

INDEX

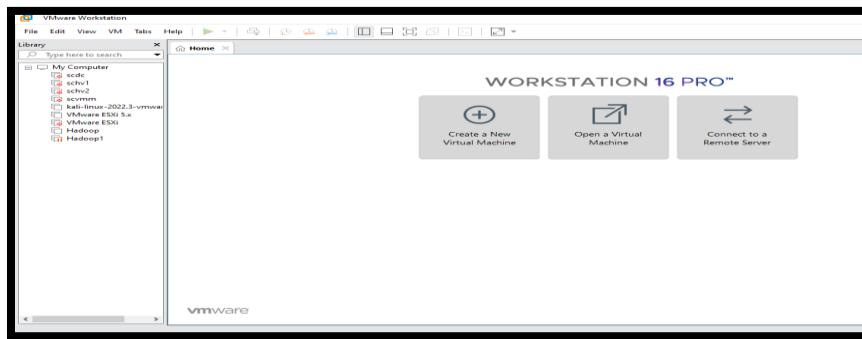
Sr.No	Title	Date	Sign
1.	Install, configure and run Hadoop and HDFS and explore HDFS.		
2.	Implement word count /frequency programs using MapReduce.		
3.	Implement an application that stores big data in Hbase/MongoDB andmanipulate it using R/python.		
4.	Implement the program using PIG		
5.	To configure Hive and implement the application using Hive.		
6.	Implement Decision tree classification techniques		
7.	Implement SVM Classification Technique.		
8.	Import the data and implement regression model.		
9.	MULTIPLE REGRESSION Model Apply multiple regression, if data havea continuous independent variable. Apply on above dataset.		
10.	Classification using Decision Tree classifier and Naïve bayes classifier.		

Practical No: 1

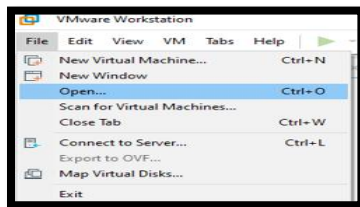
Aim: - Install, configure and run Hadoop and HDFS and explore HDFS.

1. **Hadoop:-** Hadoop represents another example of Big Data innovation on the IT infrastructure. Apache Hadoop is an open source framework that allows companies to process vast amounts of information in a highly parallelized way. Hadoop represents a specific implementation of the MapReduce paradigm and was designed by Doug Cutting and Mike Cafarella in 2005 to use data with varying structures.
2. **HDFS:-** The Hadoop Distributed File System (HDFS) is a distributed file system that runs on standard or low-end hardware. Developed by Apache Hadoop, HDFS works like a standard distributed **file system** but provides better data throughput and access through the MapReduce algorithm, high fault tolerance and native support of large data sets.

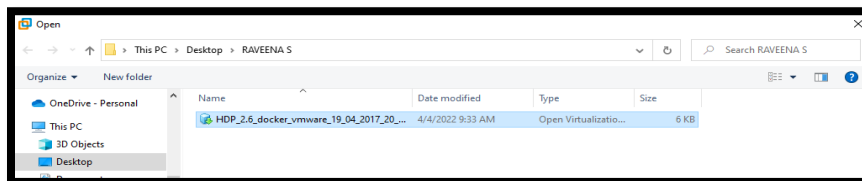
Step 1:- Open VMWare Workstation.



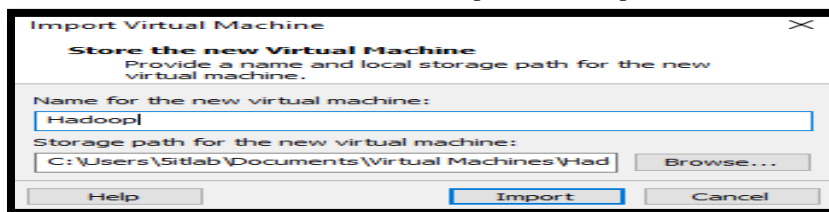
Step 2:- Select File-> Open



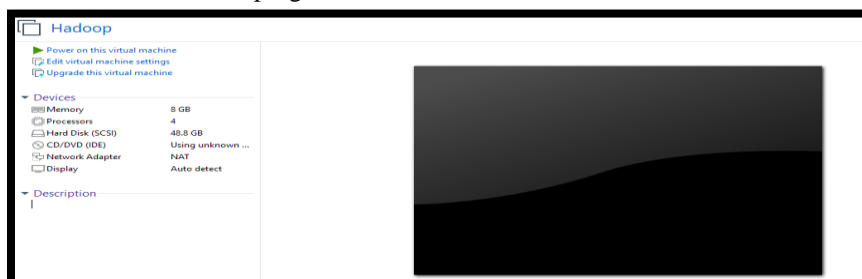
Step 3 : RAVEENA S -> Select HDP_2.6_docker_vmware

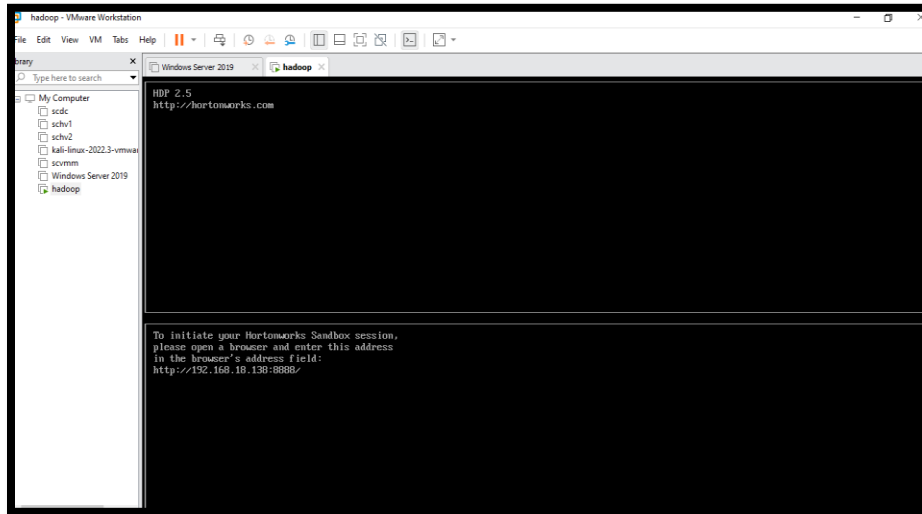


Step 4: Name for the new virtual machine:Hadoop-> Select Import

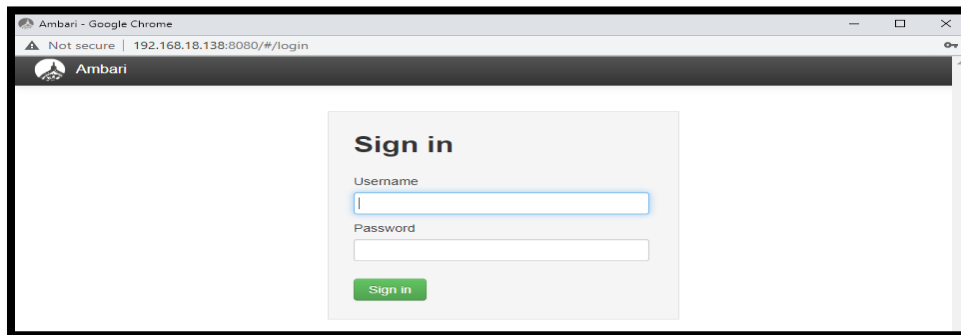
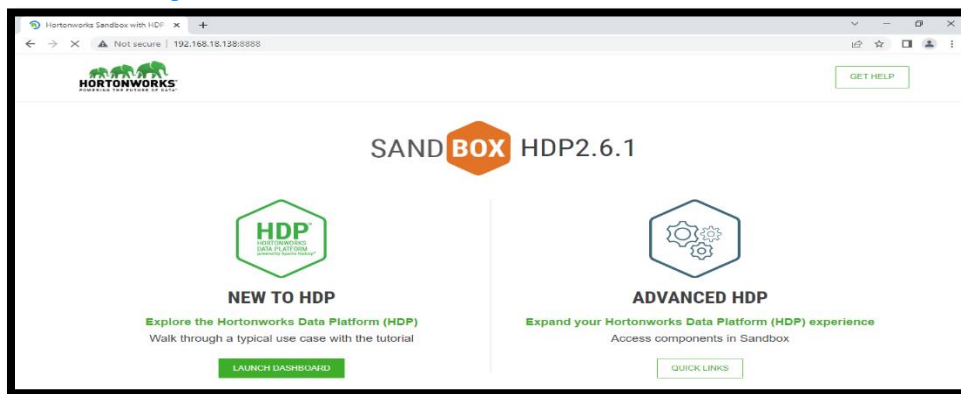


Step 5: Power on the virtual program

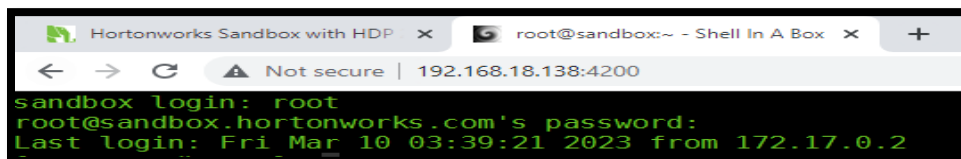




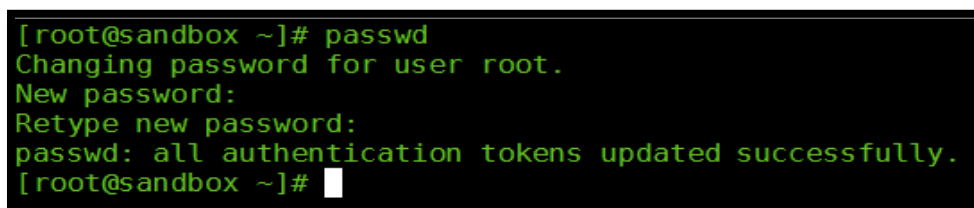
Step 6: Web link: <http://192.168.18.138:8888/>



Step 7: Web link: 192.168.18.138:4200



Step 8: RESET PASSWORD

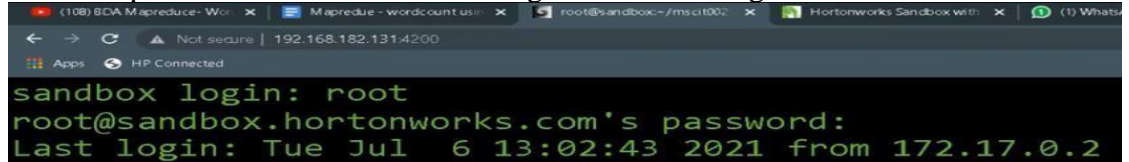


Practical No: 2

Aim:- Implement word count /frequency programs using MapReduce.

