# INDEX


| Sr. no | Aim | Date | Page No | Sign |
|---|---|---|---|---|
| 1 | Install, configure and run Hadoop and HDFS and explore HDFS. | | 2 | |
| 2 | Implement word count / frequency programs using MapReduce. | | 9 | |
| 3 | Implement an application that stores big data in Hbase / MongoDB and manipulate it using R / Python. | | 13 | |
| 4 | Implement Decision tree classification techniques. | | 20 | |
| 5 | Implement SVM classification techniques. | | 21 | |
| 6 | REGRESSION MODEL: Import data from web storage. Perform Logistic Regression to find the relationship between variables affecting student admission based on GRE score, GPA, and student rank. Check model fit using require(foreign) and require(MASS). | | 24 | |
| 7 | MULTIPLE REGRESSION MODEL: Apply multiple regressions on the above dataset, if data has a continuous independent variable. | | 26 | |
| 8 | CLASSIFICATION MODEL: a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier. | | 28 | |
| 9 | CLUSTERING MODEL: a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations. | | 30 | |

# Practical 1

**Install, configure and run Hadoop and HDFS ad explore HDFS.**

Steps to Install Hadoop

1. Install Java JDK 1.8

2. Download Hadoop and extract and place under C drive

3. Set Path in Environment Variables

4. Config files under Hadoop directory

5. Create folder datanode and namenode under data directory

6. Edit HDFS and YARN files

7. Set Java Home environment in Hadoop environment

8. Setup Complete. Test by executing start-all.cmd

There are two ways to install Hadoop, i.e.

9. Single node

10. Multi node

Here, we use multi node cluster.

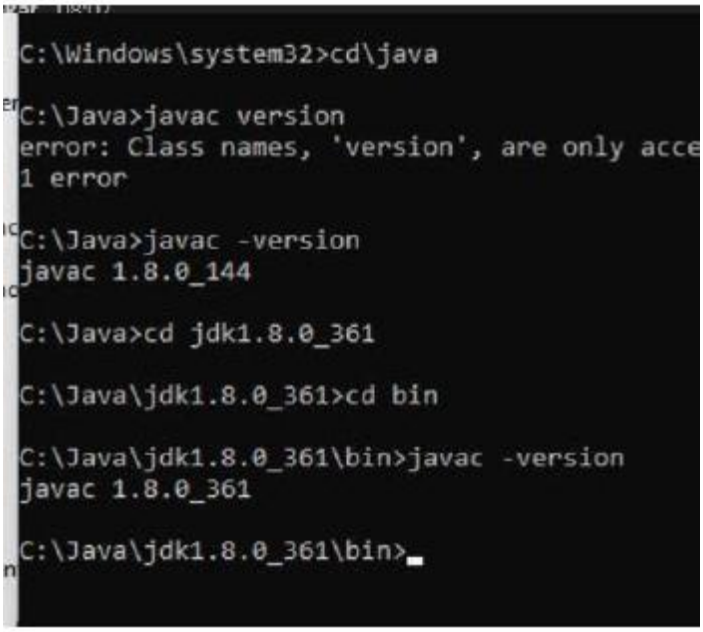
1. Install Java

• – Java JDK Link to download

o https://www.oracle.com/java/technologies/javase-jdk8-downloads.html

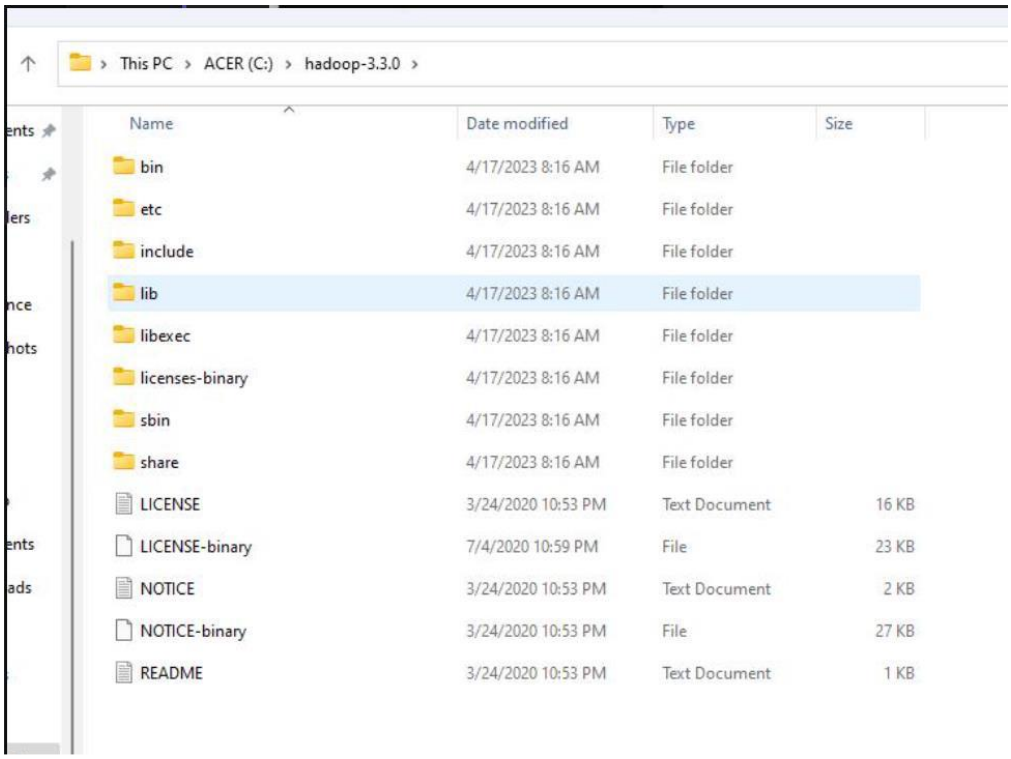
• – extract and install Java in C:\Java

• – open cmd and type -> javac -version



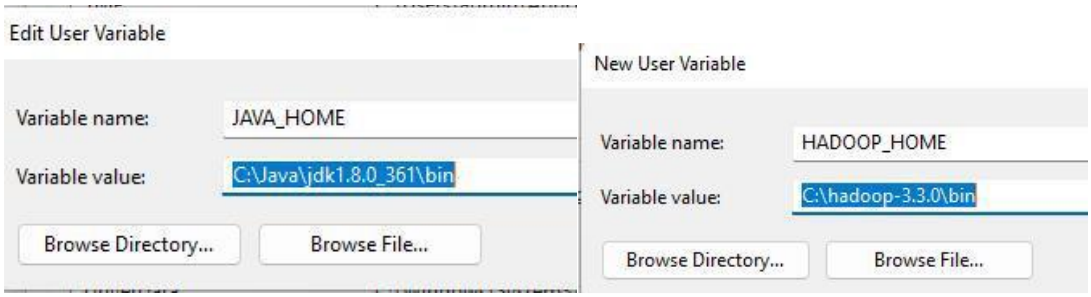
**2. Download Hadoop**

https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz

- • right click .rar.gz file -> show more options -> 7-zip->and extract to C:\Hadoop-3.3.0\
- 


**3. Set the path JAVA_HOME Environment variable**

**4. Set the path HADOOP_HOME Environment variable Click on New to both user variables and system variables. Click on user variable -> path -> edit-> add path for Hadoop and java upto 'bin'**




Click Ok, Ok, Ok.

**5. Configurations**

**Edit file C:/Hadoop-3.3.0/etc/hadoop/core-site.xml,**

paste the xml code in folder and save

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

==========

**Rename "mapred-site.xml.template" to "mapred-site.xml" and edit this file C:/Hadoop-3.3.0/etc/hadoop/mapred-site.xml, paste xml code and save this file.**

==========

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

==========

**Create folder "data" under "C:\Hadoop-3.3.0"**

**Create folder "datanode" under "C:\Hadoop-3.3.0\data"**

**Create folder "namenode" under "C:\Hadoop-3.3.0\data"**

==========

**Edit file C:\Hadoop-3.3.0/etc/hadoop/hdfs-site.xml,**

**paste xml code and save this file.**

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

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

<value>/hadoop-3.3.0/data/namenode</value>

</property>

<property>

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

<value>/hadoop-3.3.0/data/datanode</value>

</property>

</configuration>

==========

**Edit file C:/Hadoop-3.3.0/etc/hadoop/yarn-site.xml,**

**paste xml code and save this file.**

<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>

<property>

<name>yarn.resourcemanager.address</name>

<value>127.0.0.1:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>127.0.0.1:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>127.0.0.1:8031</value>

</property>

</configuration>

==========

6. **Edit file C:/Hadoop-3.3.0/etc/hadoop/hadoop-env.cmd**

Find “JAVA_HOME=%JAVA_HOME%” and replace it as

set JAVA_HOME="C:\Java\jdk1.8.0_361"

