**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**Vision**

* To achieve excellence in the domain of Artificial Intelligence and Data Science and produce globally competent professionals to solve futuristic societal challenges and industrial needs

**Mission**

* To actively engage in the implementation of innovative intelligent solutions for interdisciplinary Artificial Intelligence based applications with ethical standards
* To promote research, innovation and entrepreneurial skills through industry and academic collaboration

**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO I:** Exhibit proficiency in their career, higher studies and research with strong foundations in Mathematics, Computing, Artificial Intelligence and Data Science.

**PEO II:** Apply Artificial Intelligence and Data Science knowledge and skills to develop innovative solutions for multi-disciplinary problems, adhering to ethical standards

**PEO III:** Engage in constructive research, professional development and life-long learning with skills in emerging technologies

**PROGRAM OUTCOMES (POs)**

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineeringfundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complexengineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problemsand design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge andresearch methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, andmodern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge toassess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineeringsolutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilitiesand norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member orleader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with theengineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of theengineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engagein independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO 1:** Analyze, design and build sustainable intelligent solutions to solve challenges imposed by industry and society.

**PSO 2:** Demonstrate data analysis skills to achieve effective insights and decision making to solve real-life problems.

**PSO 3:** Apply mathematical and statistical models to solve the computational tasks, and model real-world problems using appropriate AI / ML algorithms.

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EX.NO** | **DATE** | **NAME OF THE PROGRAM** | **PAGE NO** | **SIGN** |
| 1 |  | Installation, Configuration, and Running of Hadoop and HDFS |  |  |
| 2 |  | Implementation of Word Count / Frequency Programs using MapReduce |  |  |
| 3 |  | Implementation of MR Program that processes a Weather Dataset |  |  |
| 4(a) |  | Implementation of Linear Regression |  |  |
| 4(b) | Implementation of Logistic Regression |  |
| 5(a) |  | Implementation of SVM Classification Technique |  |  |
| 5(b) | Implementation of Decision Tree Classification Technique |  |
| 6(a) |  | Implementation of Hierarchical Clustering |  |  |
| 6(b) | Implementation of Partitioning Clustering |  |
| 6(c) | Implementation of Fuzzy Clustering |  |
| 7(a) |  | Implementation of Density Based Clustering |  |  |
| 7(b) | Implementation of Model Based Clustering |  |
| 8(a) |  | Data Visualization using Pie Chart Plotting Framework |  |  |
| 8(b) | Data Visualization using Bar Chart Plotting Framework |  |
| 9(a) |  | Data Visualization using Boxplot Plotting Framework |  |  |
| 9(b) | Data Visualization using Histogram Plotting Framework |  |
| 10(a) |  | Data Visualization using Line Graph Plotting Framework |  |  |
| 10(b) | Data Visualization using Scatterplot Plotting Framework |  |
| 11(a) |  | Application to adjust the Number of Bins in the Histogram using R Language |  |  |
| 11(b) | Application To Analyze Stock Market Data Using R Language |  |
| **CONTENT BEYOND THE SYLLABUS** | | | | |
|  |  | Apache Flink |  |  |

|  |  |
| --- | --- |
| **EX.NO: 1** | **INSTALLATION, CONFIGURATION, AND RUNNING OF HADOOP AND HDFS** |
| **DATE:** |

**AIM:**

To install a single-node Hadoop cluster backed by the Hadoop Distributed File System on Ubuntu.

**PROCEDURE:**

**1. Installing Java**

**prince@prince-VirtualBox:~$ cd ~**

Update the source list

**prince@prince-VirtualBox:~$ sudo apt-get update**

The OpenJDK project is the default version of Java

that is provided from a supported Ubuntu repository.

**prince@prince-VirtualBox:~$ sudo apt-get install default-jdk**

**prince@prince-VirtualBox:~$ java -version**

java version "1.7.0\_65"

OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1)

OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)

**2. Adding a dedicated Hadoop user**

**prince@prince-VirtualBox:~$ sudoaddgrouphadoop**

Adding group `hadoop' (GID 1002) ...

Done.

**prince@prince-VirtualBox:~$ sudoadduser --ingrouphadoophduser**

Adding user `hduser' ...

Adding new user `hduser' (1001) with group `hadoop' ...

Creating home directory `/home/hduser' ...

Copying files from `/etc/skel' ...

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

Changing the user information for hduser

Enter the new value, or press ENTER for the default

Full Name []:

Room Number []:

Work Phone []:

Home Phone []:

Other []:

Is the information correct? [Y/n] Y

**3. Installing SSH**

**ssh** has two main components:

1. **ssh** : The command we use to connect to remote machines - the client.
2. **sshd** : The daemon that is running on the server and allows clients to connect to the server.

The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install **ssh** first. Use this command to do that :

**prince@prince-VirtualBox:~$ sudo apt-get install ssh**

This will install ssh on our machine. If we get something similar to the following, we can think it is setup properly:

**prince@prince-VirtualBox:~$ which ssh**

/usr/bin/ssh

**prince@prince-VirtualBox:~$ which sshd**

/usr/sbin/sshd

**4. Create and Setup SSH Certificates**

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.

So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

**prince@prince-VirtualBox:~$ suhduser**

Password:

**prince@prince-VirtualBox:~$ ssh-keygen -t rsa -P ""**

Generating public/private rsa key pair.

Enter file in which to save the key (/home/hduser/.ssh/id\_rsa):

Created directory '/home/hduser/.ssh'.

Your identification has been saved in /home/hduser/.ssh/id\_rsa.

Your public key has been saved in /home/hduser/.ssh/id\_rsa.pub.

The key fingerprint is:

50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3**hduser@prince-VirtualBox**

The key's randomart image is:

+--[ RSA 2048]----+

| .oo.o |

| . .o=. o |

| . + . o . |

| o = E |

| S + |

| . + |

| O + |

| O o |

| o.. |

+-----------------+

**hduser@prince-VirtualBox:/home/k$ cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys**

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.

We can check if ssh works:

**hduser@prince-VirtualBox:/home/k$ sshlocalhost**

The authenticity of host 'localhost (127.0.0.1)' can't be established.

ECDSA key fingerprint is e1:8b:a0:a5:75:ef:f4:b4:5e:a9:ed:be:64:be:5c:2f.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-40-generic x86\_64)

...

**5. Install Hadoop**

**hduser@prince-VirtualBox:~$ wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz**

**hduser@prince-VirtualBox:~$ tar xvzf hadoop-2.6.0.tar.gz**

We want to move the Hadoop installation to the **/usr/local/hadoop** directory using the following command:

**hduser@prince-VirtualBox:~/hadoop-2.6.0$ sudo mv \* /usr/local/hadoop**

[sudo] password for hduser:

hduser is not in the sudoers file. This incident will be reported.

Oops!... We got:

"hduser is not in the sudoers file. This incident will be reported."

This error can be resolved by logging in as a root user, and then add **hduser** to **sudo**:

**hduser@prince-VirtualBox:~/hadoop-2.6.0$ su prince**

Password:

**prince@prince-VirtualBox:/home/hduser$ sudoadduserhdusersudo**

[sudo] password for prince:

Adding user `hduser' to group `sudo' ...

Adding user hduser to group sudo

Done.

Now, the **hduser** has root priviledge, we can move the Hadoop installation to the **/usr/local/hadoop** directory without any problem:

**prince@prince-VirtualBox:/home/hduser$ sudosuhduser**

**hduser@prince-VirtualBox:~/hadoop-2.6.0$ sudo mv \* /usr/local/hadoop**

**hduser@prince-VirtualBox:~/hadoop-2.6.0$ sudochown -R hduser:hadoop /usr/local/hadoop**

**6. Setup Configuration Files**

The following files will have to be modified to complete the Hadoop setup:

**i.~/.bashrc**

**ii./usr/local/hadoop/etc/hadoop/hadoop-env.sh**

**iii./usr/local/hadoop/etc/hadoop/core-site.xml**

**iv./usr/local/hadoop/etc/hadoop/mapred-site.xml.template**

**v./usr/local/hadoop/etc/hadoop/hdfs-site.xml**

**i. ~/.bashrc**:

Before editing the **.bashrc** file in our home directory, we need to find the path where Java has been installed to set the **JAVA\_HOME** environment variable using the following command:

**hduser@prince-VirtualBox:~$ update-alternatives --config java**

There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java

Nothing to configure.

Now we can append the following to the end of **~/.bashrc**:

**hduser@prince-VirtualBox:~$ nano ~/.bashrc**

#HADOOP VARIABLES START

export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64

export HADOOP\_INSTALL=/usr/local/hadoop

export PATH=$PATH:$HADOOP\_INSTALL/bin

export PATH=$PATH:$HADOOP\_INSTALL/sbin

export HADOOP\_MAPRED\_HOME=$HADOOP\_INSTALL

export HADOOP\_COMMON\_HOME=$HADOOP\_INSTALL

export HADOOP\_HDFS\_HOME=$HADOOP\_INSTALL

export YARN\_HOME=$HADOOP\_INSTALL

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_INSTALL/lib/native

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_INSTALL/lib"

#HADOOP VARIABLES END

**hduser@prince-VirtualBox:~$ source ~/.bashrc**

note that the JAVA\_HOME should be set as the path just before the '.../bin/':

**hduser@ubuntu-VirtualBox:~$ javac -version**

javac 1.7.0\_75

**hduser@ubuntu-VirtualBox:~$ which javac**

/usr/bin/javac

**hduser@ubuntu-VirtualBox:~$ readlink -f /usr/bin/javac**

/usr/lib/jvm/java-7-openjdk-amd64/bin/javac

**ii. /usr/local/hadoop/etc/hadoop/hadoop-env.sh**

We need to set **JAVA\_HOME** by modifying **hadoop-env.sh** file.

**hduser@prince-VirtualBox:~$ nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh**

export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64

Adding the above statement in the **hadoop-env.sh** file ensures that the value of JAVA\_HOME variable will be available to Hadoop whenever it is started up.

**iii. /usr/local/hadoop/etc/hadoop/core-site.xml**:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up.   
This file can be used to override the default settings that Hadoop starts with.

**hduser@prince-VirtualBox:~$ sudomkdir -p /app/hadoop/tmp**

**hduser@prince-VirtualBox:~$ sudochownhduser:hadoop /app/hadoop/tmp**

Open the file and enter the following in between the <configuration></configuration> tag:

**hduser@prince-VirtualBox:~$ nano /usr/local/hadoop/etc/hadoop/core-site.xml**

<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>/app/hadoop/tmp</value>

<description>A base for other temporary directories.</description>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:54310</value>

<description>The name of the default file system. A URI whose

scheme and authority determine the FileSystem implementation. The

uri's scheme determines the config property (fs.SCHEME.impl) naming

theFileSystem implementation class. The uri's authority is used to

determine the host, port, etc. for a filesystem.</description>

</property>

</configuration>

**iv. /usr/local/hadoop/etc/hadoop/mapred-site.xml**

By default, the **/usr/local/hadoop/etc/hadoop/** folder contains   
**/usr/local/hadoop/etc/hadoop/mapred-site.xml.template**  
file which has to be renamed/copied with the name **mapred-site.xml**:

**hduser@prince-VirtualBox:~$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml**

The **mapred-site.xml** file is used to specify which framework is being used for MapReduce.  
We need to enter the following content in between the <configuration></configuration> tag:

<configuration>

<property>

<name>mapred.job.tracker</name>

<value>localhost:54311</value>

<description>The host and port that the MapReduce job tracker runs

at. If "local", then jobs are run in-process as a single map

and reduce task.

