INDEX

Sr No.	Name of the practical	Date	Signature
1	Install, configure and run Hadoop and HDFS and explore HDFS.		
2	Implement word count / frequency programs using MapReduce		
3	Implement an application that stores big data in Hbase/MongoDB and manipulate it using R/Python		
4	Write a Pig Script for solving counting problems.		
5	Configure the Hive and implement the application in Hive.		
6	Implement the following		
6A	Implement Decision tree classification techniques		
6B	Implement SVM classification techniques		
7	Solve the following:		
7A	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).		
7B	MULTIPLE REGRESSION MODEL Apply multiple regressions, if data have a		

	continuous independent variable. Apply on above dataset.		
8	Solve the Following:		
8A	CLASSIFICATION MODEL a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.		
8B	CLUSTERING MODEL a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations.		

PRACTICAL 1

Hadoop Installation

Aim:
Install, configure and run Hadoop and HDFS and explore HDFS on Windows

Code:

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

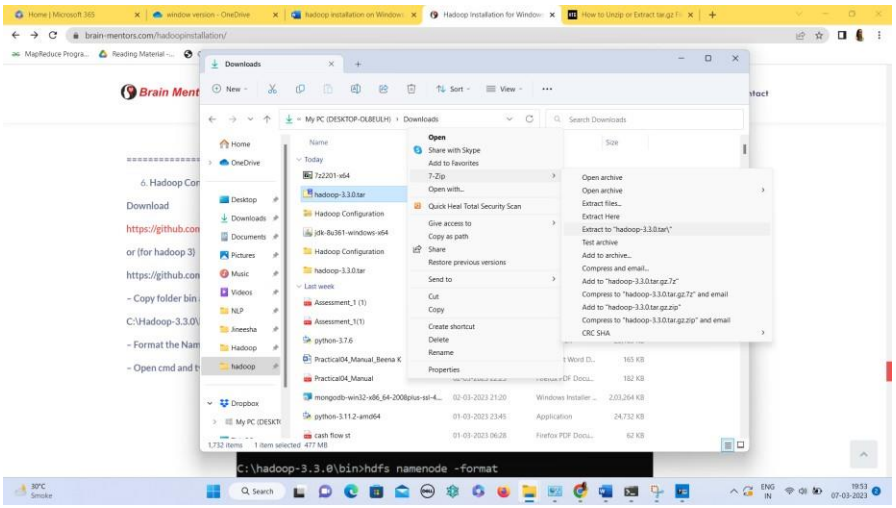
Here, we use multi node cluster.

1. **Install Java**
   11. – Java JDK Link to download
       https://www.oracle.com/java/technologies/javase-jdk8-downloads.html
   12. – extract and install Java in C:\Java
   13. – open cmd and type -> javac -version

```
C:\Users>cd Beena

C:\Users\Beena>java -version
java version "1.8.0_361"
Java(TM) SE Runtime Environment (build 1.8.0_361-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.361-b09, mixed mode)
```

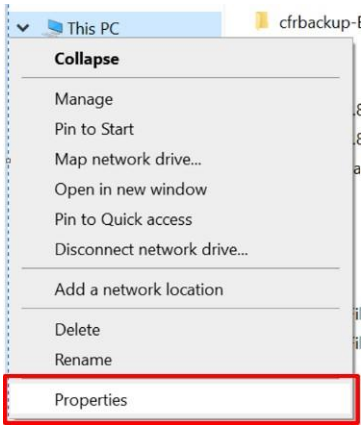
2. **Download Hadoop**
   https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz

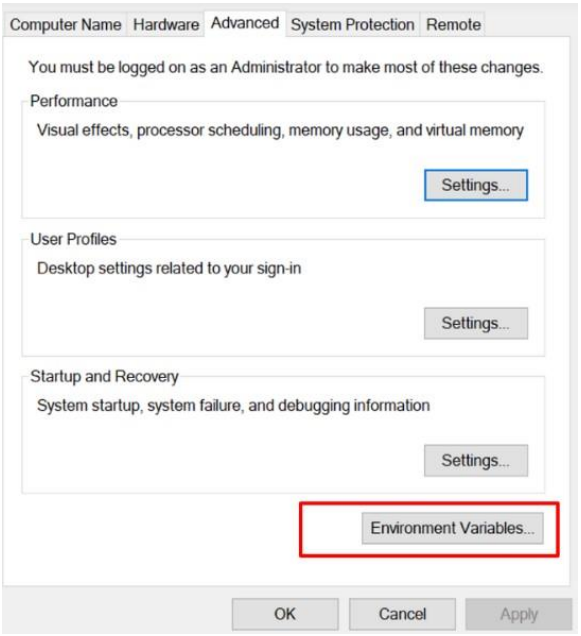
   - right click .rar.gz file -> show more options -> 7-zip->and extract to C:\Hadoop-3.3.0\



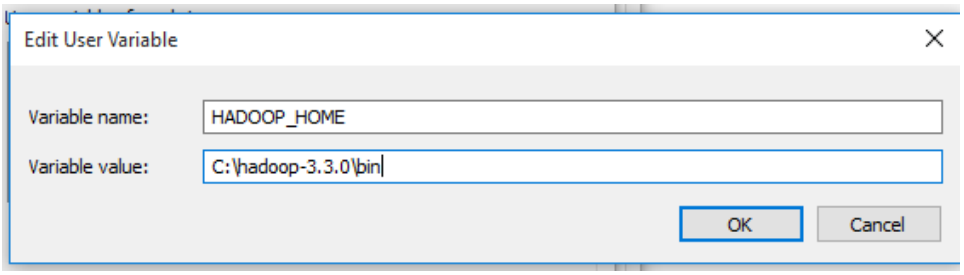
3. Set the path JAVA_HOME Environment variable

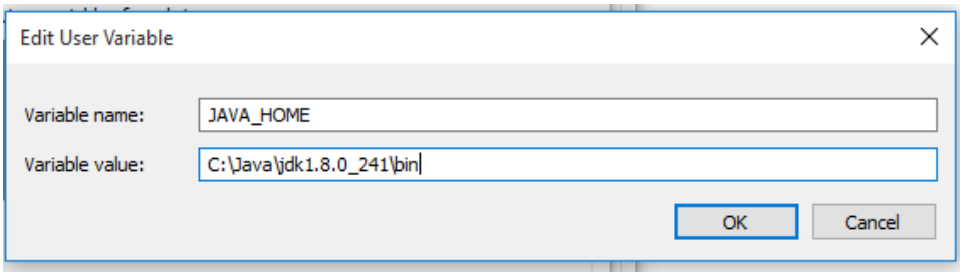
4. Set the path HADOOP_HOME Environment variable



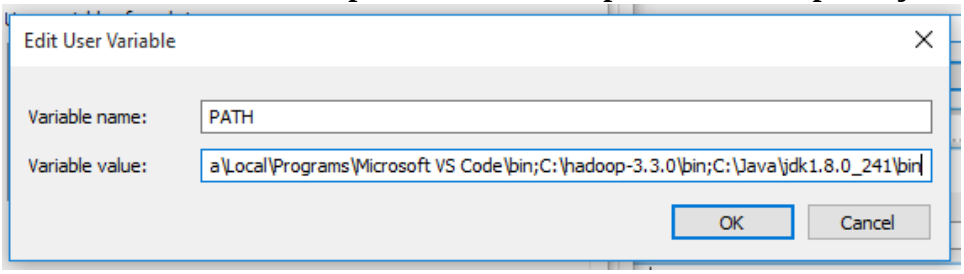


Click on **New to both user variables and system variables.**





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

This is a "redistributable" package of the Visual C runtime code for 64-bit applications, from Microsoft. It contains certain shared code that every application written with Visual C expects to have available on the Windows computer it runs on.