==========

**7. Download “redistributable” package**

Download and run VC_redist.x64.exe

8. Hadoop Configurations

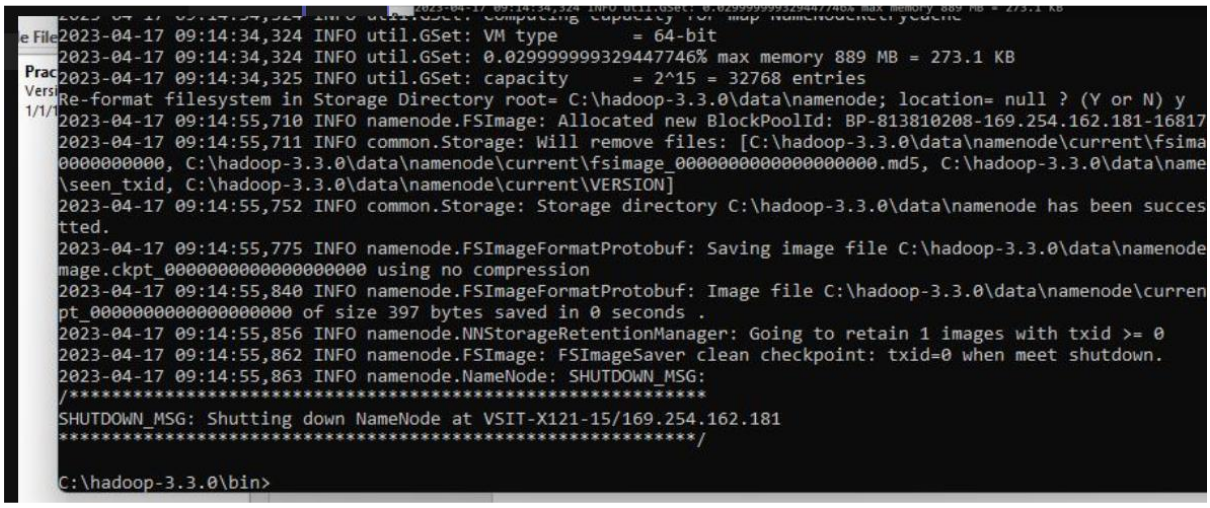
**Download bin folder from https://github.com/s911415/apache-hadoop-3.1.0-winutils**

**– Copy the bin folder to c:\hadoop-3.3.0. Replace the existing bin folder.**

**9. copy "hadoop-yarn-server-timelineservice-3.0.3.jar" from ~\hadoop-3.0.3\share\hadoop\yarn\timelineservice to ~\hadoop-3.0.3\share\hadoop\yarn folder.**

10. Format the NameNode

**– Open cmd ‘Run as Administrator’ and type command “hdfs namenode –format”**




**11. Testing**

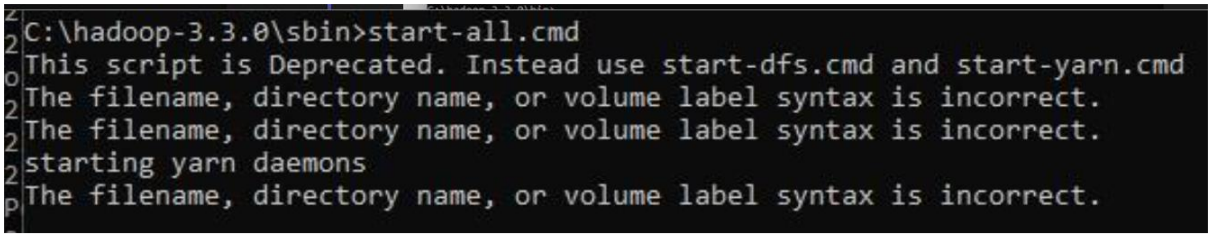
**– Open cmd ‘Run as Administrator’ and change directory to C:\Hadoop-3.3.0\sbin**