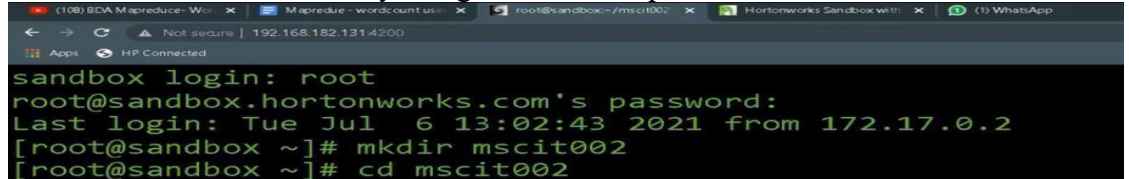
Step 1:- Start the server and note down the ip address

Step 2:- Past the ip address in the browser and login to sandbox login terminal



```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Tue Jul  6 13:02:43 2021 from 172.17.0.2
```

Step 3:-Create a file in local directory using “mkdir mscitp002”

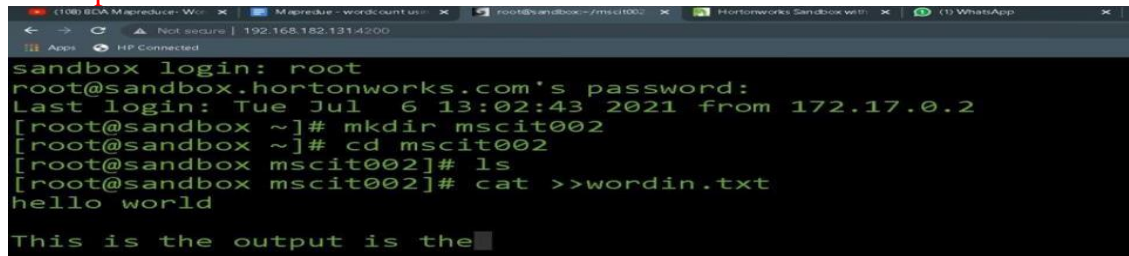


```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Tue Jul  6 13:02:43 2021 from 172.17.0.2
[root@sandbox ~]# mkdir mscitp002
[root@sandbox ~]# cd mscitp002
```

Step 4:-Cat >>wordin.txt : create a text file with data

Data : **hello world**

This is the output is the



```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Tue Jul  6 13:02:43 2021 from 172.17.0.2
[root@sandbox ~]# mkdir mscitp002
[root@sandbox ~]# cd mscitp002
[root@sandbox mscitp002]# ls
[root@sandbox mscitp002]# cat >>wordin.txt
hello world
This is the output is the
```

Step 5:-Cat >>WordCount.java : in this java file will save java packages



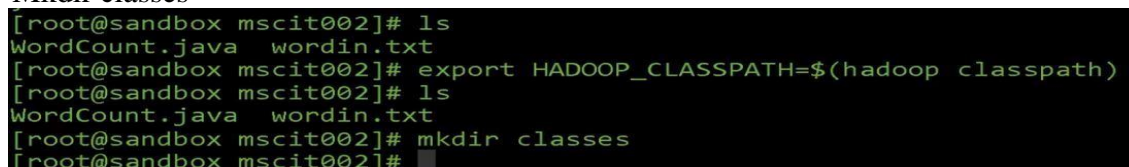
```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Tue Jul  6 13:02:43 2021 from 172.17.0.2
[root@sandbox ~]# mkdir mscitp002
[root@sandbox ~]# cd mscitp002
[root@sandbox mscitp002]# ls
[root@sandbox mscitp002]# cat >>wordin.txt
hello world
This is the output is the
[root@sandbox mscitp002]# vi wordin.txt
[root@sandbox mscitp002]# cat wordin.txt
hello world
This is the output is the
[root@sandbox mscitp002]# cat >>WordCount.java
```

Step 6:-Export HADOOP_CLASSPATH=\$(hadoop classpath) : export the classpath in to our system



```
[root@sandbox mscitp002]# ls
WordCount.java wordin.txt
[root@sandbox mscitp002]# export HADOOP_CLASSPATH=$(hadoop classpath)
[root@sandbox mscitp002]# ls
WordCount.java wordin.txt
[root@sandbox mscitp002]#
```

Step 7:-Mkdir classes



```
[root@sandbox mscitp002]# ls
WordCount.java wordin.txt
[root@sandbox mscitp002]# export HADOOP_CLASSPATH=$(hadoop classpath)
[root@sandbox mscitp002]# ls
WordCount.java wordin.txt
[root@sandbox mscitp002]# mkdir classes
[root@sandbox mscitp002]#
```

Step 8:-Javac -classpath \${HADOOP_CLASSPATH} -d classes WordCount.java : to compile the java program

```
[root@sandbox mscit002]# javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java
javac: file not found: WordCount.java
Usage: javac <options> <source files>
use -help for a list of possible options
[root@sandbox mscit002]#
```

Step 9:-Ls classes

```
[root@sandbox mscit002]# ls classes
WordCount.class WordCount$IntSumReducer.class WordCount$TokenizerMapper.class
[root@sandbox mscit002]#
```

Step 10:-Jar -cvf WordCount.jar -C classes/ . : bind all the classes together into single jar file

```
[root@sandbox mscit002]# jar -cvf WordCount.jar -C classes/ .
Manifest
WordCount$IntSumReducer.class(in = 1739) (out= 742)(deflated 57%)
WordCount$TokenizerMapper.class(in = 1736) (out= 756)(deflated 56%)
WordCount.class(in = 1491) (out= 813)(deflated 45%)
[root@sandbox mscit002]#
```

Step 11:-Hdfs dfs -mkdir /p2: create a folder in hadoop system

```
[root@sandbox mscit002]# hdfs dfs -mkdir /p2
[root@sandbox mscit002]# ls
-bash: ls: command not found
[root@sandbox mscit002]# ls
classes WordCount.jar WordCount.java wordin.txt
```

Step 12:-Hdfs dfs -put wordin.txt /p2 : uploading wordin file into p2 folder

```
classes WordCount.jar WordCount.java wordin.txt
[root@sandbox mscit002]# hdfs dfs -put wordin.txt /p2
```

Step 13:-Hdfs dfs -ls /p2

```
[root@sandbox mscit002]# hdfs dfs -ls /p2
Found 1 items
-rw-r--r-- 1 root hdfs 38 2021-07-06 13:31 /p2/wordin.txt
[root@sandbox mscit002]# p2/ /p2output
21/07/06 13:32:12 INFO client.RMProxy: Connecting to ResourceManager at sandbox.hortonworks.com/172.17.0.2:8032
21/07/06 13:32:12 INFO client.AHSProxy: Connecting to Application History server at sandbox.hortonworks.com/172.17.0.2:10200
21/07/06 13:32:13 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your
oolRunner to remedy this.
21/07/06 13:32:13 INFO input.FileInputFormat: Total input paths to process : 1
21/07/06 13:32:14 INFO mapreduce.JobSubmitter: number of splits:1
21/07/06 13:32:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1625575002861_0001
21/07/06 13:32:16 INFO impl.YarnClientImpl: Submitted application application_1625575002861_0001
21/07/06 13:32:16 INFO mapreduce.Job: The url to track the job: http://sandbox.hortonworks.com:8088/proxy/application_1625575002861_0001/
21/07/06 13:32:16 INFO mapreduce.Job: Running job: job_1625575002861_0001
```

Step 14:-Hadoop jar WordCount.jar WordCount /p2/ /p2output :executing the jar file on hadoop to get the output of the code

```
[root@sandbox mscit002]#
This 1
hello 1
is 2
output 1
the 2
world 1
```

Step 15:-Hdfs dfs -ls /p2output

```
[root@sandbox mscit002]# hdfs dfs -ls /p2output
Found 2 items
-rw-r--r-- 1 root hdfs 0 2021-07-06 13:32 /p2output/_SUCCESS
-rw-r--r-- 1 root hdfs 43 2021-07-06 13:32 /p2output/part-r-00000
[root@sandbox mscit002]#
```

Step 16:- Hdfs dfs -cat /p2output/* : print all the file

Practical No: 3

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

Step A: Install Mongo database

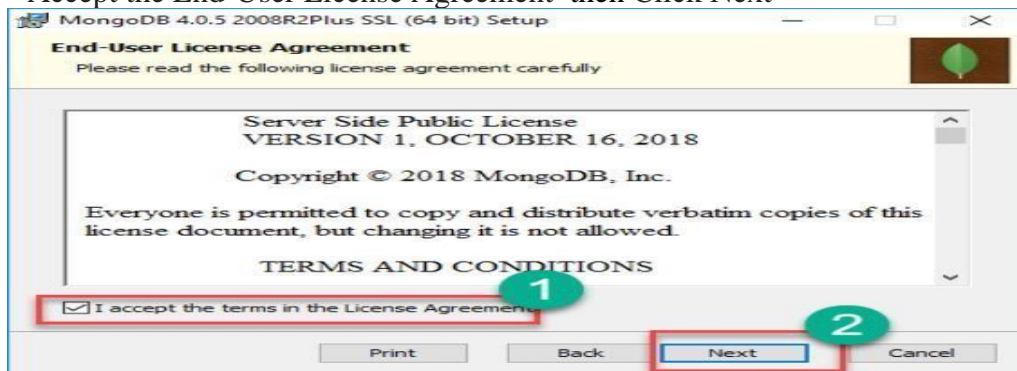
Step 1:- Goto (<https://www.mongodb.com/download-center/community>) and Download MongoDB Community Server. We will install the 64-bit version for Windows.



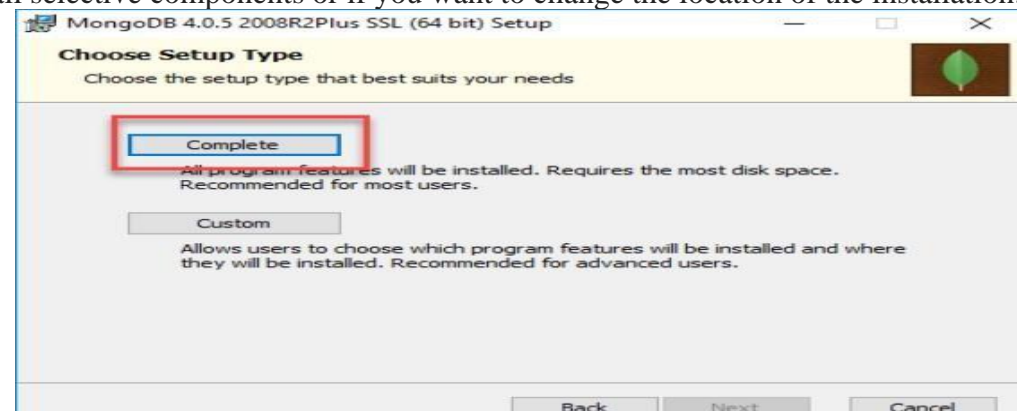
Step 2:- Once download is complete open the msi file. Click Next in the start up screen



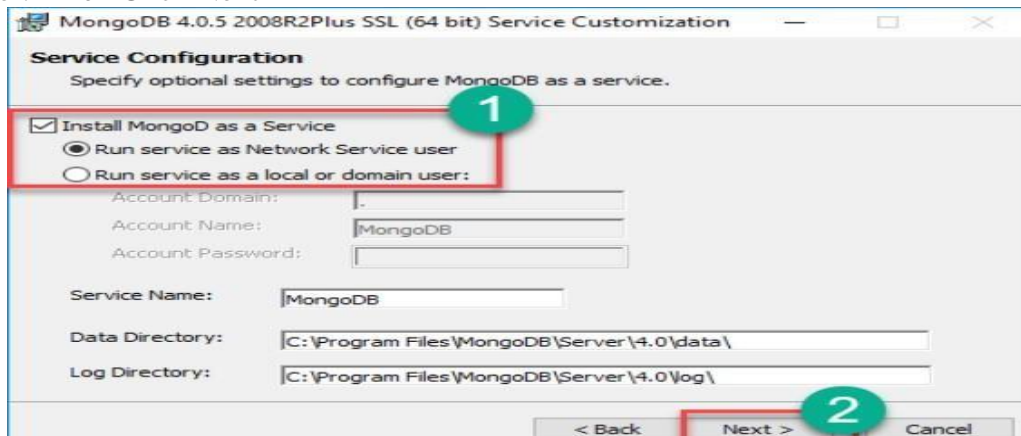
Step 3:- Accept the End-User License Agreement then Click Next