8. Hadoop Configurations

Download bin folder from

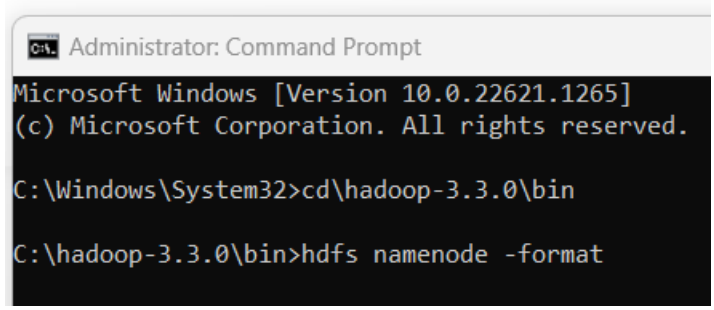
https://github.com/s911415/apache-hadoop-3.1.0-winutils

– 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"




```
2023-03-07 21:31:34,685 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.3.0\data\namenode\current\fsi
mage.ckpt_0000000000000000000 using no compression
2023-03-07 21:31:34,844 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.0\data\namenode\current\fsimage.ck
pt_0000000000000000000 of size 400 bytes saved in 0 seconds .
2023-03-07 21:31:34,860 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2023-03-07 21:31:34,869 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-03-07 21:31:34,870 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at DESKTOP-OL8EULH/192.168.1.19
************************************************************/
```

11. Testing

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

– type start-all.cmd

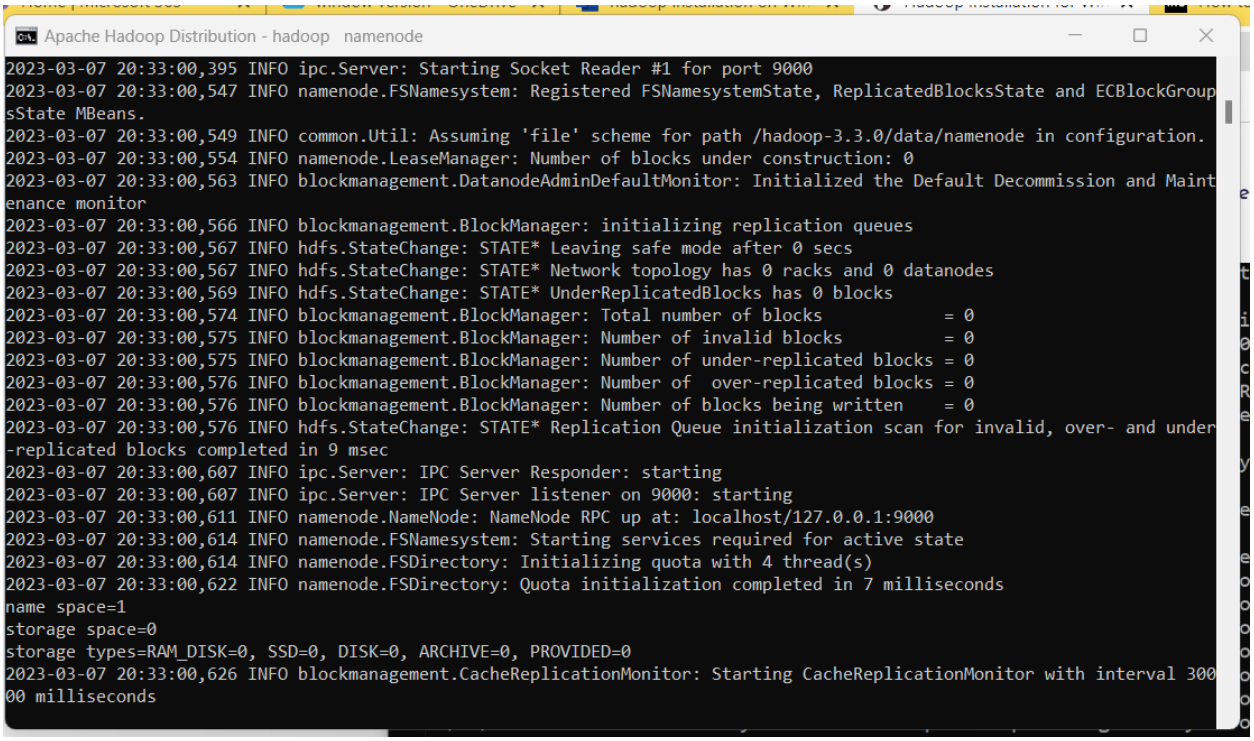
OR

- type start-dfs.cmd

– type start-yarn.cmd

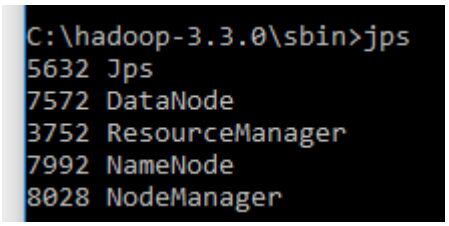
```
C:\hadoop-3.3.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
The filename, directory name, or volume label syntax is incorrect.
The filename, directory name, or volume label syntax is incorrect.
starting yarn daemons
The filename, directory name, or volume label syntax is incorrect.
```

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

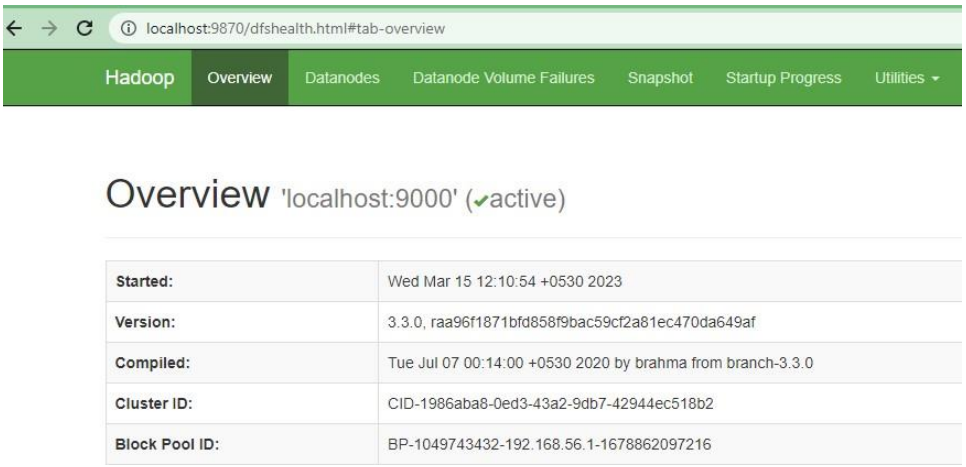


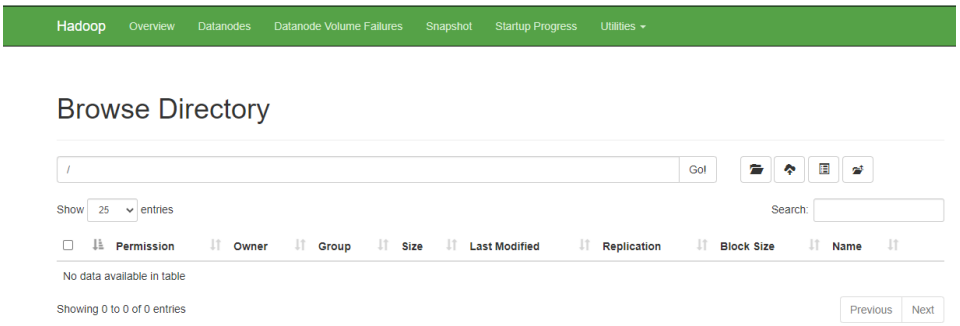
Output:

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
Browse Directory
/
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
No data available in table
Showing 0 to 0 of 0 entries
Previous
Next

PRACTICAL 2

MapReduce Implementation

Aim:

Implement word count / frequency programs using MapReduce.