**– type start-all.cmd**

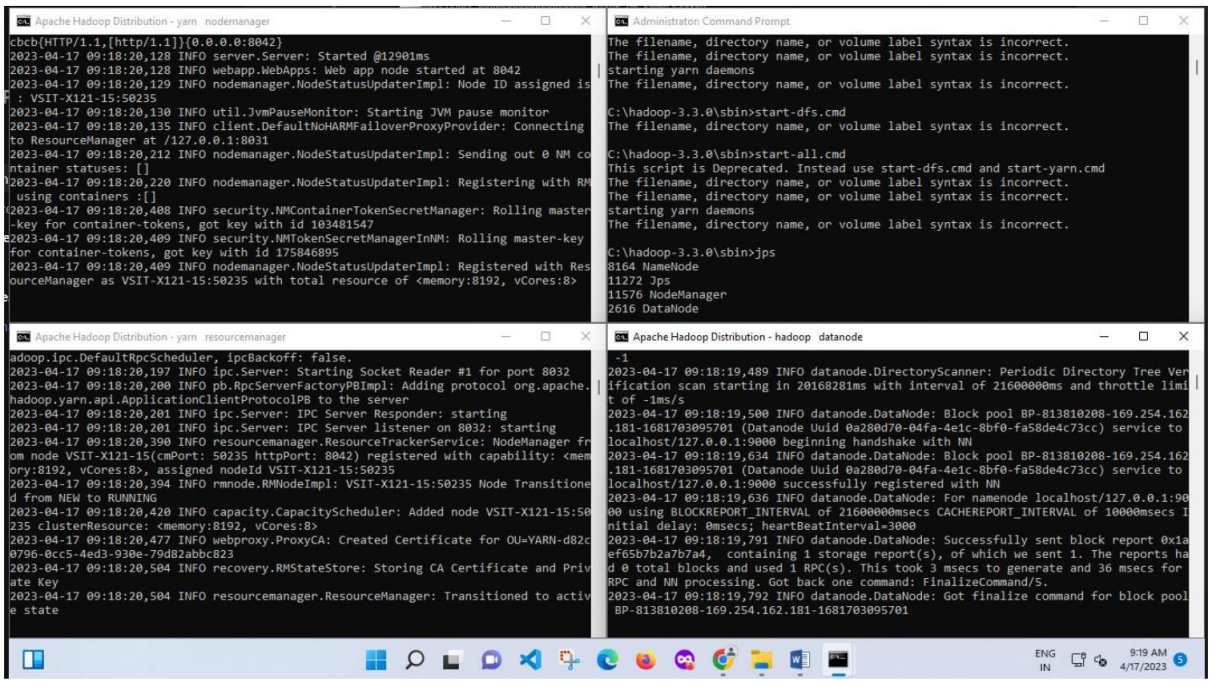
**OR**

**- type start-dfs.cmd**

**– type start-yarn.cmd**

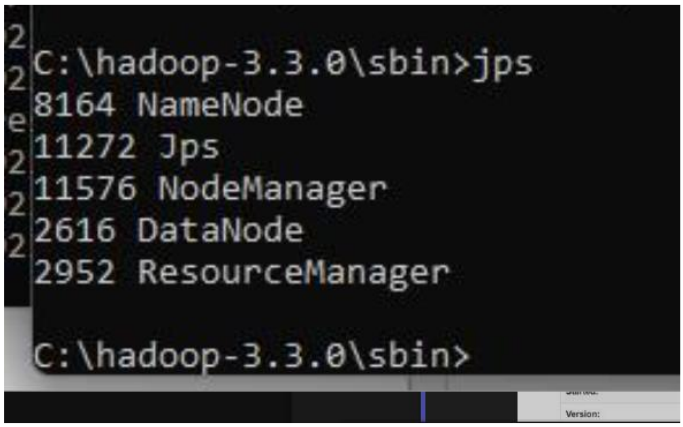



**– You will get 4 more running threads for Datanode, namenode, resouce manager and node manager**

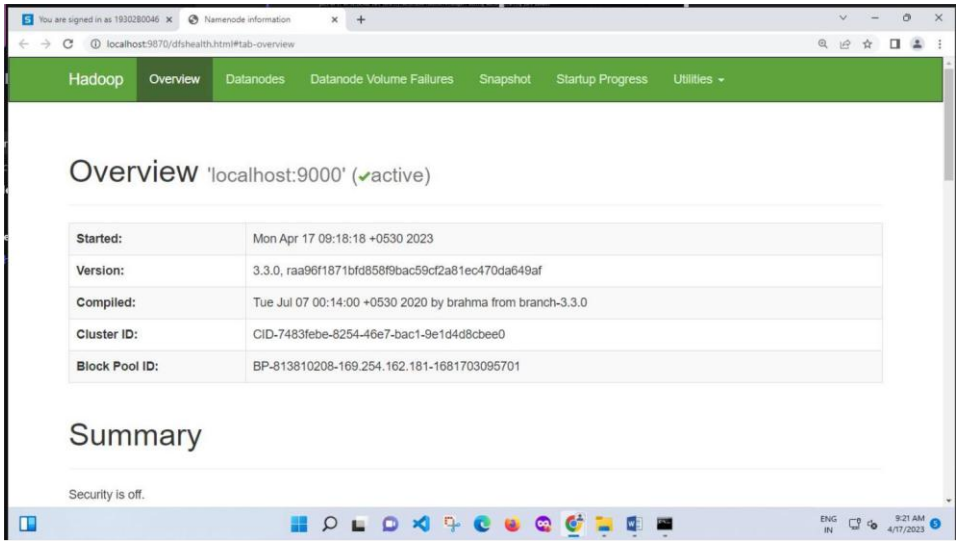


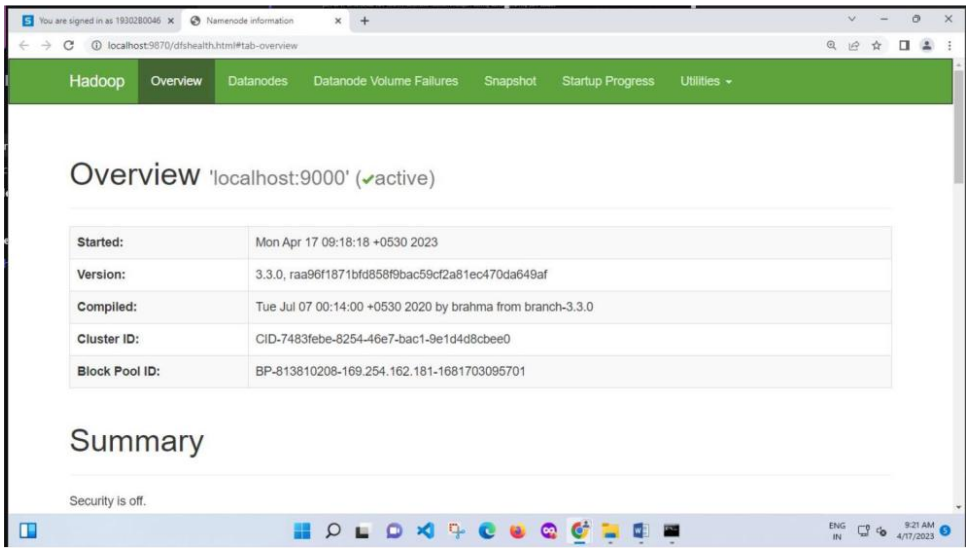

**Output:**

**12. Type JPS command to start-all.cmd command prompt, you will get following output.**




**13. Run http://localhost:9870/ from any browser**

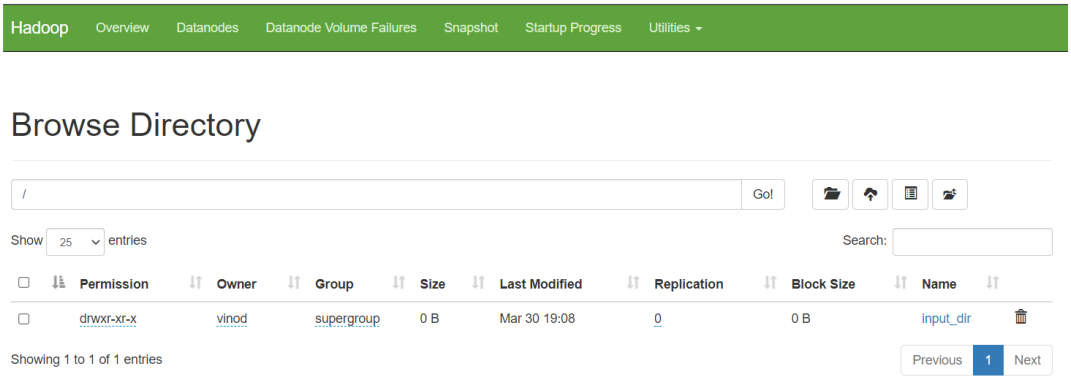

Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Overview 'localhost:9000' (active)
Started: Mon Apr 17 09:18:18 +0530 2023
Version: 3.3.0, raa96f1871bfd858f9bac59cf2a81ec470da649af
Compiled: Tue Jul 07 00:14:00 +0530 2020 by brahma from branch-3.3.0
Cluster ID: CID-7483febe-8254-46e7-bac1-9e1d4d8cbee0
Block Pool ID: BP-813810208-169.254.162.181-1681703095701
Summary
Security is off.


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Overview 'localhost:9000' (active)
Started: Mon Apr 17 09:18:18 +0530 2023
Version: 3.3.0, raa96f1871bfd858f9bac59cf2a81ec470da649af
Compiled: Tue Jul 07 00:14:00 +0530 2020 by brahma from branch-3.3.0
Cluster ID: CID-7483febe-8254-46e7-bac1-9e1d4d8cbee0
Block Pool ID: BP-813810208-169.254.162.181-1681703095701
Summary
Security is off.

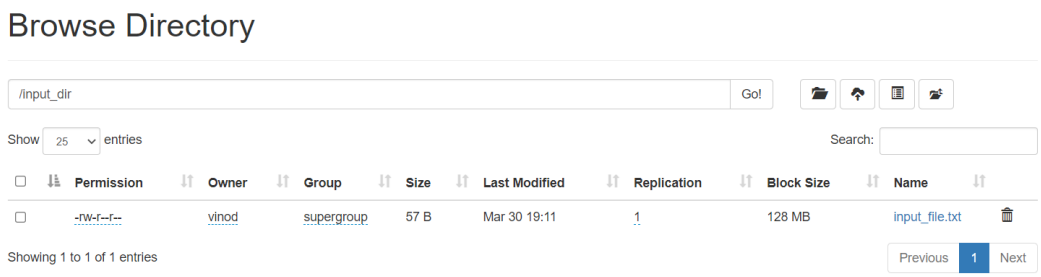
# Practical 2

## Implement word count / frequency programs using MapReduce

**Solution:**
**C:\hadoop-3.3.0\sbin>start-dfs.cmd**
**C:\hadoop-3.3.0\sbin>start-yarn.cmd**
**Open a command prompt as administrator and run the following command to create an input and output folder on the Hadoop file system, to which we will be moving the sample.txt file for our analysis.**
**C:\hadoop-3.3.0\bin>cd**
**C:\>hadoop dfsadmin -safemode leave**
**DEPRECATED: Use of this script to execute hdfs command is deprecated.**
**Instead use the hdfs command for it.**
**Safe mode is OFF**
**C:\>hadoop fs -mkdir /input_dir**
**Check it by giving the following URL at browser**
**http://localhost:9870**
**Utilities -> browse the file system**




**Copy the input text file named input_file.txt in the input directory (input_dir) of HDFS.**
**Make a file in c:\input_file.txt and write following content in it.**
**Hadoop Window version is easy compared to Ubuntu version**
**Now apply the following command at c:\>**
**C:\> hadoop fs -put C:/input_file.txt /input_dir**




**Verify input_file.txt available in HDFS input directory (input_dir).**
**C:\>Hadoop fs -ls /input_dir/**

```
C:\>hadoop fs -put C:/input_file.txt /input_dir

C:\>hadoop fs -ls /input_dir/
Found 1 items
-rw-r--r--   1 vinod supergroup         57 2023-03-30 19:11 /input_dir/input_file.txt

C:\>
```

**Verify content of the copied file**

**C:\>hadoop dfs -cat /input_dir/input_file.txt**

**You can see the file content displayed on the CMD.**

```
C:\>hadoop dfs -cat /input_dir/input_file.txt
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Hadoop Window version is easy compared to Ubuntu version.
C:\>
```

**Run MapReduceClient.jar and also provide input and out directories.**

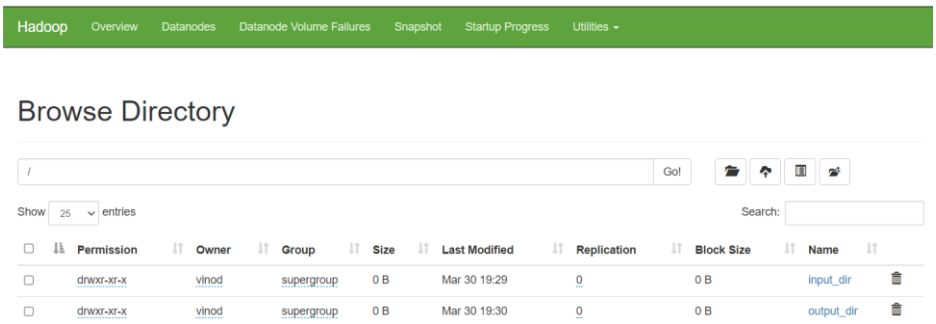
**C:\>hadoop jar C:/hadoop-3.3.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar wordcount /input_dir /output_dir**

```
                Reduce input groups=8
                Reduce shuffle bytes=103
                Reduce input records=8
                Reduce output records=8
                Spilled Records=16
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=70
                CPU time spent (ms)=219
                Physical memory (bytes) snapshot=517128192
                Virtual memory (bytes) snapshot=792633344
                Total committed heap usage (bytes)=392691712
                Peak Map Physical memory (bytes)=314761216
                Peak Map Virtual memory (bytes)=465485824
                Peak Reduce Physical memory (bytes)=202366976
                Peak Reduce Virtual memory (bytes)=327180288
        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=56
        File Output Format Counters
                Bytes Written=65

:\Windows\System32>_
```