</description>

</property>

</configuration>

**v. /usr/local/hadoop/etc/hadoop/hdfs-site.xml**

The **/usr/local/hadoop/etc/hadoop/hdfs-site.xml** file needs to be configured for each host in the cluster that is being used.   
It is used to specify the directories which will be used as the **namenode** and the **datanode** on that host.

Before editing this file, we need to create two directories which will contain the namenode and the datanode for this Hadoop installation.   
This can be done using the following commands:

**hduser@prince-VirtualBox:~$ sudomkdir -p /usr/local/hadoop\_store/hdfs/namenode**

**hduser@prince-VirtualBox:~$ sudomkdir -p /usr/local/hadoop\_store/hdfs/datanode**

**hduser@prince-VirtualBox:~$ sudochown -R hduser:hadoop /usr/local/hadoop\_store**

Open the file and enter the following content in between the <configuration></configuration> tag:

**hduser@prince-VirtualBox:~$ nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml**

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication.

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

</description>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop\_store/hdfs/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop\_store/hdfs/datanode</value>

</property>

</configuration>

**7. Format the NewHadoop File system**

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current** directory   
under **/usr/local/hadoop\_store/hdfs/namenode** folder:

**hduser@prince-VirtualBox:~$ hadoopnamenode -format**

DEPRECATED: Use of this script to execute hdfs command is deprecated.

Instead use the hdfs command for it.

15/04/18 14:43:03 INFO namenode.NameNode: STARTUP\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

STARTUP\_MSG: Starting NameNode

STARTUP\_MSG: host = laptop/192.168.1.1

STARTUP\_MSG: args = [-format]

STARTUP\_MSG: version = 2.6.0

STARTUP\_MSG: classpath = /usr/local/hadoop/etc/hadoop

...

STARTUP\_MSG: java = 1.7.0\_65

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

15/04/18 14:43:03 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT]

15/04/18 14:43:03 INFO namenode.NameNode: createNameNode [-format]

15/04/18 14:43:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Formatting using clusterid: CID-e2f515ac-33da-45bc-8466-5b1100a2bf7f

15/04/18 14:43:09 INFO namenode.FSNamesystem: No KeyProvider found.

15/04/18 14:43:09 INFO namenode.FSNamesystem: fsLock is fair:true

15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.block.invalidate.limit=1000

15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.namenode.datanode.registration.ip-hostname-check=true

15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.namenode.startup.delay.block.deletion.sec is set to 000:00:00:00.000

15/04/18 14:43:10 INFO blockmanagement.BlockManager: The block deletion will start around 2015 Apr 18 14:43:10

15/04/18 14:43:10 INFO util.GSet: Computing capacity for map BlocksMap

15/04/18 14:43:10 INFO util.GSet: VM type = 64-bit

15/04/18 14:43:10 INFO util.GSet: 2.0% max memory 889 MB = 17.8 MB

15/04/18 14:43:10 INFO util.GSet: capacity = 2^21 = 2097152 entries

15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.block.access.token.enable=false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: defaultReplication = 1

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplication = 512

15/04/18 14:43:10 INFO blockmanagement.BlockManager: minReplication = 1

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplicationStreams = 2

15/04/18 14:43:10 INFO blockmanagement.BlockManager: shouldCheckForEnoughRacks = false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: replicationRecheckInterval = 3000

15/04/18 14:43:10 INFO blockmanagement.BlockManager: encryptDataTransfer = false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxNumBlocksToLog = 1000

15/04/18 14:43:10 INFO namenode.FSNamesystem: fsOwner = hduser (auth:SIMPLE)

15/04/18 14:43:10 INFO namenode.FSNamesystem: supergroup = supergroup

15/04/18 14:43:10 INFO namenode.FSNamesystem: isPermissionEnabled = true

15/04/18 14:43:10 INFO namenode.FSNamesystem: HA Enabled: false

15/04/18 14:43:10 INFO namenode.FSNamesystem: Append Enabled: true

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map INodeMap

15/04/18 14:43:11 INFO util.GSet: VM type = 64-bit

15/04/18 14:43:11 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB

15/04/18 14:43:11 INFO util.GSet: capacity = 2^20 = 1048576 entries

15/04/18 14:43:11 INFO namenode.NameNode: Caching file names occuring more than 10 times

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map cachedBlocks

15/04/18 14:43:11 INFO util.GSet: VM type = 64-bit

15/04/18 14:43:11 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB

15/04/18 14:43:11 INFO util.GSet: capacity = 2^18 = 262144 entries

15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.9990000128746033

15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0

15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension = 30000

15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache on namenode is enabled

15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map NameNodeRetryCache

15/04/18 14:43:11 INFO util.GSet: VM type = 64-bit

15/04/18 14:43:11 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB

15/04/18 14:43:11 INFO util.GSet: capacity = 2^15 = 32768 entries

15/04/18 14:43:11 INFO namenode.NNConf: ACLs enabled? false

15/04/18 14:43:11 INFO namenode.NNConf: XAttrs enabled? true

15/04/18 14:43:11 INFO namenode.NNConf: Maximum size of anxattr: 16384

15/04/18 14:43:12 INFO namenode.FSImage: Allocated new BlockPoolId: BP-130729900-192.168.1.1-1429393391595

15/04/18 14:43:12 INFO common.Storage: Storage directory /usr/local/hadoop\_store/hdfs/namenode has been successfully formatted.

15/04/18 14:43:12 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid>= 0

15/04/18 14:43:12 INFO util.ExitUtil: Exiting with status 0

15/04/18 14:43:12 INFO namenode.NameNode: SHUTDOWN\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SHUTDOWN\_MSG: Shutting down NameNode at laptop/192.168.1.1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Note that **hadoopnamenode -format** command should be executed once before we start using Hadoop.   
If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

**8. Starting Hadoop**

Now it's time to start the newly installed single node cluster.   
We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

**prince@prince-VirtualBox:~$ cd /usr/local/hadoop/sbin**

**prince@prince-VirtualBox:/usr/local/hadoop/sbin$ ls**

distribute-exclude.sh start-all.cmd stop-balancer.sh

hadoop-daemon.sh start-all.sh stop-dfs.cmd

hadoop-daemons.sh start-balancer.sh stop-dfs.sh

hdfs-config.cmd start-dfs.cmd stop-secure-dns.sh

hdfs-config.sh start-dfs.sh stop-yarn.cmd

httpfs.sh start-secure-dns.sh stop-yarn.sh

kms.sh start-yarn.cmd yarn-daemon.sh

mr-jobhistory-daemon.sh start-yarn.sh yarn-daemons.sh

refresh-namenodes.sh stop-all.cmd

slaves.sh stop-all.sh

**prince@prince-VirtualBox:/usr/local/hadoop/sbin$ sudosuhduser**

**hduser@prince-VirtualBox:/usr/local/hadoop/sbin$ start-all.sh**

**hduser@prince-VirtualBox:~$ start-all.sh**

This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh

15/04/18 16:43:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting namenodes on [localhost]

localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-laptop.out

localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-laptop.out

Starting secondary namenodes [0.0.0.0]

0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-laptop.out

15/04/18 16:43:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

starting yarn daemons

startingresourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-laptop.out

localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-laptop.out

We can check if it's really up and running:

**hduser@prince-VirtualBox:/usr/local/hadoop/sbin$ jps**

9026 NodeManager

7348 NameNode

9766 Jps

8887 ResourceManager

7507 DataNode

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

Another way to check is using **netstat**:

**hduser@prince-VirtualBox:~$ netstat -plten | grep java**

(Not all processes could be identified, non-owned process info

will not be shown, you would have to be root to see it all.)

tcp 0 0 0.0.0.0:50020 0.0.0.0:\* LISTEN 1001 1843372 10605/java

tcp 0 0 127.0.0.1:54310 0.0.0.0:\* LISTEN 1001 1841277 10447/java

tcp 0 0 0.0.0.0:50090 0.0.0.0:\* LISTEN 1001 1841130 10895/java

tcp 0 0 0.0.0.0:50070 0.0.0.0:\* LISTEN 1001 1840196 10447/java

tcp 0 0 0.0.0.0:50010 0.0.0.0:\* LISTEN 1001 1841320 10605/java

tcp 0 0 0.0.0.0:50075 0.0.0.0:\* LISTEN 1001 1841646 10605/java

tcp6 0 0 :::8040 :::\* LISTEN 1001 1845543 11383/java

tcp6 0 0 :::8042 :::\* LISTEN 1001 1845551 11383/java

tcp6 0 0 :::8088 :::\* LISTEN 1001 1842110 11252/java

tcp6 0 0 :::49630 :::\* LISTEN 1001 1845534 11383/java

tcp6 0 0 :::8030 :::\* LISTEN 1001 1842036 11252/java

tcp6 0 0 :::8031 :::\* LISTEN 1001 1842005 11252/java

tcp6 0 0 :::8032 :::\* LISTEN 1001 1842100 11252/java

tcp6 0 0 :::8033 :::\* LISTEN 1001 1842162 11252/java

**9. Stopping Hadoop**

**$ pwd**

/usr/local/hadoop/sbin

**$ ls**

distribute-exclude.sh httpfs.sh start-all.sh start-yarn.cmd stop-dfs.cmd yarn-daemon.sh

hadoop-daemon.sh mr-jobhistory-daemon.sh start-balancer.sh start-yarn.sh stop-dfs.sh yarn-daemons.sh

hadoop-daemons.sh refresh-namenodes.sh start-dfs.cmd stop-all.cmd stop-secure-dns.sh

hdfs-config.cmd slaves.sh start-dfs.sh stop-all.sh stop-yarn.cmd

hdfs-config.sh start-all.cmd start-secure-dns.sh stop-balancer.sh stop-yarn.sh

We run **stop-all.sh** or (**stop-dfs.sh** and **stop-yarn.sh**) to stop all the daemons running on our machine:

**hduser@prince-VirtualBox:/usr/local/hadoop/sbin$ pwd**

/usr/local/hadoop/sbin

**hduser@prince-VirtualBox:/usr/local/hadoop/sbin$ ls**

distribute-exclude.sh httpfs.sh start-all.cmd start-secure-dns.sh stop-balancer.sh stop-yarn.sh

hadoop-daemon.sh kms.sh start-all.sh start-yarn.cmd stop-dfs.cmd yarn-daemon.sh

hadoop-daemons.sh mr-jobhistory-daemon.sh start-balancer.sh start-yarn.sh stop-dfs.sh yarn-daemons.sh

hdfs-config.cmd refresh-namenodes.sh start-dfs.cmd stop-all.cmd stop-secure-dns.sh

hdfs-config.sh slaves.sh start-dfs.sh stop-all.sh stop-yarn.cmd

**hduser@prince-VirtualBox:/usr/local/hadoop/sbin$ stop-all.sh**

This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh

15/04/18 15:46:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Stopping namenodes on [localhost]

localhost: stopping namenode

localhost: stopping datanode

Stopping secondary namenodes [0.0.0.0]