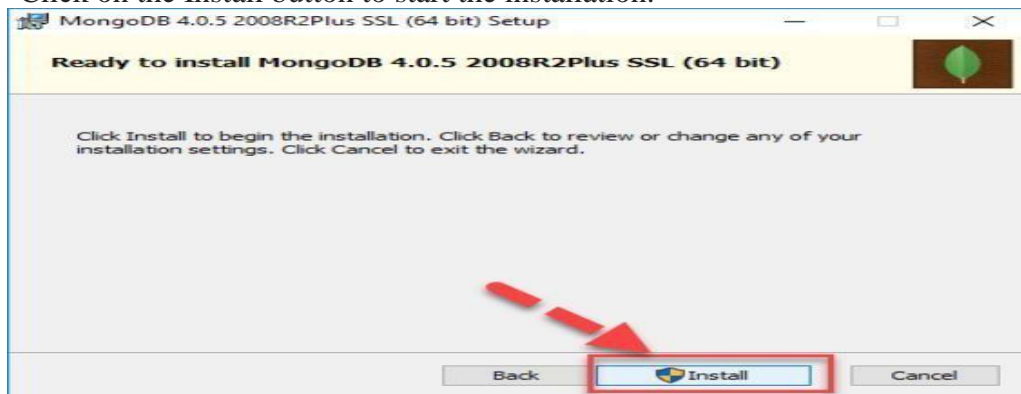
Step 4:- Click on the "complete" button to install all of the components. The custom option can be used to install selective components or if you want to change the location of the installation.



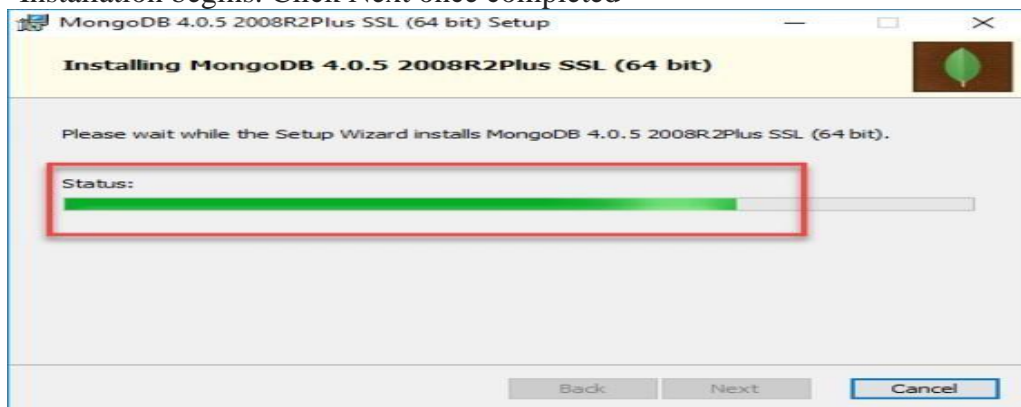
Step 5:- Select “Run service as Network Service user”. make a note of the data directory, we’ll need this later. Then Click Next



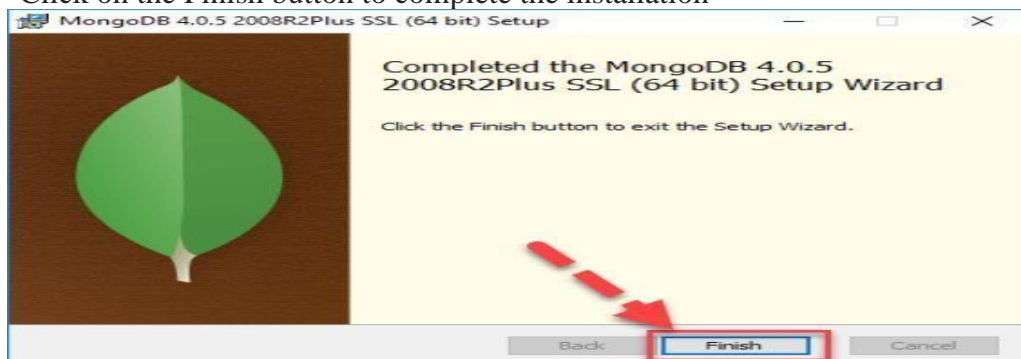
Step 6:- Click on the Install button to start the installation.



Step 7:- Installation begins. Click Next once completed



Step 8:- Click on the Finish button to complete the installation



Test Mongoddb

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 item

- **Create the directory where MongoDB will store it's files.** From the command prompt run md \data\db . This is the default location. However, other locations can be specified using the --dbpath parameter. See [the Mongo docs](#) for more information.

C:\>md data

C:\>md data\db

C:\Program Files\MongoDB\Server\4.05\bin>mongod.exe --dbpath "C:\data"

- **Start the mongod daemon** by running C:\mongodb\bin\mongod.exe in the Command Prompt. Or by running, C:\path\to\mongodb\bin\mongod.exe
- **Connect to MongoDB using the Mongo shell** While the MongoDB daemon is running, from a different Command prompt window run C:\mongodb\bin\mongo.exe
- C:\Program Files\MongoDB\Server\4.05\bin>mongod.exe --dbpath "C:\data"
- C:\Program Files\MongoDB\Server\4.05\bin>mongo.exe

Step B: - Install PyMongo

C:\Users\YourName\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install pymongo

Now you have downloaded and installed a mongoDB driver.

Test PyMongo

demo_mongoddb_test.py:

import pymongo

Program 1: Creating a Database

```
Python 3.8.1rc1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1rc1 (tags/v3.8.1rc1:b00a2b5, Dec 10 2019, 01:13:53) [MSC v.1916 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import pymongo
>>> myclient = pymongo.MongoClient("mongodb://localhost:27017/")
>>> mydb = myclient["mybigdata"]
>>> print(myclient.list_database_names())
['admin', 'config', 'local']
b>>>
```

Program 2: Creating a Collection

```
Python 3.8.1rc1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1rc1 (tags/v3.8.1rc1:b00a2b5, Dec 10 2019, 01:13:53) [MSC v.1916 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import pymongo
>>> myclient = pymongo.MongoClient("mongodb://localhost:27017/")
>>> mydb = myclient["mybigdata"]
>>> print(myclient.list_database_names())
['admin', 'config', 'local']
>>> import pymongo
>>> myclient = pymongo.MongoClient("mongodb://localhost:27017/")
>>> mydb = myclient["mybigdata"]
>>> mycol=mydb["student"]
SyntaxError: invalid character in identifier
>>> mycol=mydb["student"]
>>> print(mydb.list_collection_names())
[]
>>>
```

Program 3: Insert into Collection

```
Python 3.8.1rc1 Shell
File Edit Shell Debug Options Window Help
['admin', 'config', 'local']
>>> import pymongo
>>> myclient = pymongo.MongoClient("mongodb://localhost:27017/")
>>> mydb = myclient["mybigdata"]
>>> mycol=mydb["student"]
SyntaxError: invalid character in identifier
>>> mycol=mydb["student"]
>>> print(mydb.list_collection_names())
[]
>>> import pymongo
>>> myclient = pymongo.MongoClient("mongodb://localhost:27017/")
>>> mydb = myclient["mybigdata"]
>>> mycol = mydb["student"]
>>> mydict={"name":"Shoaib","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

```
>>> import pymongo
>>> myclient = pymongo.MongoClient("mongodb://localhost:27017/")
>>> mydb = myclient["mybigdata"]
>>> mycol = mydb["student"]
>>> mylist=[{"name":"shoaib","address":"mumbai"}, {"name":"asgar","address":"mumbai"}, {"name":"ronak","address":"pune"}]
>>> x=mycol.insert_many(mylist)
```

Test in Mongodb to check database and data inserted in collection

- If you want to check your database list, use the command **show dbs** in mongo command prompt
- If you want to use a database with name mybigdata, then use database statement would be as follow: **use mybigdata**
- If you want to check collection in mongodb use the command **show collections**
- If you want to display all the data from collection: **db.collection_name.find()** or **db.collection_name.find().pretty()**



```
Select C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe

> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
mybigdata  0.000GB
> use mybigdata
switched to db mybigdata
> show collections
student
> db.collection_name.find()
> db.collection_name.find().pretty()
> db.collection_student.find().pretty()
> db.student.find().pretty()
{
  "_id" : ObjectId("5e969f29f342c45e06640b0b"),
  "name" : "Shoaib",
  "address" : "Mumbai"
}
{
  "_id" : ObjectId("5e96a06ff342c45e06640b0d"),
  "name" : "shoaib",
  "address" : "mumbai"
}
{
  "_id" : ObjectId("5e96a06ff342c45e06640b0e"),
  "name" : "asgar",
  "address" : "mumbai"
}
{
  "_id" : ObjectId("5e96a06ff342c45e06640b0f"),
  "name" : "ronak",
  "address" : "pune"
}
>
```

Practical No: 4

Aim: Implement the program using PIG.

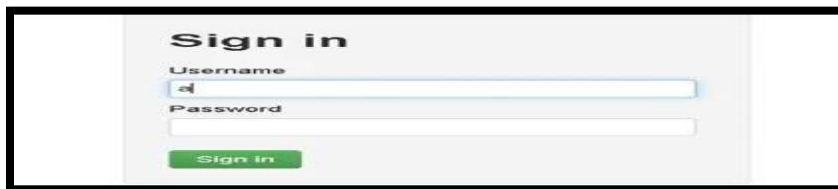
Description:

Step1:- Login to (gui) interface using the ip address.

Step2:- Click on the launch dashboard.



Step 3:- We need to put on the login details which include Username & Password



Step4:- The next step is to click on the pig component and select the pig client.



Step 5:- We need to make a directory using the “mkdir” command & change the directory using the “cd” directory.

```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Tue Jul 6 13:42:10 2021 from 172.17.0.2
[root@sandbox ~]# mkdir mscitp5
[root@sandbox ~]# cd mscitp5
[root@sandbox mscitp5]#
```

Step 6:- To create a file we need to use the “cat” command and name the file as “student.txt” and insert data in it.

```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Tue Jul 6 13:42:10 2021 from 172.17.0.2
[root@sandbox ~]# mkdir mscitp5
[root@sandbox ~]# cd mscitp5
[root@sandbox mscitp5]# cd mscitp5/
-bash: cd: mscitp5/: No such file or directory
[root@sandbox mscitp5]# cat >>student.txt
```

Step 7:- Create another file named as “Program” and save the file with the extension as “.pig”.

```
[root@sandbox mscitp5]# cat >>program.pig
student = LOAD 'student.txt' USING PigStorage
as (id:int, firstname:chararray, lastname:
student_order = ORDER student BY age DESC;
```

Step 8:- Before executing we need to upload the “student” file in the “Hadoop Distributed File System”(Hdfs).

```
004,Preethi,Agarwal,21,9848022330,Pune
[root@sandbox mscitp5]# hdfs dfs -put student.txt /user/root/
```

Step 9:- To run the program we use keyword pig program.pig.

