to implement the ID3 algorithm for decision tree learning using the concept of entropy and information gain.

EXP NO:09 **DATE:**

BACKPROPAGATION ALGORITHM

AIM: To implement a simple Artificial Neural Network (ANN) using NumPy for predicting output from given input data by performing forward and backward propagation (training process) with one hidden layer.

ALGORITHM:

STEP 1: Input training data X and target output y.

STEP 2: Normalize input data X and output y.

STEP 3: Initialize network parameters:

- Input layer size = 2
- Hidden layer size = 3
- Output layer size = 1
- Randomly assign weights W1 and W2.

STEP 4: Forward Propagation:

- Compute z = X * W1
- Apply activation: z2 = sigmoid(z)
- Compute z3 = z2 * W2
- Apply activation: o = sigmoid(z3) (final output)

STEP 5: Backward Propagation:

- Compute output delta: o_delta = error * sigmoidPrime(o)
- Compute hidden layer error: z2_error = o_delta * W2^T
- Compute hidden delta: z2_delta = z2_error * sigmoidPrime(z2)

STEP 6: Update Weights:

- W1 = W1 + X^T * z2_delta
- W2 = W2 + z2^T * o_delta

STEP 7: Repeat Steps 5–8 for multiple iterations (training epochs).

STEP 8: Output the final predicted values and loss.

PROGRAM:

```
import numpy as np
# X = (hours sleeping, hours studying)
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
# y = score on test
y = np.array(([92], [86], [89]), dtype=float)

# scale units
X = X/np.amax(X, axis=0)       # maximum of X array
y = y/100                      # max test score is 100

class Neural_Network(object):
   def __init__(self):
      # Parameters
      self.inputSize = 2
      self.outputSize = 1
      self.hiddenSize = 3
                        # Weights
      self.W1 = np.random.randn(self.inputSize, self.hiddenSize)
        # (3x2) weight matrix from input to hidden layer
      self.W2 = np.random.randn(self.hiddenSize, self.outputSize)
        # (3x1) weight matrix from hidden to output layer

   def forward(self, X):
      #forward propagation through our network
      self.z = np.dot(X, self.W1)
        # dot product of X (input) and first set of 3x2 weights
      self.z2 = self.sigmoid(self.z)
        # activation function
      self.z3 = np.dot(self.z2, self.W2)
          # dot product of hidden layer (z2) and second set of 3x1 weights
      o = self.sigmoid(self.z3)
        # final activation function
      return o

   def sigmoid(self, s):
      return 1/(1+np.exp(-s))
 # activation function
   def sigmoidPrime(self, s):
      return s * (1 - s)
# derivative of sigmoid
```

```
        def backward(self, X, y, o):
        # backward propgate through the network
        self.o_error = y - o
          # error in output
        self.o_delta = self.o_error*self.sigmoidPrime(o)
          # applying derivative of sigmoid to
        self.z2_error = self.o_delta.dot(self.W2.T)
          # z2 error: how much our hidden layer weights contributed to output error
        self.z2_delta = self.z2_error*self.sigmoidPrime(self.z2)
          # applying derivative of sigmoid to z2 error
        self.W1 += X.T.dot(self.z2_delta)
           # adjusting first set (input --> hidden) weights
        self.W2 += self.z2.T.dot(self.o_delta)
          # adjusting second set (hidden --> output) weights

    def train (self, X, y):
        o = self.forward(X)
        self.backward(X, y, o)

NN = Neural_Network()
print ("\nInput: \n" + str(X))
print ("\nActual Output: \n" + str(y))
print ("\nPredicted Output: \n" + str(NN.forward(X)))
print ("\nLoss: \n" + str(np.mean(np.square(y - NN.forward(X)))))
          # mean sum squared loss)
NN.train(X, y)
```

OUTPUT:

```
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]

Actual Output:
[[0.92]
 [0.86]
 [0.89]]

Predicted Output:
[[0.55398066]
 [0.49831918]
 [0.50254468]]

Loss:
0.13830159742519685
```

RESULT: Thus the program To implement a simple Artificial Neural Network (ANN) using NumPy for predicting output from given input data by performing forward and backward propagation (training process) with one hidden layer.

EXP NO:10 **DATE:**

NAÏVE BAYESIAN CLASSIFIER

AIM: To implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

ALGORITHM:

STEP 1: Import required libraries (pandas, sklearn.tree, LabelEncoder, GaussianNB).
STEP 2: Load dataset (tennisdata.csv) into a pandas DataFrame.
STEP 3: Separate features (X) and target (y) from the dataset.
STEP 4: Encode categorical feature columns (Outlook, Temperature, Humidity, Windy) using LabelEncoder.
STEP 5: Encode the target column (PlayTennis) using LabelEncoder.
STEP 6: Split the dataset into training and testing sets using train_test_split().
STEP 7: Create a GaussianNB classifier.
STEP 8: Train the classifier using the training data (fit).
STEP 9: Predict the output for the test data.
STEP 10: Calculate and display the model accuracy using accuracy_score().

PROGRAM:

```
# import necessary libarities
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

# load data from CSV
data = pd.read_csv('tennisdata.csv')
print("THe first 5 values of data is :\n",data.head())

# obtain Train data and Train output
X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())

y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())

# Convert then in numbers
le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)

le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)

le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)

le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)

print("\nNow the Train data is :\n",X.head())

le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20)

classifier = GaussianNB()
classifier.fit(X_train,y_train)

from sklearn.metrics import accuracy_score
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))
```

OUTPUT:

```
 The First 5 values of train data is
      Outlook Temperature Humidity  Windy
 0     Sunny         Hot     High  False
 1     Sunny         Hot     High   True
 2  Overcast         Hot     High  False
 3     Rainy        Mild     High  False
 4     Rainy        Cool   Normal  False

THe first 5 values of data is :
     Outlook Temperature Humidity  Windy PlayTennis
0     Sunny         Hot     High  False         No
1     Sunny         Hot     High   True         No
2  Overcast         Hot     High  False        Yes
3     Rainy        Mild     High  False        Yes
4     Rainy        Cool   Normal  False        Yes

The first 5 values of Train output is
 0     No
1     No
2    Yes
3    Yes
4    Yes
Name: PlayTennis, dtype: object

 Now the Train data is :
     Outlook  Temperature  Humidity  Windy
 0        2            1         0      0
 1        2            1         0      1
 2        0            1         0      0
 3        1            2         0      0
 4        1            0         1      0

Now the Train output is
 [0 0 1 1 1 0 1 0 1 1 1 1 1 0]

Accuracy is: 1.0
```

RESULT: Thus the program To implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.