0.0.0.0: no secondarynamenode to stop

15/04/18 15:46:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

stopping yarn daemons

stoppingresourcemanager

localhost: stopping nodemanager

noproxyserver to stop

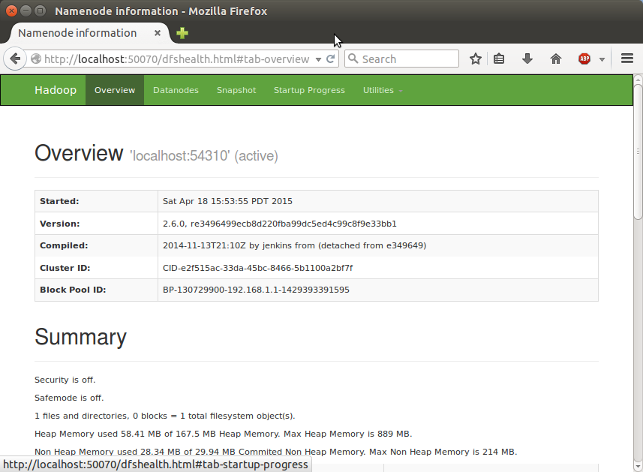
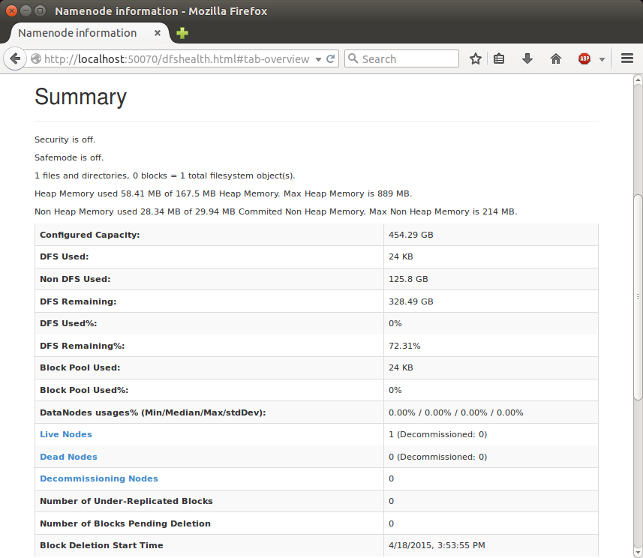
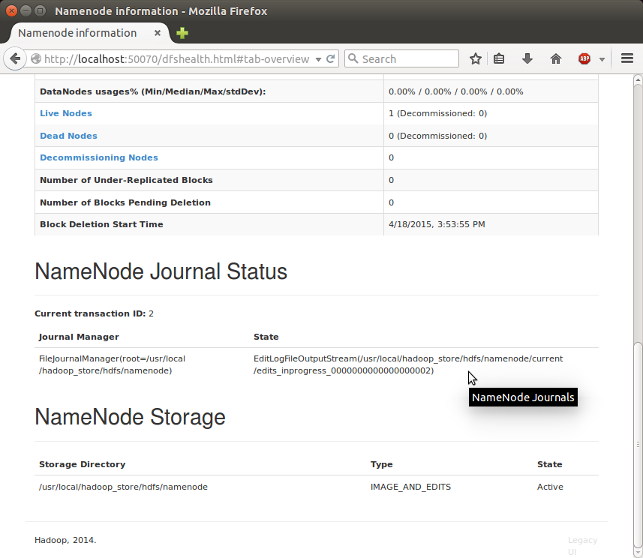
**10.Hadoop Web Interfaces**

Let's start the Hadoop again and see its Web UI:

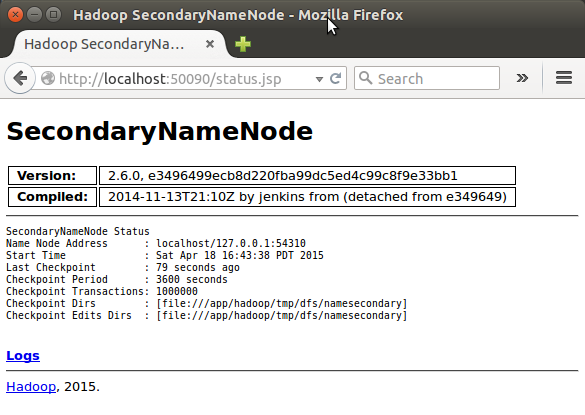
**hduser@prince-VirtualBox:/usr/local/hadoop/sbin$ start-all.sh**

**http://localhost:50070/ - web UI of the NameNode daemon**

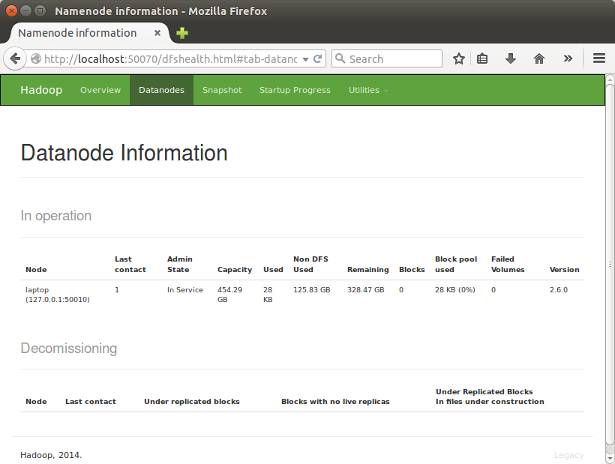
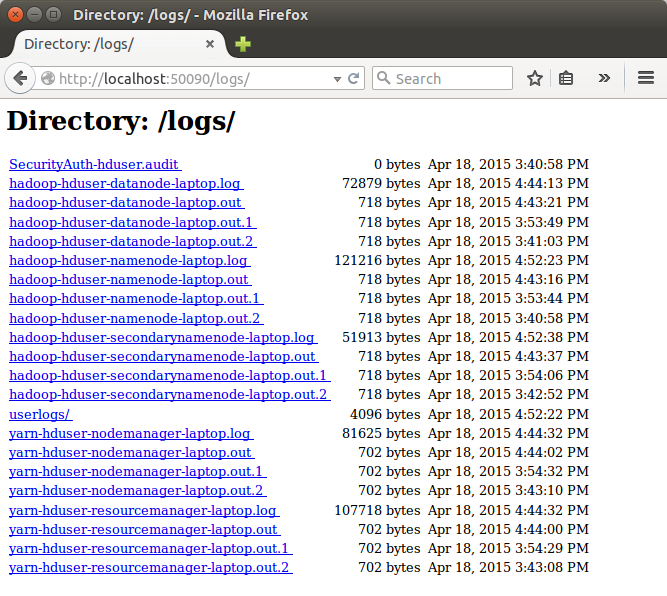
**OUTPUT:**

**Secondary Name Node**



**Data Node**

**RESULT:**

Thus, the single-node Hadoop cluster backed by the Hadoop Distributed File System on Ubuntu is installed successfully.

|  |  |
| --- | --- |
| **EX.NO:2** | **IMPLEMENTATION OF WORD COUNT / FREQUENCY PROGRAMS USING MAPREDUCE** |
| **DATE:** |

**AIM:**

To write a java program for counting the number of occurrences of each word in a text file using the MapReduce concepts.

**PROCEDURE:**

1. Install hadoop.

2. Start all services using the command.

hduser@prince-VirtualBox:/usr/local/hadoop/bin$ jps

3242 Jps

hduser@prince-VirtualBox:/usr/local/hadoop/bin$ **start-all.sh**

This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh

16/09/15 15:38:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting namenodes on [localhost]

localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-prince-VirtualBox.out

localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-prince-VirtualBox.out

Starting secondary namenodes [0.0.0.0]

0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-prince-VirtualBox.out

16/09/15 15:39:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

starting yarn daemons

startingresourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-prince-VirtualBox.out

localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-prince-VirtualBox.out

hduser@prince-VirtualBox:/usr/local/hadoop/bin$ **jps**

16098 NameNode

16214 DataNode

16761 NodeManager

16636 ResourceManager

16429 SecondaryNameNode

19231 Jps

**PROGRAM CODING:**

hduser@prince-VirtualBox:/usr/local/hadoop/bin$ **nano wordcount7.java**

importjava.io.IOException;

importjava.util.StringTokenizer;

importorg.apache.hadoop.conf.Configuration;

importorg.apache.hadoop.fs.Path;

importorg.apache.hadoop.io.IntWritable;

importorg.apache.hadoop.io.Text;

importorg.apache.hadoop.mapreduce.Job;

importorg.apache.hadoop.mapreduce.Mapper;

importorg.apache.hadoop.mapreduce.Reducer;

importorg.apache.hadoop.mapreduce.lib.input.FileInputFormat;

importorg.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class wordcount7 {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

StringTokenizeritr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

privateIntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritableval : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(wordcount7.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**TO COMPILE:**

hduser@prince-VirtualBox:/usr/local/hadoop/bin$ **hadoopcom.sun.tools.javac.Main wordcount7.java**

**TO CREATE A JAR FILE:**

hduser@prince-VirtualBox:/usr/local/hadoop/bin$ **jar cf wc2.jar wordcount7\*.java**

**TO CREATE A DIRECTORY IN HDFS:**

hduser@prince-VirtualBox:/usr/local/hadoop/bin$**hadoopdfs -mkdir /deepika**

**TO LOAD INPUT FILE:**

hduser@prince-VirtualBox:/usr/local/hadoop/bin$**hdfs -put /home/prince/Downloads/wc.txt /deepika/wc1**

**TO EXECUTE:**

@prince-VirtualBox:/usr/local/hadoop/bin$ **hadoop jar wc2.jar wordcount7 /deepika/wc1.txt /deepika/out2**

16/09/16 14:34:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

16/09/16 14:34:17 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id

16/09/16 14:34:17 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=

16/09/16 14:34:17 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

16/09/16 14:34:17 INFO input.FileInputFormat: Total input paths to process : 1

16/09/16 14:34:17 INFO mapreduce.JobSubmitter: number of splits:1

16/09/16 14:34:18 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_local364071501\_0001

16/09/16 14:34:18 INFO mapreduce.Job: The url to track the job: http://localhost:8080/

16/09/16 14:34:18 INFO mapreduce.Job: Running job: job\_local364071501\_0001

16/09/16 14:34:18 INFO mapred.LocalJobRunner: OutputCommitter set in config null

16/09/16 14:34:19 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter

16/09/16 14:34:19 INFO mapred.LocalJobRunner: Waiting for map tasks

16/09/16 14:34:19 INFO mapred.LocalJobRunner: Starting task: attempt\_local364071501\_0001\_m\_000000\_0

16/09/16 14:34:19 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]

16/09/16 14:34:19 INFO mapred.MapTask: Processing split: hdfs://localhost:54310/deepika/wc1:0+712

16/09/16 14:34:19 INFO mapreduce.Job: Job job\_local364071501\_0001 running in ubermode : false

16/09/16 14:34:23 INFO mapreduce.Job: map 0% reduce 0%

16/09/16 14:34:24 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)

16/09/16 14:34:24 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100

16/09/16 14:34:24 INFO mapred.MapTask: soft limit at 83886080

16/09/16 14:34:24 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600

16/09/16 14:34:24 INFO mapred.MapTask: kvstart = 26214396; length = 6553600

16/09/16 14:34:24 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer

16/09/16 14:34:26 INFO mapred.LocalJobRunner:

16/09/16 14:34:26 INFO mapred.MapTask: Starting flush of map output

16/09/16 14:34:26 INFO mapred.MapTask: Spilling map output

16/09/16 14:34:26 INFO mapred.MapTask: bufstart = 0; bufend = 1079; bufvoid = 104857600

16/09/16 14:34:26 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26214032(104856128); length = 365/6553600

16/09/16 14:34:26 INFO mapred.MapTask: Finished spill 0

16/09/16 14:34:26 INFO mapred.Task: Task:attempt\_local364071501\_0001\_m\_000000\_0 is done. And is in the process of committing

16/09/16 14:34:26 INFO mapred.LocalJobRunner: map

16/09/16 14:34:26 INFO mapred.Task: Task 'attempt\_local364071501\_0001\_m\_000000\_0' done.