```
004,Preethi,Agarwal,21,9848022330,Pune
[root@sandbox mscitp5]# hdfs dfs -put student.txt /user/root/
[root@sandbox mscitp5]# pig program.pig
```

Output:-

```
2021-07-06 14:24:16,536 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(7,Komal,Nayak,24,9848022334,trivendram)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
(5,Trupthi,Mohanthy,23,9848022336,Bhuvaneshwar)
(6,Archana,Mishra,23,9848022335,Chennai)
2021-07-06 14:24:16,824 [main] INFO org.apache.pig.Main - Pig script completed in 49 seconds and 925 milliseconds (49925 ms)
2021-07-06 14:24:16,837 [main] INFO org.apache.pig.backend.hadoop.executionengine.tez.TezLauncher - Shutting down thread pool
2021-07-06 14:24:16,860 [pool-1-thread-1] INFO org.apache.pig.backend.hadoop.executionengine.tez.TezSessionManager - Shutting down Tez sa
client05d49100f
```

Practical No: 5

Aim: To configure Hive and implement the application using Hive.

Description:

Step 1:- We have to login to GUI using the ip address given.

Step 2:- The next step is to launch the dashboard & Put the login details the username and the root password allotted to you.

```

HDP 2.5
http://hortonworks.com

To initiate your Hortonworks Sandbox session,
please open a browser and enter this address.
In the browser's address field:
http://192.168.102.131:8080/

sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Tue Jul 6 14:36:39 2021 from 172.17.0.2
[root@sandbox ~]#

```

Step 3:- We will create a folder using the “mkdir” command .

Step 4:- Create a file using the “cat” command inside the folder.

```

login: root
ndbox.hortonworks.com's password:
gin: Tue Jul 6 14:36:39 2021 from 172.17.0.2
andbox ~]# mkdir p6mscit
andbox ~]# cd p6mscit
andbox p6mscit]# cat >>data.txt

```

Step 5:- Save your work on the terminal using “:wq”.

Step 6:- Next step after creating file is to open “HIVE terminal”. To open hive terminal we will use the “HIVE” keyword.

```

[root@sandbox p6mscit]# ls
data.txt
[root@sandbox p6mscit]# hive

```

Step 7:- Create a table using the query after the hive terminal is opened.

```

Logging initialized using configuration in file:/etc/hive/2.6.0.3-8/0/hive-log4j.properties
hive> CREATE TABLE IF NOT EXISTS employee ( eid int, fname String, lname String, age int, c
>
> COMMENT 'Employee details'
>
> ROW FORMAT DELIMITED
>
> FIELDS TERMINATED BY ','
>
> LINES TERMINATED BY '\\n'
>
hive>

```

Step 8:- Using the “Select” query keyword to we will check whether the data is stored in the table or not.

```

hive> select * from employee;
OK
Time taken: 1.3 seconds
hive>

```

Step 9:- The next step is to load the data.

Step 10:- Use the “SELECT” keyword to display the data.

Output :-

```

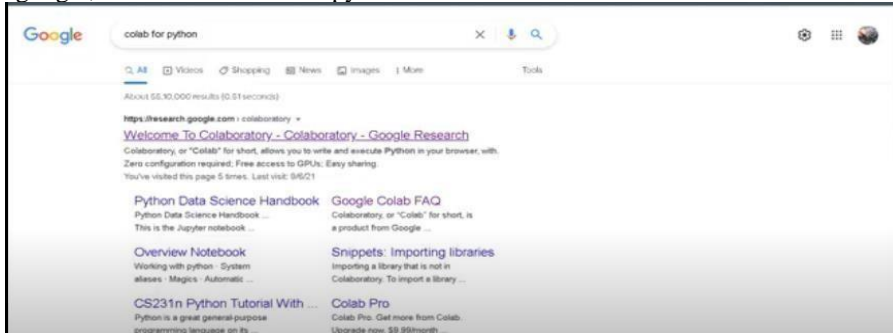
hive> Select * from employee where age >23;
OK
1      Rajiv   Reddy    21      9848022337    Hyderabad
2      siddarth Battacharya 22      9848022338    Kolkata
3      Rajesh   Khanna   22      9848022339    Delhi
4      Preethi  Agarwal  21      9848022330    Pune
5      Trupthi  Mohanthy 23      9848022336    Bhuwaneshwar
6      Archana  Mishra   23      9848022335    Chennai
7      Komal    Nayak    24      9848022334    trivendram
8      Bharathi Nambiayar 24      9848022333    Chennai
Time taken: 0.728 seconds, Fetched: 2 row(s)
hive>

```

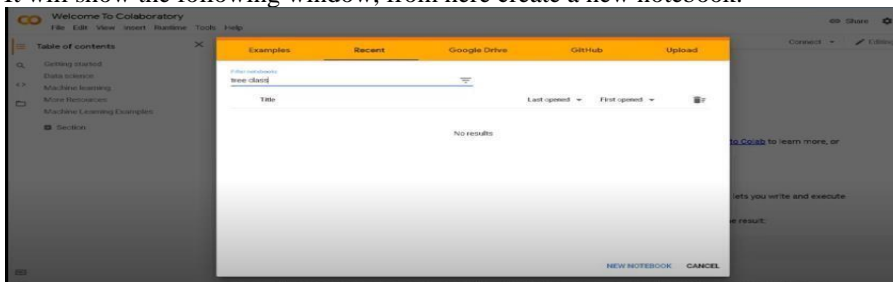
Practical No: 6

Aim: Implement Decision tree classification techniques

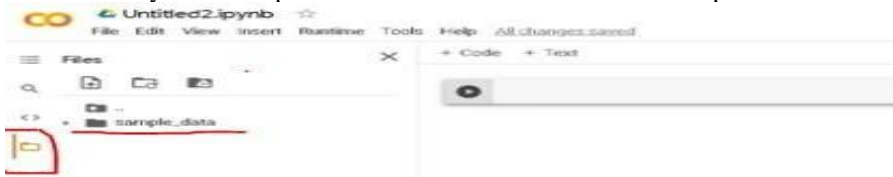
Step 1:- Go to google, Search for "colab for python" and click on the first link.



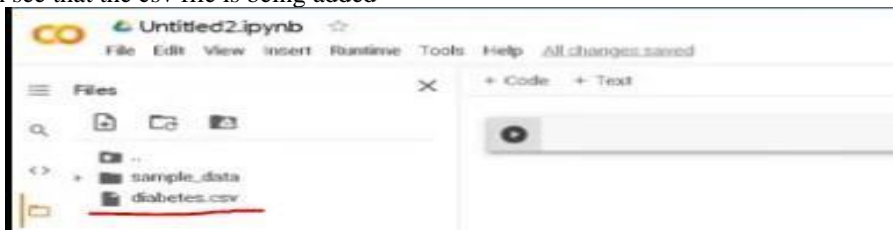
Step 2:- It will show the following window, from here create a new notebook.



Step 3:- The window will be open now we need to add the csv file. Click on the " folder icon" which is on the left side bar - Right click on directory -- select upload - brows the csv file and click on "open"



You can see that the csv file is being added



Step 4:- add the following code inside the box and click on the "Run" button

```
import pandas as pd
import io
import image io from sklearn.tree
import export_graphviz
import graphviz
```

Step 5:- click on "+Code" to add further code and write following code to retrieve data from csv file.

```
diabetes_data = pd.read_csv('diabetes.csv', encoding='latin-1')
diabetes_data.head()
```

Click on "Run". It will show the data in the csv file as follows

	Pregnancies	PlasmaGlucose	DiastolicBloodPressure	TricepsThickness	SerumInsulin	BMI	DiabetesPedigree	Age	Diabetic
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	127	40	35	168	43.1	2.298	33	1

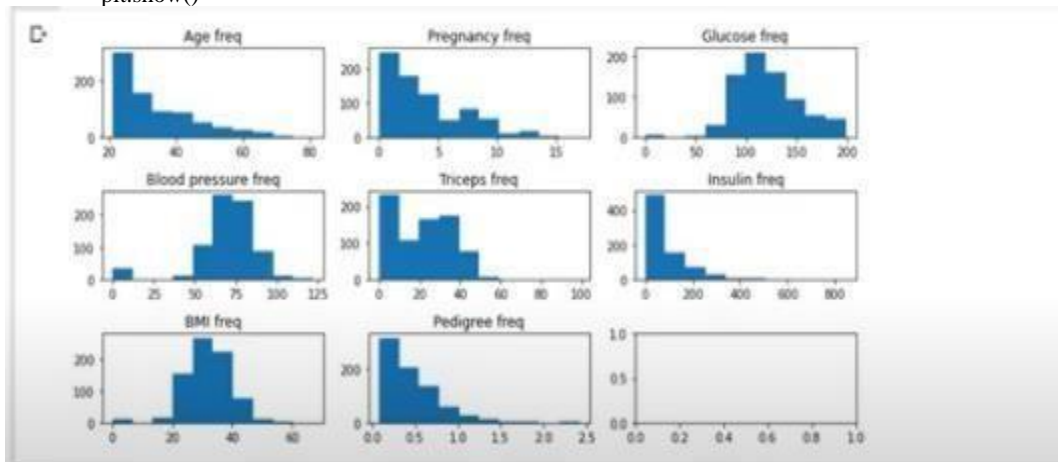
Step 6:- click on "+code", write following code and click on run

```
%matplotlib inline
```

```

import matplotlib.pyplot as plt
#plot histograms to reveal how the data is skewed
fig, axes = plt.subplots( rows=3, ncols=3 )
( ageHist, pregHist, glucoseHist, bldPressHist, triThickHist, insulHist, bmiHist, pedigrHist, placeholder ) =
axes.flatten()
fig.set_size_inches( 10, 5 )
ageHist.hist( diabetes_data[ "Age" ] )
ageHist.set_title( 'Age freq' )
pregHist.hist( diabetes_data[ "Pregnancies" ] )
pregHist.set_title( 'Pregnancy freq' )
glucoseHist.hist( diabetes_data[ "PlasmaGlucose" ] )
glucoseHist.set_title( 'Glucose freq' )
bldPressHist.hist( diabetes_data[ "DiastolicBloodPressure" ] )
bldPressHist.set_title( 'Blood pressure freq' )
triThickHist.hist( diabetes_data[ "TricepsThickness" ] )
triThickHist.set_title( 'Triceps freq' )
insulHist.hist( diabetes_data[ "SerumInsulin" ] )
insulHist.set_title( 'Insulin freq' )
bmiHist.hist( diabetes_data[ "BMI" ] )
bmiHist.set_title( 'BMI freq' )
pedigrHist.hist( diabetes_data[ "DiabetesPedigree" ] )
pedigrHist.set_title( 'Pedigree freq' )
plt.tight_layout()
plt.show()

```



Step 7:- click on "+code", write following code and click on run