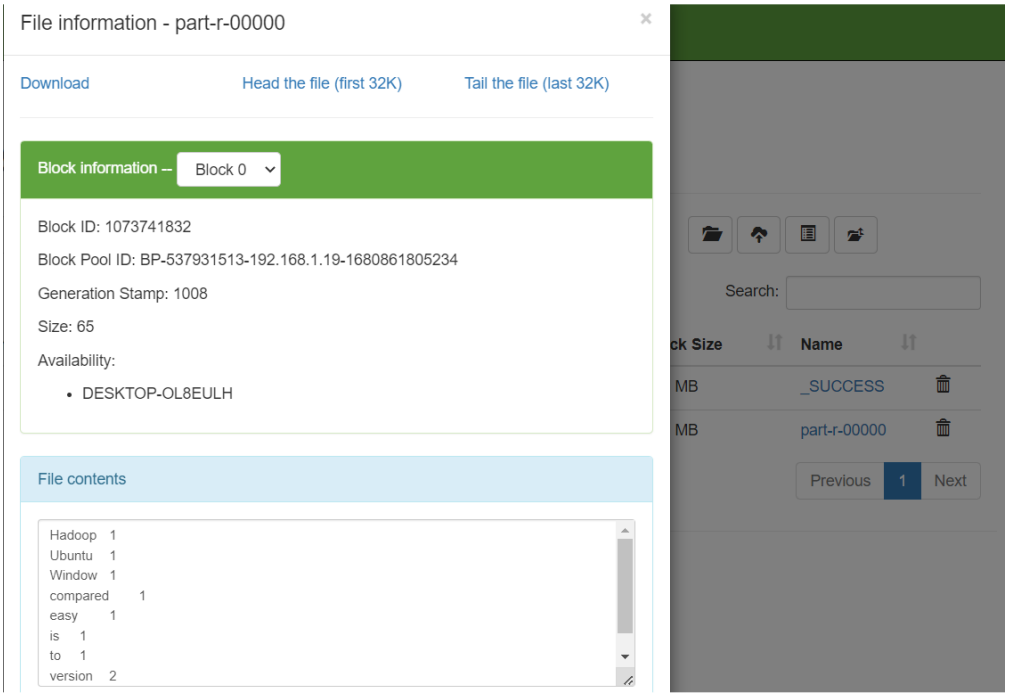
**In case, there is some error in executing then copy the file MapReduceClient.jar in C:\ and run the program with the jar file using existing MapReduceClient.jar file as:**

**C:\> hadoop jar C:/MapReduceClient.jar wordcount /input_dir /output_dir**

**Now, check the output_dir on browser as follows:**




**Click on output_dir → part-r-00000 → Head the file (first 32 K) and check the file content as the output.**



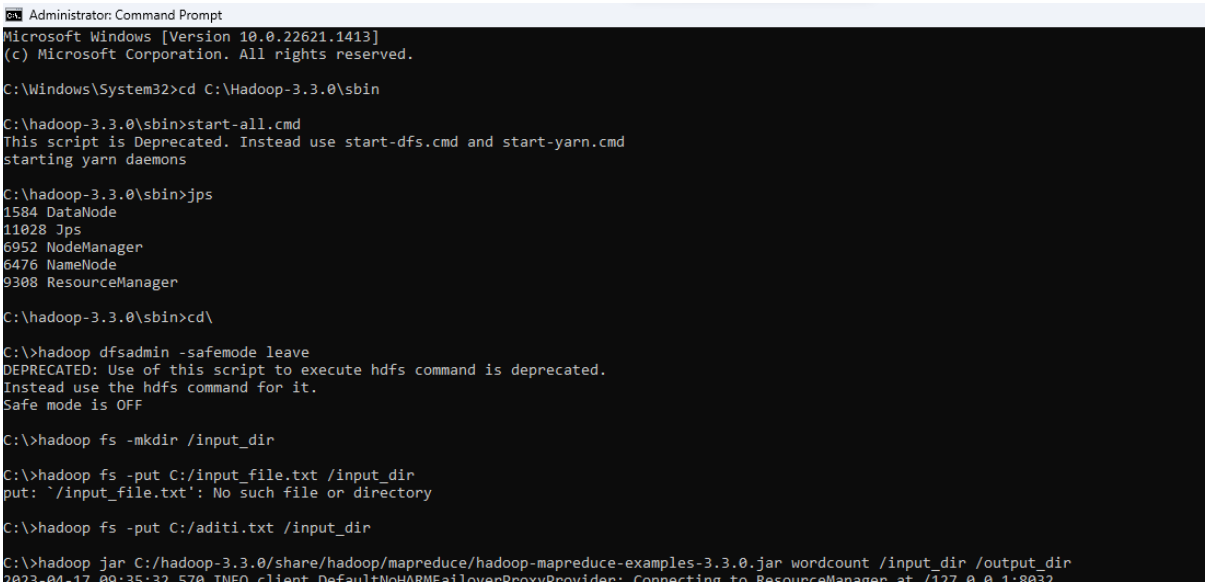

Alternatively, you may type the following command on CMD window as:

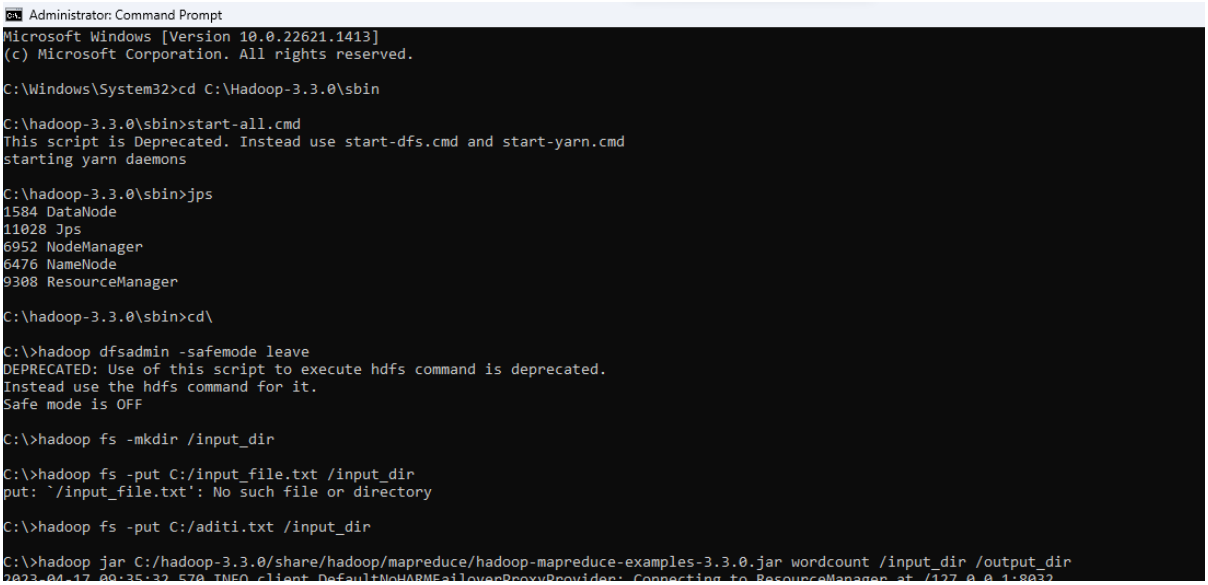
C:\> hadoop dfs -cat /output_dir/*

You can get the following output


**Output:**


part-r-00000
aditi.txt
File
Edit
View
aditi
arman
karan
rajesh
aditi
arman
ashmi
seema
arman
karan
karan
karan


Administrator: Command Prompt
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\System32>cd C:\Hadoop-3.3.0\sbin
C:\hadoop-3.3.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons
C:\hadoop-3.3.0\sbin>jps
1584 DataNode
11028 Jps
6952 NodeManager
6476 NameNode
9308 ResourceManager
C:\hadoop-3.3.0\sbin>cd\
C:\>hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Safe mode is OFF
C:\>hadoop fs -mkdir /input_dir
C:\>hadoop fs -put C:/input_file.txt /input_dir
put: `/input_file.txt': No such file or directory
C:\>hadoop fs -put C:/aditi.txt /input_dir
C:\>hadoop jar C:/hadoop-3.3.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar wordcount /input_dir /output_dir


Administrator: Command Prompt
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\System32>cd C:\Hadoop-3.3.0\sbin
C:\hadoop-3.3.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons
C:\hadoop-3.3.0\sbin>jps
1584 DataNode
11028 Jps
6952 NodeManager
6476 NameNode
9308 ResourceManager
C:\hadoop-3.3.0\sbin>cd\
C:\>hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Safe mode is OFF
C:\>hadoop fs -mkdir /input_dir
C:\>hadoop fs -put C:/input_file.txt /input_dir
put: `/input_file.txt': No such file or directory
C:\>hadoop fs -put C:/aditi.txt /input_dir
C:\>hadoop jar C:/hadoop-3.3.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar wordcount /input_dir /output_dir

# Practical 3

## Implement an application that stores big data in Hbase / MongoDB and manipulate it using R / Python

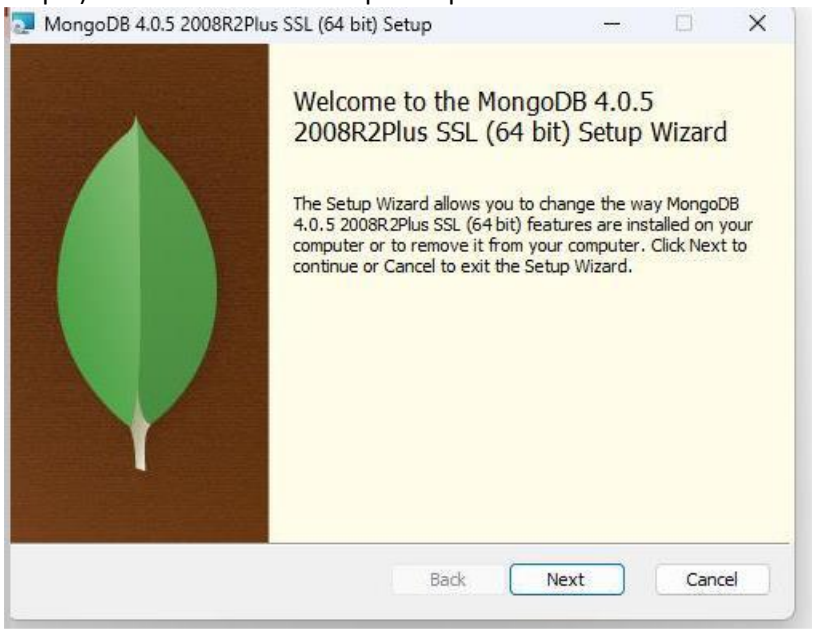
a. PyMongo

b. Mongo Database

**Step A: Install Mongo database**

Step 1) Go to (https://www.mongodb.com/download-center/community) and Download MongoDB Community Server. We will install the 64-bit version for Windows.