16/09/16 14:34:26 INFO mapred.LocalJobRunner: Finishing task: attempt\_local364071501\_0001\_m\_000000\_0

16/09/16 14:34:26 INFO mapred.LocalJobRunner: map task executor complete.

16/09/16 14:34:26 INFO mapred.LocalJobRunner: Waiting for reduce tasks

16/09/16 14:34:26 INFO mapred.LocalJobRunner: Starting task: attempt\_local364071501\_0001\_r\_000000\_0

16/09/16 14:34:26 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]

16/09/16 14:34:26 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin: org.apache.hadoop.mapreduce.task.reduce.Shuffle@2ee9ab75

16/09/16 14:34:26 INFO mapreduce.Job: map 100% reduce 0%

16/09/16 14:34:26 INFO reduce.MergeManagerImpl: MergerManager: memoryLimit=363285696, maxSingleShuffleLimit=90821424, mergeThreshold=239768576, ioSortFactor=10, memToMemMergeOutputsThreshold=10

16/09/16 14:34:26 INFO reduce.EventFetcher: attempt\_local364071501\_0001\_r\_000000\_0 Thread started: EventFetcher for fetching Map Completion Events

16/09/16 14:34:26 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle output of map attempt\_local364071501\_0001\_m\_000000\_0 decomp: 1014 len: 1018 to MEMORY

16/09/16 14:34:27 INFO reduce.InMemoryMapOutput: Read 1014 bytes from map-output for attempt\_local364071501\_0001\_m\_000000\_0

16/09/16 14:34:27 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 1014, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory ->1014

16/09/16 14:34:27 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning

16/09/16 14:34:27 INFO mapred.LocalJobRunner: 1 / 1 copied.

16/09/16 14:34:27 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs

16/09/16 14:34:27 INFO mapred.Merger: Merging 1 sorted segments

16/09/16 14:34:27 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 991 bytes

16/09/16 14:34:27 INFO reduce.MergeManagerImpl: Merged 1 segments, 1014 bytes to disk to satisfy reduce memory limit

16/09/16 14:34:27 INFO reduce.MergeManagerImpl: Merging 1 files, 1018 bytes from disk

16/09/16 14:34:27 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce

16/09/16 14:34:27 INFO mapred.Merger: Merging 1 sorted segments

16/09/16 14:34:27 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 991 bytes

16/09/16 14:34:27 INFO mapred.LocalJobRunner: 1 / 1 copied.

16/09/16 14:34:27 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords

16/09/16 14:34:30 INFO mapred.Task: Task:attempt\_local364071501\_0001\_r\_000000\_0 is done. And is in the process of committing

16/09/16 14:34:30 INFO mapred.LocalJobRunner: 1 / 1 copied.

16/09/16 14:34:30 INFO mapred.Task: Task attempt\_local364071501\_0001\_r\_000000\_0 is allowed to commit now

16/09/16 14:34:30 INFO output.FileOutputCommitter: Saved output of task 'attempt\_local364071501\_0001\_r\_000000\_0' to hdfs://localhost:54310/deepika/out2/\_temporary/0/task\_local364071501\_0001\_r\_000000

16/09/16 14:34:30 INFO mapred.LocalJobRunner: reduce > reduce

16/09/16 14:34:30 INFO mapred.Task: Task 'attempt\_local364071501\_0001\_r\_000000\_0' done.

16/09/16 14:34:30 INFO mapred.LocalJobRunner: Finishing task: attempt\_local364071501\_0001\_r\_000000\_0

16/09/16 14:34:30 INFO mapred.LocalJobRunner: reduce task executor complete.

16/09/16 14:34:30 INFO mapreduce.Job: map 100% reduce 100%

16/09/16 14:34:31 INFO mapreduce.Job: Job job\_local364071501\_0001 completed successfully

16/09/16 14:34:31 INFO mapreduce.Job: Counters: 38

File System Counters

FILE: Number of bytes read=8552

FILE: Number of bytes written=507858

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=1424

HDFS: Number of bytes written=724

HDFS: Number of read operations=13

HDFS: Number of large read operations=0

HDFS: Number of write operations=4

Map-Reduce Framework

Map input records=10

Map output records=92

Map output bytes=1079

Map output materialized bytes=1018

Input split bytes=99

Combine input records=92

Combine output records=72

Reduce input groups=72

Reduce shuffle bytes=1018

Reduce input records=72

Reduce output records=72

Spilled Records=144

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=111

CPU time spent (ms)=0

Physical memory (bytes) snapshot=0

Virtual memory (bytes) snapshot=0

Total committed heap usage (bytes)=242360320

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=712

File Output Format Counters

Bytes Written=724

**INPUT FILE:**

**wc1.txt**

STEPS:

1. Open an editor and type WordCount program and save as WordCount.java

2. Set the path as export HADOOP\_CLASSPATH=${JAVA\_HOME}/lib/tools.jar

3. To compile the program, bin/hadoopcom.sun.tools.javac.Main WordCount.java

4. Create a jar file, jar cf wc.jar WordCount\*.class

5. Create input files input.txt,input1.txt and input2.txt and create a directory in hdfs, /mit/wordcount/input

6. Move these i/p files to hdfs system, bin/hadoopfs –put /opt/hadoop-2.7.0/input.txt /mit/wordcount/input/input.txt repeat this step for other two i/p files.

7. To execute, bin/hadoop jar wc.jar WordCount /mit/wordcount/input /mit/wordcount/output.

8. The mapreduce result will be available in the output directory.

**OUTPUT:**

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

/mit/wordcount/input 2

/mit/wordcount/input/input.txt 1

/mit/wordcount/output. 1

/opt/hadoop-2.7.0/input.txt 1

1. 1

2. 1

3. 1

4. 1

5. 1

6. 1

7. 1

8. 1

Create 2

HADOOP\_CLASSPATH=${JAVA\_HOME}/lib/tools.jar 1

Move 1

Open 1

STEPS: 1

Set 1

The 1

To 2

WordCount 2

WordCount\*.class 1

WordCount.java 2

a 2

an 1

and 4

as 2

available 1

be 1

bin/hadoop 3

cf 1

com.sun.tools.javac.Main 1

compile 1

create 1

directory 1

directory. 1

editor 1

execute, 1

export 1

file, 1

files 2

files. 1

for 1

fs 1

hdfs 1

hdfs, 1

i/p 2

in 2

input 1

input.txt,input1.txt 1

input2.txt 1

jar 3

mapreduce 1

other 1

output 1

path 1

program 1

program, 1

repeat 1

result 1

save 1

step 1

system, 1

the 3

these 1

this 1

to 1

two 1

type 1

wc.jar 2

will 1

–put 1

**RESULT:**

Thus, the java program for counting the number of occurrences of each word in a text file using the MapReduce concepts is executed successfully.

|  |  |
| --- | --- |
| **EX.NO:3** | **IMPLEMENTATION OF** **AN MR PROGRAM THAT PROCESSES A WEATHER DATASET** |
| **DATE:** |

**AIM:**

To implement an MR program that processes a weather dataset.

**PROGRAM:**

**AverageMapper.java**

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*; import java.io.IOException;

public class AverageMapper extends Mapper <LongWritable, Text, Text, IntWritable>

{

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException

{

String line = value.toString(); String year = line.substring(15,19); int temperature;

if (line.charAt(87)=='+')

temperature = Integer.parseInt(line.substring(88, 92));

else

temperature = Integer.parseInt(line.substring(87, 92));

String quality = line.substring(92, 93);

if(temperature != MISSING && quality.matches("[01459]")) context.write(new Text(year),new IntWritable(temperature));

}

}

**AverageReducer.java**

import org.apache.hadoop.mapreduce.\*;

import java.io.IOException;

public class AverageReducer extends Reducer <Text, IntWritable,Text, IntWritable >

{

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException

{

int max\_temp = 0; int count = 0;

for (IntWritable value : values)

{

max\_temp += value.get(); count+=1;

}

context.write(key, new IntWritable(max\_temp/count));

} }

**AverageDriver.java**

import org.apache.hadoop.io.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AverageDriver

{

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

{

if (args.length != 2)

{

System.err.println("Please Enter the input and output parameters"); System.exit(-1);

}

Job job = new Job(); job.setJarByClass(AverageDriver.class); job.setJobName("Max temperature");

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

job.setMapperClass(AverageMapper.class); job.setReducerClass(AverageReducer.class);

job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);

System.exit(job.waitForCompletion(true)?0:1);

}

}

**RESULT:**

Thus, the MR program that processes a weather dataset is implemented and executed successfully.

|  |  |
| --- | --- |
| **EX.NO:4a** | **IMPLEMENTATION OF LINEAR REGRESSION** |
| **DATE:** |

**AIM:**

To implement the linear regression using R Language.

**PROCEDURE:**

1. Linear regression is used to predict a quantitative outcome variable (y) on the basis of one or multiple predictor variables (x).

2. The goal is to build a mathematical formula that defines y as a function of the x variable.

3. When you build a regression model, you need to assess the performance of the predictive model.

4. Two important metrics are commonly used to assess the performance of the predictive regression model:

5. Root Mean Squared Error, which measures the model prediction error. It corresponds to the average difference between the observed known values of the outcome and the predicted value by the model. RMSE is computed as RMSE = mean((observeds - predicteds)^2) %>% sqrt(). The lower the RMSE, the better the model.

6. R-square, representing the squared correlation between the observed known outcome values and the predicted values by the model. The higher the R2, the better the model.

**PROGRAM:**

X=c(151,174,138,186,128,136,179,163,152,131)

Y=c(63,81,56,91,47,57,76,72,62,48)

plot(X,Y)

relation=lm(Y~X)

print(relation)

print(summary(relation))

a=data.frame(X=170)

result=predict(relation,a)

print(result)

png(file="linearregression.png")

plot(Y,X,col="green",main="Height & Weight Regression",abline(lm(X~Y)),

cex=1.3,pch=16,Xlab="Weight in kg",Ylab="Height in cm")

dev.off()

**OUTPUT:**

> a=data.frame(X=170)

>result=predict(relation,a)

>print(result)

1

76.22869

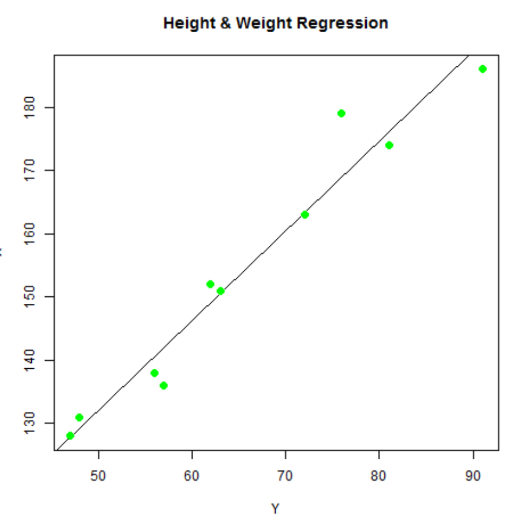
>png(file="linearregression.png")

>plot(Y,X,col="green",main="Height & Weight Regression",abline(lm(X~Y)),

cex=1.3,pch=16,Xlab="Weight in kg",Ylab="Height in cm")

>dev.off()

RStudioGD



**RESULT:**

Thus, the implementation of linear regression was executed and verified successfully.

|  |  |
| --- | --- |
| **EX.NO:4b** | **IMPLEMENTATION OF LOGISTIC REGRESSION** |
| **DATE:** |

**AIM:**

To implement the logistic regression using R programming language.