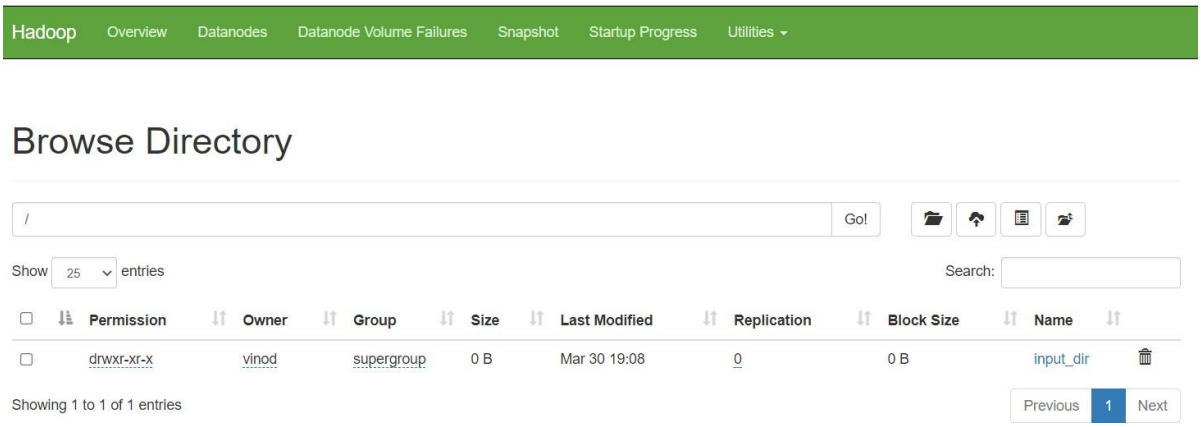
Steps:

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



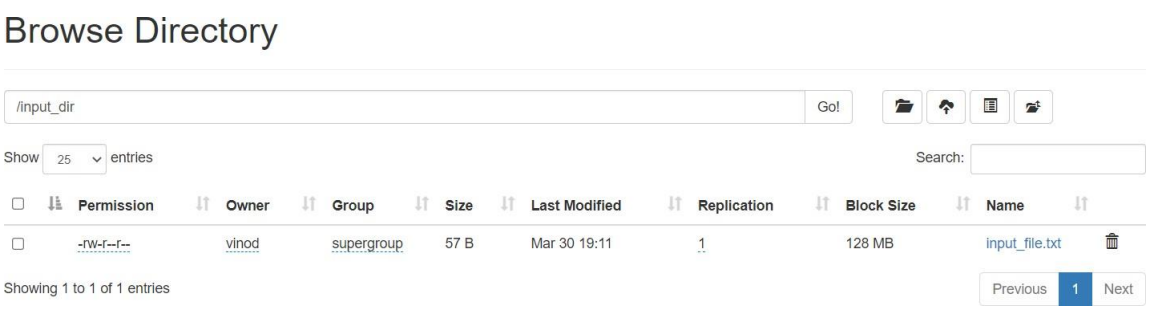
Copy the input text file named input_file.txt in the input directory (input_dir)of HDFS.

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.

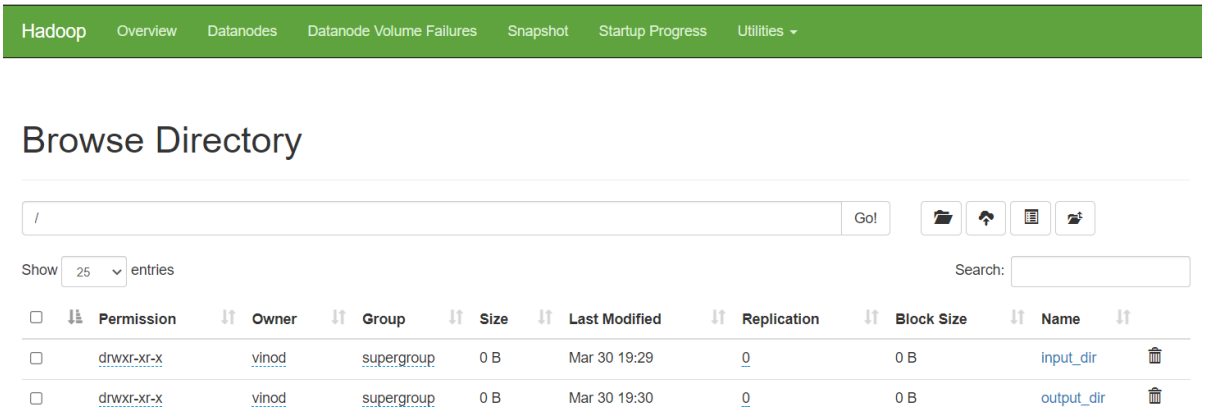
C:\>hadoop jar C:/hadoop-3.3.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar wordcount /input_dir /output_dir

```
                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>_
```

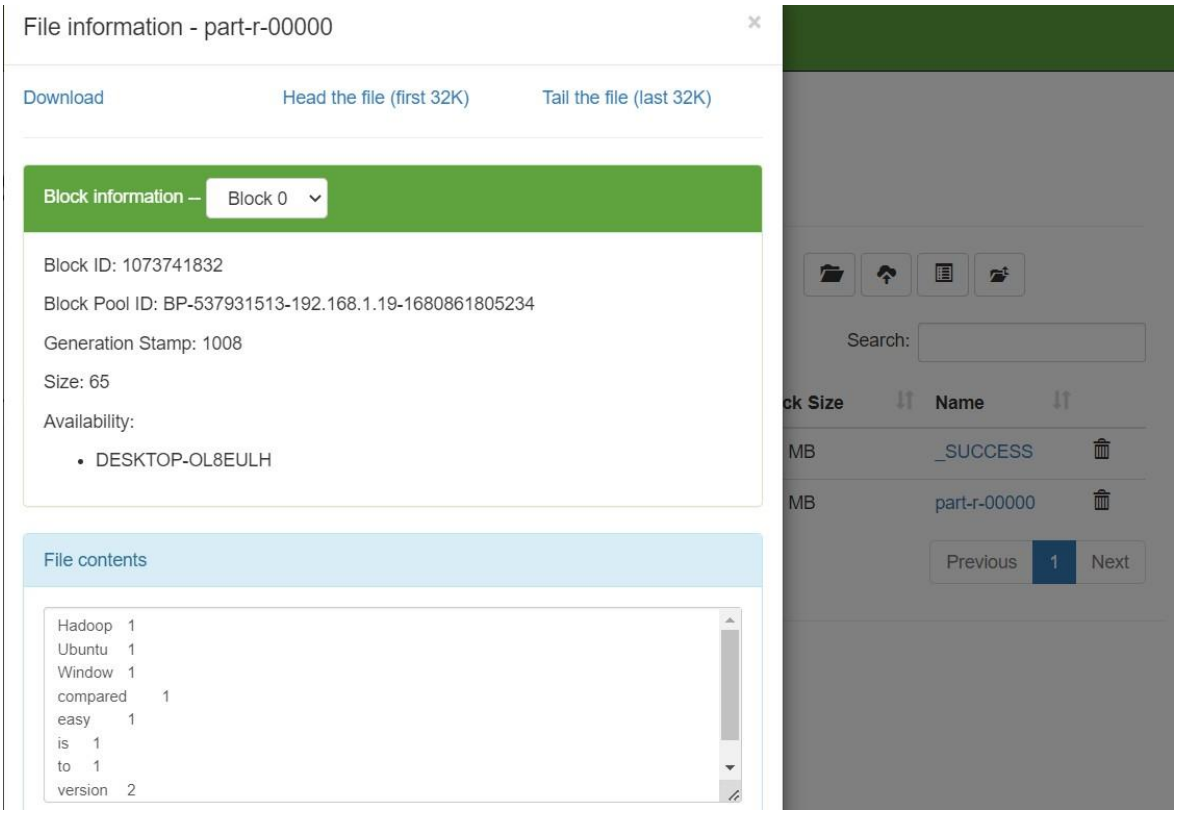
In case, there is some error in executing then copy the file MapReduceClient.jar in C:\ and run the program with the jar file using existing MapReduceClient.jar file as:

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

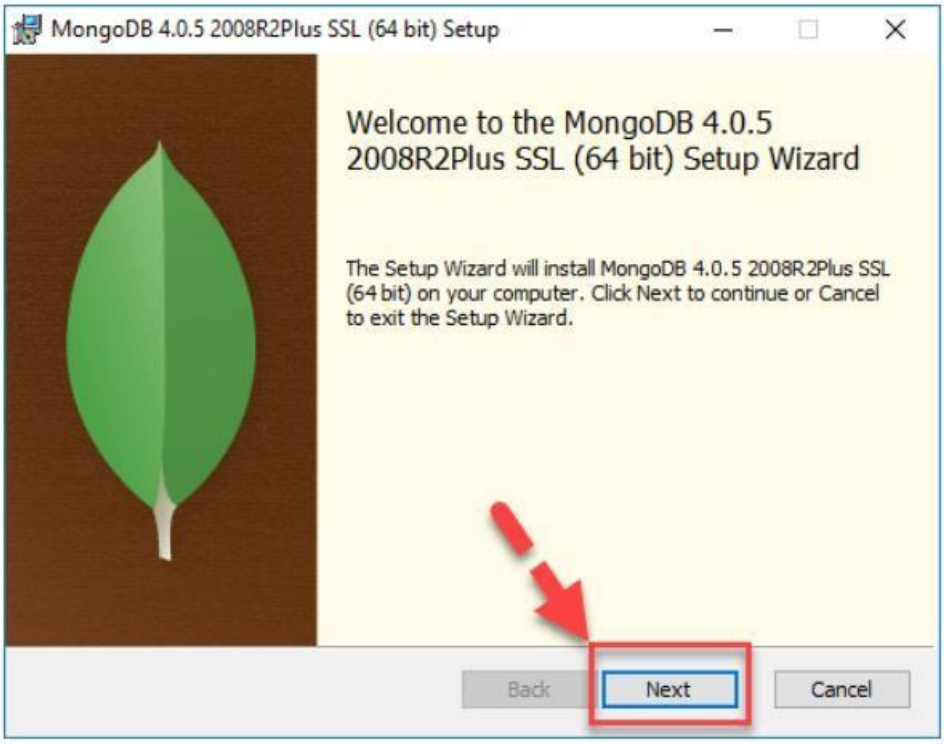
```
C:\Windows\System32>hadoop dfs -cat /output_dir/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Hadoop  1
Ubuntu  1
Window  1
compared        1
easy    1
is      1
to      1
version 2

C:\Windows\System32>
```

PRACTICAL 3

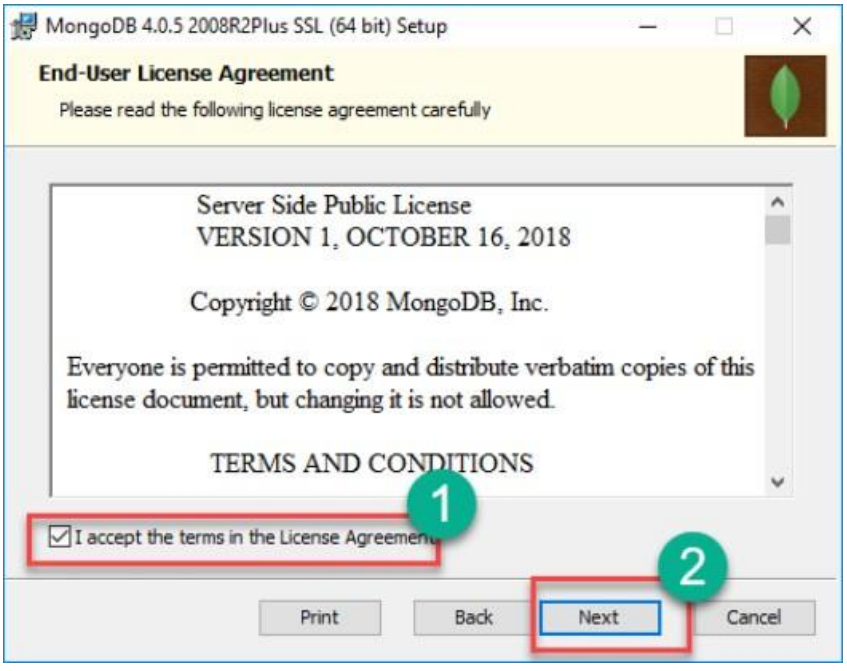
Aim:
Implement an application that stores big data in Hbase / MongoDB and manipulate it using R / Python

Requirements
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.

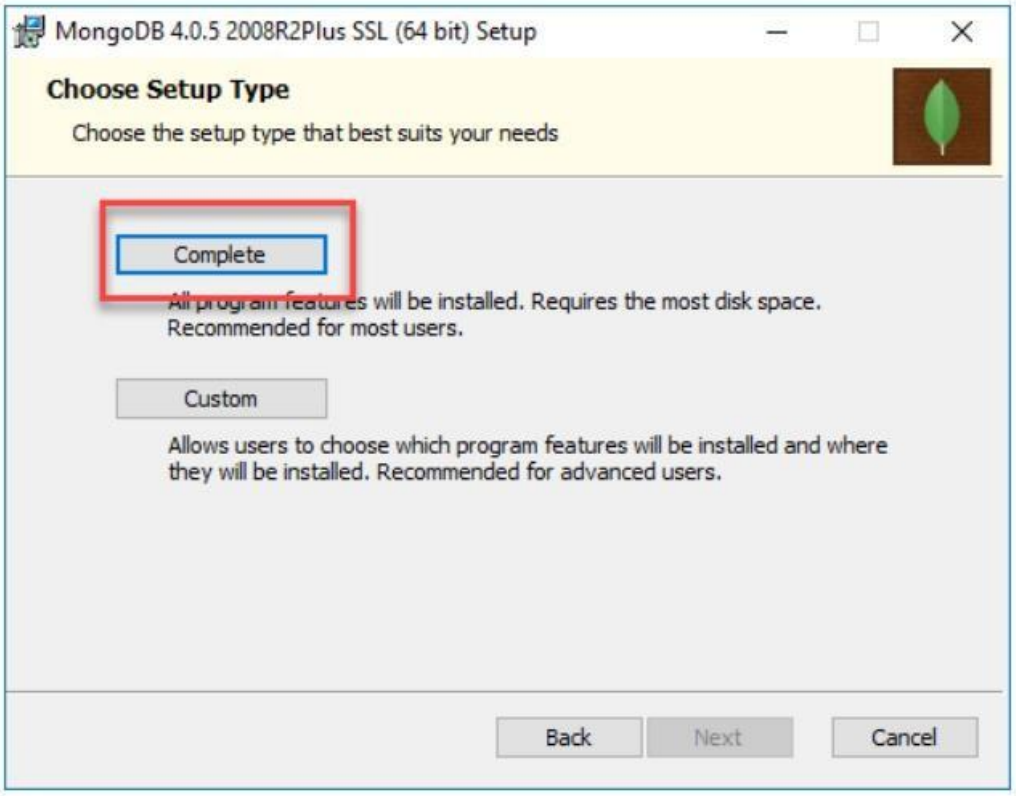
Step 2) Once download is complete open the msi file. Click Next in the start up screen



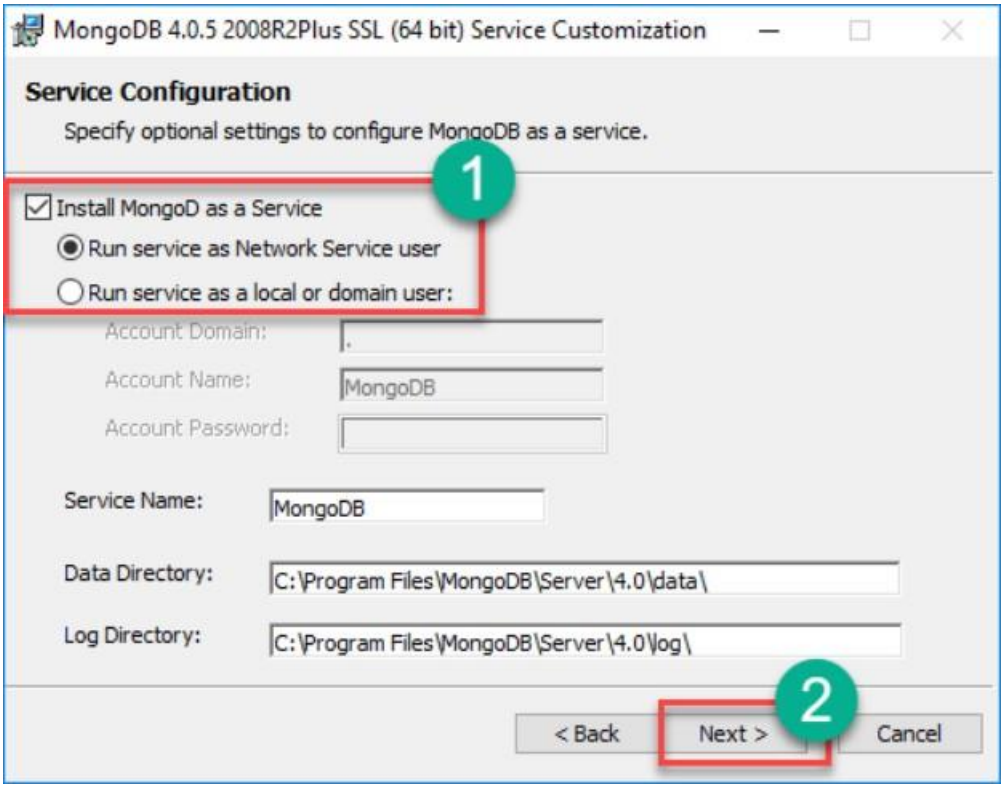
Step 3)
1. Accept the End-User License Agreement
2. Click Next