```

#data has too many young people
#use log to flatten Age out a bit
import numpy as np
diabetes_data = diabetes_data.assign(log_Age = lambda x: np.log( x[ 'Age' ] ) )
#apply zscore for other features: glucose, blood pressure
#triceps thickness, insulin, BMI
from scipy.stats import zscore
diabetes_data = diabetes_data.assign(zscore_glucose = zscore(diabetes_data[ 'PlasmaGlucose' ] ) )
diabetes_data = diabetes_data.assign(zscore_pressure = zscore(diabetes_data[ 'DiastolicBloodPressure' ] ) )
diabetes_data = diabetes_data.assign(zscore_thick = zscore( diabetes_data[ 'TricepsThickness' ] ) )
diabetes_data = diabetes_data.assign( zscore_insulin = zscore( diabetes_data[ 'SerumInsulin' ] ) )
diabetes_data = diabetes_data.assign( zscore_bmi = zscore( diabetes_data[ 'BMI' ] ) )
#apply min-max for other features: pregnancy, diabetes pedigree
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
minMaxData = pd.DataFrame(scaler.fit_transform(diabetes_data.loc[:, [ 'Pregnancies', 'DiabetesPedigree' ] ] ), columns =
[ 'minMaxPreg', 'minMaxPedigree' ] )
diabetes_data = pd.concat( [ diabetes_data, minMaxData ], axis = 1, join = 'inner' ) diabetes_data.head()

```


Insulin	BMI	DiabetesPedigree	Age	Diabetic	log_Age	zscore_glucose	zscore_pressure	zscore_thick	zscore_insulin	zscore_bmi	minMaxPreg
0	33.6	0.627	50	1	3.912023	0.848324	0.149641	0.907270	-0.692891	0.204013	0.352941
0	26.6	0.351	31	0	3.433987	-1.123396	-0.160546	0.530902	-0.692891	-0.684422	0.058824
0	23.3	0.672	32	1	3.465736	1.943724	-0.263941	-1.288212	-0.692891	-1.103255	0.470588
94	28.1	0.167	21	0	3.044522	-0.998208	-0.160546	0.154533	0.123302	-0.494043	0.058824
168	43.1	2.288	33	1	3.496508	0.504055	-1.504687	0.907270	0.765836	1.409746	0.000000

Step 8:- click on "+code", write following code and click on run

```
#remove unneeded features
diabetes_copy = diabetes_data.copy(deep=True)
del diabetes_copy[ 'Age' ]
del diabetes_copy[ 'PlasmaGlucose' ]
del diabetes_copy[ 'DiastolicBloodPressure' ]
del diabetes_copy[ 'TricepsThickness' ]
del diabetes_copy[ 'SerumInsulin' ]
del diabetes_copy[ 'BMI' ]
del diabetes_copy[ 'DiabetesPedigree' ]
del diabetes_copy[ 'Pregnancies' ]
del diabetes_copy[ 'PatientID' ]
diabetes_copy.head()
```

	Diabetic	log_Age	zscore_glucose	zscore_pressure	zscore_thick	zscore_insulin	zscore_bmi	minMaxPreg	minMaxPedigree
0	1	3.912023	0.848324	0.149641	0.907270	-0.692891	0.204013	0.352941	0.234415
1	0	3.433987	-1.123396	-0.160546	0.530902	-0.692891	-0.684422	0.058824	0.116567
2	1	3.465736	1.943724	-0.263941	-1.288212	-0.692891	-1.103255	0.470588	0.253629
3	0	3.044522	-0.998208	-0.160546	0.154533	0.123302	-0.494043	0.058824	0.038002
4	1	3.496508	0.504055	-1.504687	0.907270	0.765836	1.409746	0.000000	0.943638

Step 9:- click on "+code", write following code (write it in differnt boxes) and click on run

```
#split data into 70% training 30% testing
from sklearn.model_selection import train_test_split
train, test = train_test_split( diabetes_copy, test_size = 0.3 )
#select features to train and test features = ["log_Age","zscore_glucose",
"zscore_pressure","zscore_thick","zscore_insulin","zscore_bmi","minMaxPreg","minMaxPedigree" ]
X_train = train[ features ]
Y_train = train[ "Diabetic" ]
X_test = test[ features ]
Y_test = test[ "Diabetic" ]
```

Step 10:- Click on "+code", write following code and click on run

```
#train a Boosted Decision tree model to predict Diabetic (0 or 1)
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
#this will create 200 weak learner decision trees and use them to build a strong classifier
bdt = AdaBoostClassifier( DecisionTreeClassifier( max_depth = 4 ) ,
algorithm="SAMME", n_estimators=200 )
dt = bdt.fit( X_train, Y_train )
```

Step 11:- Click on "+code", write following code and click on run

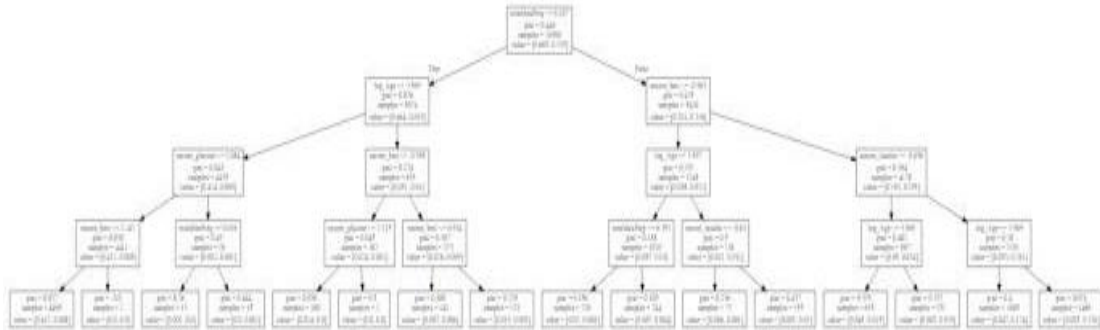
```
from sklearn import tree import graphviz
%matplotlib inline
def show_tree( theTree, features, width, height ):
the_data = tree.export_graphviz( theTree, out_file = None, feature_names = features )
# find first { in string
indexDot = the_data.find( '{' )
# hack to resize graph because kaggle doesn't have a way to change size on graphviz dot string the_data =
the_data[:indexDot + 1 ] + '\n graph [size="%d,%d"]; ' % (width, height) + the_data[ indexDot + 1 : ]
graph = graphviz.Source( the_data )
return graph
# modified function to handle ensemble instead of a single decision tree
def show_tree2( theTree, index, features, path ):
f = io.StringIO()
```

```

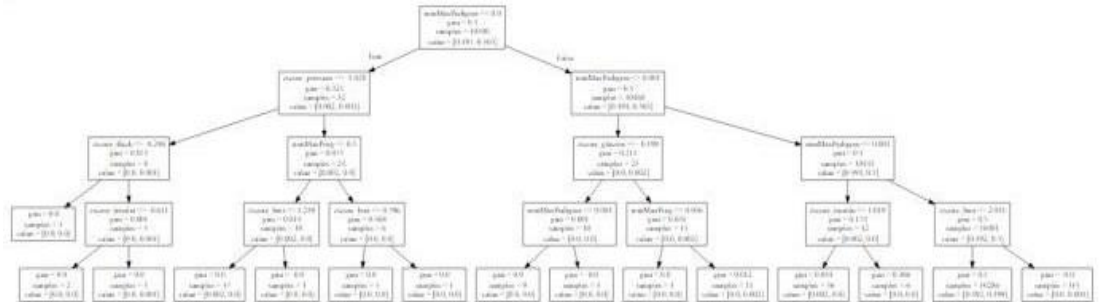
export_graphviz( theTree[ index ], out_file=f, feature_names=features )
pydotplus.graph_from_dot_data(f.getvalue() ).write_png( path )
#img = misc.imread( path )
img = imageio.imread( path )
plt.rcParams["figure.figsize" ] = ( 20, 20 )
plt.imshow( img )

```

Step 12:- Click on "+code", write following code and click on run
 show_tree(dt[0], features, 13, 13)



Step 13:- Click on "+code", write following code and click on run
 show_tree(dt[199], features, 13, 13)



Step 14:- Click on "+code", write following code and Click on run for name, importance in zip(X_train.columns, dt.feature_importances_): print(name, importance)

```

log_Age 0.124571684219427
zscore_glucose 0.1245835695739108
zscore_pressure 0.09221842594216138
zscore_thick 0.11122959738150234
zscore_insulin 0.1439953125487751
zscore_bmi 0.1597462464336025
minMaxPreg 0.12441826712731291
minMaxPedigree 0.11923689677330782

```

Step 15:- Click on "+code", write following code and Click on run

```

#adapted from a function by paultimothymooney
def plot_feature_importances( decTree, trainingFeatures ):
    featureList = trainingFeatures.columns.values
    # sort both arrays by importance but get only the sorted feature names
    featureList = [ x for _, x in sorted( zip( decTree.feature_importances_, featureList ) ) ]    featureSize = len( featureList )
    plt.barh( range( featureSize ), sorted( decTree.feature_importances_ ) )
    plt.ylabel( "Feature" )
    plt.yticks( np.arange( featureSize ), featureList )
    plt.xlabel( "Importance" )
    from sklearn import model_selection
    def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, n_jobs=1, train_sizes =np.linspace(.1, 1.0, 5)):
        """ Plots a learning curve. http://scikit-learn.org/stable/modules/learning\_curve.html """
        plt.figure().set_size_inches( 5, 5 )
        plt.title(title)
        if ylim is not None:
            plt.ylim(*ylim)
            plt.xlabel("Training examples")
            plt.ylabel("Score")

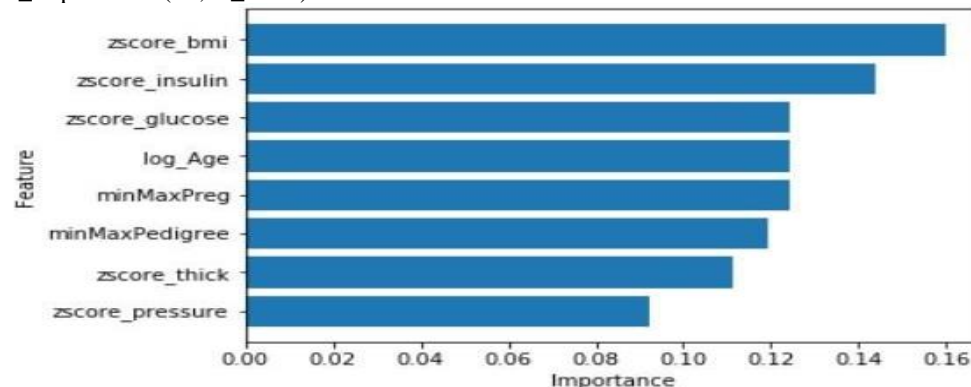
```

```

train_sizes, train_scores, test_scores = model_selection.
learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
plt.grid()
plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std,
alpha=0.1, color="r")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, alpha=0.1,
color="g")
plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score") plt.plot(train_sizes,
test_scores_mean, 'o-', color="g", label="Cross-validation score") plt.legend(loc="best")
return plt
def plot_ROC( falsePositiveRate, truePositiveRate, areaUnderCurve ):
fig = plt.figure()
fig.set_size_inches( 15, 5 )
rocCurve = fig.add_subplot( 1, 2, 1 )
rocCurve.plot( falsePositiveRate, truePositiveRate, color = 'darkgreen', lw = 2, label='ROC curve
(area = %0.2f)' % areaUnderCurve )
rocCurve.plot( [0, 1], [0, 1], color = 'navy', lw = 1, linestyle = '--' )
rocCurve.grid()
plt.xlim( [0.0, 1.0] )
rocCurve.set_xticks( np.arange( -0.1, 1.0, 0.1 ) )
plt.ylim( [0.0, 1.05] )
rocCurve.set_yticks( np.arange( 0, 1.05, 0.1 ) )
plt.xlabel('False Positive Rate' )
plt.ylabel( 'True Positive Rate' )
plt.title('ROC' )
rocCurve.legend( loc = "lower right" )
return plt

```

Step 16:- Click on "+code", write following code and Click on run
plot_feature_importances(dt, X_train)



Step 17:- Click on "+code", write following code and Click on run

```

Y_pred = dt.predict( X_test )
Y_probab = dt.predict_proba( X_test )