**PROCEDURE:**

1. Logistic regression is used to predict the class of individuals based on one or multiple predictor variables (x).

2. It is used to model a binary outcome, that is a variable, which can have only two

possible values: 0 or 1, yes or no, diseased or non-diseased.

3. Logistic regression belongs to a family, named Generalized Linear Model (GLM),

developed for extending the linear regression model to other situations.

4. Other synonyms are binary logistic regression, binomial logistic regression and logit model.

5. Logistic regression does not return directly the class of observations. It allows us to estimate the probability (p) of class membership. The probability will range between

0 and 1.

**PROGRAM:**

input=mtcars[,c("am","cyl","hp","wt")]

am.data=glm(formula=am~cyl+hp+wt,data=input,family = binomial)

print(summary(am.data))

**OUTPUT:**

Call:

glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.17272 -0.14907 -0.01464 0.14116 1.27641

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 19.70288 8.11637 2.428 0.0152 \*

cyl 0.48760 1.07162 0.455 0.6491

hp 0.03259 0.01886 1.728 0.0840 .

wt -9.14947 4.15332 -2.203 0.0276 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 43.2297 on 31 degrees of freedom

Residual deviance: 9.8415 on 28 degrees of freedom

AIC: 17.841

Number of Fisher Scoring iterations: 8

**RESULT:**

Thus, the implementation of logistic regression was executed and verified successfully.

|  |  |
| --- | --- |
| **EX.NO:5a** | **IMPLEMENTATION OF SVM CLASSIFICATION TECHNIQUE** |
| **DATE:** |

**AIM:**

To implement SVM Classification using R Language.

**PROCEDURE:**

1. To use SVM in R, we have a package e1071.

2. The package is not preinstalled, hence one needs to run the line

“install.packages(“e1071”) to install the package.

3. Then import the package contents using the library command--library(e1071)

**PROGRAM:**

x=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)

y=c(3,4,5,4,8,10,10,11,14,20,23,24,32,34,35,37,42,48,53,60)

#Create a data frame of the data

train=data.frame(x,y)

#Plot the dataset

plot(train,pch=16)

#Linear regression

model<- lm(y ~ x, train)

#Plot the model using abline

abline(model)

#SVM

library(e1071)

#Fit a model. The function syntax is very similar to lm function

model\_svm<- svm(y ~ x , train)

#Use the predictions on the data

pred<- predict(model\_svm, train)

#Plot the predictions and the plot to see our model fit

points(train$x, pred, col = "blue", pch=4)

error<- model$residuals

lm\_error<- sqrt(mean(error^2)) # 3.832974

predictions (pred)

error\_2 <- train$y - pred

svm\_error<- sqrt(mean(error\_2^2)) # 2.696281

svm\_tune<- tune(svm, y ~ x, data = train,

ranges = list(epsilon = seq(0,1,0.01), cost = 2^(2:9)))

print(svm\_tune)

best\_mod<- svm\_tune$best.model

best\_mod\_pred<- predict(best\_mod, train)

error\_best\_mod<- train$y - best\_mod\_pred

# this value can be different on your computer

# because the tune method randomly shuffles the data

best\_mod\_RMSE<- sqrt(mean(error\_best\_mod^2)) # 1.290738

plot(svm\_tune)

plot(train,pch=16)

points(train$x, best\_mod\_pred, col = "blue", pch=4)

**OUTPUT:**

Chart, scatter chart

Description automatically generated Chart, scatter chart

Description automatically generated

**RESULT:**

Thus, the implementation of SVM was executed and verified successfully.

|  |  |
| --- | --- |
| **EX.NO:5b** | **IMPLEMENTATION OF DECISSION TREE CLASSIFICATION TECHNIQUE** |
| **DATE:** |

**AIM:**

To implement decision tree classification using R Language.

**PROCEDURE:**

1. Install party packages

a. install.packages("party")

i. it has the ctree function.

2. Create the input data

i. # Create the input data frame.

ii. input.dat <- readingSkills[c(1:105),]

3. Give the chart file a name.

i. png(file = "decision\_tree.png")

4. Create the tree.

i. output.tree <- ctree( nativeSpeaker ~ age + shoeSize + score, data = input.dat)

5. Plot the tree.

i. plot(output.tree)

6. Save the file

i. dev.off()

**PROGRAM:**

library(party)

input.dat <- readingSkills[c(1:105),]

png(file = "decision\_tree.png")

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

plot(output.tree)

dev.off()

**OUTPUT:**

null device

1

Loading required package: methods

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

Diagram

Description automatically generated

**RESULT:**

Thus, the implementation of decision tree classification was executed and verified

successfully.

|  |  |
| --- | --- |
| **EX.NO:6a** | **IMPLEMENTATION OF HIERARCHICAL CLUSTERING** |
| **DATE:** |

**AIM:**

To implement clustering techniques using hierarchical clustering.

**PROCEDURE:**

1. Hierarchical clustering is an alternative approach to partitioning clustering for identifying groups in the dataset.

2. It does not require to pre-specify the number of clusters to be generated.

3. The result of hierarchical clustering is a tree-based representation of the objects, which is JYYalso known as dendrogram.

4. Observations can be subdivided into groups by cutting the dendrogram at a desired

similarity level.

5. R code to compute and visualize hierarchical clustering.

**PROGRAM:**

install.packages("factoextra")

install.packages("cluster")

install.packages("magrittr")

library("factoextra")

library("cluster")

library("magrittr")

res.hc <- USArrests %>%

scale() %>%

# Scale the data

dist(method = "euclidean") %>% # Compute dissimilarity matrix

hclust(method = "ward.D2") # Compute hierachical clustering

# Visualize using factoextra

# Cut in 4 groups and color by groups

fviz\_dend(res.hc, k = 4, # Cut in four groups

cex = 0.5, # label size

k\_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),

color\_labels\_by\_k = TRUE, # color labels by groups

rect = TRUE # Add rectangle around groups

)

**OUTPUT:**

Chart, box and whisker chart

Description automatically generated

**RESULT:**

Thus, the implementation of clustering techniques using hierarchical clustering was

executed and verified successfully.

|  |  |
| --- | --- |
| **EX.NO:6b** | **IMPLEMENTATION OF PARTITIONING CLUSTERING** |
| **DATE:** |

**AIM:**

To implement the clustering techniques using partitioning clustering.

**PROCEDURE:**

1. Partitioning algorithms are clustering techniques that subdivide the data sets into a set of

k groups, where k is the number of groups pre-specified by the analyst.

2. There are different types of partitioning clustering methods. The most popular is the K-

means clustering (MacQueen 1967), in which, each cluster is represented by the center or means of the data points belonging to the cluster. The K-means method is sensitive to outliers.

3. An alternative to k-means clustering is the K-medoids clustering or PAM (Partitioning

Around Medoids, Kaufman & Rousseeuw, 1990), which is less sensitive to outliers compared to k-means.

4. Determining the optimal number of clusters: use factoextra::fviz\_nbclust()

5. Compute and visualize k-means clustering.

**PROGRAM:**

install.packages("factoextra")

install.packages("magrittr")

install.packages("cluster")

library("factoextra")

library("magrittr")

library("cluster")

set.seed(123)

km.res<-kmeans(my\_data, 3, nstart=25)

# Visualize

library("factoextra")

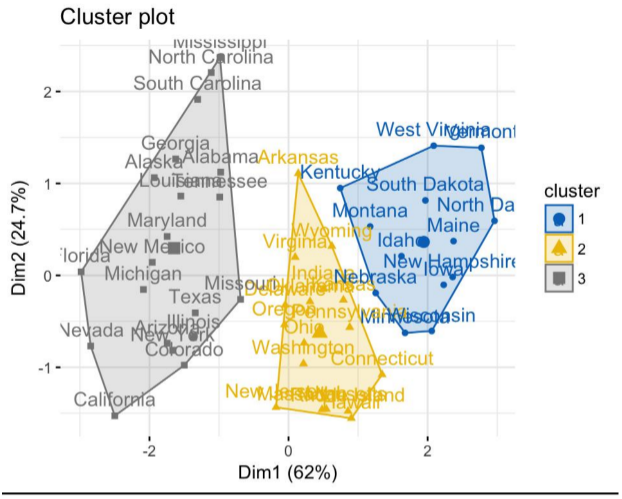
fviz\_cluster(km.res, data=my\_data,

ellipse.type="convex",

palette="jco",

ggtheme=theme\_minimal())

**OUTPUT:**



**RESULT:**

Thus, the implementation of clustering techniques using partitioning clustering was

executed and verified successfully.

|  |  |
| --- | --- |
| **EX.NO:6c** | **IMPLEMENTATION OF FUZZY CLUSTERING** |
| **DATE:** |

**AIM:**

To implement the clustering techniques using fuzzy clustering.

**PROCEDURE:**

1. Fuzzy clustering is also known as soft method. Standard clustering approaches produce partitions (K-means, PAM), in which each observation belongs to only one cluster. This is known as hard clustering.

2. In Fuzzy clustering, items can be a member of more than one cluster. The Fuzzy

c-means method is the most popular fuzzy clustering algorithm.

3. Cluster for computing fuzzy clustering

4. factoextra for visualizing clusters.

5. The function fanny()can be used to compute fuzzy clustering.

6. Compute and visualize fuzzy clustering using the combination

of cluster and factoextra R packages.

**PROGRAM:**

install.packages("factoextra")

install.packages("magrittr")

install.packages("cluster")

library("factoextra")

library("magrittr")

library("cluster")

library(cluster)

df<-scale(USArrests)# Standardize the data

res.fanny<-fanny(df, 2)# Compute fuzzy clustering with k = 2

head(res.fanny$membership, 3)# Membership coefficients

res.fanny$coeff# Dunn's partition coefficient

head(res.fanny$clustering)# Observation groups

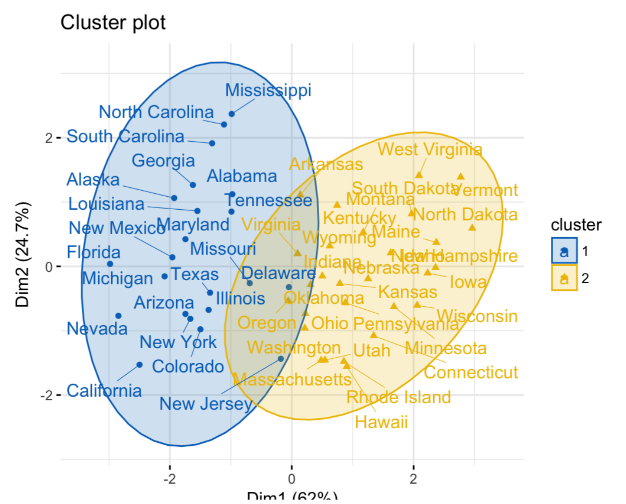
library(factoextra)

fviz\_cluster(res.fanny, ellipse.type="norm", repel=TRUE,

palette="jco", ggtheme=theme\_minimal(),

legend="right")

**OUTPUT:**



**RESULT:**

Thus, the implementation of clustering techniques using fuzzy clustering was executed

and verified successfully.

|  |  |
| --- | --- |
| **EX.NO:7a** | **IMPLEMENTATION OF DENSITY BASED CLUSTERING** |
| **DATE:** |

**AIM:**

To implement the clustering techniques using density-based clustering.