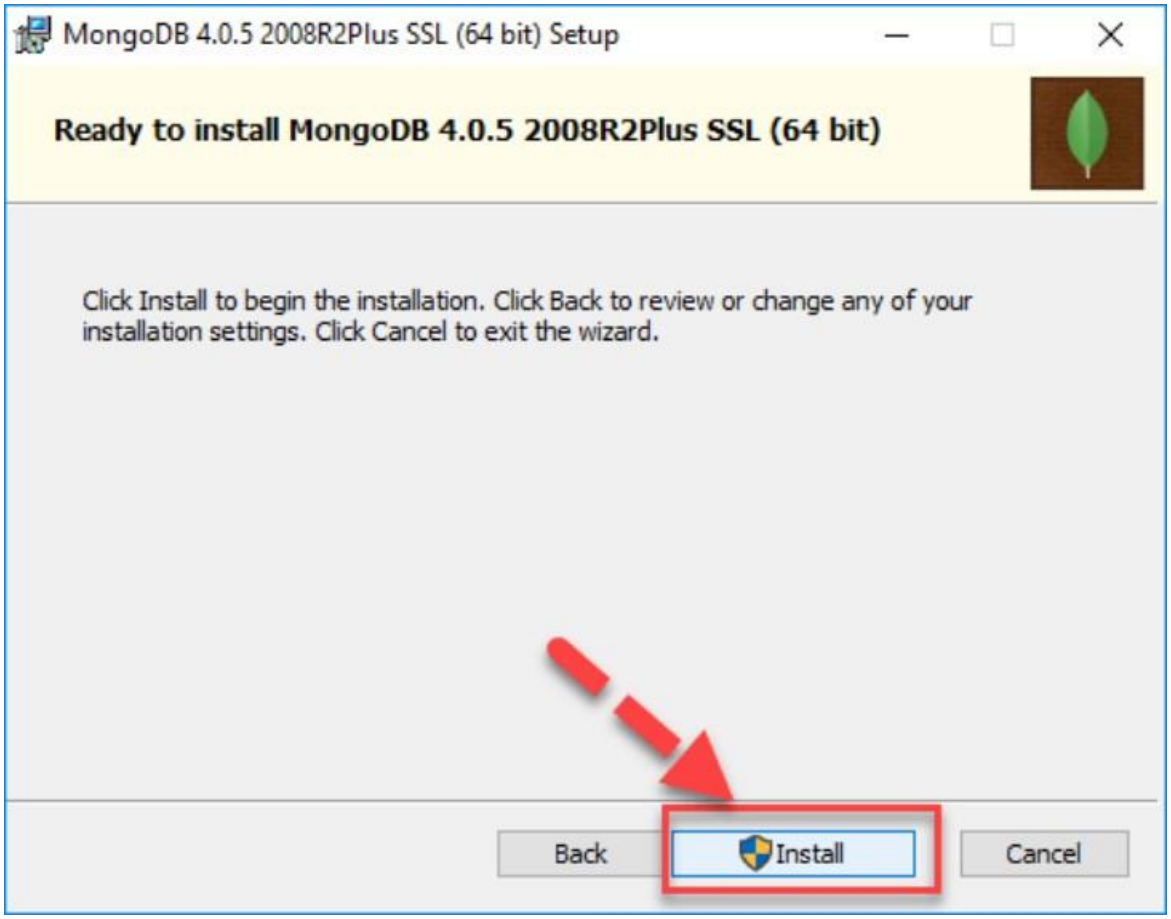
Step 4) Click on the "complete" button to install all of the components. The custom option can be used to install selective components or if you want to change the location of the installation.



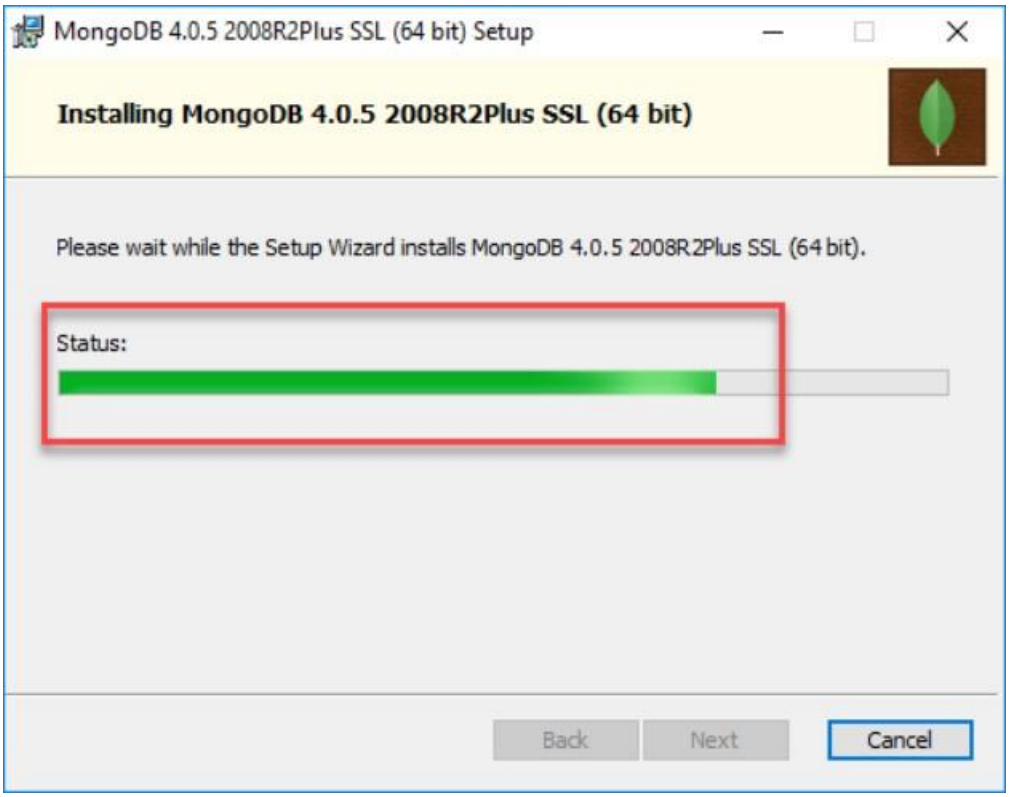
Step 5)
1. Select “Run service as Network Service user”. make a note of the data directory, we”ll need this later.
2. Click Next