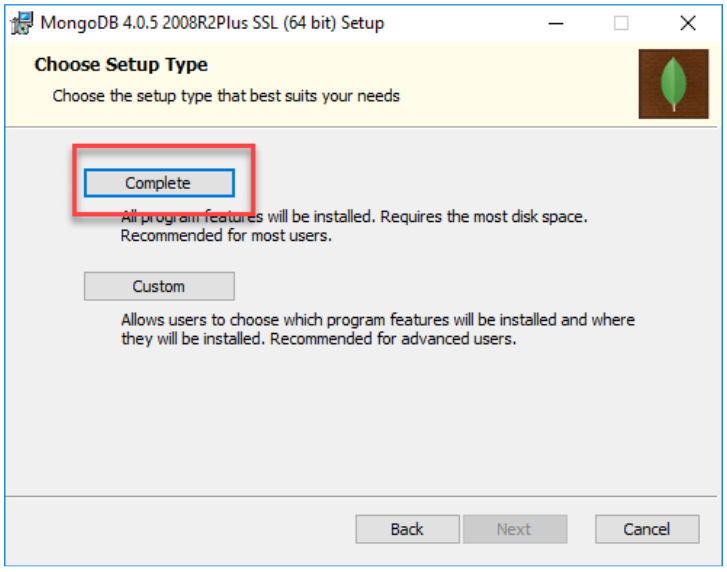
Step 2) Once download is complete open the msi file. Click Next in the start up screen




Step 3)

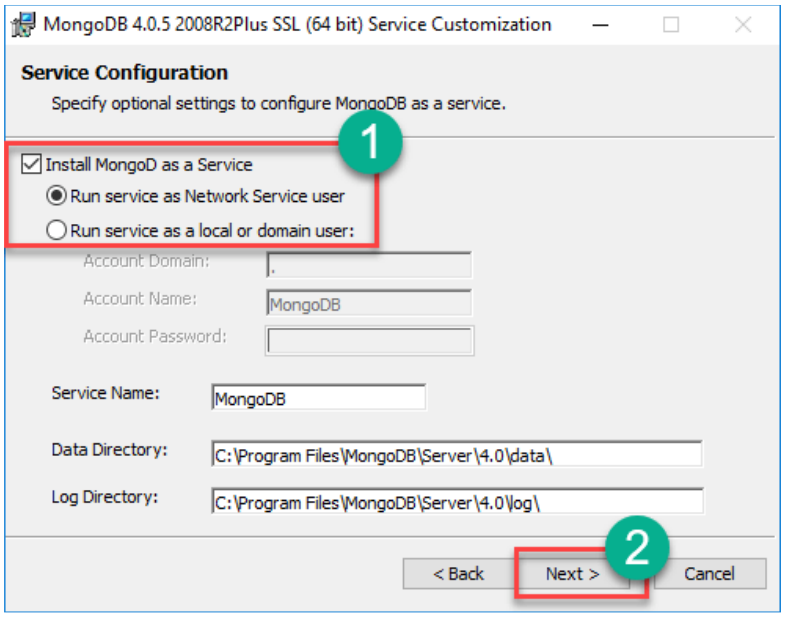
1. Accept the End-User License Agreement
2. Click Next

Step 4) Click on the "complete" button to install all of the components. The custom option can be used to install selective components or if you want to change the location of the installation.

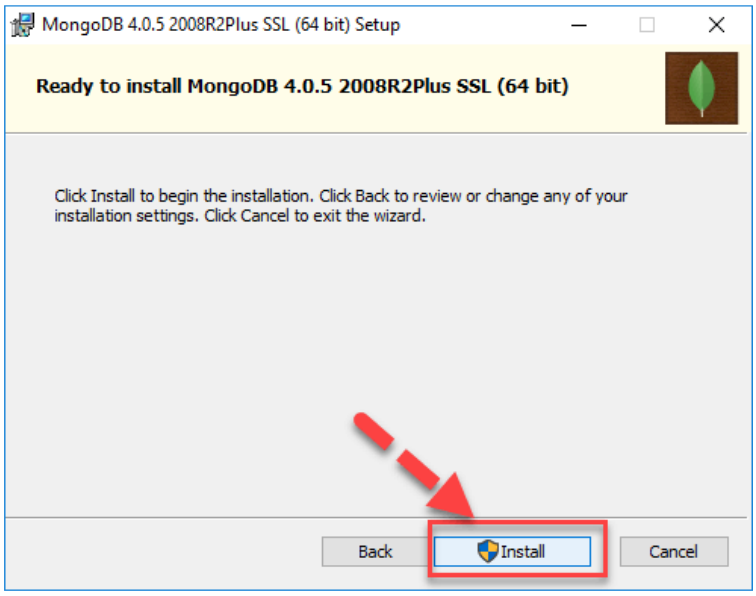


Step 5)

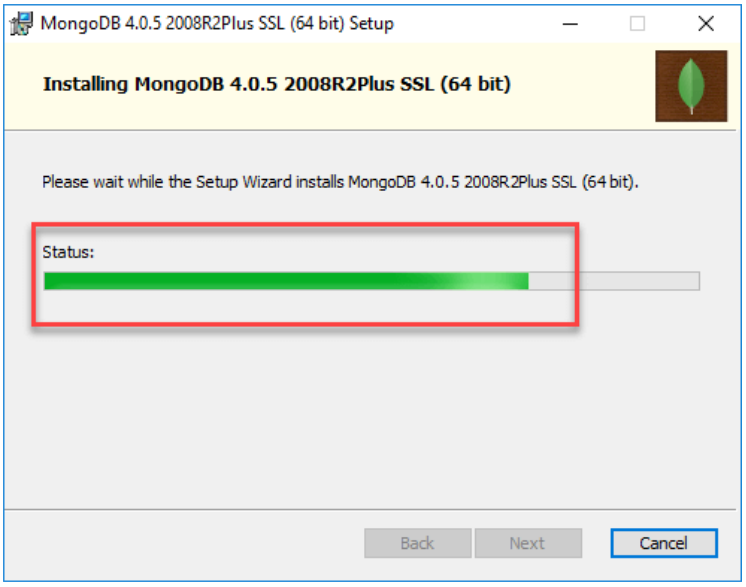
1. Select “Run service as Network Service user”. make a note of the data directory,



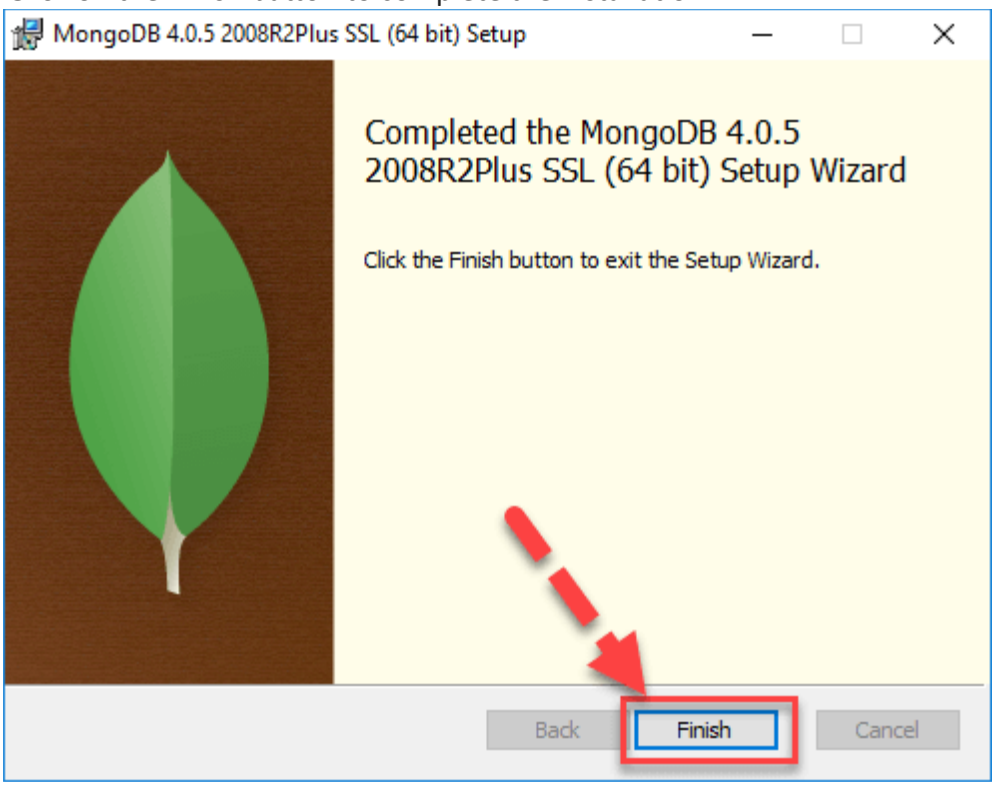

**Step 6)** Click on the Install button to start the installation.




**Step 7)** Installation begins. Click Next once completed.

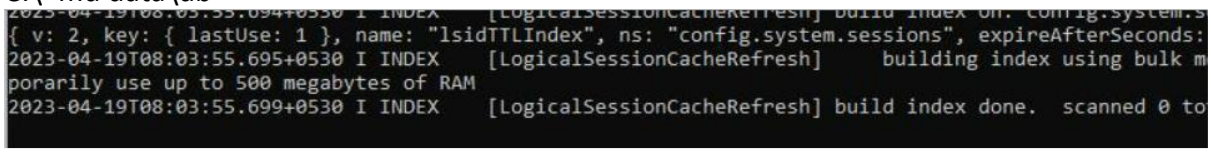



**Step 8) Click** on the Finish button to complete the installation




**Test Mongodb**

**Step 1**) Go to " C:\Program Files\MongoDB\Server\4.0\bin" and double click on **mongo.exe.**
Alternatively, you can also click on the MongoDB desktop icon.
Create the directory where MongoDB will store its files.
Open command prompt window and apply following commands
C:\users\admin> cd\
C:\>md data\db



**Step 2) Execute mongodb**
Open another command prompt window.
C:\> cd C:\Program Files\MongoDB\Server\4.0\bin
C:\Program Files\MongoDB\Server\4.0\bin> mongod
*In case if it gives an error then run the following command:*
*C:\Program Files\MongoDB\Server\4.0\bin> mongod –repair*

**Step 3) Connect to MongoDB using the Mongo shell**
Let the MongoDB daemon to run.
Open another command prompt window and run the following commands:
C:\users\admin> cd C:\Program Files\MongoDB\Server\4.0\bin
C:\Program Files\MongoDB\Server\4.0\bin>mongo

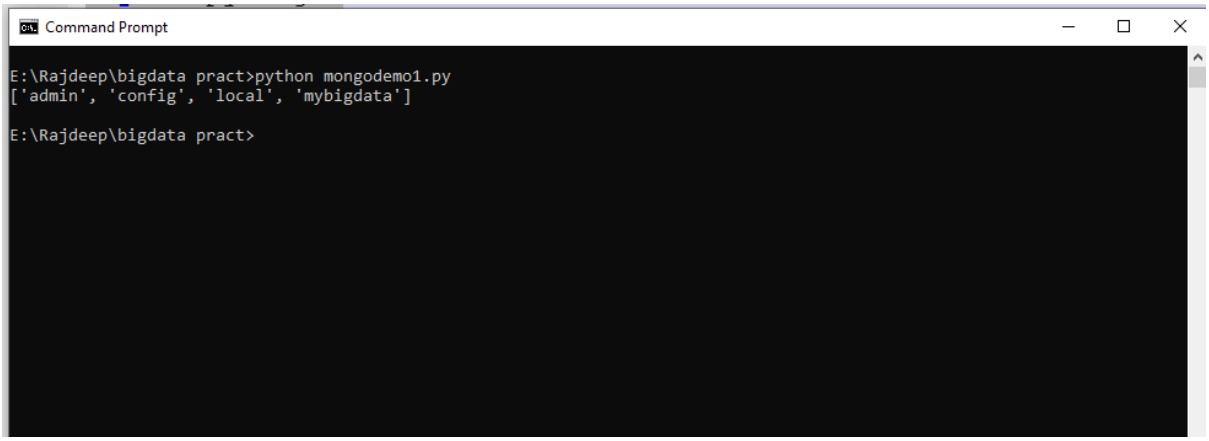
```
> show dbs
VSIT    0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

**Step 4) Install PyMongo**
Open another command prompt window and run the following commands:
Check the python version on your desktop / laptop and copy that path from window explorer
C:\users\admin>cd C:\Program Files\Python311\Scripts
C:\Program Files\<Python38>\Scripts > python -m pip install pymongo
**Step 5) Test PyMongo**
Run the following command from python command prompt
**import pymongo**
Now, either create a file in Python IDLE or run all commands one by one in sequence on Python cell

**Program 1:** Displaying the database name:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
print(myclient.list_database_names())
```

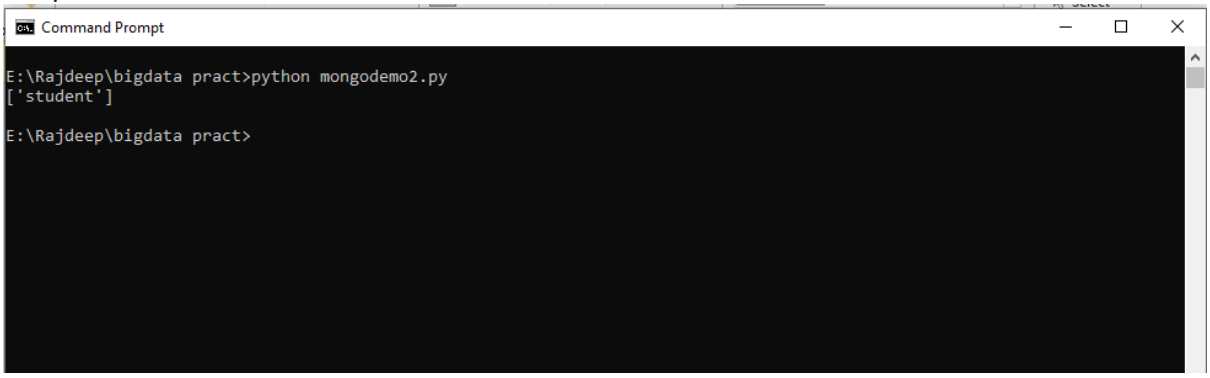
Output:




**Program 2:** Creating collection:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb["student"]
print(mydb.list_collection_names())
```

Output:




**Program 3**: Inserting Data

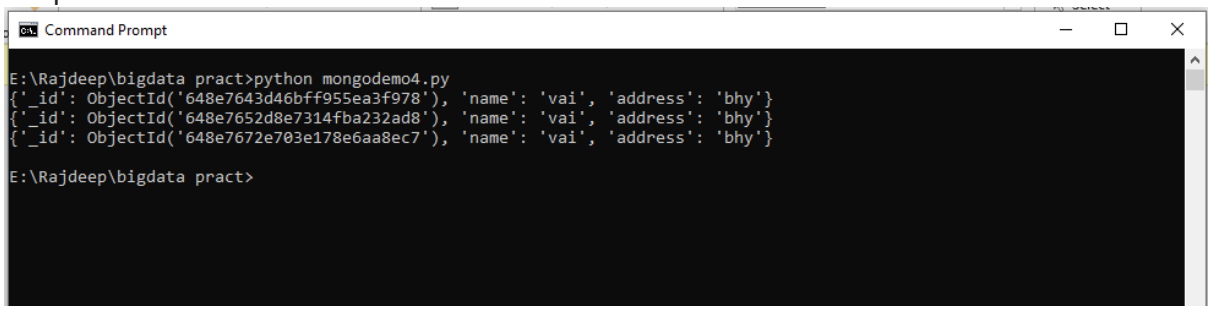
```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb["student"]
mydict={"name":"vai", "address":"bhy"}
x=mycol.insert_one(mydict)
print("Data inserted !")
```

Output:


**Program 4**: Insert Multiple data into Collection

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb["student"]
mylist=[{"name":"Ganesh", "address":"Mumbai"}, {"name":"Varun", "address":"Mumbai"},
{"name":"Prasoon", "address":"Pune"}, {"name":"Satish", "address":"Pune"},]
x=mycol.insert_many(mylist)
print("Data inserted !")
```

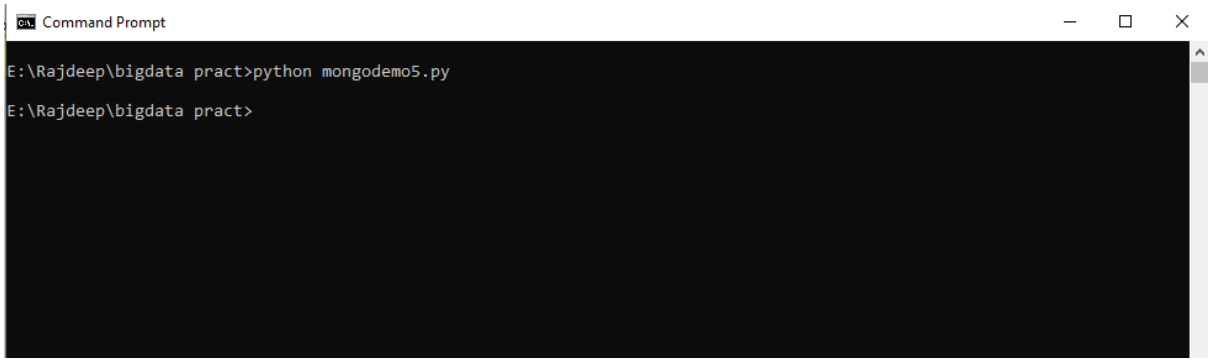
Output:




**Program 5:** Displaying the collection data:

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol = mydb["student"]

myquery = { "name": "Vai" }

mydoc = mycol.find(myquery)

for x in mydoc:
  print(x)
```

Output:


Command Prompt
E:\Rajdeep\bigdata pract>python mongodemo5.py
E:\Rajdeep\bigdata pract>

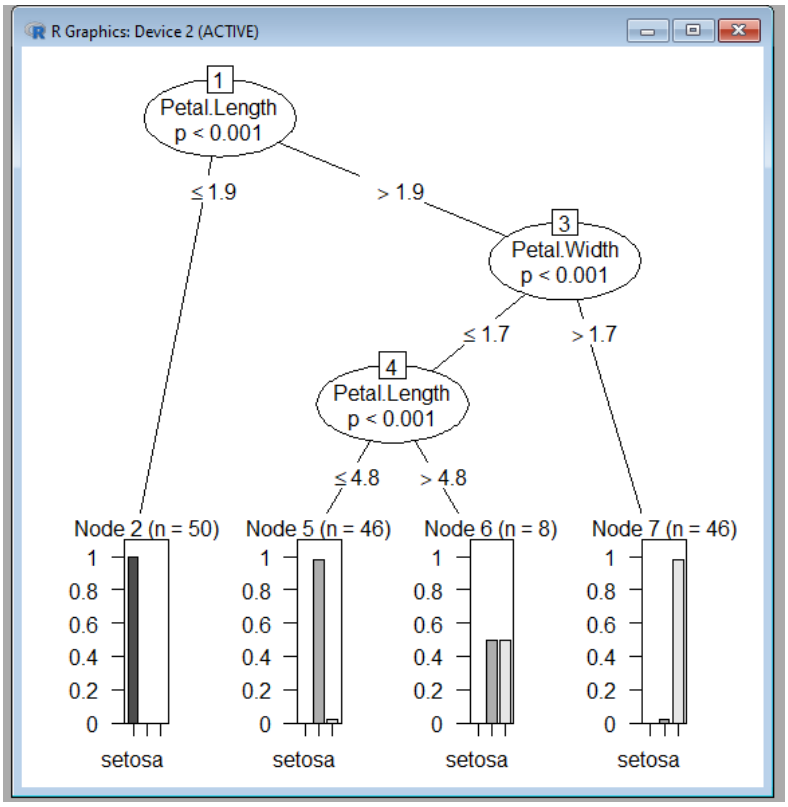
# Practical 4

## Implement Decision tree classification techniques

### Code-

```
library("party")
print(head(readingSkills))
str(iris)
iris_ctree <- ctree(Species ~ Sepal.Width + Sepal.Length + Petal.Length + Petal.Width,
data=iris)
print (iris_ctree)
plot(iris_ctree)
```

### output



# Practical 5

## Implement SVM classification techniques

## Code-

```
# Importing the dataset
dataset = read.csv(' E:/NIKHILESH/social.csv')

# Selecting relevant columns: Age, EstimatedSalary, Purchased
dataset = dataset[3:5]

# Encoding the target feature as factor
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

# Splitting the dataset into the Training set and Test set
install.packages('caTools')  # Run only once
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])

# Fitting SVM to the Training set
install.packages('e1071')  # Run only once
library(e1071)
classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
```

```
              kernel = 'linear')

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-3])

# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
print("Confusion Matrix:")
print(cm)

# Visualising the Training set results
install.packages("ElemStatLearn")  # Run only once
library(ElemStatLearn)

# Plotting function (for training and test sets)
plot_svm <- function(set, title) {
 X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
 X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
 grid_set = expand.grid(X1, X2)
 colnames(grid_set) = c('Age', 'EstimatedSalary')
 y_grid = predict(classifier, newdata = grid_set)
 plot(set[, -3],
      main = title,
      xlab = 'Age', ylab = 'Estimated Salary',
      xlim = range(X1), ylim = range(X2))
 contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
 points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
 points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
}

# Plot training and test results
```

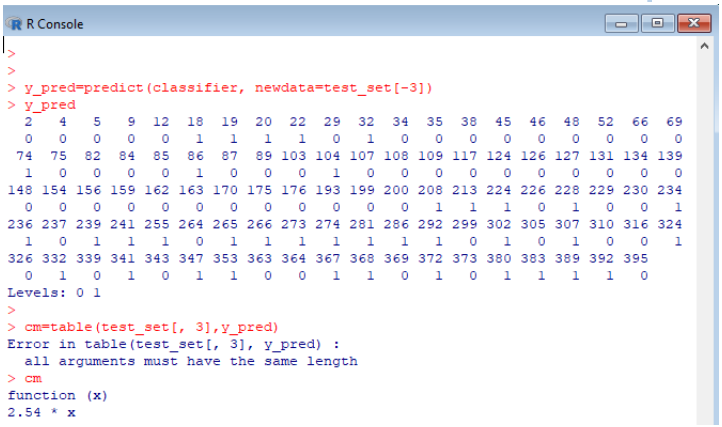
plot_svm(training_set, 'SVM Classification (Training set)')

```
> set = training_set
> X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
> X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
```

```
> grid_set = expand.grid(X1, X2)
> colnames(grid_set) = c('Age', 'EstimatedSalary')
> y_grid = predict(classifier, newdata = grid_set)
> plot(set[, -3],
+      main = 'SVM (Training set)',
+      xlab = 'Age', ylab = 'Estimated Salary',
+      xlim = range(X1), ylim = range(X2))
```

```
> contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
> points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'coral1', 'aquamarine'))
> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```

**Output:**

# Practical 6

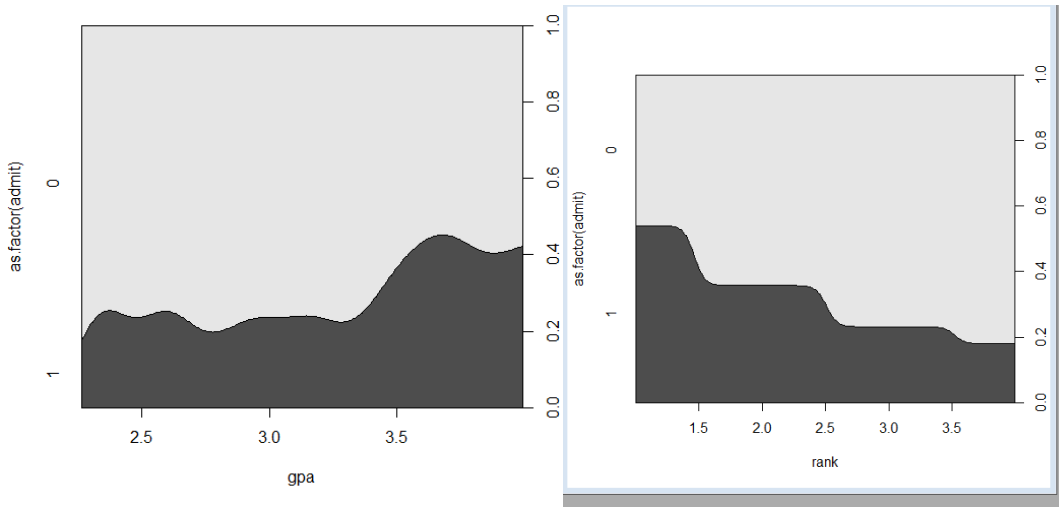
**REGRESSION MODEL Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. require (foreign), require(MASS).**

Linear regression practical
**code-**

```
# Load dataset
college <-
read.csv("https://raw.githubusercontent.com/ropensci/datapack/main/inst/extdata/pkg-example/binary.csv")
head(college)
nrow(college)

# Install and load caTools
install.packages("caTools")  # Run only once
library(caTools)

# Split dataset
set.seed(123)
split <- sample.split(college$admit, SplitRatio = 0.75)
training_reg <- subset(college, split == TRUE)
test_reg <- subset(college, split == FALSE)

# Fit logistic regression model
fit_logistic_model <- glm(admit ~ ., data = training_reg, family = "binomial")

# View coefficients
coef(fit_logistic_model)["gre"]
coef(fit_logistic_model)["gpa"]
coef(fit_logistic_model)["rank"]

# Predict probabilities on test data
predict_reg <- predict(fit_logistic_model, newdata = test_reg, type = "response")

# Plot Conditional Density
cdplot(as.factor(admit) ~ gpa, data = college)
cdplot(as.factor(admit) ~ gre, data = college)
cdplot(as.factor(admit) ~ rank, data = college)

# Convert probabilities to binary predictions
predict_binary <- ifelse(predict_reg > 0.5, 1, 0)
```

# Confusion Matrix
table(Actual = test_reg$admit, Predicted = predict_binary)

**output**



# Practical 7

## MULTIPLE REGRESSION MODEL: Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.

## Code-

**Explain Multiple regression in detail.**

```
# Load dataset

college <- read.csv("https://raw.githubusercontent.com/csquared/udacity-dlnd/master/nn/binary.csv")

head(college)

nrow(college)


# Install and load caTools (only run install once)

install.packages("caTools")  # Only the first time

library(caTools)


# Data splitting

set.seed(123)

split <- sample.split(college$admit, SplitRatio = 0.75)

training_reg <- subset(college, split == TRUE)

test_reg <- subset(college, split == FALSE)

# Fit logistic regression model

fit_MRegressor_model <- glm(formula = admit ~ gre + gpa + rank, data = training_reg, family = binomial)

# Predict probabilities on test set

predict_reg <- predict(fit_MRegressor_model, newdata = test_reg, type = "response")

head(predict_reg)

# Classify predictions (threshold = 0.5)

predict_class <- ifelse(predict_reg > 0.5, 1, 0)
```

**# Plot conditional density plots**

**cdplot(as.factor(admit) ~ gpa, data = college)**

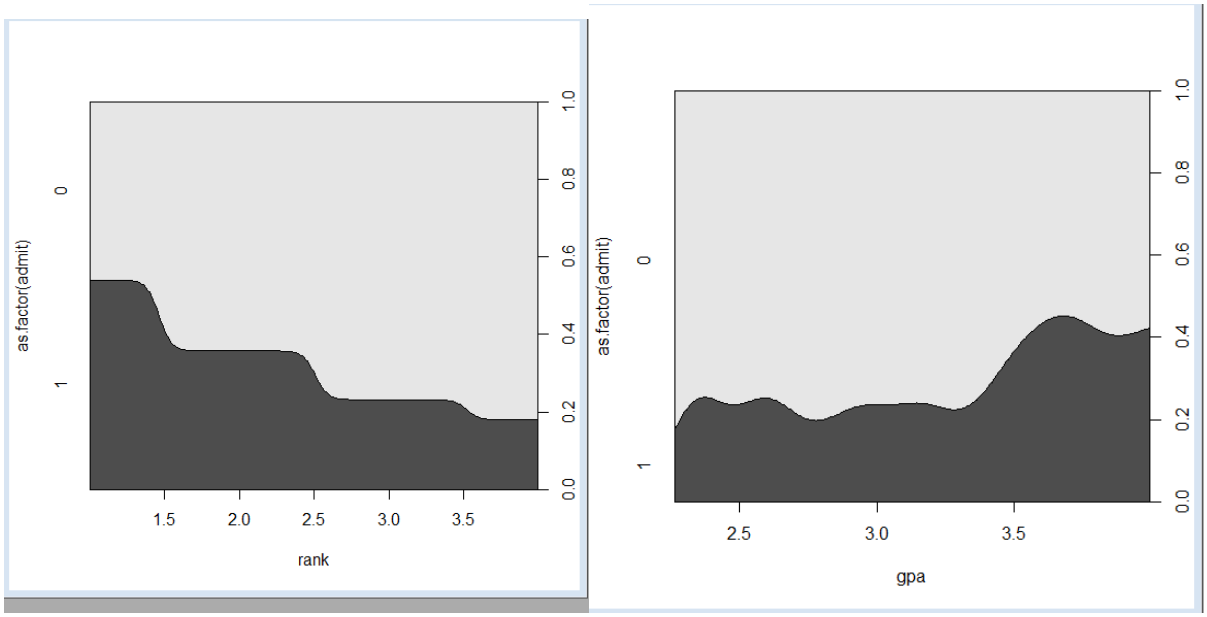
**cdplot(as.factor(admit) ~ gre, data = college)**

**cdplot(as.factor(admit) ~ rank, data = college)**

**# Confusion matrix**

**table(Actual = test_reg$admit, Predicted = predict_class)**

**Output**



# Practical 8

CLASSIFICATION MODEL a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.

## Navebyse

```
data(iris)
str(iris)
install packages("e1071")
install packages("caTools")
install packages("caret")
library(e1071)
library(caTools)
library(caret)
split <- sample.split(iris,SplitRatio=0.7)
train_c1 <-subset(iris,split=="TRUE")
test_c1 <- subset(iris,split == "FALSE")
train_scale <- scale(train_c1[, 1:4])
test_scale <- scale(test_c1[,1:4])

set.seed(120)
classifier_c1 <- naiveBayes(Species ~ ., data = train_c1)
classifier_c1

y_pred <- predict(classifier_c1, newdata= test_c1)
cm <- table(test_c1$Species, y_pred)
cm
confusionMatrix(cm) data(iris)
str(iris)
install packages("e1071")
install packages("caTools")
install packages("caret")
library(e1071)
library(caTools)
library(caret)
split <- sample.split(iris,SplitRatio=0.7)
train_c1 <-subset(iris,split=="TRUE")
test_c1 <- subset(iris,split == "FALSE")
train_scale <- scale(train_c1[, 1:4])
test_scale <- scale(test_c1[,1:4])

set.seed(120)
classifier_c1 <- naiveBayes(Species ~ ., data = train_c1)
classifier_c1
```

**output** -

```
Naive Bayes Classifier for Discrete Predictors

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

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

Conditional probabilities:
            Sepal.Length
Y                [,1]      [,2]
  setosa     5.050000 0.3739353
  versicolor 5.833333 0.5141671
  virginica  6.626667 0.5545631

            Sepal.Width
Y                [,1]      [,2]
  setosa     3.443333 0.3838852
  versicolor 2.753333 0.3191944
  virginica  2.976667 0.3058998
```

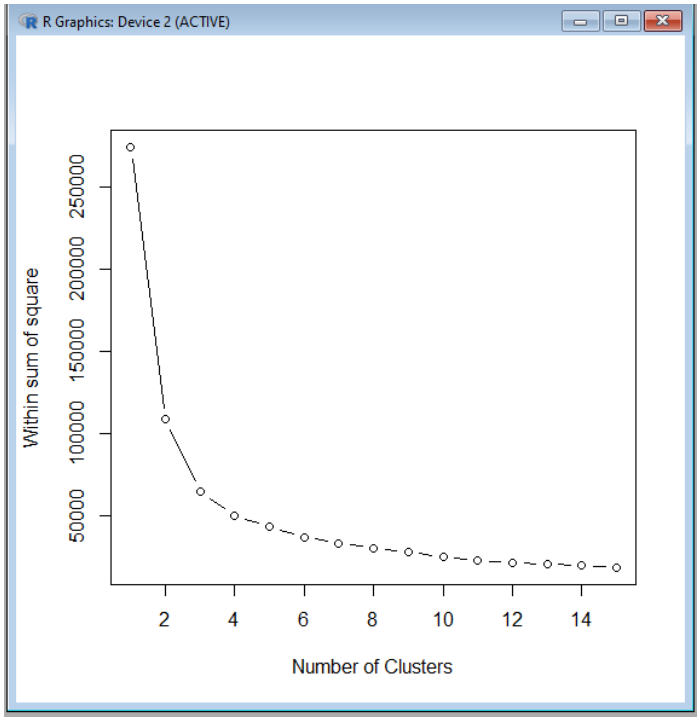
y_pred <- predict(classifier_c1, newdata= test_c1)
cm <- table(test_c1$Species, y_pred)
cm
confusionMatrix(cm)
**output**-

```
cm
           y_pred
            setosa versicolor virginica
setosa          20          0         0
versicolor       0         18         2
virginica        0          1        19
confusionMatrix(cm)
```

# Practical 9

## CLUSTERING MODEL a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations.

```
install.packages("plyr")
install.packages("ggplot2")
install.packages("cluster")
install.packages("lattice")
install.packages("grid")
install.packages("gridExtra")
library(plyr)
library(ggplot2)
library(cluster)
library(lattice)
library(grid)
library(gridExtra)
grade_input=as.data.frame(read.csv("E:/Rajdeep/bigdata
pract/dataset/grades_km_input.csv"))
kmdata_orig=as.matrix(grade_input[, c ("Student","English","Math","Science")])
kmdata=kmdata_orig[,2:4]
kmdata[1:10,]
wss=numeric(15)
for(k in 1:15)wss[k]=sum(kmeans(kmdata,centers=k,nstart=25)$withinss)
plot(1:15,wss,type="b",xlab="Number of Clusters",ylab="Within sum of square")
km = kmeans(kmdata,3,nstart=25)
km
c( wss[3] , sum(km$withinss))
df=as.data.frame(kmdata_orig[,2:4])
df$cluster=factor(km$cluster)
centers=as.data.frame(km$centers)
g1=ggplot(data=df, aes(x=English, y=Math, color=cluster )) +
geom_point() + theme(legend.position="right") +
geom_point(data=centers,aes(x=English,y=Math, color=as.factor(c(1,2,3))),size=10,
alpha=.3, show.legend =FALSE)
g2=ggplot(data=df, aes(x=English, y=Science, color=cluster )) +
geom_point () +geom_point(data=centers,aes(x=English,y=Science,
color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend=FALSE)
g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +
geom_point () + geom_point(data=centers,aes(x=Math,y=Science,
color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend=FALSE)
tmp=ggplot_gtable(ggplot_build(g1))
grid.arrange(arrangeGrob(g1 + theme(legend.position="none"),g2 +
theme(legend.position="none"),g3 + theme(legend.position="none"),top ="High School
Student Cluster Analysis" ,ncol=1))
```




Aprori Practical

code-

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

data(Groceries)
Groceries

summary(Groceries)
class(Groceries)

rules = apriori(Groceries, parameter = list(supp = 0.02, conf = 0.2))
summary (rules)

inspect(rules[1:10])

arules::itemFrequencyPlot(Groceries, topN = 20,
col = brewer.pal(8, 'Pastel2'),
main = 'Relative Item Frequency Plot',
type = "relative",
ylab = "Item Frequency (Relative)")

itemsets = apriori(Groceries, parameter = list(minlen=2, maxlen=2,support=0.02,
target="frequent itemsets"))
summary(itemsets)
inspect(itemsets)
```

```
itemsets_3 = apriori(Groceries, parameter = list(minlen=3, maxlen=3,support=0.02,
target="frequent itemsets"))
summary(itemsets_3)
```

```
inspect(itemsets_3)
```

**output-**