**PROCEDURE:**

1. It can be used to identify clusters of any shape in a data set containing noise and outliers.

2. Clusters are dense regions in the data space, separated by regions of lower density of

points.

3. The simulated data set multishapes is used.

4. The function fviz\_cluster() is used to visualize the clusters.

5. First, install factoextra: install.packages(“factoextra”); then compute and visualize k-means

clustering using the data set multishapes.

1. The goal is to identify dense regions, which can be measured by the number of objects

close to a given point.

**PROGRAM:**

install.packages("factoextra")

install.packages("magrittr")

install.packages("cluster")

library("factoextra")

library("magrittr")

library("cluster")

install.packages("fpc")

install.packages("dbscan")

install.packages("factoextra")

# Load the data

data("multishapes", package="factoextra")

df<-multishapes[, 1:2]

# Compute DBSCAN using fpc package

library("fpc")

set.seed(123)

db<-fpc::dbscan(df, eps=0.15, MinPts=5)

# Plot DBSCAN results

library("factoextra")

fviz\_cluster(db, data=df, stand=FALSE,

ellipse=FALSE,

show.clust.cent=FALSE,

geom="point",palette="jco", ggtheme=theme\_classic())

**OUTPUT:**

Chart, scatter chart

Description automatically generated

**RESULT:**

Thus, the implementation of clustering techniques using density-based clustering was

executed and verified successfully.

|  |  |
| --- | --- |
| **EX.NO:7b** | **IMPLEMENTATION OF MODEL BASED CLUSTERING** |
| **DATE:** |

**AIM:**

To implement the clustering techniques using model-based clustering.

**PROCEDURE:**

* 1. In model-based clustering, the data are viewed as coming from a distribution that is

mixture of two ore more clusters.

2. It finds best fit of models to data and estimates the number of clusters.

3. Install the mclust package as follow: install.packages(“mclust”).

4. Model-based clustering results can be drawn using the base function plot.Mclust() .

5. fviz\_mclust() uses a principal component analysis to reduce the dimensionnality of the

data.

**PROGRAM:**

install.packages("factoextra")

install.packages("cluster")

install.packages("magrittr")

library("cluster")

library("factoextra")

library("magrittr")

library("mclust")

data("diabetes")

head(diabetes, 3)

library(factoextra)

# BIC values used for choosing the number of clusters

fviz\_mclust(mc, "BIC", palette="jco")

# Classification: plot showing the clustering

fviz\_mclust(mc, "classification", geom="point",

pointsize=1.5, palette="jco")

# Classification uncertainty

fviz\_mclust(mc, "uncertainty", palette="jco")

**OUTPUT:**

## ----------------------------------------------------

## Gaussian finite mixture model fitted by EM algorithm

## ----------------------------------------------------

##

## Mclust VVV (ellipsoidal, varying volume, shape, and

orientation) model with 3 components:

##

## log.likelihood n df BIC ICL

## -169 145 29 -483 -501

##

## Clustering table:

## 1 2 3

## 81 36 28

Chart

Description automatically generated Chart

Description automatically generated Diagram

Description automatically generated

**RESULT:**

Thus, the implementation of clustering techniques using model-based clustering was

executed and verified successfully.

|  |  |
| --- | --- |
| **EX.NO:8a** | **DATA VISUALIZATION USING PIE CHART PLOTTING FRAMEWORK** |
| **DATE:** |

**AIM:**

To visualize data using pie chart using plotty framework.

**PROCEDURE:**

1. In R the pie chart is created using the pie() function which takes positive

numbers as a vector input.

2. The additional parameters are used to control labels, color, title etc.

3. The basic syntax for creating a pie-chart using the R is −

i. pie(x, labels, radius, main, col, clockwise)

4. Following is the description of the parameters used −

a. 4.a x is a vector containing the numeric values used in the pie chart.

b. 4.b labels is used to give description to the slices.

c. 4.c radius indicates the radius of the circle of the pie chart.(value between −1 and

+1).

d. 4.d main indicates the title of the chart.

e. 4.e col indicates the color palette.

f. 4.f clockwise is a logical value indicating if the slices are drawn clockwise or anti

clockwise.

5. We will use parameter main to add a title to the chart and another parameter is col which will make use of rainbow colour pallet while drawing the chart. The length of the pallet should be same as the number of values we have for the chart. Hence we use length(x).

**PROGRAM:**

# Create data for the graph.

x <- c(21, 62, 10, 53)

labels<- c("London", "New York", "Singapore", "Mumbai")

# Give the chart file a name.

png(file = "city\_title\_colours.jpg")

# Plot the chart with title and rainbow color pallet.

pie(x, labels, main = "City pie chart", col = rainbow(length(x)))

# Save the file.

dev.off()

**OUTPUT:**

Graphical user interface, chart, application, pie chart

Description automatically generated

**RESULT:**

Thus, the data is visualized using pie chart using the plotty framework.

|  |  |
| --- | --- |
| **EX.NO:8b** | **DATA VISUALIZATION USING BAR PLOT PLOTTING FRAMEWORK** |
| **DATE:** |

**AIM:**

To visualize data using bar plot using plotty framework.

**PROCEDURE:**

1. R uses the function barplot() to create bar charts. R can draw both vertical and horizontal

bars in the bar chart. In bar chart each of the bars can be given different colors.

2. The basic syntax to create a bar-chart in R is −

i. barplot(H, xlab, ylab, main, names.arg, col)

3. Following is the description of the parameters used −

a. H is a vector or matrix containing numeric values used in bar chart.

b. xlab is the label for x axis.

c. ylab is the label for y axis.

d. main is the title of the bar chart.

e. names.arg is a vector of names appearing under each bar.

f. col is used to give colors to the bars in the graph.

4. The main parameter is used to add title. The col parameter is used to add colors to the

bars. The args.name is a vector having same number of values as the input vector

to describe the meaning of each bar.

**PROGRAM:**

# Create the data for the chart.

H <- c(7,12,28,3,41)

M <- c("Mar","Apr","May","Jun","Jul")

# Give the chart file a name.

png(file = "barchart\_months\_revenue.png")

# Plot the bar chart.

barplot(H,names.arg = M,xlab = "Month",ylab = "Revenue",col = "blue",

main = "Revenue chart",border = "red")

# Save the file.

dev.off()

**OUTPUT:**

Graphical user interface, application

Description automatically generated

**RESULT:**

Thus, the data is visualized using bar plot using the plotty framework.

|  |  |
| --- | --- |
| **EX.NO:9a** | **DATA VISUALIZATION USING BOX PLOT PLOTTING FRAMEWORK** |
| **DATE:** |

**AIM:**

To visualize data using box plot using plotty framework.

**PROCEDURE:**

1. Boxplots are created in R by using the boxplot() function.

2. The basic syntax to create a boxplot in R is

boxplot(x, data, notch, varwidth, names, main)

3. Following is the description of the parameters used

a. x is a vector or a formula.

b. data is the data frame.

c. notch is a logical value. Set as TRUE to draw a notch.

d. Var width is a logical value. Set as true to draw width of the box proportionate to

the sample size.

e. names are the group labels which will be printed under each boxplot.

f. main is used to give a title to the graph.

**PROGRAM:**

# Give the chart file a name.

png(file = "boxplot.png")

# Plot the chart.

boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders",

ylab = "Miles Per Gallon", main = "Mileage Data")

# Save the file.

dev.off()

**OUTPUT:**

Chart, box and whisker chart

Description automatically generated

**RESULT:**

Thus, the data is visualized using box plot using the plotty framework.

|  |  |
| --- | --- |
| **EX.NO:9b** | **DATA VISUALIZATION USING HISTOGRAM PLOTTING FRAMEWORK** |
| **DATE:** |

**AIM:**

To visualize data using histogram using plotty framework.

**PROCEDURE:**

1. R creates histogram using hist() function. This function takes a vector as an input and

uses some more parameters to plot histograms.

2. The basic syntax for creating a histogram using R is −

i. hist(v,main,xlab,xlim,ylim,breaks,col,border)

3. Following is the description of the parameters used −

a. v is a vector containing numeric values used in histogram.

b. main indicates title of the chart.

c. col is used to set color of the bars.

d. border is used to set border color of each bar.

e. xlab is used to give description of x-axis.

f. xlim is used to specify the range of values on the x-axis.

g. ylim is used to specify the range of values on the y-axis.

h. breaks is used to mention the width of each bar.

**PROGRAM:**

# Create data for the graph.

v <- c(9,13,21,8,36,22,12,41,31,33,19)

# Give the chart file a name.

png(file = "histogram.png")

# Create the histogram.

hist(v,xlab = "Weight",col = "yellow",border = "blue")

# Save the file.

dev.off()

**OUTPUT:**

Chart, histogram

Description automatically generated

**RESULT:**

Thus, the data is visualized using histogram using the plotty framework.

|  |  |
| --- | --- |
| **EX.NO:10a** | **DATA VISUALIZATION USING LINE GRAPH PLOTTING FRAMEWORK** |
| **DATE:** |

**AIM:**

To visualize data using line graph using plotty framework.

**PROCEDURE:**

1. The plot() function in R is used to create the line graph.

2. The basic syntax to create a line chart in R is −

i. plot(v,type,col,xlab,ylab)

3. Following is the description of the parameters used −

a. v is a vector containing the numeric values.

b. type takes the value "p" to draw only the points, "l" to draw only the lines and "o" to

draw both points and lines.

c. xlab is the label for x axis.

d. ylab is the label for y axis.

e. main is the Title of the chart.

f. col is used to give colors to both the points and lines.

4. We add color to the points and lines, give a title to the chart and add labels to the axes.

**PROGRAM:**

# Create the data for the chart.

v <- c(7,12,28,3,41)

# Give the chart file a name.

png(file = "line\_chart\_label\_colored.jpg")

# Plot the bar chart.

plot(v,type = "o", col = "red", xlab = "Month", ylab = "Rain fall",main = "Rain fall chart")

# Save the file.

dev.off()

**OUTPUT:**

Chart, line chart

Description automatically generated

**RESULT:**

Thus, the data is visualized using line graph using the plotty framework

|  |  |
| --- | --- |
| **EX.NO:10b** | **DATA VISUALIZATION USING SCATTER PLOT PLOTTING FRAMEWORK** |
| **DATE:** |

**AIM:**

To visualize data using scatter plot using plotty framework.