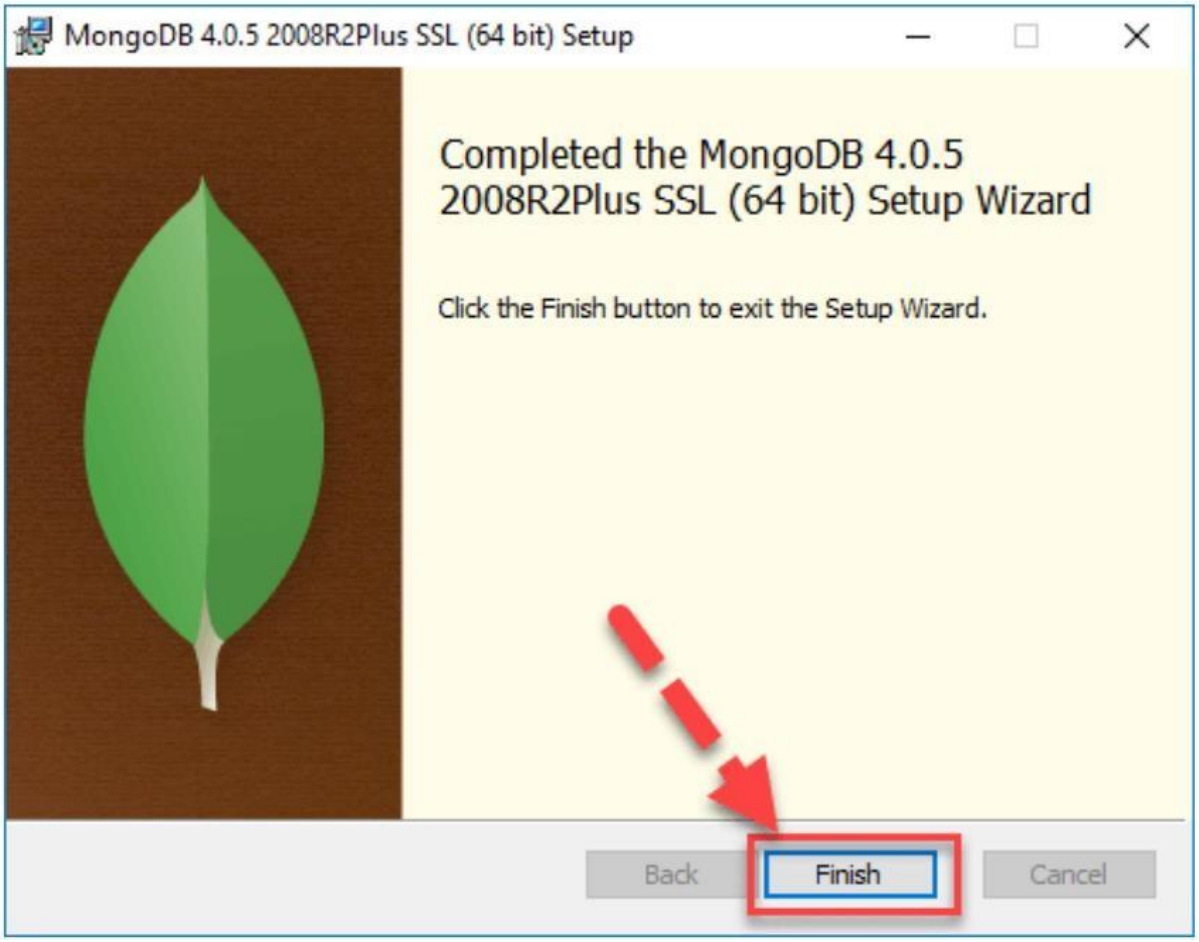
Step 6) Click on the Install button to start the installation.



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

```
{ v: 2, key: { lastUse: 1 }, name: "lsidTTLIndex", ns: "config.system.sessions", expireAfterSeconds: 1800 }
2023-03-03T09:46:21.011+0530 I INDEX    [LogicalSessionCacheRefresh]     building index using bulk method; build may tem
porarily use up to 500 megabytes of RAM
2023-03-03T09:46:21.044+0530 I INDEX    [LogicalSessionCacheRefresh] build index done.  scanned 0 total records. 0 secs
2023-03-03T09:46:21.045+0530 I COMMAND  [LogicalSessionCacheRefresh] command config.$cmd command: createIndexes { create
Indexes: "system.sessions", indexes: [ { key: { lastUse: 1 }, name: "lsidTTLIndex", expireAfterSeconds: 1800 } ], $db: "
config" } numYields:0 reslen:114 locks:{ Global: { acquireCount: { r: 2, w: 2 } }, Database: { acquireCount: { w: 2, W:
1 } }, Collection: { acquireCount: { w: 2 } } } protocol:op_msg 254ms
```

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

```
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
mybigdata  0.000GB
> use mybigdata
```

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

```
C:\Program Files\Python38\Scripts>python -m pip install pymongo
Defaulting to user installation because normal site-packages is not writeable
Collecting pymongo
  Downloading pymongo-4.3.3-cp38-cp38-win_amd64.whl (382 kB)
     |████████████████████████████████| 382 kB 3.3 MB/s
Collecting dnspython<3.0.0,>=1.16.0
  Downloading dnspython-2.3.0-py3-none-any.whl (283 kB)
     |████████████████████████████████| 283 kB ...
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.3.0 pymongo-4.3.3
WARNING: You are using pip version 20.2.1; however, version 23.0.1 is available.
You should consider upgrading via the 'C:\Program Files\Python38\python.exe -m pip install --upgrade pip' command.
```

Note: # **-m** option is for <module-name>

Now you have downloaded and installed a mongoDB driver.

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: Creating a Database: create_dp.py

import pymongo

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

```
['admin', 'config', 'local']
```

Progam 2: Creating a Collection: create_collection.py

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

```
[]
```

Progam 3: Insert into Collection: insert_into_collection.py

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb["student"]
mydict={"name":"Beena", "address":"Mumbai"}
x=mycol.insert_one(mydict) # insert_one(containing the name(s) and value(s) of each field
```

Program 4: Insert Multiple data into Collection: insert_many.py

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb["student"]
mylist=[{"name":"Khyati", "address":"Mumbai"}, {"name":"Kruti", "address":"Mumbai"},
{"name":"Nidhi", "address":"Pune"}, {"name":"Komal", "address":"Pune"},]
x=mycol.insert_many(mylist)
```

Step 6) Test in Mongodb to check database and data inserted in collection

a. If you want to check your database list, use the command show dbs in mongo command prompt

> show dbs

```
admin      0.000GB
config     0.000GB
local      0.000GB
mybigdata  0.000GB
```

b. If you want to use a database with name mybigdata, then use database statement would be as follow:

> use mybigdata

```
switched to db mybigdata
```

c. If you want to check collection in mongodb use the command show collections
> show collections

```
student
```

d. If you want to display the first row from collection: db.collection_name.find()
> db.student.findOne()

```
> db.student.findOne()
{
        "_id" : ObjectId("640178face663db608cef72f"),
        "name" : "Beena",
        "address" : "Mumbai"
}
```

e. If you want to display all the data from collection: db.collection_name.find()
> db.student.find()

```
> db.student.find()
{ "_id" : ObjectId("640178face663db608cef72f"), "name" : "Beena", "address" : "Mumbai" }
{ "_id" : ObjectId("640179336ce317082c266dc1"), "name" : "Khyati", "address" : "Mumbai" }
{ "_id" : ObjectId("640179336ce317082c266dc2"), "name" : "Kruti", "address" : "Mumbai" }
{ "_id" : ObjectId("640179336ce317082c266dc3"), "name" : "Nidhi", "address" : "Pune" }
{ "_id" : ObjectId("640179336ce317082c266dc4"), "name" : "Komal", "address" : "Pune" }
```

f. count number of rows in a collection
> db.student.count()

```
5
```

Site for R packages documentation:
https://cran.r-project.org/web/packages/available_packages_by_name.html

PRACTICAL 4

Aim : Write a Pig Script for solving counting problems.

Steps :

cat> /home/cloudera/input.csv

cat /home/cloudera/input.csv

pig -x local

lines = load '/home/cloudera/input.csv' as (line:chararray);

words = foreach lines GENERATE FLATTEN(TOKENIZE(line)) as woed;

grouped = GROUP words by woed;

wordcount = foreach grouped GENERATE group, COUNT(words);

dump wordcount;

PRACTICAL 5

Aim: Install Hive and use Hive Create and store structured databases.