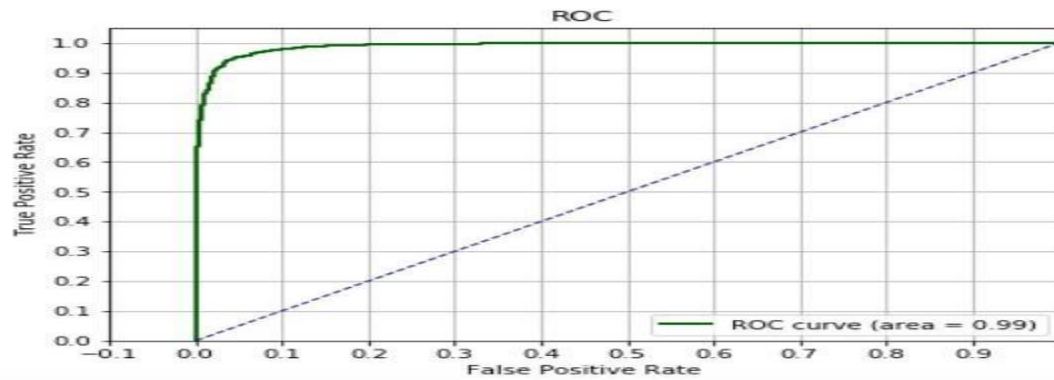
```

Step 18:- Click on "+code", write following code and Click on run

```

# see evaluation metrics
import numpy as np
from sklearn import metrics
# Receiver Operating Characteristic - precision/recall lift
# Y_probab[:,1] is the array of probabilities for Diabetic = 1 at the leaf level
fpr, tpr, thresholds = metrics.roc_curve( Y_test.values, Y_probab[:,1] )
auc = metrics.auc( fpr, tpr )
# plot ROC curve plot_ROC( fpr, tpr, auc )
plt.show()
# alternative using scikitplot: conda install -c conda-forge scikit-plot
#import scikitplot as skplt
#skplt.metrics.plot_roc_curve( Y_test, Y_probab )

```



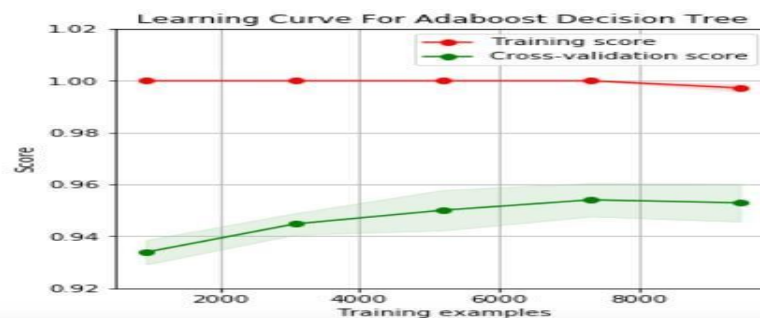
Step 19:- Click on "+code" , write following code and Click on run

```
# score the model on the test data
def scoreModel( Y_test, Y_pred ):
# show accuracy, precision and recall
from sklearn.metrics import accuracy_score
score = accuracy_score( Y_test, Y_pred )
print( "Accuracy: %.3f " % round( score, 3 ) )
from sklearn.metrics import precision_score
precScore = precision_score( Y_test, Y_pred, average = 'binary' )
print( "Precision: %.3f " % round( precScore, 3 ) )
from sklearn.metrics import recall_score
recScore = recall_score( Y_test, Y_pred, average = 'binary' )
print( "Recall: %.3f " % round( recScore, 3 ) )
# confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix( Y_test, Y_pred )
print( "True positives: %d False negatives: %d" % ( cm[ 1, 1 ], cm[ 1, 0 ] ) )
print( "False positives: %d True negatives: %d" % ( cm[ 0, 1 ], cm[ 0, 0 ] ) )
# AUC (area under the curve)
auc = metrics.auc( fpr, tpr )
print( "AUC: %.3f" % auc ) scoreModel( Y_test, Y_pred )
```

```
Accuracy: 0.954
Precision: 0.933
Recall: 0.928
True positives: 1376 False negatives: 186
False positives: 99 True negatives: 2919
AUC: 0.990
```

Step 20:- Click on "+code" , write following code and Click on run

```
# this is to prevent 100s of DeprecationWarnings for something that is scheduled to
# be fixed on the scikit-learn release of August 2018
from sklearn import preprocessing import warnings
warnings.filterwarnings(action='ignore', category=DeprecationWarning)
the LearningCurve = plot_learning_curve( dt, 'Learning Curve For Adaboost Decision Tree', X_train, Y_train, ( 0.92,
1.02 ), 10 )
theLearningCurve.figure().set_size_inches( 5, 5 )
theLearningCurve.show()
```



```
<matplotlib.figure.Figure at 0x7f68454acd30>
```

Step 21:- Click on "+code" , write following code and Click on run

```
#copied score of Light GBM using dart for comparison:
```

```
#Accuracy: 0.945
#Precision: 0.918 <- lower (less of the reported positives are true positives, that is, more false positives)
#Recall: 0.913 <- lower (less true positives are reported as positives, that is, more false negatives)
#True positives: 1339 False negatives: 128 #False positives: 119 True negatives: 2914
#AUC: 0.990
# score of LightGBM using gbdt:
#Accuracy: 0.946
#Precision: 0.918
#Recall: 0.915
#True positives: 1342 False negatives: 125 #False positives: 120 True negatives: 2913
#AUC: 0.990
```

Step 22:- Click on "+code" , write following code and Click on run

```
#TODO: find a MART gradient boosting library
#install LightGBM:
#run brew install cmake
#brew install gcc
#(wait about 50 minutes for the above to complete)
#git clone --recursive https://github.com/Microsoft/LightGBM ; cd LightGBM
#export CXX=g++-7 CC=gcc-7
#mkdir build ; cd build #cmake ..
#make -j4
#line below fails: No module named 'lightgbm' unless fixed with:
#conda config --add channels conda-forge
#conda install lightgbm
import lightgbm as lgb #create dataset for LightGBM
lgb_train = lgb.Dataset( X_train, Y_train )
lgb_eval = lgb.Dataset( X_test, Y_test, reference = lgb_train )
#configuration: http://lightgbm.readthedocs.io/en/latest/Parameters.html
params = {
    'task': 'train',
    'boosting_type': 'dart',
    # gbdt = Gradient Boosting Decision Tree
    #dart = Dropouts meet Multiple Additive Regression Trees
    'objective': 'binary',
    'metric': { 'auc', 'root_mean_squared_error' },
    'max_depth': 4,
    'learning_rate': .5,
    'feature_fraction': 0.9,
    'bagging_fraction': 1,
    'bagging_freq': 0, # no bagging
    'verbose': 0,
    'min_data_in_leaf': 20, # adjust this to handle overfitting
    'num_machines': 1, # can do parallel processing via network
    'device': 'cpu', # can use GPUs via OpenCL
    'gpu_platform_id': -1,
    'gpu_device_id': -1 }
# http://lightgbm.readthedocs.io/en/latest/Python-API.html
gbm = lgb.train( params, lgb_train, num_boost_round = 200, valid_sets = lgb_eval, early_stopping_rounds = 5 )
Y_pred = gbm.predict( X_test, num_iteration = gbm.best_iteration )
#for name, importance in zip( X_train.columns, gbm.feature_importances_ ):
#print( name, importance )
#from sklearn.metrics import mean_squared_error
#print( 'The rmse of prediction is:', mean_squared_error( Y_test, Y_pred ) ** 0.5 )
```



```
[1] valid_0's auc: 0.939829 valid_0's rmse: 0.363242
Training until validation scores don't improve for 5 rounds.
[2] valid_0's auc: 0.953096 valid_0's rmse: 0.309809
[3] valid_0's auc: 0.960434 valid_0's rmse: 0.28454
[4] valid_0's auc: 0.968318 valid_0's rmse: 0.262193
[5] valid_0's auc: 0.971413 valid_0's rmse: 0.252086
[6] valid_0's auc: 0.975202 valid_0's rmse: 0.241616
[7] valid_0's auc: 0.978691 valid_0's rmse: 0.232582
[8] valid_0's auc: 0.979392 valid_0's rmse: 0.232901
[9] valid_0's auc: 0.980354 valid_0's rmse: 0.226956
[10] valid_0's auc: 0.982054 valid_0's rmse: 0.222244
[11] valid_0's auc: 0.982796 valid_0's rmse: 0.218347
[12] valid_0's auc: 0.983039 valid_0's rmse: 0.219096
[13] valid_0's auc: 0.9841 valid_0's rmse: 0.215327
[14] valid_0's auc: 0.985632 valid_0's rmse: 0.209051
[15] valid_0's auc: 0.986824 valid_0's rmse: 0.204967
```

Step 23:- Click on "+code" , write following code and Click on run scoreModel(Y_test, Y_pred.round())


```

Accuracy: 0.953
Precision: 0.933
Recall: 0.924
True positives: 1370 False negatives: 112
False positives: 99 True negatives: 2919
AUC: 0.990

```

Step 24:- Click on "+code", write following code and Click on run

```

# another way of using LightGBM
gbm = lgb.LGBMClassifier(boosting_type = 'gbdt', max_depth = 4, learning_rate = 0.5, n_estimators = 200, objective =
'binary', min_child_samples = 20 )
dt = gbm.fit( X_train, Y_train )
Y_pred = dt.predict( X_test )
Y_probas = gbm.predict_proba( X_test )

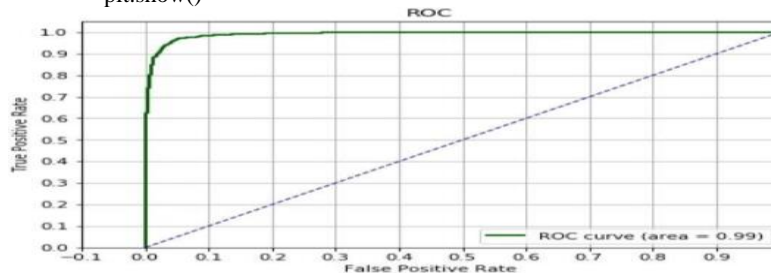
```

Step 25:- Click on "+code", write following code and Click on run

```

# Receiver Operating Characteristic - precision/recall lift
# Y_probas[:,1] is the array of probabilities for Diabetic = 1 at the leaf level fpr, tpr,
thresholds = metrics.roc_curve( Y_test, Y_probas[:,1] )
auc = metrics.auc( fpr, tpr )
# plot ROC curve
plot_ROC( fpr, tpr, auc )
plt.show()

```



Step 26:- Click on "+code", write following code and Click on run `scoreModel(Y_test, Y_pred.round())`

```

Accuracy: 0.958
Precision: 0.939
Recall: 0.933
True positives: 1383 False negatives: 99
False positives: 90 True negatives: 2928
AUC: 0.991

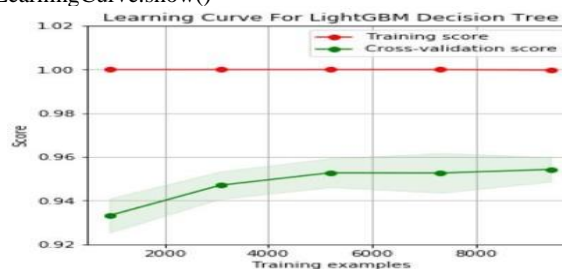
```

Step 27:- Click on "+code", write following code and Click on run

```

# this is to prevent 100s of DeprecationWarnings for something that is scheduled to
# be fixed on the scikit-learn release of August 2018
from sklearn import preprocessing import warnings
warnings.filterwarnings(action='ignore', category=DeprecationWarning)
theLearningCurve = plot_learning_curve( dt, 'Learning Curve For LightGBM Decision Tree', X_train, Y_train, ( 0.92, 1.02
), 10 )
theLearningCurve.figure().set_size_inches( 5, 5 )
theLearningCurve.show()

```