**PROCEDURE:**

1. The simple scatterplot is created using the plot() function.

2. The basic syntax for creating scatterplot in R is −

i. plot(x, y, main, xlab, ylab, xlim, ylim, axes)

3. Following is the description of the parameters used −

a. x is the data set whose values are the horizontal coordinates.

b. y is the data set whose values are the vertical coordinates.

c. main is the tile of the graph.

d. xlab is the label in the horizontal axis.

e. ylab is the label in the vertical axis.

f. xlim is the limits of the values of x used for plotting.

g. ylim is the limits of the values of y used for plotting.

h. axes indicates whether both axes should be drawn on the plot.

**PROGRAM:**

# Get the input values.

input<- mtcars[,c('wt','mpg')]

# Give the chart file a name.

png(file = "scatterplot.png")

# Plot the chart for cars with weight between 2.5 to 5 and mileage between 15 and 30.

plot(x = input$wt,y = input$mpg,

xlab = "Weight",

ylab = "Milage",

xlim = c(2.5,5),

ylim = c(15,30),

main = "Weight vsMilage")

# Save the file.

dev.off()

**OUTPUT:**

Graphical user interface, chart

Description automatically generated

**RESULT:**

Thus, the data is visualized using scatter plot using the plotty framework.

|  |  |
| --- | --- |
| **EX.NO:11a** | **APPLICATION TO ADJUST THE NUMBER OF BINS IN THE HISTOGRAM USING R LANGUAGE** |
| **DATE:** |

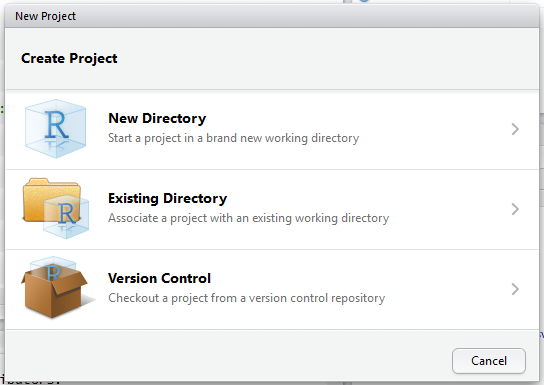
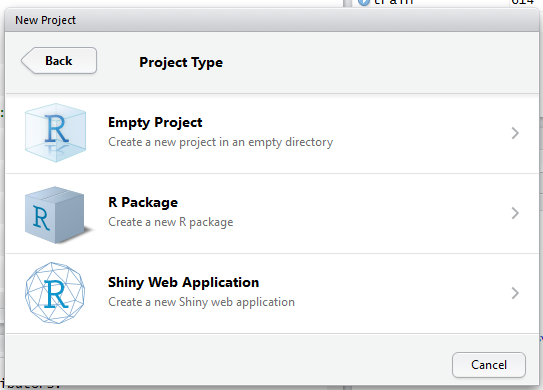
**AIM:**

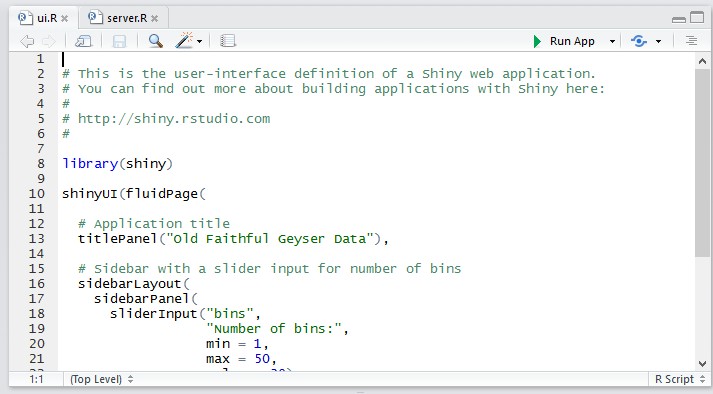
To implement the application to adjust the number of bins in the histogram using r

language.

## PROCEDURE:

Any shiny app is built using two components:

1. **UI.R:** This file creates the user interface in a shiny application. It provides interactivity to the shiny app by taking the input from the user and dynamically displaying the generated output on the screen.
2. **Server.R:** This file contains the series of steps to convert the input given by user into the desired output to be displayed.
   1. Before we proceed further you need to set up Shiny in your system. Follow these steps to get started.
3. Create a new project in R Studio
4. Select type as Shiny web application.
5. It creates two scripts in R Studio named ui.R and server R.



1. Each file needs to be coded separately and the flow of input and output between two is possible.

## PROGRAM:

# This is a Shiny web application. You can run the application by clicking # the 'Run App' button above.

#

# Find out more about building applications with Shiny here: #

# <http://shiny.rstudio.com/>

library(shiny)

# Define UI for application that draws a histogram ui<- fluidPage(

# Application title

titlePanel("Old Faithful Geyser Data"),

# Sidebar with a slider input for number of bins sidebarLayout(

sidebarPanel( sliderInput("bins",

"Number of bins:",

min = 1,

max = 50,

value = 30)

),

# Show a plot of the generated distribution mainPanel(

plotOutput("distPlot")

)

)

)

# Define server logic required to draw a histogram server<- function(input, output) {

output$distPlot<- renderPlot({

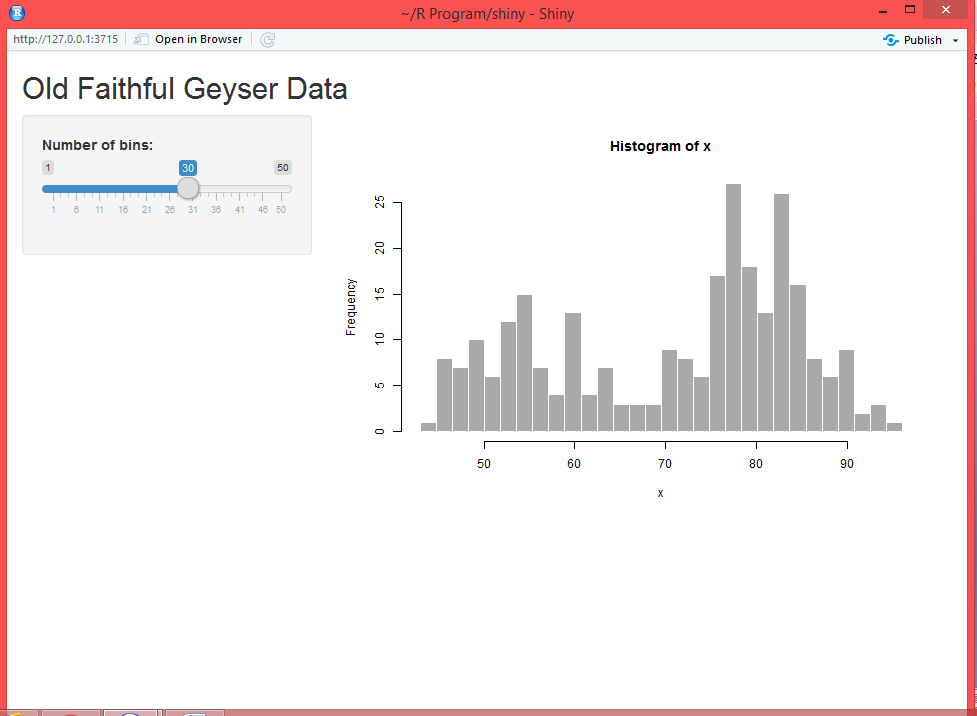
# generate bins based on input$bins from ui.R x <- faithful[, 2]

bins<- seq(min(x), max(x), length.out = input$bins + 1)

# draw the histogram with the specified number of bins hist(x, breaks = bins, col = 'darkgray', border = 'white')

})}

**Output**



## RESULT:

Thus, the application to adjust the number of bins in the histogram using r is implemented.

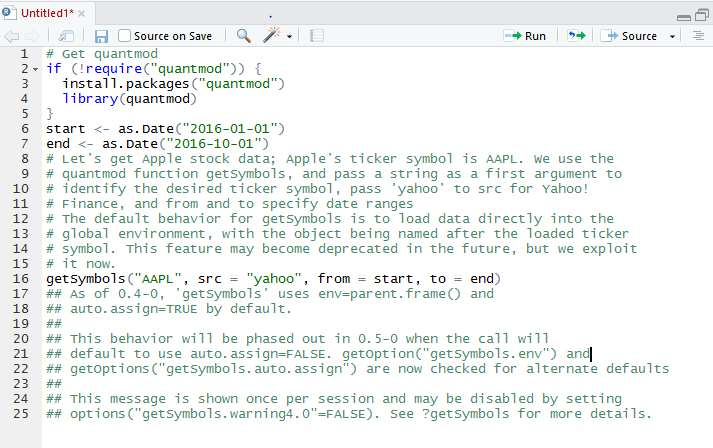
|  |  |
| --- | --- |
| **EX.NO:11b** | **APPLICATION TO ANALYZE STOCK MARKET DATA USING R LANGUAGE** |
| **DATE:** |

**AIM:**

To create an application to analyze Stock Market Data using R language.

## PROCEDURE:

**1a.**Toanalyze stock data, Stock data can be obtained from Yahoo! Finance (http://finance.yahoo.com) by using the quantmod package provides easy access to Yahoo! Finance.



**1b**.getSymbols() can create a object called AAPL in the global environment.



**2a**.The class of AAPL object can be obtained with the command



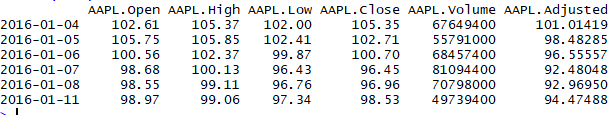
**2b**.AAPL is of the xts class (which is also a zoo-class object). xts objects (provided in the xts package) are seen as improved versions of the ts object for storing time series data.



**3a.**In this stock data’s are stored based on time-based indexing and can provide custom attributes, along with allowing multiple (presumably related) time series with the same time index to be stored in the same object.



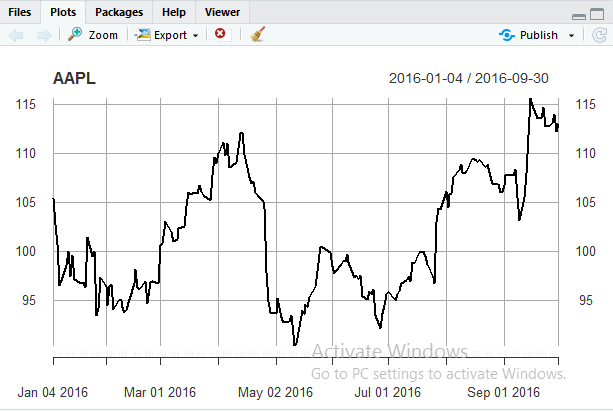
**3b**.Yahoo! Finance provides six series with each security. Open is the price of the stock at the beginning of the trading day, high is the highest price of the stock on that trading day, low the lowest price of the stock on that trading day, and close the price of the stock at closing time. Volume indicates how many stocks were traded. Adjusted is the closing price of the stock that adjusts the price of the stock for corporate actions.