Steps:

```
cat > /home/cloudera/employee.txt
        1~Sachine~Pune~Product Engineering~100000~Big Data
        2~Gaurav~Banglore~Sales~90000~CRM
        3~Manish~Chennai~Recruiter~125000~HR
        4~Bhushan~Hyderabad~Developer~50000~BFSI
cat  /home/cloudera/employee.txt
sudo -u hdfs hadoop fs -put /home/cloudera/employee.txt /inputdirectroy
hdfs dfs -ls /
hdfs dfs -ls /inputdirectory
hadoop fs -cat /inputdirectory/employee.txt
hive
show databases;
create database organization;
show databases;
use organization;
show tables;
hive> create table employee(
   > id int,
   > name string,
   > city string,
   > department string,
   > salary int,
   > domain string)
   > row format delimited
   > fields terminated by '~';

show tables;
select * from employee;
```

```
show tables;
load data inpath '/inputdirectory/employee.txt' overwrite into table employee;
show tables;
select * from employee;
```

PRACTICAL 6

PRACTICAL 6A

Aim:
a) Implement Decision Tree classification technique using Social_Network_Ads.csv dataset.

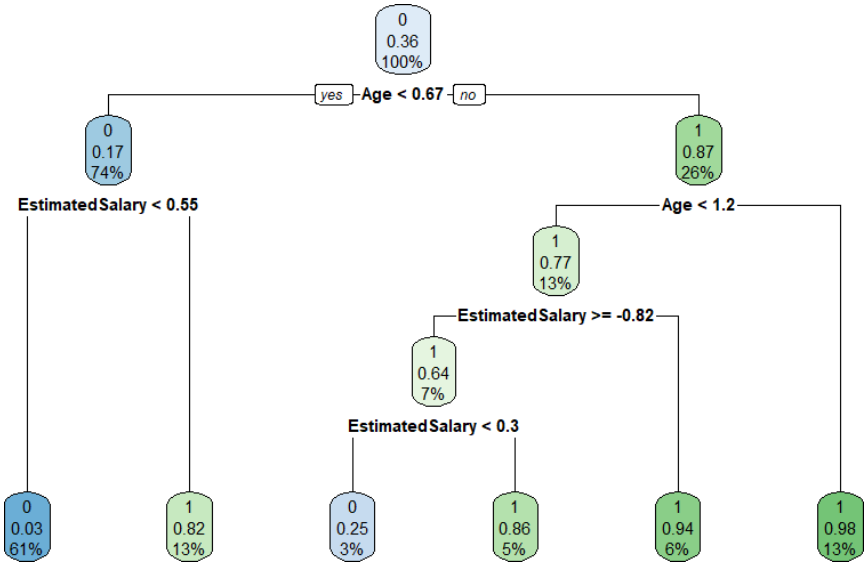
Code:

```
# Decision Tree Classification

# Importing the dataset
dataset = read.csv("C:\\2022-23\\BDA PRACTICAL 2023\\Social_Network_Ads.csv")
#print(dataset)
dataset = dataset[3:5] # columns 3 4 ad 5
print(dataset)

# Encoding the target feature as factor(just like a vector having levels
# levels to convey that only two possible values for purchased - 0 & 1
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
print (dataset$Purchased)

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

# Feature Scaling - scale() method centers and/or scales the columns of a numeric matrix.
training_set[-3] = scale(training_set[-3]) # scaling first 2 columns, don't consider 3rd column
test_set[-3] = scale(test_set[-3])
#print(test_set[-3])

# Fitting Decision Tree Classification to the Training set
install.packages('rpart')
library(rpart) # for partitioning tree
install.packages('rpart.plot')
library(rpart.plot)

classifier = rpart(formula = Purchased ~ .,data = training_set)

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

# Making the Confusion Matrix
```

```
cm = table(test_set[, 3], y_pred)
print(cm)

y_grid = predict(classifier, newdata = grid_set, type = 'class')

# Plotting the tree
#extra=106 class model with a binary response
#extra=104 class model with a response having more than two levels
rpart.plot(classifier, extra = 106)
```

Output:

```
> cm = table(test_set[, 3], y_pred)
> print(cm)
   y_pred
     0  1
  0 53 11
  1  6 30
```



PRACTICAL 6B

SVM Classification

Aim:

b) Implement SVM Classification technique using Social_Network_Ads.csv dataset. Evaluate the performance of classifier.

Code:

```
# Support Vector Machine (SVM)

# Importing the dataset
dataset = read.csv('C:\\2022-23\\BDA PRACTICAL 2023\\Social_Network_Ads.csv')
dataset = dataset[3:5]
print(dataset)
print(dataset$Purchased)
# Splitting the dataset into the Training set and Test set
install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
print(training_set)
test_set = subset(dataset, split == FALSE)
print(test_set)
# Feature Scaling
training_set[-3] = scale(training_set[-3]) # [-3] means 3rd index will be dropped
test_set[-3] = scale(test_set[-3])
print(training_set[-3])
print (test_set[-3])
# Fitting SVM to the Training set
install.packages('e1071')
library(e1071)
classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')
print (classifier)

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

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

Output:

```
> cm = table(test_set[, 3], y_pred)
> print (cm)
   y_pred
     0  1
  0 57  7
  1 13 23
```

PRACTICAL 7

PRACTICAL 7A

Logistic Regression

Aim:

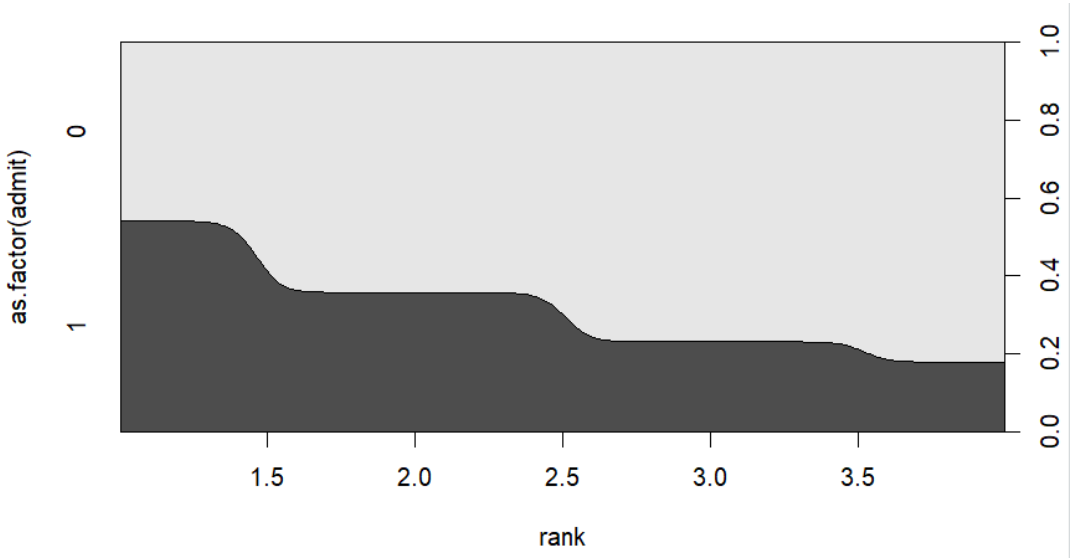
a) Import data from web storage – binary.csv. Name the dataset and do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not.

Code:

```
#fetch the data
college <- read.csv("https://raw.githubusercontent.com/csquared/udacity-
dlnd/master/nn/binary.csv")
head(college)
nrow(college)

install.packages("caTools")    # For Logistic regression
library(caTools)

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

training_reg <- subset(college, split == "TRUE")
test_reg <- subset(college, split == "FALSE")