```
<matplotlib.figure.Figure at 0x7f68a5d88390>
```

NOTE: If you want to execute the the program on idle then first you need to install following packages

```

pip install pandas
pip install -U scikit-learn
pip install graphviz
pip install pydotplus

```

Practical No: 7

Aim: Implement SVM Classification Technique.

Support Vector Machine (SVM):

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space that is it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

Diagram: -



Step 1:- We need to firstly install Pandas and Scikit-Learn Library Using Command in Python IDLE.

- Pandas cmd=pip install pandas.
- Scikit Learn cmd =pip install -U Scikit – Learn.

Step 2:- Import a datasets

```

Python Shell 385
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcb0, May 3 2021, 17:27:52) [MSC v.1
928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inform
ation.
>>>
= RESTART: C:/Users/prati/OneDrive/Desktop/raveena maam/practical
8a/practical8.py
>>>
= RESTART: C:/Users/prati/OneDrive/Desktop/raveena maam/practical
8a/practical8.py
>>>
= RESTART: C:/Users/prati/OneDrive/Desktop/raveena maam/practical
8a/practical8.py
>>>
features: ['mean radius' 'mean texture' 'mean perimeter' 'mean ar
ea'
'mean smoothness' 'mean compactness' 'mean concavity'
'mean concave points' 'mean symmetry' 'mean fractal dimension'
'radius error' 'texture error' 'perimeter error' 'area error'
'smoothness error' 'compactness error' 'concavity error'
'concave points error' 'symmetry error' 'fractal dimension error'
'worst radius' 'worst texture' 'worst perimeter' 'worst area'
'worst smoothness' 'worst compactness' 'worst concavity'
'worst concave points' 'worst symmetry' 'worst fractal dimension'
]
Labels: ['malignant' 'benign']
>>>
  
```

Step 3:- Data.Shape

```

practical8.py - C:/Users/prati/OneDrive/Desktop/raveena maam/practical 8a/practical8.py (385)
File Edit Format Run Options Window Help
from sklearn import datasets
cancer = datasets.load_breast_cancer()
print("Features: ", cancer.feature_names)
print("Labels: ", cancer.target_names)
cancer.data.shape
print(cancer.data[0:5])

practical8.py (385)
File Edit Shell Debug Options Window Help
= RESTART: C:/Users/prati/OneDrive/Desktop/raveena maam/practical
8a/practical8.py
>>>
= RESTART: C:/Users/prati/OneDrive/Desktop/raveena maam/practical
8a/practical8.py
>>>
= RESTART: C:/Users/prati/OneDrive/Desktop/raveena maam/practical
8a/practical8.py
Features: ['mean radius' 'mean texture' 'mean perimeter' 'mean ar
ea'
'mean smoothness' 'mean compactness' 'mean concavity'
'mean concave points' 'mean symmetry' 'mean fractal dimension'
'radius error' 'texture error' 'perimeter error' 'area error'
'smoothness error' 'compactness error' 'concavity error'
'concave points error' 'symmetry error' 'fractal dimension error'
'worst radius' 'worst texture' 'worst perimeter' 'worst area'
'worst smoothness' 'worst compactness' 'worst concavity'
'worst concave points' 'worst symmetry' 'worst fractal dimension'
]
Labels: ['malignant' 'benign']
>>>
= RESTART: C:/Users/prati/OneDrive/Desktop/raveena maam/practical
8a/practical8.py
Features: ['mean radius' 'mean texture' 'mean perimeter' 'mean ar
ea'
'mean smoothness' 'mean compactness' 'mean concavity'
'mean concave points' 'mean symmetry' 'mean fractal dimension'
'radius error' 'texture error' 'perimeter error' 'area error'
'smoothness error' 'compactness error' 'concavity error'
'concave points error' 'symmetry error' 'fractal dimension error'
'worst radius' 'worst texture' 'worst perimeter' 'worst area'
'worst smoothness' 'worst compactness' 'worst concavity'
'worst concave points' 'worst symmetry' 'worst fractal dimension'
]
Labels: ['malignant' 'benign']
>>>
  
```

Step 4:- Train_Test_Split()

Split arrays or matrices into random train and test subsets. Quick utility that wraps input validation and next (ShuffleSplit().split(X, y)) and application to input data

Step 5:- Importing SVM

Step 6:- Import Metrics from sklearn.

The `sklearn.metrics` module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

[illegible]

Step 7:-Calling Accuracy, Precision, Recall.

Accuracy: Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations

Recall: Recall is the ratio of correctly predicted positive observations to the all observations in actualclass.

[illegible]

Practical No: 8**Aim: Import the data and implement regression model.****Description:**

Step 1:- We need to import the following libraries Pandas, matplotlib, seaborn, scatter matrix.

Pandas: Used for manipulating numerical table and time series in python.

Matplotlib: It is used for data visualization and graph plotting in python.

Scatter(): Scatter plots are used to determine relationships between the datasets.

Scatter matrix: A scatter matrix consists of several pair-wise scatter plots of variables presented in a matrix format.

Seaborn: Seaborn is a high level data visualization tool used along with matplotlib. Histogram is a visualization output which can be produced using Seaborn library.

Step 2:- After visualization using the seaborn library we will box plot for each numeric variable.

Step 3:- Using pylab library we will produce a histogram for each numeric input variable.

Pylab:- Pylab is a module in matplotlib library it works as a plotting library.

Step 4:- Next step is to produce the scatter matrix for each input variable.

Step 5:- Implement logistic regression on the dataset. Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable.

Step 6:- The accuracy of the training set will be produced through logistic regression.

Code: import pandas as pd

import matplotlib.pyplot as plt

fruits = pd.read_table('fruit_data_with_colors.txt')

fruits.head()

print(fruits.shape)

print(fruits['fruit_name'].unique())

print(fruits.groupby('fruit_name').size())

import seaborn as sns

sns.countplot(fruits['fruit_name'],label="Count")

plt.show()

fruits.drop('fruit_label', axis=1).plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False, figsize=(9,9),title='Box Plot for each input variable')

plt.savefig('fruits_box')

plt.show()

from pandas.tools.plotting import scatter_matrix

from matplotlib import cm

feature_names = ['mass', 'width', 'height','color_score']

X = fruits[feature_names]

y = fruits['fruit_label']

cmap = cm.get_cmap('gnuplot')

scatter =pd.scatter_matrix(X,c = y,marker = 'o',s=40,hist_kwds={'bins':15},figsize=(9,9),cmap = cmap)

plt.suptitle('Scatter-matrix for each input variable')

plt.savefig('fruits_scatter_matrix')

from sklearn.linear_model import LogisticRegression

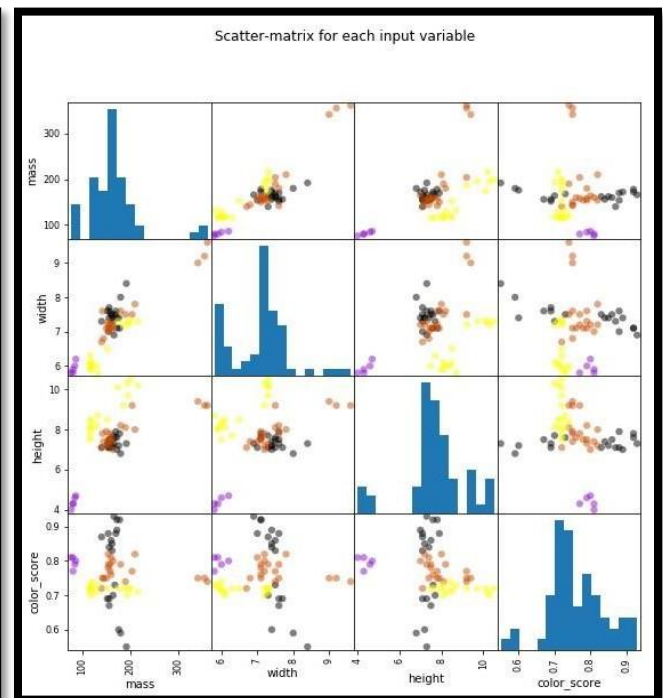
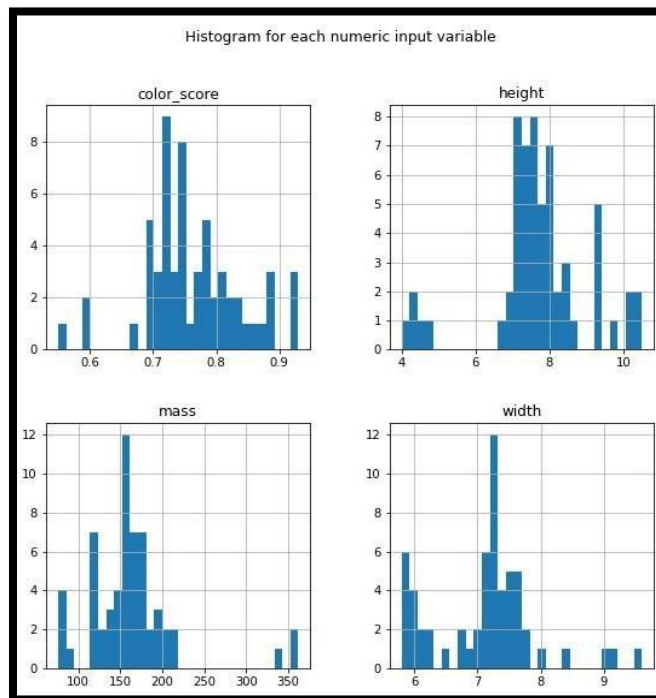
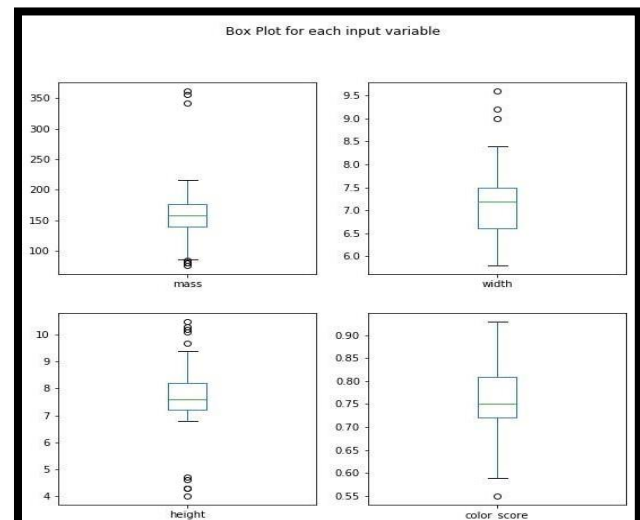
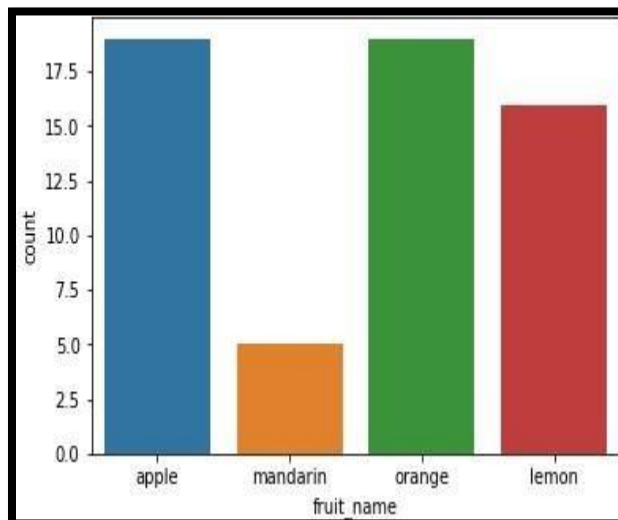
logreg = LogisticRegression()

logreg.fit(X_train, y_train)

print('Accuracy of Logistic regression classifier on training set: {:.2f}'.format(logreg.score(X_train, y_train)))

print('Accuracy of Logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))

Output:-



Accuracy of Logistic regression classifier on training set: 0.70

Accuracy of Logistic regression classifier on test set: 0.40

Accuracy of Logistic regression classifier on test set: 0.40

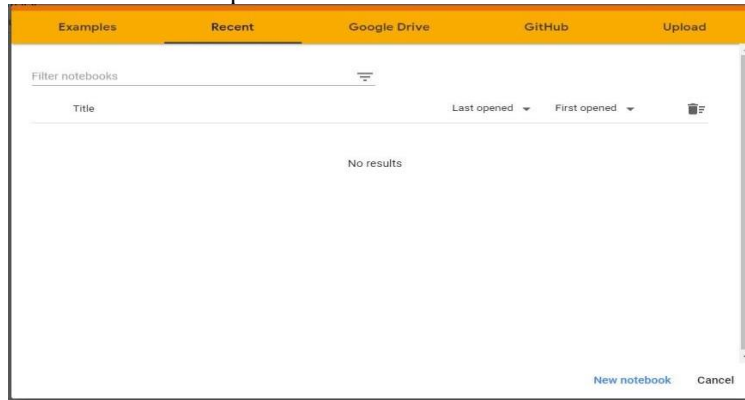
Practical No: 9

Aim: -MULTIPLE REGRESSION Model Apply multiple regression,if data have a continuous independent variable. Apply on above dataset.

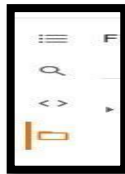
Multiple Regression is a set of techniques that describes-line relationships between two or more independent variables or predictor variables and one dependent or criterion variable.

Step 1:- If you have not install the python, Use online python virtual machine visit the website Google Colaboratory.

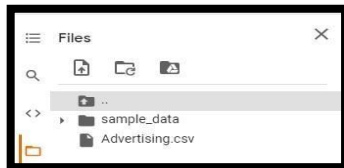
Step 2:- There will be a window open to create a new notebook click on it.



Step 3:- Click on the Option menu select third symbol that is file like structure.



Step 4:- Right click on there select upload and Select the Dataset from the drive.



Step 5:- Import all the library that are used for programme

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Step 6:- Code to Read the Data file