**4a**.Stock data series can be visualized using base R plotting with



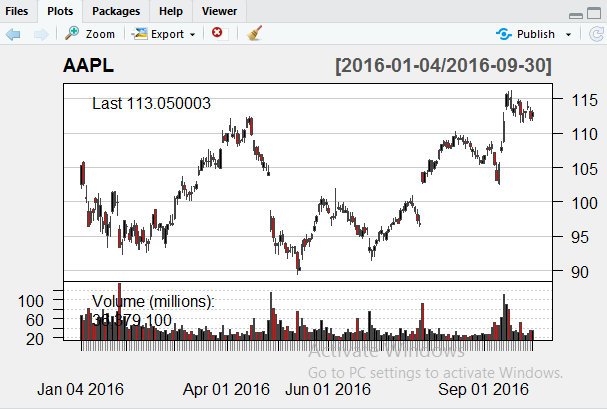
**4b**.Visualization is obtained as



**5a**. Financial data is often plotted with the function called candle Chart() from quant mod to create a chart.



**5b**.With this function, plotting of variables with separate lines as follows



## RESULT:

Thus, an application to analyze Stock Market Data using R language is created successfully.

|  |  |
| --- | --- |
| **DATE:** | **APACHE FLINK** |

**Aim:**

To demonstrate how to submit a sample Flink job and monitor its execution using the Flink web UI.

**Procedure:**

1: Download and extract the latest Flink binary release.

2: Start a local cluster using the provided script.

3: Submit a sample Flink job using the CLI tool.

4: Verify the job output and monitor its execution.

5: Access the Flink web UI to view data flow plan and timeline.

**Process:**

## Downloading Flink

Flink runs on all UNIX-like environments, i.e. Linux, Mac OS X, and Cygwin (for Windows).

System need to have **Java 11** installed. To check the Java version installed, type in terminal:

**$ java -version**

Next, [download the latest binary release](https://flink.apache.org/downloads.html) of Flink, then extract the archive:

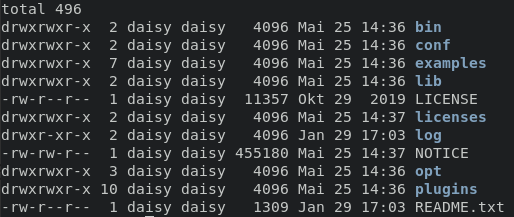
**$ tar -xzf flink-\*.tgz**

## Browsing the project directory

Navigate to the extracted directory and list the contents by issuing:

**$ cd flink-\* && ls -l**

**OUTPUT:**



## Starting and stopping a local cluster

To start a local cluster, run the bash script that comes with Flink:

**$ ./bin/start-cluster.sh**

**OUTPUT**

output

Flink is now running as a background process. Check its status with the following command:

**$ ps aux | grep flink**

Able to navigate to the web UI at [localhost:8081](http://localhost:8081/) to view the Flink dashboard and see that the cluster is up and running.

To quickly stop the cluster and all running components, the provided script:

**$ ./bin/stop-cluster.sh**

## Submitting a Flink job

Flink provides a CLI tool, **bin/flink**, that can run programs packaged as Java ARchives (JAR) and control their execution. Submitting a [job](https://nightlies.apache.org/flink/flink-docs-release-1.19/docs/concepts/glossary/#%ef%ac%82ink-job) means uploading the job’s JAR ﬁle and related dependencies to the running Flink cluster and executing it.

Flink releases come with example jobs, which you can ﬁnd in the **examples/** folder.

To deploy the example word count job to the running cluster, issue the following command:

**$ ./bin/flink run examples/streaming/WordCount.jar**

Verify the output by viewing the logs:

**$ tail log/flink-\*-taskexecutor-\*.out**

**OUTPUT:**

**(**nymph,1**)**

**(**in,3**)**

**(**thy,1**)**

**(**orisons,1**)**

**(**be,4**)**

**(**all,2**)**

**(**my,1**)**

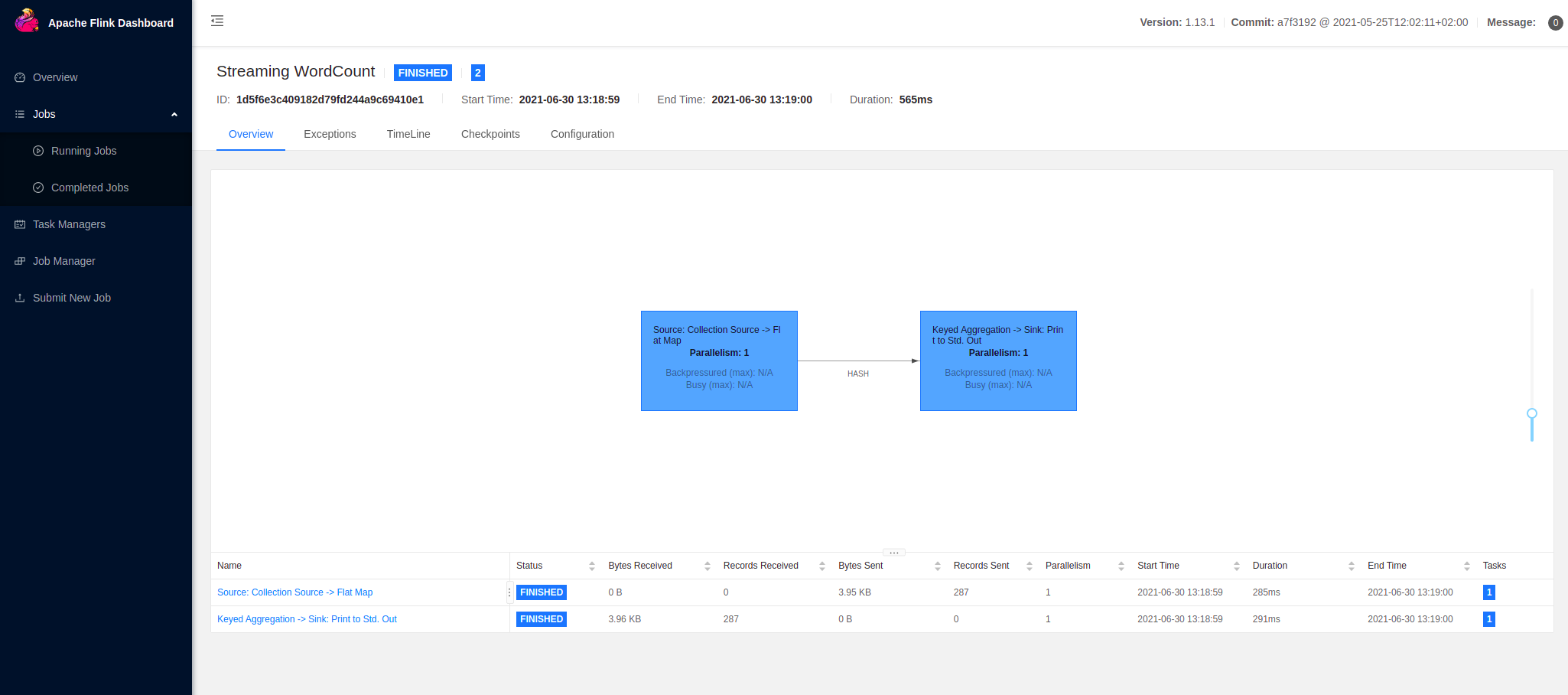
**(**sins,1**)**

**(**remember,1**)**

**(**d,4**)**

Additionally, we can check Flink’s [web UI](http://localhost:8081/) to monitor the status of the cluster and running job.

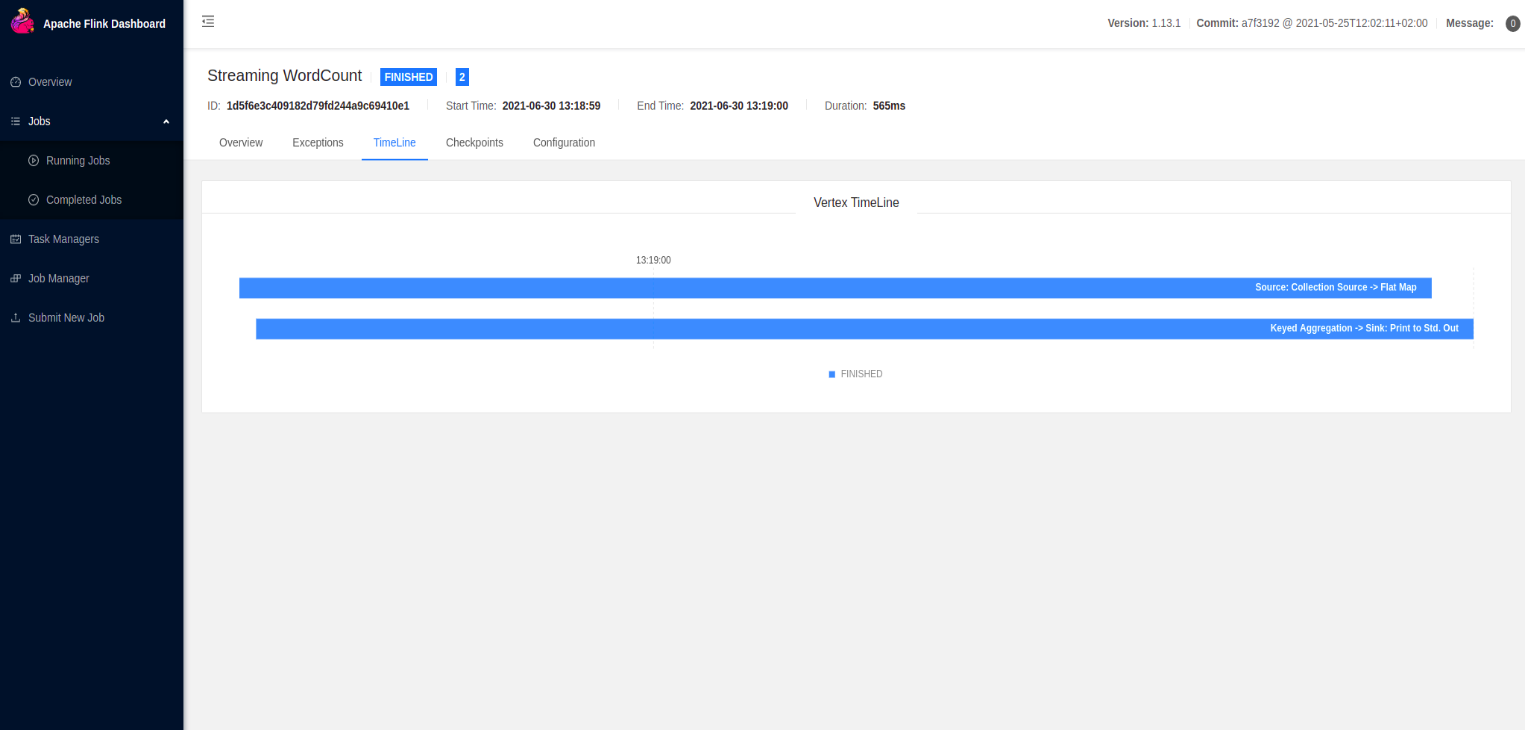
The data flow plan for the execution:



The job execution, Flink has two operators.

* The ﬁrst is the source operator which reads data from the collection source.
* The second operator is the transformation operator which aggregates counts of words

The timeline of the job execution:



**Result:**

The successful execution of a sample Flink job and its monitoring via the Flink web UI confirms proper functionality and proficiency in Flink application management.