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

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

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


# Changing probabilities
predict_reg <- ifelse(predict_reg >0.5, 1, 0)
predict_reg


# Evaluating model accuracy
# using confusion matrix
table(test_reg$admit, predict_reg)
```

Output:

```
> table(test_reg$admit, predict_reg)
   predict_reg
     0  1
  0 70  2
  1 21  7
```



PRACTICAL 7B

Multiple Regression Model

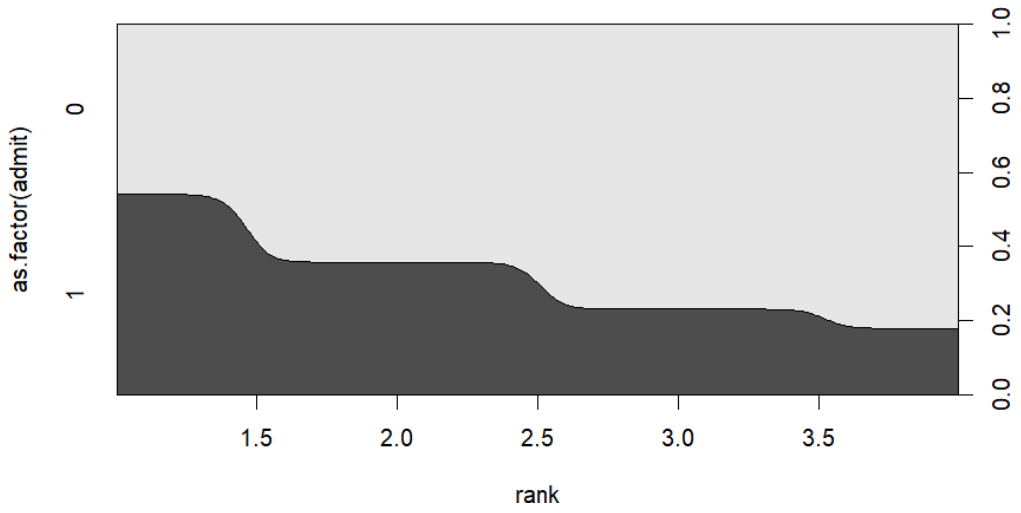
Aim:

b) Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset – binary.csv.

Code:

```
#fetch the data
college <- read.csv("https://raw.githubusercontent.com/csquared/udacity-
dlnd/master/nn/binary.csv")
head(college)
nrow(college)

install.packages("caTools")    # For Logistic regression
library(caTools)

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

training_reg <- subset(college, split == "TRUE")
test_reg <- subset(college, split == "FALSE")

# Training model
fit_MRegressor_model <- lm(formula = admit ~ gre+gpa+rank,
                  data = training_reg)

# Predict test data based on model
predict_reg <- predict(fit_MRegressor_model,
                   newdata = test_reg)
predict_reg

cdplot(as.factor(admit)~  gpa,  data=college)
cdplot(as.factor(admit)~  gre,  data=college)
cdplot(as.factor(admit)~ rank, data=college)
```

Output:


as.factor(admit)
0
1
0.0
0.2
0.4
0.6
0.8
1.0
1.5
2.0
2.5
3.0
3.5
rank

PRACTICAL 8

PRACTICAL 8A

Classification Model
Naïve Bayes Classification

Aim:

a) Implement Naïve Bayes Classification technique using Social_Network_Ads.csv dataset. Evaluate the performance of classifier.

Code:

```
# Naive Bayes
# Importing the dataset
dataset = read.csv('C:\\2022-23\\BDA PRACTICAL 2023\\Social_Network_Ads.csv')
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')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
# Feature Scaling
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])
# Fitting Naive Bayes to the Training set
install.packages('e1071')
library(e1071)
classifier = naiveBayes(x = training_set[-3],
                        y = training_set$Purchased)
# 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(cm)
```

Output:

```
> cm = table(test_set[, 3], y_pred)
> print(cm)
   y_pred
     0  1
  0 57  7
  1  7 29
```

PRACTICAL 8B

K-Means Clustering

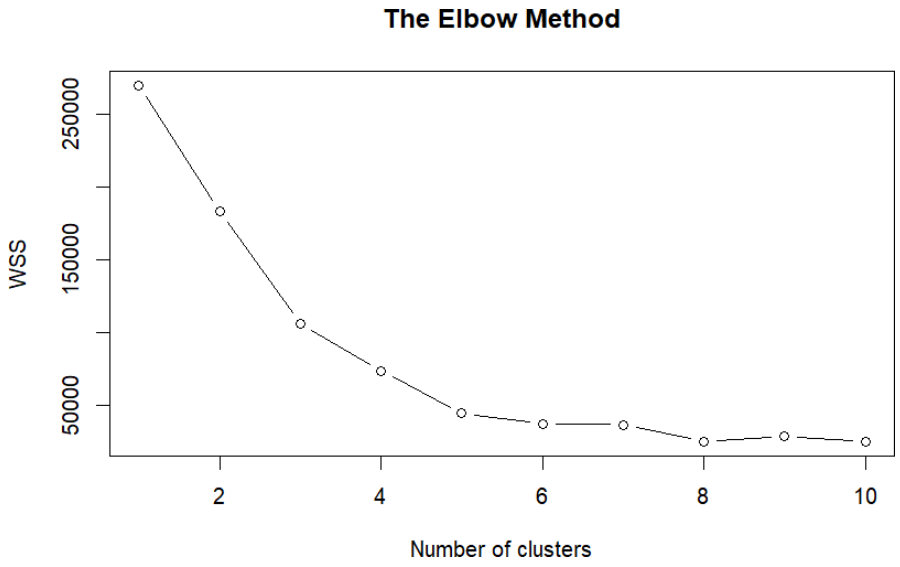
Aim:

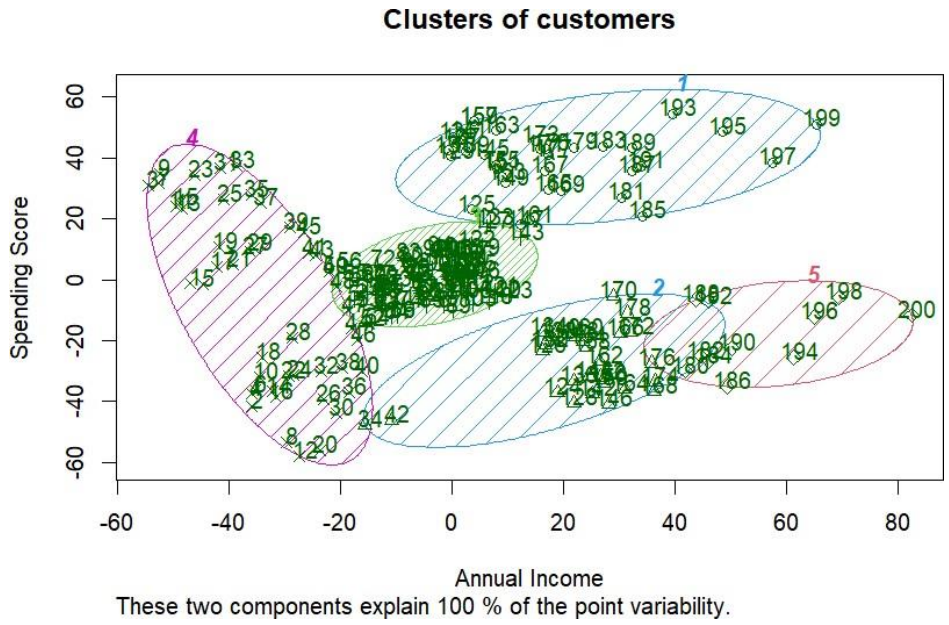
a) Clustering algorithms for unsupervised classification. Read a datafile Mall_Customers.csv and apply k-means clustering. Plot the cluster data using R visualizations.

Code:

```
# K-Means Clustering

# Importing the dataset
dataset = read.csv('C:\\2022-23\\BDA PRACTICAL 2023\\Mall_Customers.csv')
head(dataset)
dataset = dataset[4:5]
head(dataset)

wcss = vector()
for (i in 1:10) wcss[i] = sum(kmeans(dataset, i)$withinss)
plot(1:10,
    wcss,
    type = 'b',
    main = paste('The Elbow Method'),
    xlab = 'Number of clusters',
    ylab = 'WSS')


# Fitting K-Means to the dataset with no of clusters = 5
kmeans = kmeans(x = dataset, centers = 5)
y_kmeans = kmeans$cluster

# Visualising the clusters
library(cluster)
clusplot(dataset,
        y_kmeans,
        lines =  0,
        shade = TRUE,
        color = TRUE,
        labels = 2,
        main = paste('Clusters of customers'),
        xlab = 'Annual Income',
        ylab = 'Spending Score')
```

Output:





These two components explain 100 % of the point variability.