```
[4] dataset = pd.read_csv("Advertising.csv")
```

read_csv is used for reading the excel file where Advertising.csv is an excel file.

Step 7:- Code to read the head of the file

```
dataset.head()
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

head() is used to read the first 5 row of the data file.

Step 8:- Initialize all the dataset with a variable x, y

```
x = dataset[['TV', 'Radio', 'Newspaper']]
y = dataset['Sales']
```

Step 9 :- Importing train_test_split to split the data that we have initialize in variables

```
[8] from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 100)
```

sklearn.model_selection :- Utility that wraps input validation and next (ShuffleSplit(), Split(X,y)) and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

train_test_split :- Is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

test_size :- This parameter decides the size of the **data** that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset. If you're specifying this parameter, you can ignore the next parameter.

random_state:- Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls.

Step 10:- Importing LinearRegression

```
[9] from sklearn.linear_model import LinearRegression
slr = LinearRegression()
slr.fit(x_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

The class **sklearn.linear_model.LinearRegression** will be used to perform linear and polynomial regression and make predictions accordingly.

.fit(), you calculate the optimal values of the weights w_0 and w_1 , using the existing input and output (x and y) as the arguments. In other words, **.fit()** fits the model. It returns self, which is the variable model itself. That's why you can replace the last two statements with this one:

Step 11:- Printing Intercepts, & Coefficients

```
[10] print("Intercept: ",slr.intercept_)
print("Coefficients")
list(zip(x,slr.coef_))

Intercept: 2.652789668879496
Coefficients
[('TV', 0.04542559602399794),
 ('Radio', 0.18975772766893614),
 ('Newspaper', 0.004603078953112072)]
```

Intercept and Coefficients A linear regression line has the equation $Y = mx + c$, where m is the coefficient of independent variable and c is the intercept.

Step 12:- Predict the value

```
[11] y_pred_slr = slr.predict(x_test)
```

.predict() :- Given a trained model. Predict the label of a new set of data. This method accepts one argument. The new data X_{new} and returns the learned label for each object in array.

Step 13:- Printing the predict value

```
[12] print("Prediction for test set:{}".format(y_pred_slr))

Prediction for test set:[10.62160072 20.00625302 16.91850882 19.17040746 20.94974131 13.12284284
11.80740696 12.32019766 20.57806782 20.95662688 10.79096475 19.54868702
6.42403866 15.23133391 8.97226257 7.89897862 16.23599497 12.02636477
17.09702178 11.26080277 16.97826292 9.75655721 20.82389762 17.20916742
15.13816239 21.97290698 19.20181841 10.07501899 19.39017185 14.8673761
14.36798893 7.55604543 9.96742165 14.76342565 7.20995576 13.60003295
7.49088656 11.70865932 13.46091883 15.2229793 17.18088277 13.56738329
14.30942267 13.72909849 11.88559349 8.77039705 12.1244102 19.20252289
9.08376601 5.15367352 16.22852749 18.14111213 12.94835466 16.86274503
17.86462435 12.33930625 4.3575739 11.25904494 16.11560622 13.56602169]
```

Step 14:- Calculated Actual value and predict value

```
[13] slr_diff = pd.DataFrame({'Actual value':y_test,'Predicted value':y_pred_slr})
slr_diff.head()
```

	Actual value	Predicted value
126	6.6	10.621601
104	20.7	20.006253
99	17.2	16.918509
92	19.4	19.170407
111	21.8	20.949741

DataFrame is 2-D size-mutable, potentially heterogeneous tabular data structure with labeled axes in rows and column.

Step 15:- Calculating Mean Absolute Error, Mean Square Error, Root Mean Square Error

```
from sklearn import metrics
meanAbsErr = metrics.mean_absolute_error(y_test,y_pred_slr)
meanSqErr = metrics.mean_squared_error(y_test,y_pred_slr)
rootMeanSqErr = np.sqrt(metrics.mean_squared_error(y_test,y_pred_slr))
print('R squared:[.2f]'.format(slr.score))
print('Mean Absolute Error:',meanAbsErr)
print('Mean Square Error:',meanSqErr)
print('Root mean Square Error:',rootMeanSqErr)

R squared:[.2f]
Mean Absolute Error: 1.0638483124072033
Mean Square Error: 1.8506819941636972
Root mean Square Error: 1.3603977338130555
```

Practical No: 10

Aim: Classification using Decision Tree classifier and Naïve bayes classifier.

Description:

Step1: For classification in python we need to import the following libraries

Pandas: Used for manipulating numerical table and time series in python.

Matplotlib: It is used for data visualization and graph plotting in python.

Scatter (): Scatter plots are used to determine relationships between the datasets.

GaussianNB: It is a supervised classification learning algorithm which supports continuous data.

DecisionTreeClassifier: It is a library which falls under the category of supervised learning algorithm and it works for continuous as well as categorical dataset.

KNeighborsClassifier: It will look for 5 nearest neighbour in a dataset.

Step2: Read_table() is used through importing pandas this function is used to read the csv file which is in table format.

Step3: Now import the Decision Tree Classifier. The fit() method takes the training data as arguments, which can be one array in the case of unsupervised learning, or two arrays in the case of supervised learning.

Step4: Similarly import Kneighbors and naïve bayes on the dataset for classification.

Step5: Print the accuracy of DecisionTreeClassifier. Similarly print accuracy of KNeighborsClassifier and naïve bayesclassifier.

Step6: Here plt() is a mathematical function used in graph which helps in visualization.

```
Code:
import pandas as pd
import matplotlib.pyplot as plt
fruits = pd.read_table('fruit_data_with_colors.txt')
fruits.head()

from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train,y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'.format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'.format(clf.score(X_test, y_test)))

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier() knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'.format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'.format(knn.score(X_test, y_test)))

from sklearn.naive_bayes import GaussianNB
Bgnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'.format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'.format(gnb.score(X_test, y_test)))

k_range = range(1, 20)
scores = []
for k in k_range:knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    scores.append(knn.score(X_test, y_test))

plt.figure()
plt.xlabel('k')
plt.ylabel('accuracy')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20])
```

Output:-



Aim: K-means clustering using sklearn in python.

Step 1: For k-means clustering in python we need to import the following libraries NumPy, Pandas, Matplotlib and from sklearn.cluster we will import k-means.

Numpy: Used for mathematical operations in python.

Pandas: Used for manipulating numerical table and time series in python.

Matplotlib: It is used for data visualization and graph plotting in python.

Step2: To read the dataset in python we use the read_csv() method and we will store it in dataframe.

Step3: Now we select columns in dataset and provide them column index for this we use iloc() function on dataframe.

Step 4: Now we will use the Elbow method to find the optimal number of clusters in a dataset. We will plot the graph between “Number of clusters” and “WCSS”.

Step 5: After finding the optimal value we will use k-means for data visualization with matplotlib library using the “scatter()” function.

Scatter (): Used to draw a scatter plot and used to represent the relationship among the variables. This will help us in visualising the clusters.

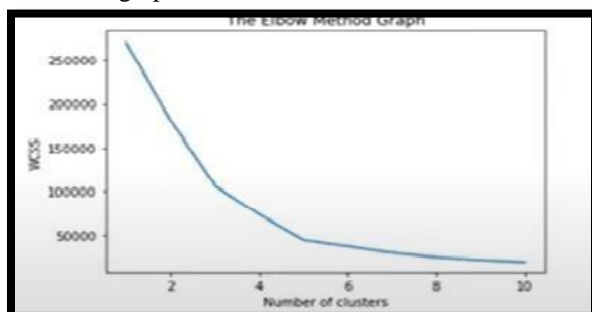
Step6: Now we will plot the centres of each cluster using the keyword “kmeans.cluster_centers”.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset= pd.read_csv('Mall_Customers.csv')
X=dataset.iloc[:, [3,4]].values
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method Graph')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300, n_init=10, random_state=0)
y_kmeans = kmeans.fit_predict(X)
plt.scatter(X[y_kmeans==0, 0], X[y_kmeans==0, 1], s=100, c='red', label='Cluster 1')
plt.scatter(X[y_kmeans==1, 0], X[y_kmeans==1, 1], s=100, c='blue', label='Cluster 2')
plt.scatter(X[y_kmeans==2, 0], X[y_kmeans==2, 1], s=100, c='green', label='Cluster 3')
plt.scatter(X[y_kmeans==3, 0], X[y_kmeans==3, 1], s=100, c='cyan', label='Cluster 4')
plt.scatter(X[y_kmeans==4, 0], X[y_kmeans==4, 1], s=100, c='magenta', label='Cluster 5')
plt.scatter(kmeans.cluster_centers_[0, 0], kmeans.cluster_centers_[0, 1], s=300, c='yellow', label='Centroids')
plt.title('Clusters of Customers')
plt.xlabel('Annual ncome(k$)')
plt.ylabel('Spending Score(1-100)')
plt.show()
```

Output:

The elbow graph:



Kmeans clustering output:

