

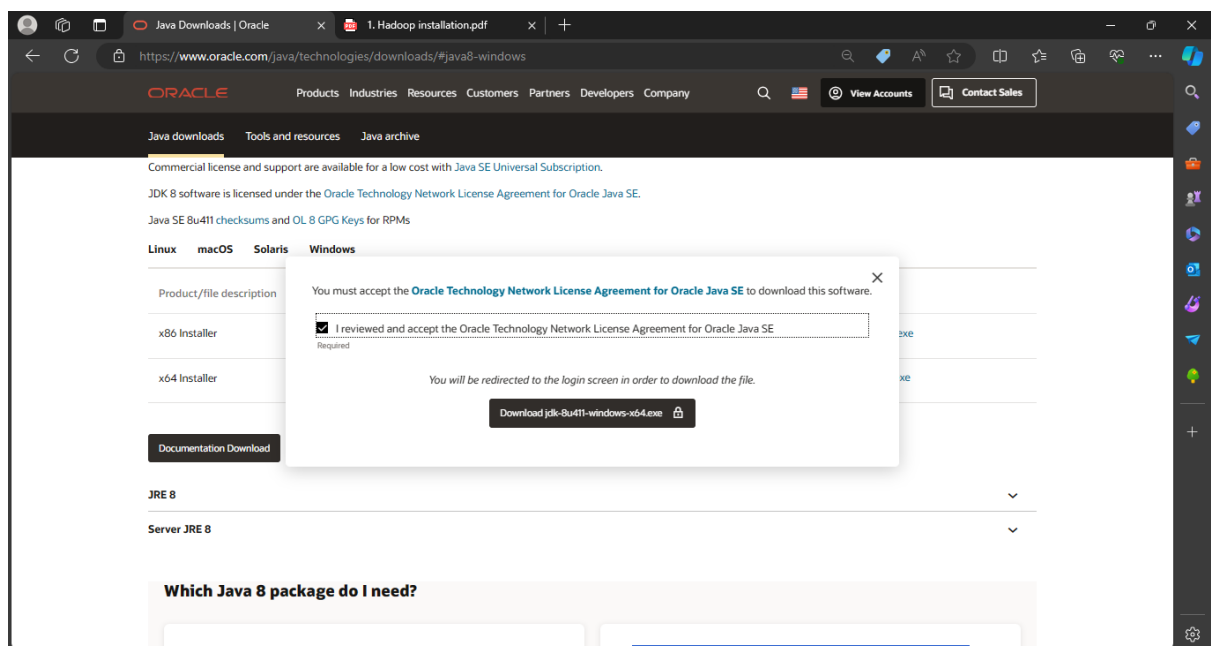
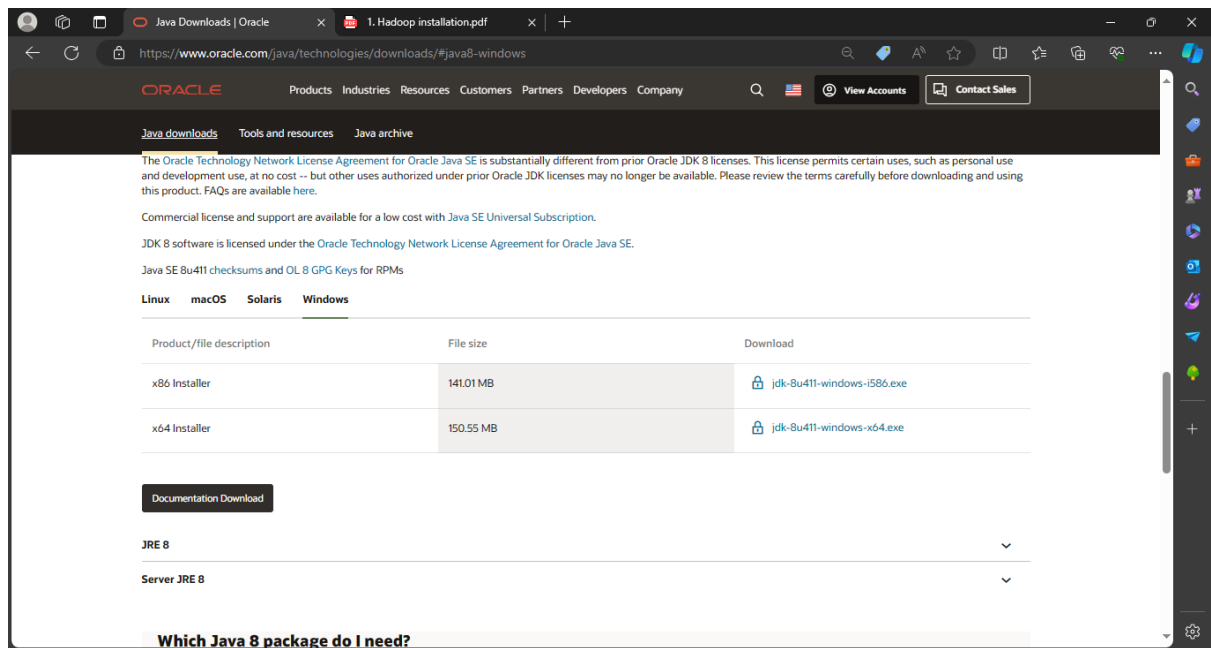
## Practical 1

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

### Step 1: Download Java from oracle website

Search for java SE development kit 8 download

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



## Oracle registration

Provide your email address and password if already registered, else register.

Jdk is downloaded

## Step 2: Download Hadoop for the local system

<https://hadoop.apache.org/releases.html>

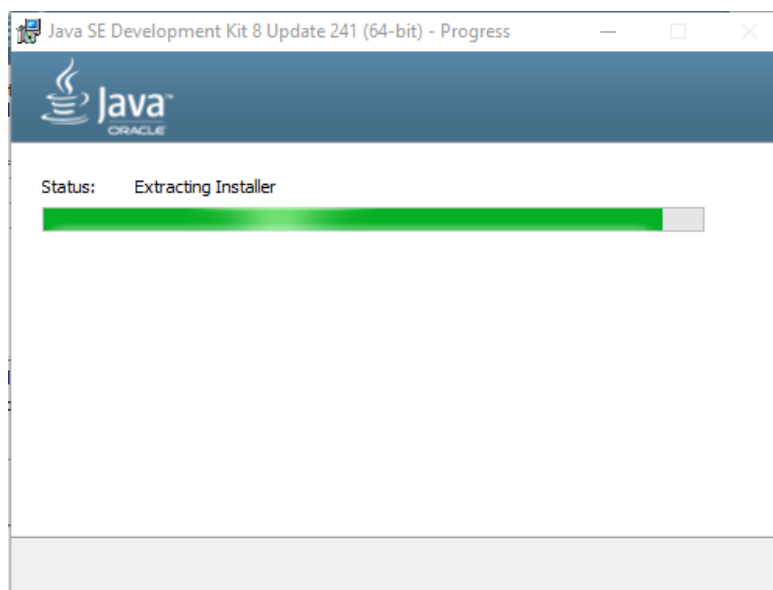
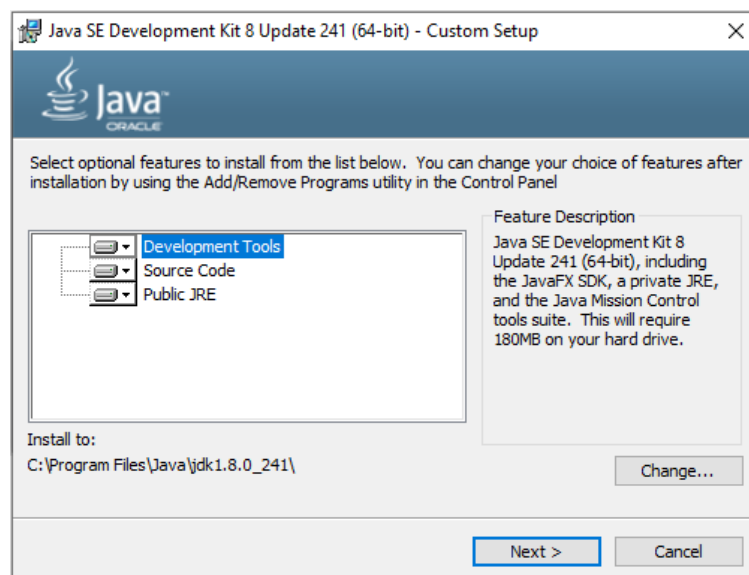
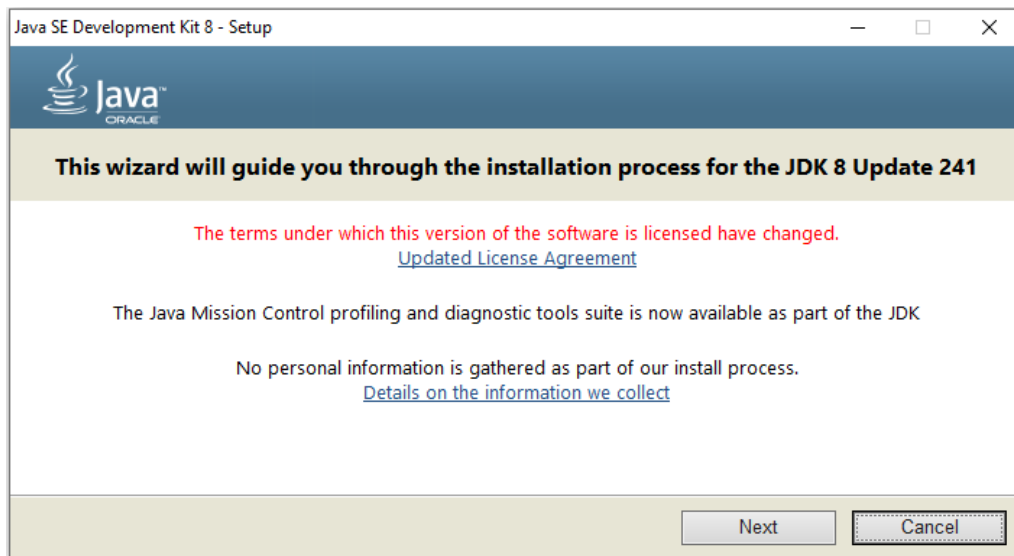
The screenshot shows the Apache Hadoop website's release page for version 3.1.3. The page title is "Release 3.1.3 available". The text states: "This is the third stable release of Apache Hadoop 3.1 line. It contains 246 bug fixes, improvements and enhancements since 3.1.2." It also mentions that users are encouraged to read the "overview of major changes" since 3.1.2 and to check "release notes and changelog" for details of bug fixes, improvements, and other enhancements since the previous 3.1.2 release. The date "2019 Oct 21" is displayed. On the right side, there are four buttons: "Download tar.gz" (green), "(checksum signature)" (blue), "Download src" (orange), and "(checksum signature)" (blue). Below these buttons is a "Documentation" button (blue). At the bottom of the page, there is a footer with the Apache Software Foundation logo and text: "Apache Hadoop, Hadoop, Apache, the Apache feather logo, and the Apache Hadoop project logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and other countries. Copyright © 2006-2024 The Apache Software Foundation. Privacy policy".

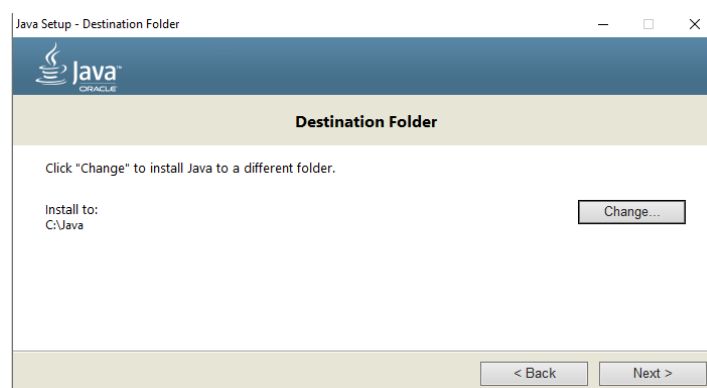
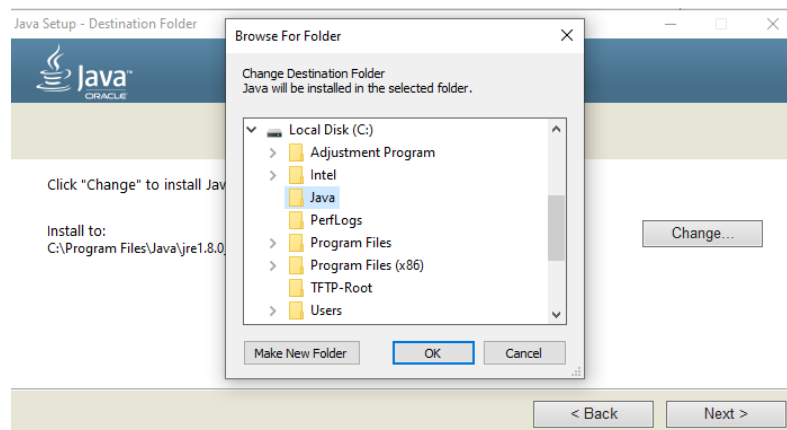
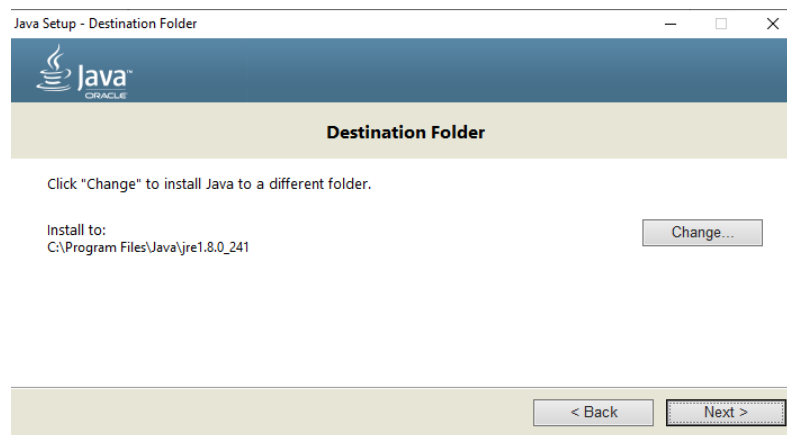
Hadoop downloaded

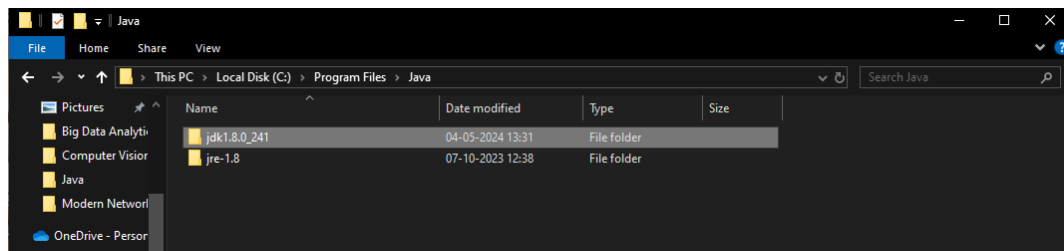
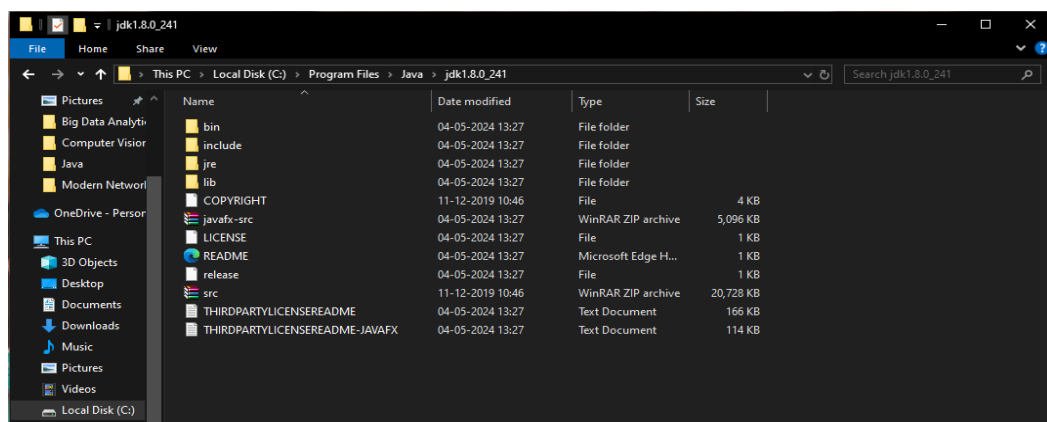
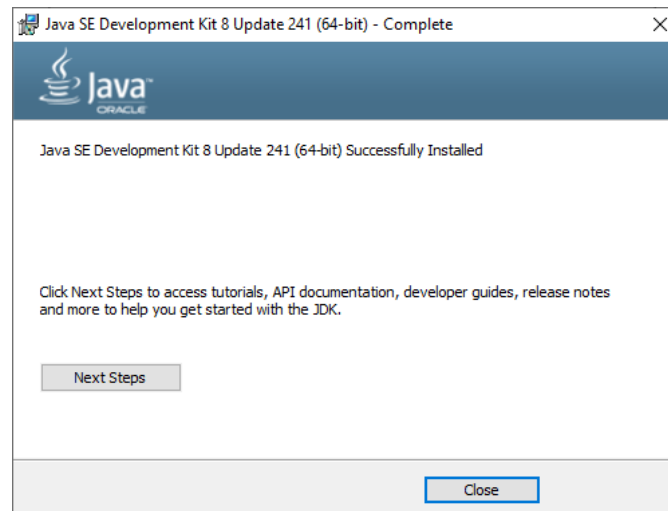
Move files to c drive

The screenshot shows a Windows File Explorer window titled "Local Disk (C:)". The left sidebar shows the "This PC" view with "Local Disk (C:)" selected. The main pane displays a list of files and folders. The list includes folders like "Adjustment Program", "data", "hadoop", "HadoopConfiguration-FIXbin", "Intel", "Java", "PerfLogs", "Program Files", "Program Files (x86)", "TFTP-Root", "tmp", "Users", "VKHCG", "Windows", and files like ".cmd", "clnunit", "hadoop-3.1.3.tar", and "HadoopConfiguration-FIXbin". The "hadoop-3.1.3.tar" file is highlighted, showing its size as 3,30,153 KB. The "HadoopConfiguration-FIXbin" file is also visible, showing its size as 876 KB.

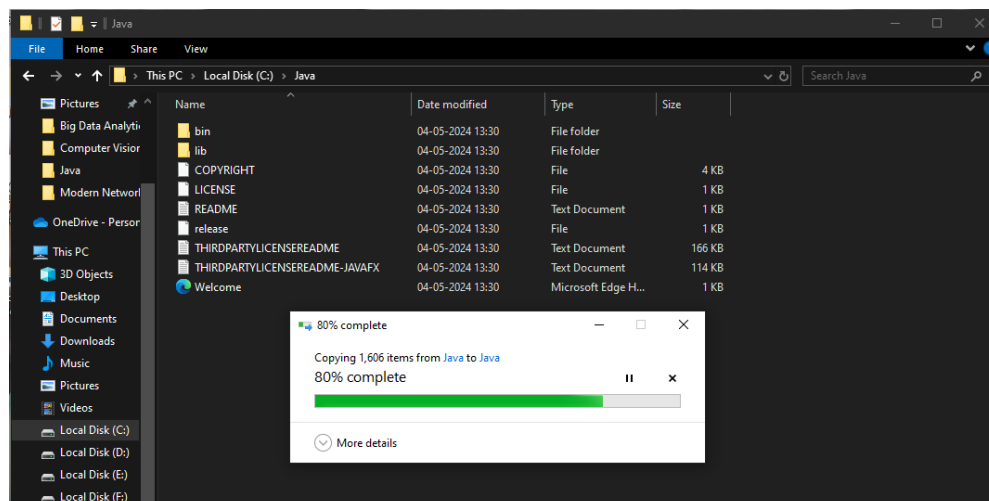
### Step 3: Install Java

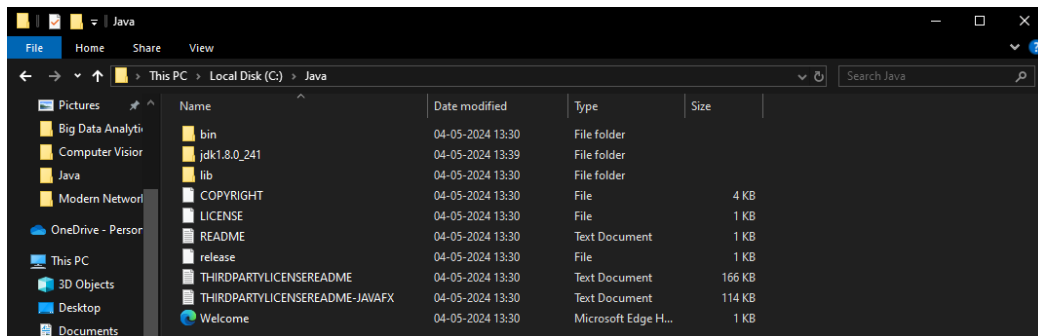






Paste it here.



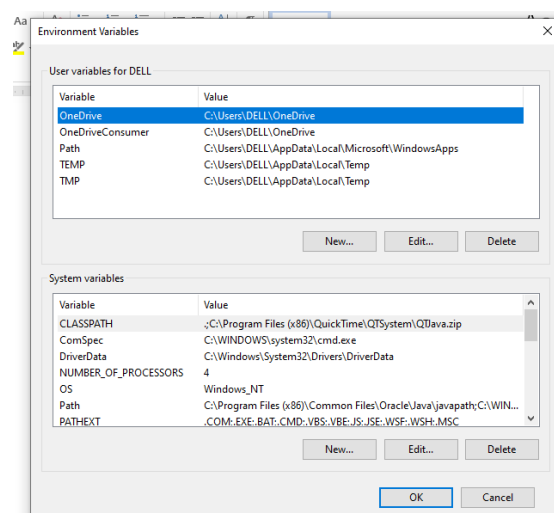
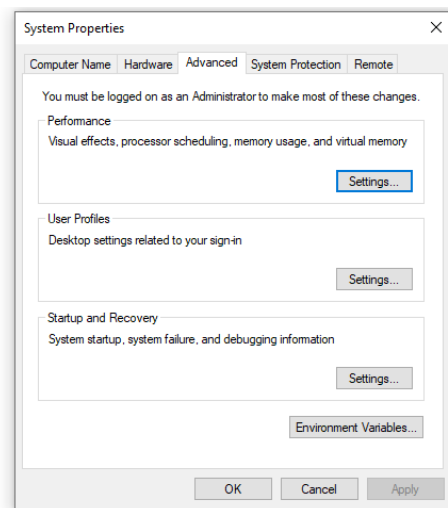


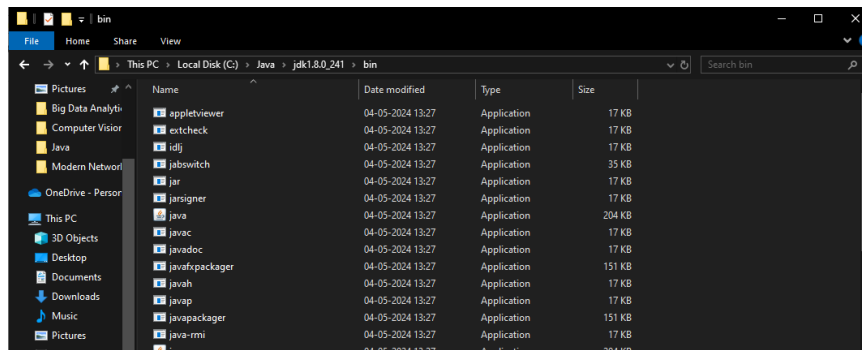
Now java will be available at:

C:\Java

#### Step 4: Setting environment variables for java

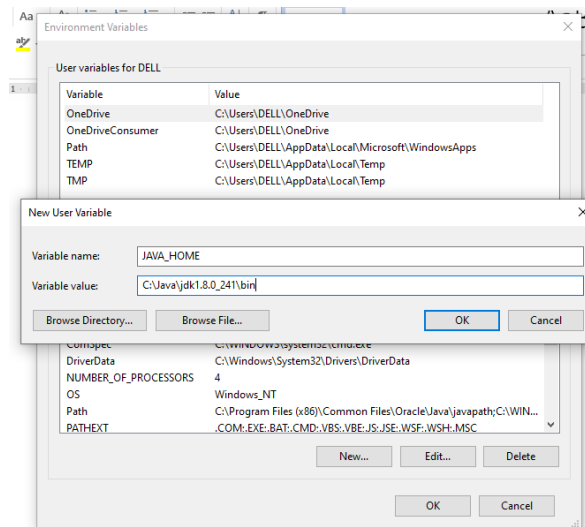
Windows->settings->system->environment variables for system->edit the system environment variables





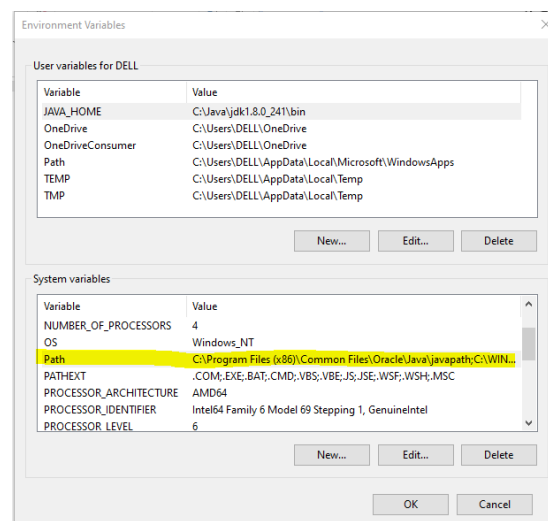
## Set java home and path for java

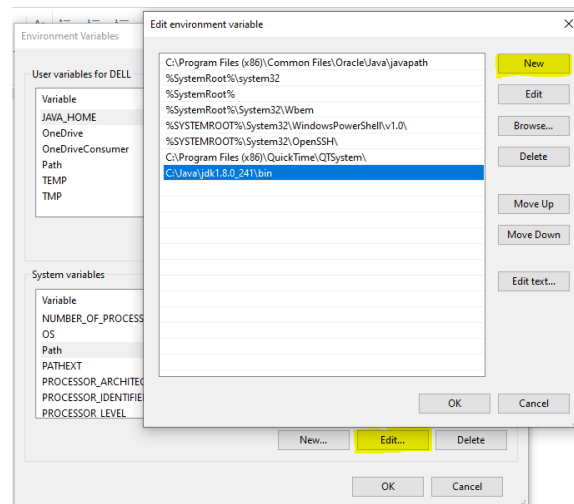
### Set user variables



### Set system variables

### System variable->path->edit->new





Click Ok-> OK->close

Java successfully installed

### Step 5: Java version checking

Go to command prompt

```
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>javac
Usage: javac <options> <source files>
where possible options include:
  -g               Generate all debugging info
  -g:none          Generate no debugging info
  -g:{lines,vars,source} Generate only some debugging info
  -nowarn          Generate no warnings
  -verbose         Output messages about what the compiler is doing
  -deprecation     Output source locations where deprecated APIs are used
  -classpath <path> Specify where to find user class files and annotation processors
  -cp <path>       Specify where to find user class files and annotation processors
  -sourcepath <path> Specify where to find input source files
  -bootclasspath <path> Override location of bootstrap class files
  -extdirs <dirs>   Override location of installed extensions
  -endorseddirs <dirs> Override location of endorsed standards path
  -proc:{none,only} Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery proce
ss
  -processorpath <path> Specify where to find annotation processors
  -parameters     Generate metadata for reflection on method parameters
  -d <directory>   Specify where to place generated class files
  -s <directory>   Specify where to place generated source files
  -h <directory>   Specify where to place generated native header files
  -implicit:{none,class} Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding> Specify character encoding used by source files
  -source <release> Provide source compatibility with specified release
  -target <release> Generate class files for specific VM version
```

Check java version

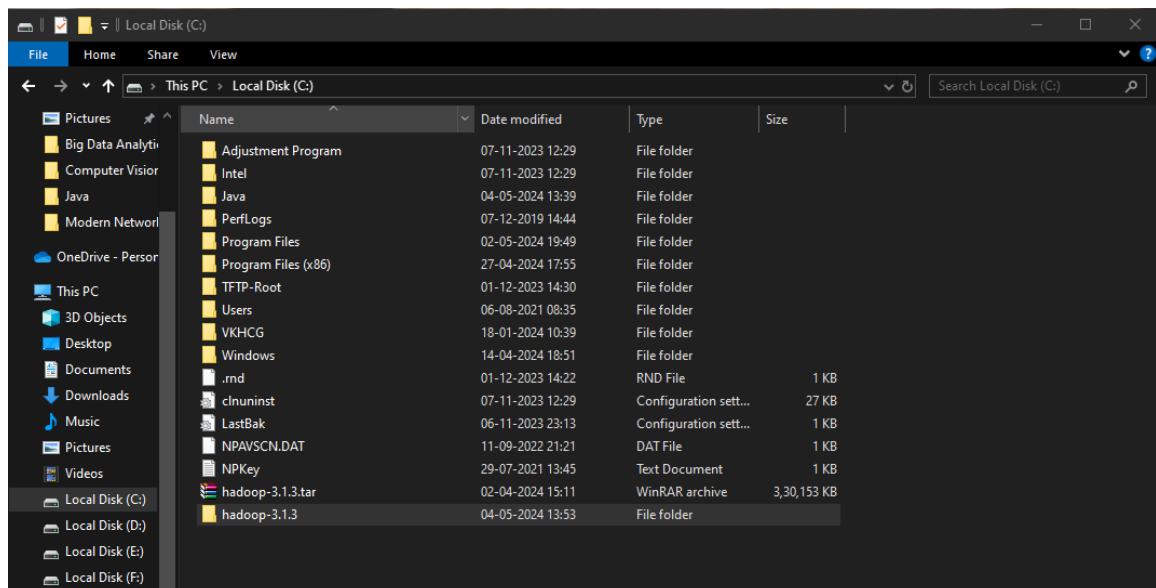
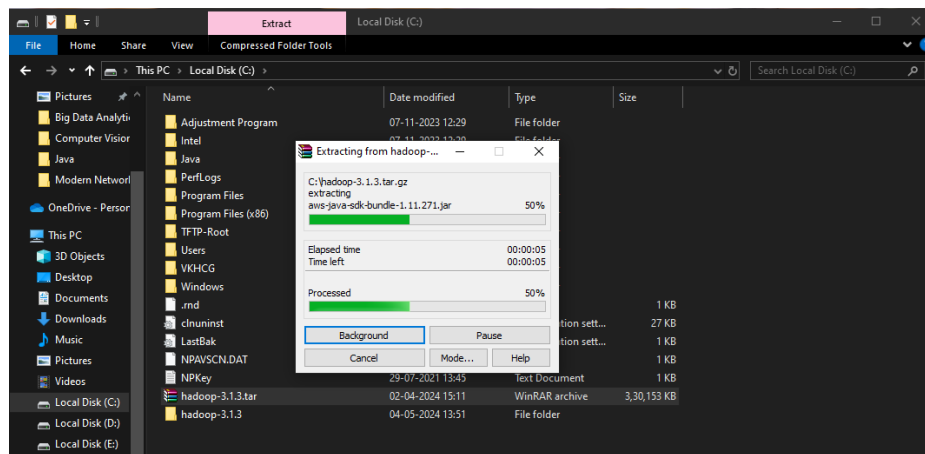
```
C:\Users\DELL> java -version
java version "1.8.0_241"
Java(TM) SE Runtime Environment (build 1.8.0_241-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)

C:\Users\DELL>
```

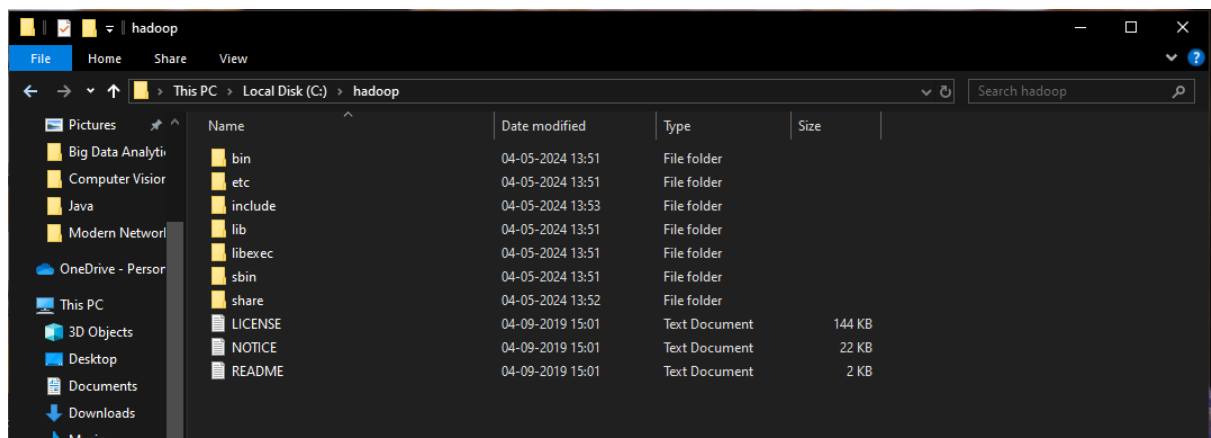


## Step 6: Install Hadoop now to our local system

Unzip Hadoop setup file

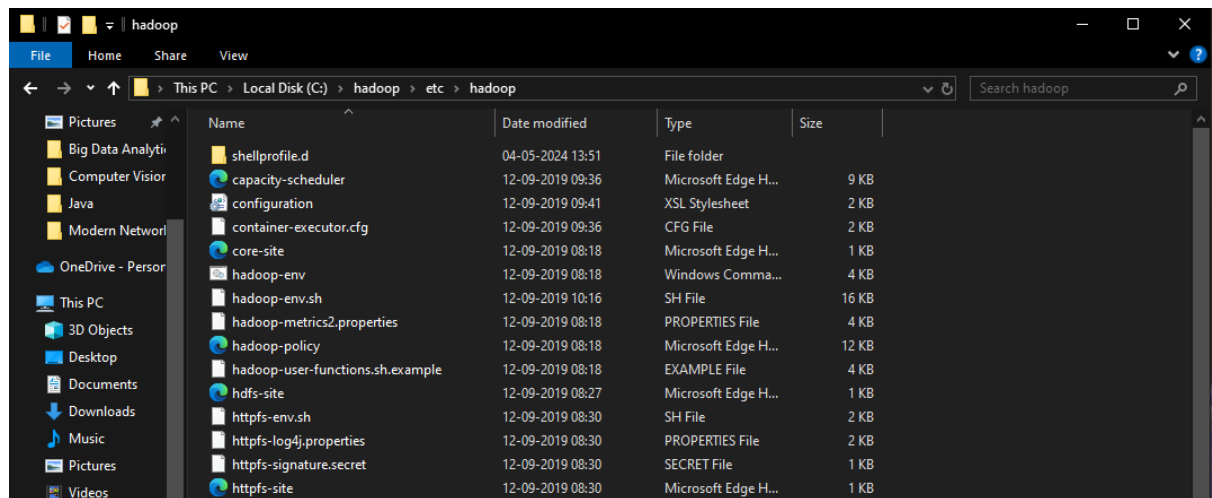


Rename Hadoop-3.1.3 folder as hadoop



## Step 7: Configuration of Hadoop

hadoop->etc-> hadoop



### Important files:

Core-site.xml

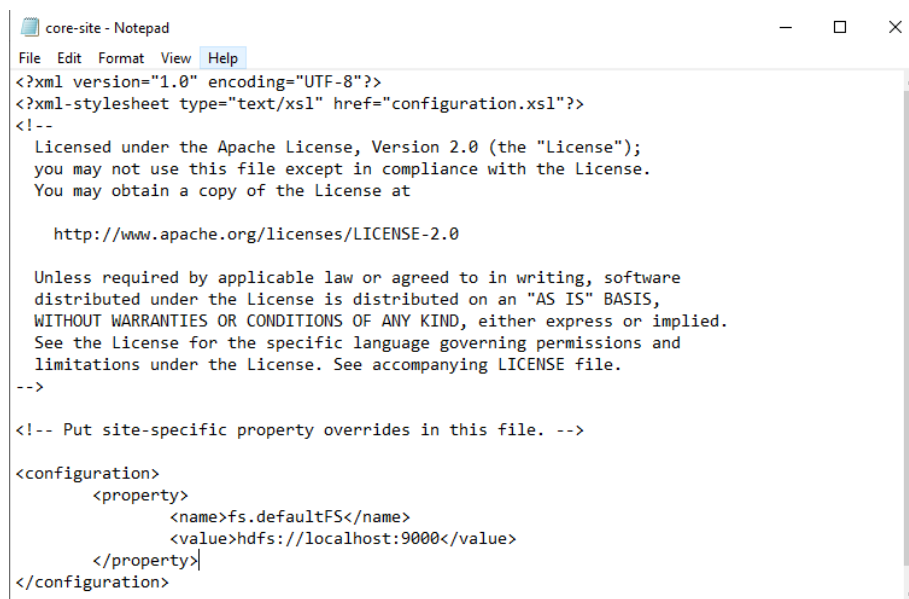
Hdfs-site.xml

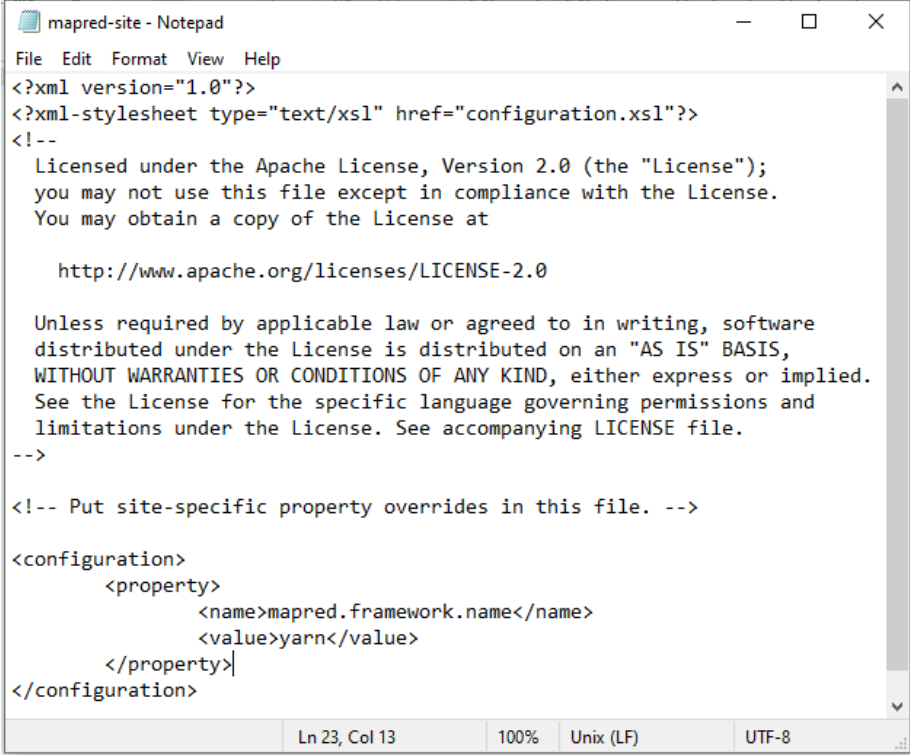
Mapred-site.xml

Yarn-site.xml

Hadoop-env windows command prompt file

Open all these files in notepad





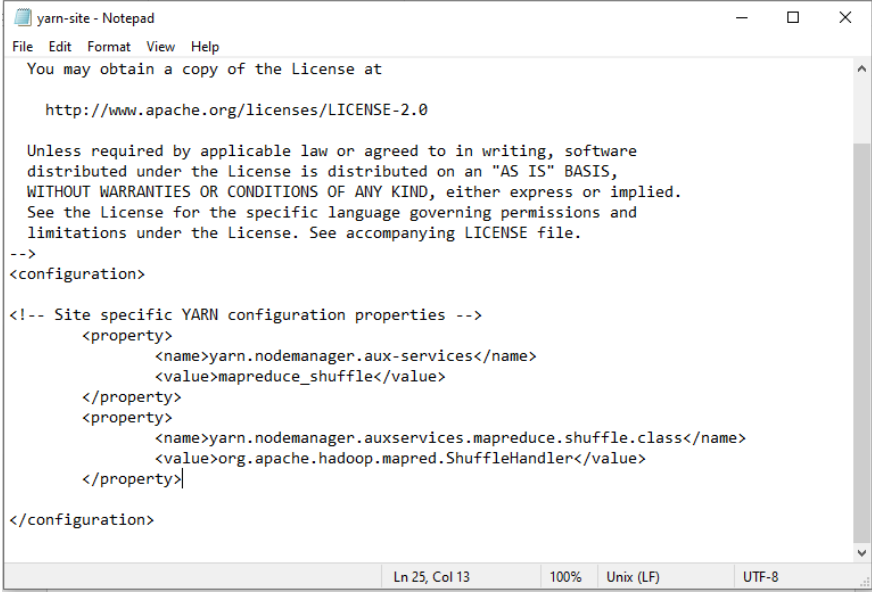
```
mapred-site - Notepad
File Edit Format View Help
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl">
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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



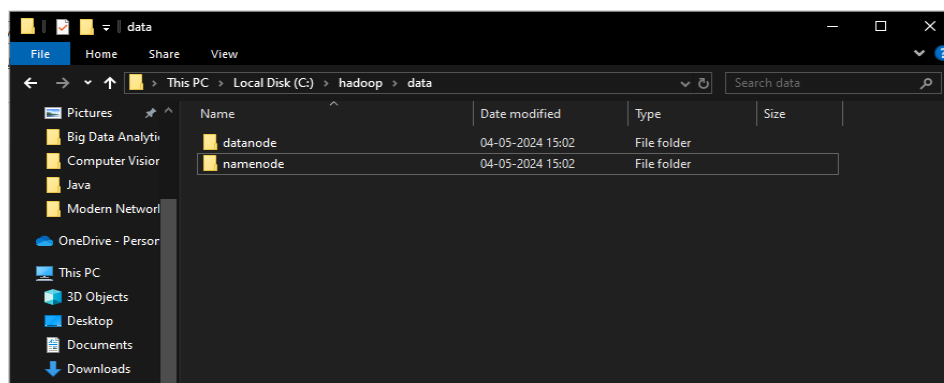
```
yarn-site - Notepad
File Edit Format View Help
You may obtain a copy of the License at

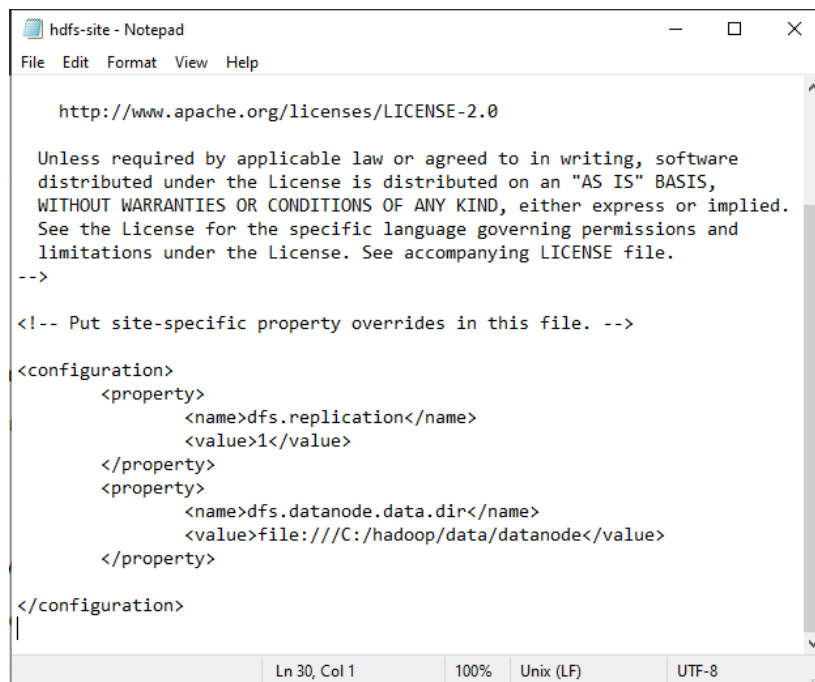
    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<configuration>

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

## Create data directory and subdirectories in hadoop





```

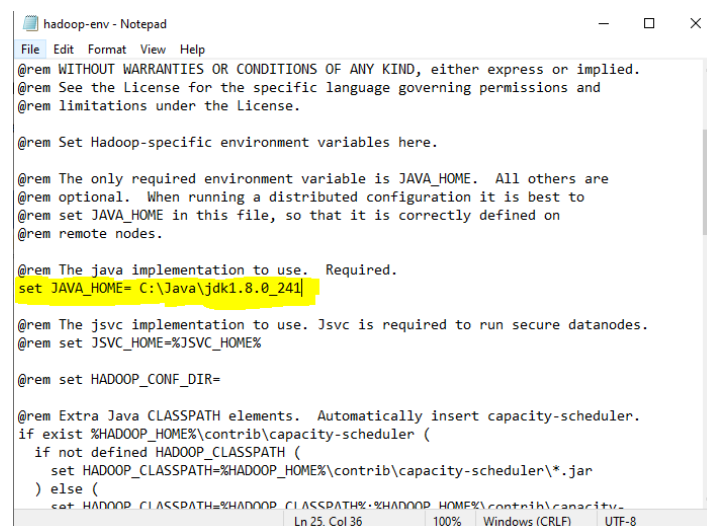
http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///C:/hadoop/data/datanode</value>
  </property>
</configuration>

```



```

@rem WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
@rem See the License for the specific language governing permissions and
@rem limitations under the License.

@rem Set Hadoop-specific environment variables here.

@rem The only required environment variable is JAVA_HOME. All others are
@rem optional. When running a distributed configuration it is best to
@rem set JAVA_HOME in this file, so that it is correctly defined on
@rem remote nodes.

@rem The java implementation to use. Required.
set JAVA_HOME= C:\Java\jdk1.8.0_241

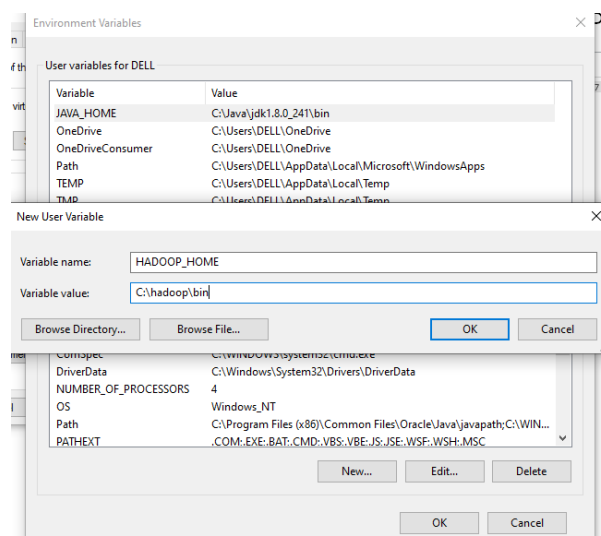
@rem The jsvc implementation to use. Jsvc is required to run secure datanodes.
@rem set JSVC_HOME=%JSVC_HOME%

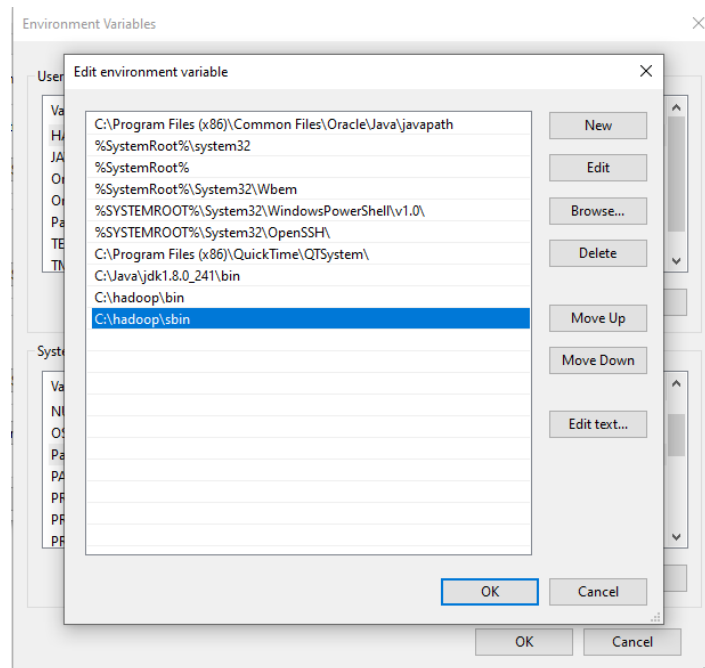
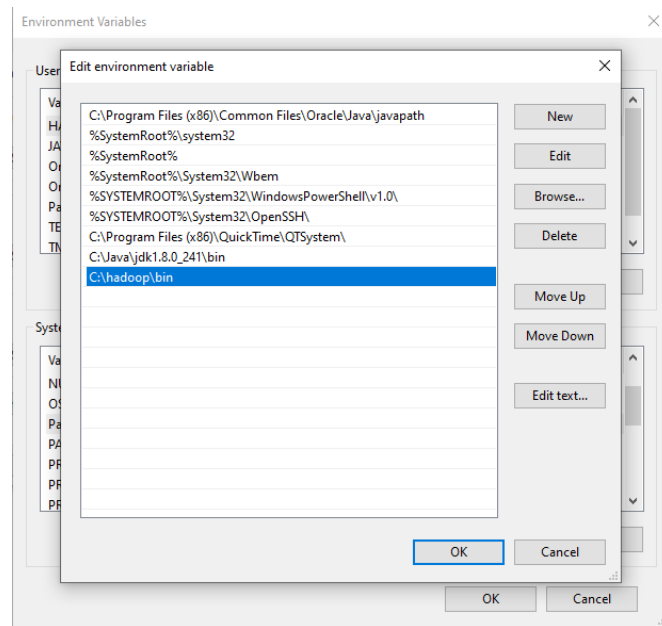
@rem set HADOOP_CONF_DIR=

@rem Extra Java CLASSPATH elements. Automatically insert capacity-scheduler.
if exist %HADOOP_HOME%\contrib\capacity-scheduler (
  if not defined HADOOP_CLASSPATH (
    set HADOOP_CLASSPATH=%HADOOP_HOME%\contrib\capacity-scheduler\*.jar
  ) else (
    set HADOOP_CLASSPATH=%HADOOP_CLASSPATH%;%HADOOP_HOME%\contrib\capacity-

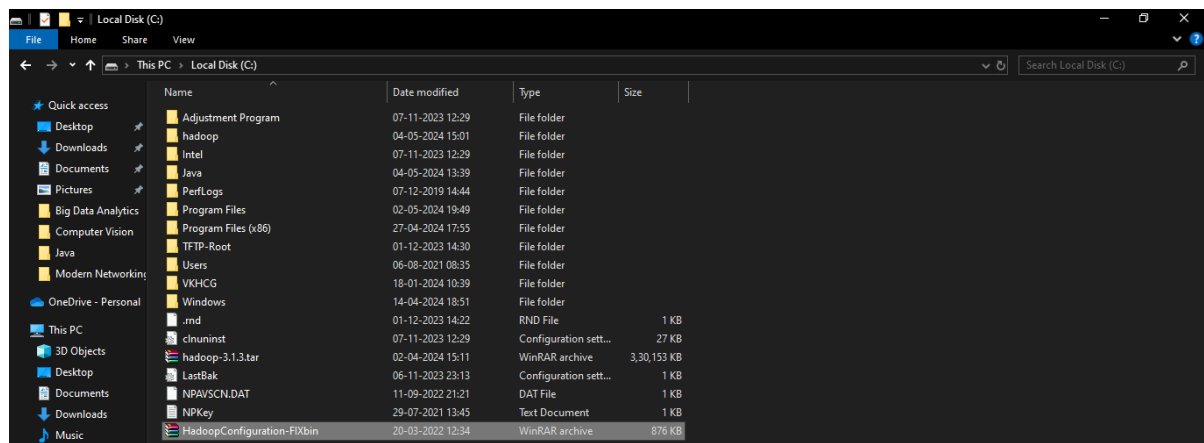
```

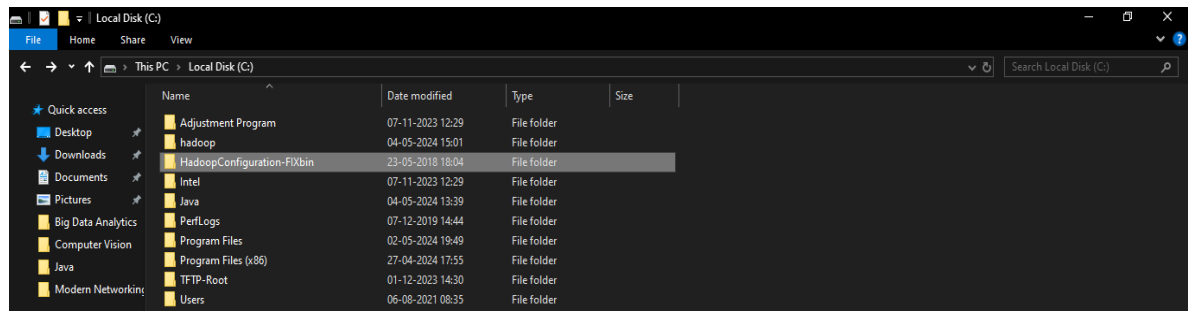
## Set home and path for Hadoop



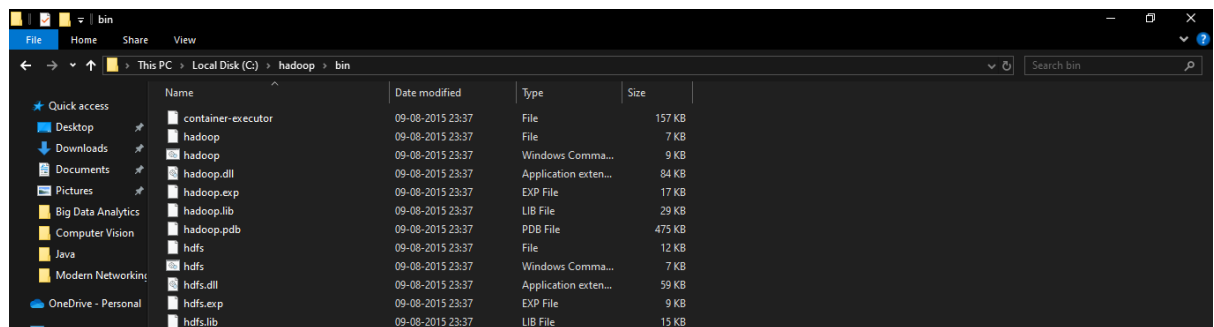


## Other configuration files





Copy bin folder from HadoopConfiguration-Fixbin folder and replace hadoop\bin folder with this



## Step 8: Verification of Hadoop installation

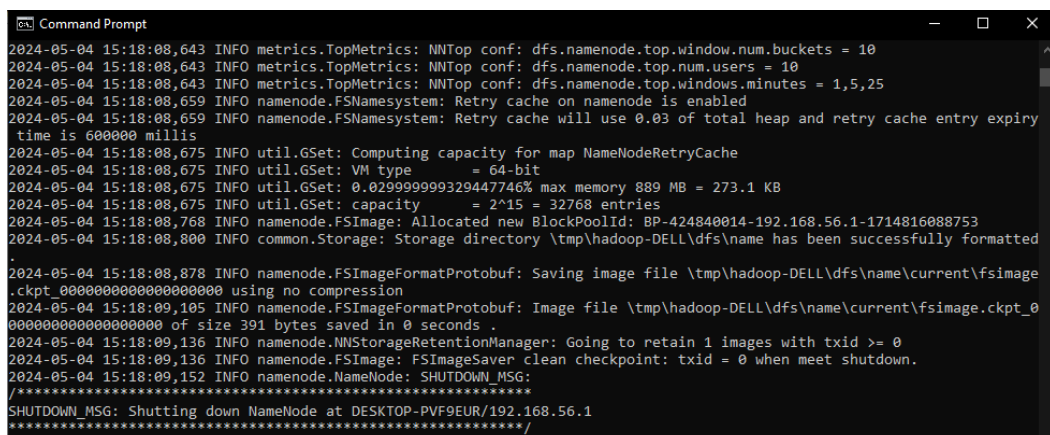
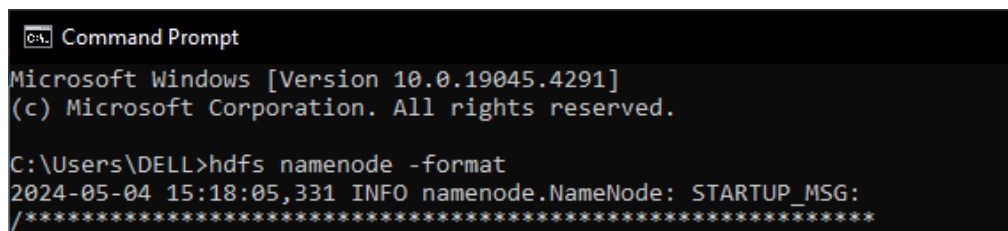
Go to command prompt.

Type:

hdfs namenode -format

set of files pop up on the terminal.

That means successful installation of Hadoop.





```
start-yarn.cmd
```

```

C:\> Command Prompt
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd..

C:\Users>cd..

C:\>cd hadoop

C:\hadoop>cd sbin

C:\hadoop\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop\sbin>start-yarn.cmd
starting yarn daemons

C:\hadoop\sbin>

```

```

ResourceLocalizationService.java:429)
    at org.apache.hadoop.service.AbstractService.stop(AbstractService.java:220)

Apache Hadoop Distribution

    at com.google.inject.servlet.ServletModule.configure(ServletModule.java:55)
    at com.google.inject.AbstractModule.configure(AbstractModule.java:62)
    at com.google.inject.spi.Elements$RecordingBinder.install(Elements.java:340)
    at com.google.inject.spi.Elements.getElements(Elements.java:110)
    at com.google.inject.internal.InjectorShell$Builder.build(InjectorShell.java:138)
    at com.google.inject.internal.InternalInjectorCreator.build(InternalInjectorCreator.java:104)
    at com.google.inject.Guice.createInjector(Guice.java:96)
    at com.google.inject.Guice.createInjector(Guice.java:73)
    at com.google.inject.Guice.createInjector(Guice.java:62)
    at org.apache.hadoop.yarn.webapp.WebApps$Builder.build(WebApps.java:387)
    at org.apache.hadoop.yarn.webapp.WebApps$Builder.start(WebApps.java:432)
    at org.apache.hadoop.yarn.server.resourcemanager.ResourceManager.startWebApp(ResourceManager.java:1203)
    at org.apache.hadoop.yarn.server.resourcemanager.ResourceManager.serviceStart(ResourceManager.java:1312)
    at org.apache.hadoop.service.AbstractService.start(AbstractService.java:194)
    at org.apache.hadoop.yarn.server.resourcemanager.ResourceManager.main(ResourceManager.java:1507)
Caused by: java.lang.ClassNotFoundException: org.apache.hadoop.yarn.server.timelineservice.collector.TimelineCollectorManager
    at java.net.URLClassLoader.findClass(URLClassLoader.java:382)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:355)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
    ... 36 more

2024-05-04 15:23:06,330 INFO resourcemanager.ResourceManager: Transitioning to standby state
2024-05-04 15:23:06,330 INFO resourcemanager.ResourceManager: Transitioned to standby state
2024-05-04 15:23:06,330 INFO resourcemanager.ResourceManager: SHUTDOWN_MSG:
C:\hadoop\*****
SHUTDOWN_MSG: Shutting down ResourceManager at DESKTOP-PVF9EUR\192.168.56.1
*****

```



## Practical 2

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

**Theory:** MongoDB is an open-source and the leading NoSQL database. It is a document-oriented database that offers high performance, easy scalability, and high availability. It uses documents and collections to organize data rather than relations. This makes it an ideal database management system for the storage of unstructured data.

MongoDB uses replica sets to ensure there is a high availability of data. Each replica set is made up of two or more replicas of data. This gives its users the ability to access their data at any time. The replica sets also create fault tolerance. MongoDB scales well to accommodate more data. It uses the sharing technique to scale horizontally and meet the changing storage needs of its users. MongoDB was developed to help developers unleash the power of data and software.

MongoDB is an unstructured database. It stores data in the form of documents. MongoDB is able to handle huge volumes of data very efficiently and is the most widely used NoSQL database as it offers rich query language and flexible and fast access to data.

#### **The Architecture of a MongoDB Database**

The information in MongoDB is stored in documents. Here, a document is analogous to rows in structured databases.

- Each document is a collection of key-value pairs
- Each key-value pair is called a field
- Every document has an `_id` field, which uniquely identifies the documents
- A document may also contain nested documents
- Documents may have a varying number of fields (they can be blank as well)

These documents are stored in a collection. A collection is literally a collection of documents in MongoDB. This is analogous to tables in traditional databases.

Unlike traditional databases, the data is generally stored in a single collection in MongoDB, so there is no concept of joins (except \$lookup operator, which performs left-outer-join like operation). MongoDB has the nested document instead.

PyMongo is a Python library that enables us to connect with MongoDB. It allows us to perform basic operations on the MongoDB database.

We have chosen Python to interact with MongoDB because it is one of the most commonly used and considerably powerful languages for data science. PyMongo allows us to retrieve the data with dictionary-like syntax.

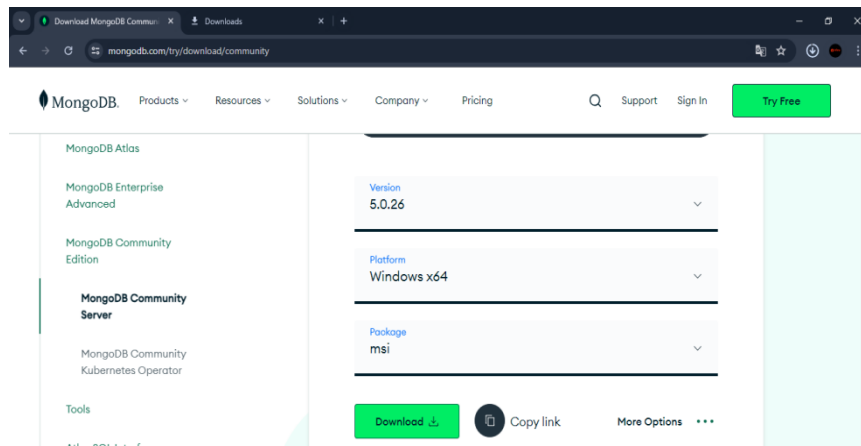
We can also use the dot notation to access MongoDB data. Its easy syntax makes our job a lot easier. Additionally, PyMongo's rich documentation is always standing there with a helping hand. We will use this library for accessing MongoDB.

### Steps of the installation:

#### Step 1: Download MongoDB

Go to official website: MongoDB Community server

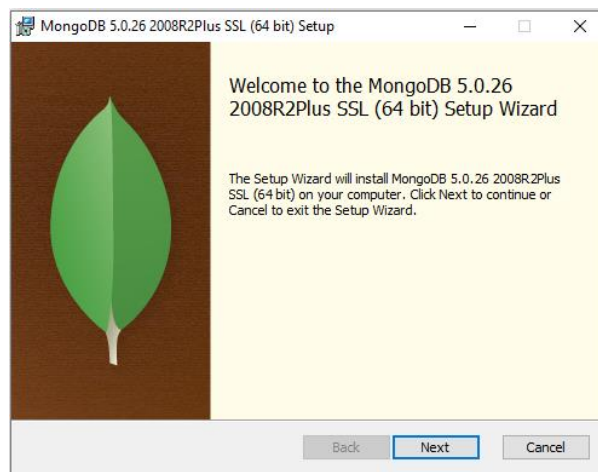
<https://www.mongodb.com/try/download/community>

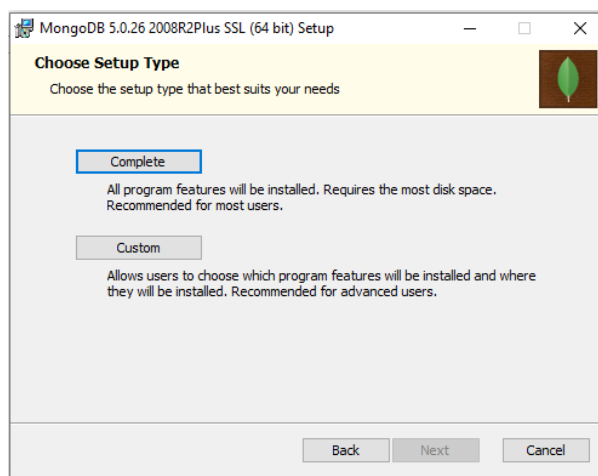
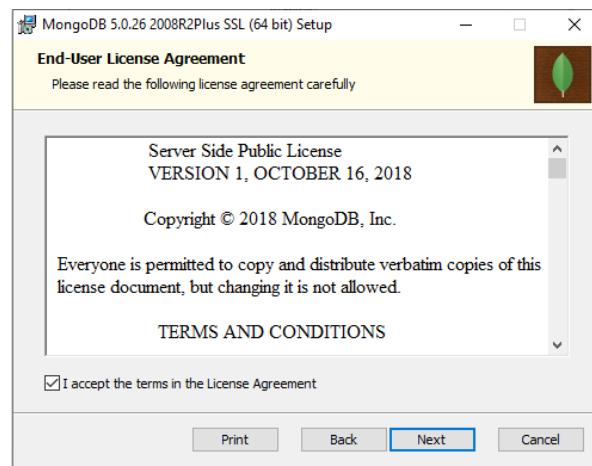


Click on download

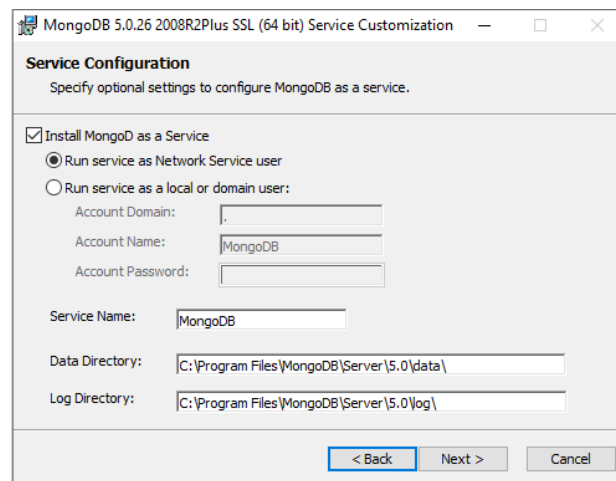
#### Step 2: Install MongoDB

It will download msi file. Click on it and Start the installation.

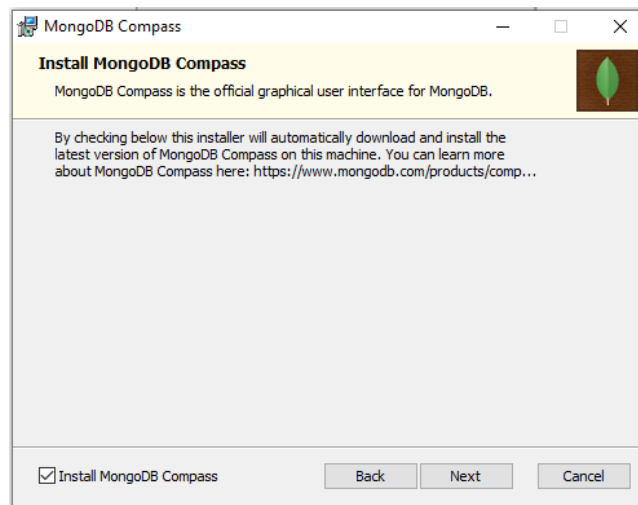




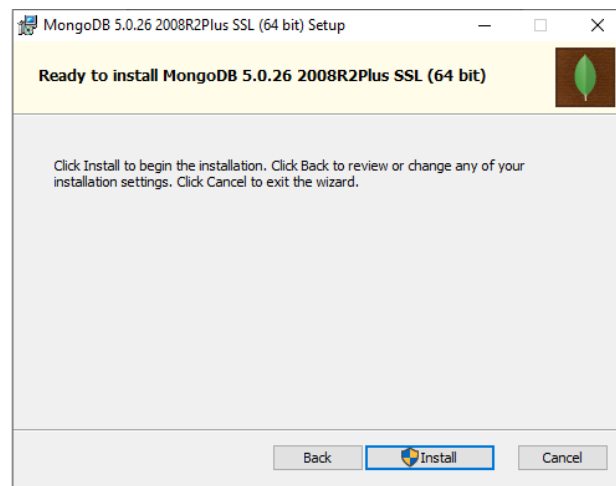
Click on complete option



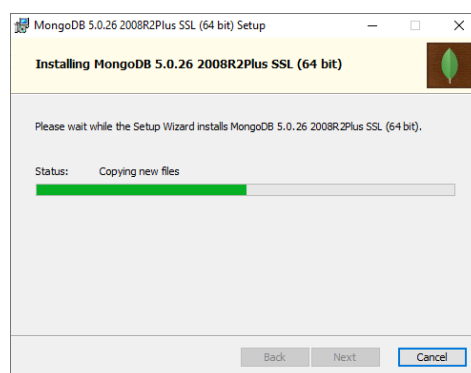
Keep all these setting as it is. Click on next.

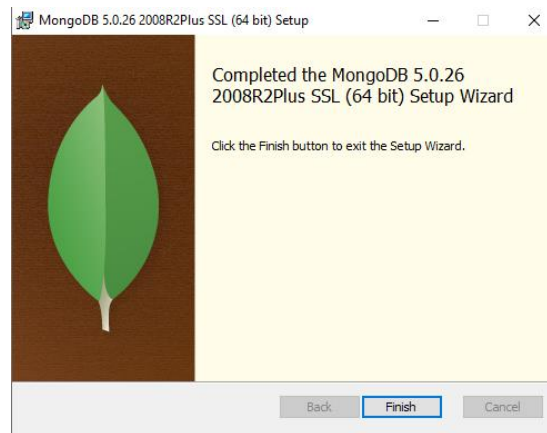


Click on next



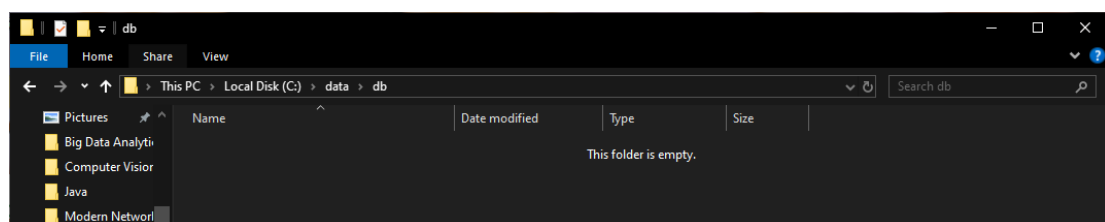
Click on install





### Step 3: Verify MongoDB Installation

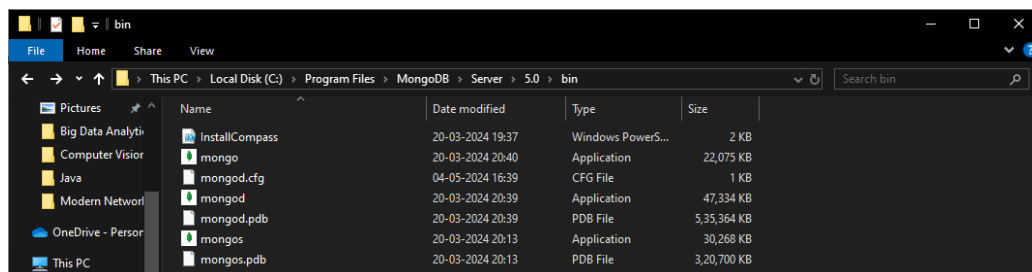
First we will create C:\data\db directory



Now go to

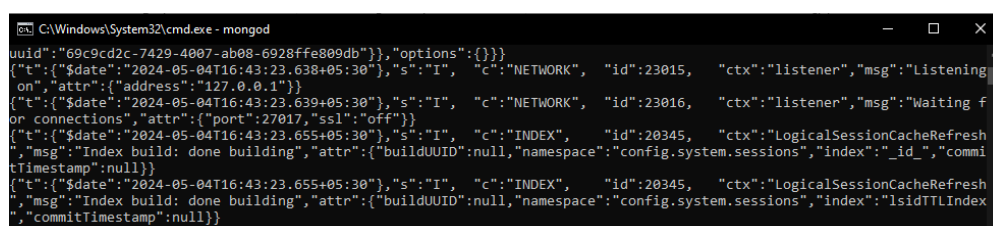
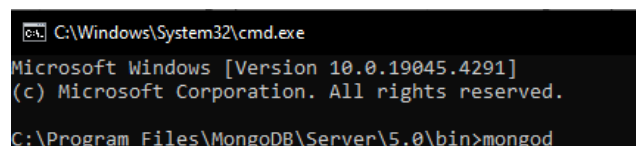
C:\Program Files\MongoDB\Server\5.0\bin

Start command prompt from this location



To start mongo db server

Enter mongod command



Mongo daemon is started now

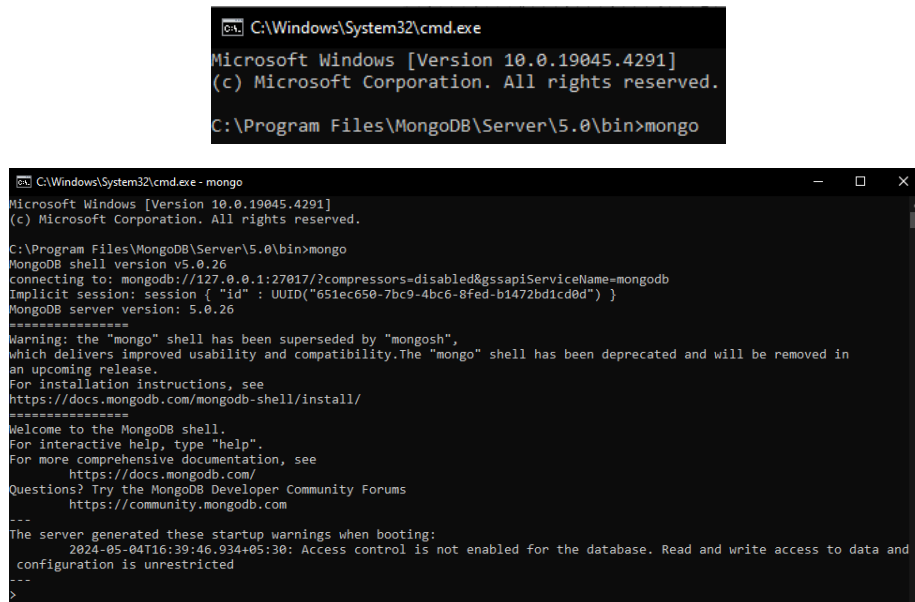
To open the mongo shell

Go to

C:\Program Files\MongoDB\Server\5.0\bin

Start command prompt from this location

Fire the command: mongo



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\MongoDB\Server\5.0\bin>mongo

C:\Windows\System32\cmd.exe - mongo
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\MongoDB\Server\5.0\bin>mongo
MongoDB shell version v5.0.26
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("651ec650-7bc9-4bc6-8fed-b1472bd1cd0d") }
MongoDB server version: 5.0.26

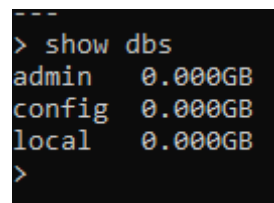
*****
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
*****
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
https://community.mongodb.com

---
The server generated these startup warnings when booting:
  2024-05-04T16:39:46.934+05:30: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
---
>
```

Mongo shell is started

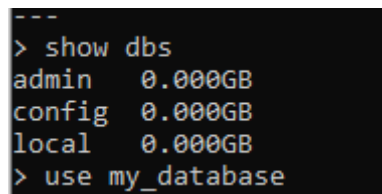
To see all the default databases:

>show dbs



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

To create new database named my\_database:



```
---
> show dbs
admin    0.000GB
config  0.000GB
local   0.000GB
> use my_database
```

To create collection in the database:

And insert json values into it:

```
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> use my_database
switched to db my_database
> db.books.insert({"name":"It Ends With Us"})
```

```
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> use my_database
switched to db my_database
> db.books.insert({"name":"It Ends With Us"})
WriteResult({ "nInserted" : 1 })
```

To see entered collection:

```
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> use my_database
switched to db my_database
> db.books.insert({"name":"It Ends With Us"})
WriteResult({ "nInserted" : 1 })
> show collections;
books
>
```

To see all the documents in the collection:

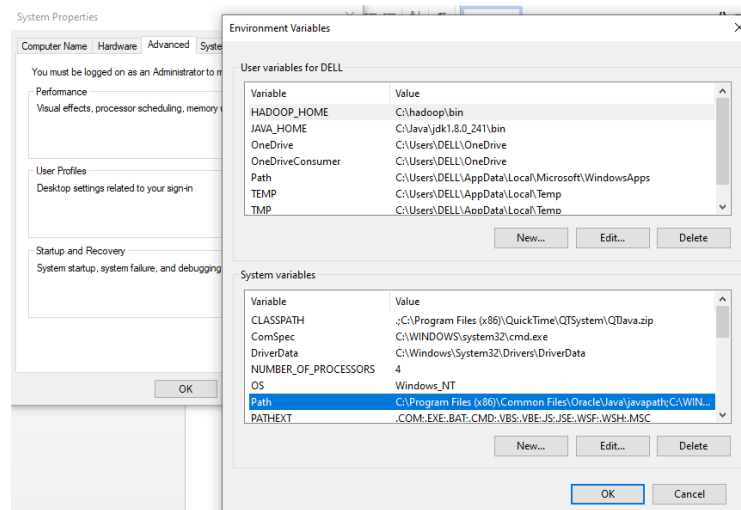
```
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> use my_database
switched to db my_database
> db.books.insert({"name":"It Ends With Us"})
WriteResult({ "nInserted" : 1 })
> show collections;
books
> db.books.find()
{ "_id" : ObjectId("66361991164b7d2f82dae287"), "name" : "It Ends With Us" }
```

To set path of MongoDB server and shell

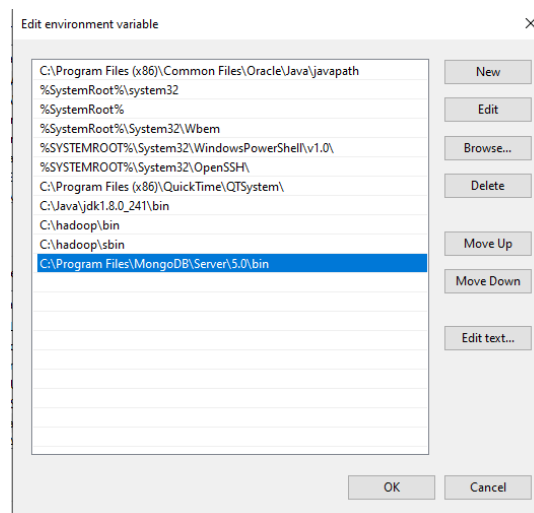
Copy in clipboard: C:\Program Files\MongoDB\Server\5.0\bin

Go to environment variables

Add mongodb path to system path variable



Click on New



Paste the path

Click on ok..ok

Run mongod and mongo command from command prompt once again from anywhere, it will

start mongo server and mongo shell

```

Command Prompt - mongod
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>mongod
{"t":{"$date":"2024-05-04T16:54:11.769+05:30"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"-", "msg":"Automatically di
abling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2024-05-04T16:54:12.838+05:30"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"thread1", "msg":"Initializ
d wire specification", "attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":13},"incomingIntern
alClient":{"minWireVersion":0,"maxWireVersion":13},"outgoing":{"minWireVersion":0,"maxWireVersion":13},"isInternalClient"
:true}}}
{"t":{"$date":"2024-05-04T16:54:12.843+05:30"},"s":"W", "c":"ASIO", "id":22601, "ctx":"thread1", "msg":"No Transpo
rtLayer configured during NetworkInterface startup"}
{"t":{"$date":"2024-05-04T16:54:12.844+05:30"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"thread1", "msg":"Implicit T
CP FastOpen in use."}
{"t":{"$date":"2024-05-04T16:54:12.847+05:30"},"s":"W", "c":"ASIO", "id":22601, "ctx":"thread1", "msg":"No Transpo
rtLayer configured during NetworkInterface startup"}
{"t":{"$date":"2024-05-04T16:54:12.847+05:30"},"s":"I", "c":"REPL", "id":5123008, "ctx":"thread1", "msg":"Successful
ly registered PrimaryOnlyService", "attr":{"service":"TenantMigrationDonorService", "ns":"config.tenantMigrationDonors"}}

```

```

Command Prompt - mongo
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>mongo
MongoDB shell version v5.0.26
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("33197e8b-5bb2-40e6-b993-b5edc01bd410") }
MongoDB server version: 5.0.26

=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
---
The server generated these startup warnings when booting:
  2024-05-04T16:39:46.934+05:30: Access control is not enabled for the database. Read and write access to data and
  configuration is unrestricted
---
>

```

To get the effect of running mongod as service, Restart your windows operating system.

Restarted the machine

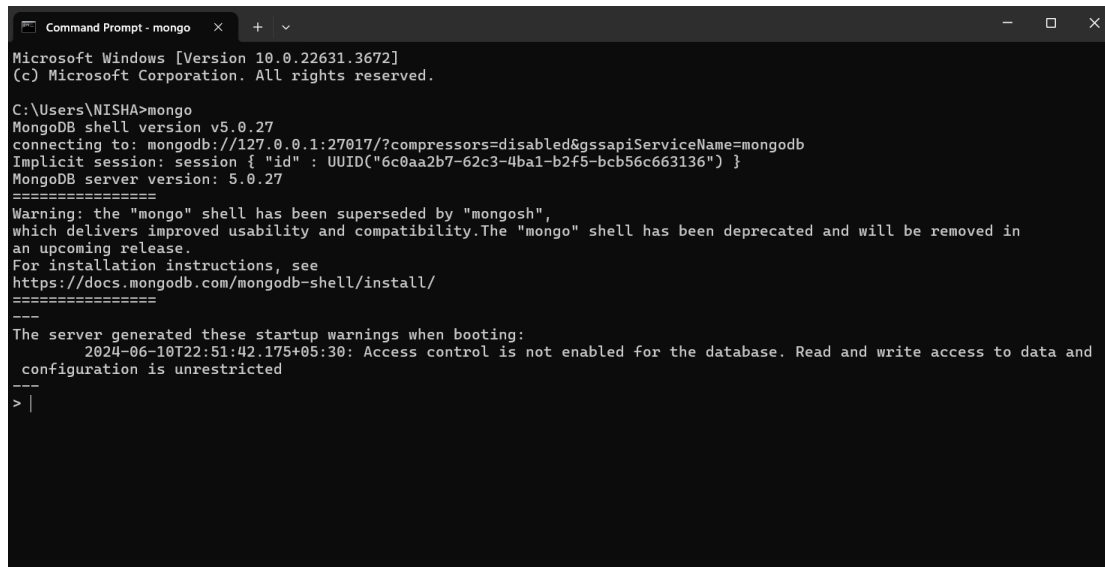


To check mongod service is running automatically:

Open command prompt

Give mongo command without running mongod command in another terminal.

It will start mongod server.



```
Command Prompt - mongo
Microsoft Windows [Version 10.0.22631.3672]
(c) Microsoft Corporation. All rights reserved.

C:\Users\NISHA>mongo
MongoDB shell version v5.0.27
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6c0aa2b7-62c3-4ba1-b2f5-bcb56c663136") }
MongoDB server version: 5.0.27
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
---
The server generated these startup warnings when booting:
  2024-06-10T22:51:42.175+05:30: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
---
> |
```

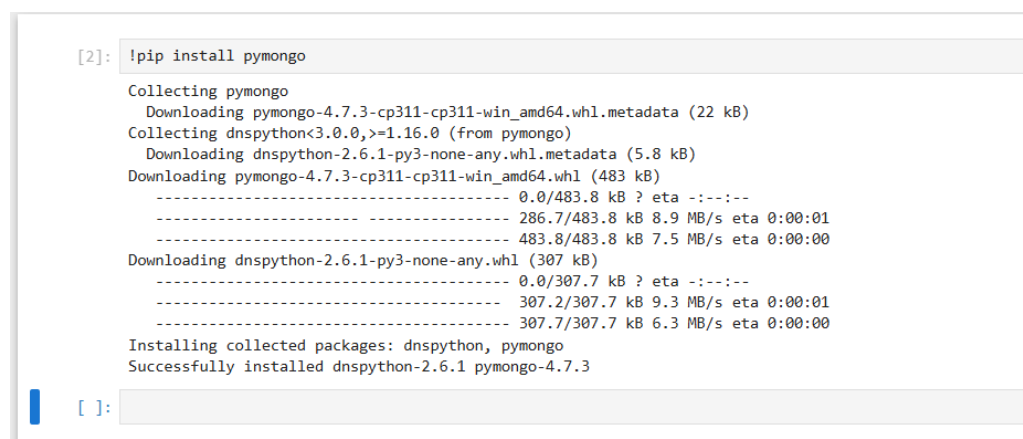
#### Step 4: Install MongoDB Python on Windows

We will be performing a few key basic operations on a MongoDB database in Python using

the PyMongo library.

Install package to use MongoDB

To install this package run the command !pip install pymongo on Jupyter Notebook



```
[2]: !pip install pymongo

Collecting pymongo
  Downloading pymongo-4.7.3-cp311-cp311-win_amd64.whl.metadata (22 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.6.1-py3-none-any.whl.metadata (5.8 kB)
  Downloading pymongo-4.7.3-cp311-cp311-win_amd64.whl (483 kB)
----- 0.0/483.8 kB ? eta ----
----- 286.7/483.8 kB 8.9 MB/s eta 0:00:01
----- 483.8/483.8 kB 7.5 MB/s eta 0:00:00
  Downloading dnspython-2.6.1-py3-none-any.whl (307 kB)
----- 0.0/307.7 kB ? eta ----
----- 307.2/307.7 kB 9.3 MB/s eta 0:00:01
----- 307.7/307.7 kB 6.3 MB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.6.1 pymongo-4.7.3

[ ]:
```

**Step 5: Verify MongoDB Python Connection**

To retrieve the data from a MongoDB database, we will first connect to it. Write and execute

the below code in your spider anaconda

```
import pymongo
```

```
mongo_uri = "mongodb://localhost:27017/"
```

```
client = pymongo.MongoClient(mongo_uri)
```

Let's see the available databases:

```
print(client.list_database_names())
```

We will use the my\_database database for our purpose. Let's set the cursor to the same

database:

```
db = client.my_database
```

connect to analysis database

The list\_collection\_names command shows the names of all the available collections:

```
print(db.list_collection_names())
```

Let's see the number of books we have. We will connect to the customers collection and then

print the number of documents available in that collection:

```
table=db.books
```

```
print(table.count_documents({})) #gives the number of documents in the table
```

## CODE

```
import pymongo

mongo_uri = "mongodb://localhost:27017/"

client = pymongo.MongoClient(mongo_uri)

print(client.list_database_names())

db = client.my_database

print(db.list_collection_names())

table=db.books

print(table.count_documents({}))

print('Ayush Patel - 53004230035')
```

```
[2]: import pymongo
      mongo_uri = "mongodb://localhost:27017/"
      client = pymongo.MongoClient(mongo_uri)
      print(client.list_database_names())
      db = client.my_database
      print(db.list_collection_names())
      table=db.books
      print(table.count_documents({}))
      print('Ayush Patel - 53004230035')

      ['admin', 'config', 'local', 'my_database']
      ['books']
      1
      Ayush Patel - 53004230035
```

```
[ ]:
```

### Practical 3

**AIM: Implement Regression Model to import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables. Also check if the model is fit or not.**

#### Theory:

A Regression models are used to describe relationships between variables by fitting a line to the observed data. Regression allows you to estimate how a dependent variable change as the independent variable(s) change.

#### Linear Regression:

Linear regression quantifies the relationship between one or more predictor variable(s) and one outcome variable. Linear regression is commonly used for predictive analysis and modelling. For example, it can be used to quantify the relative impacts of age, gender, and diet (the predictor variables) on height (the outcome variable).

#### Simple linear regression formula:

The formula for a simple linear regression is:

$$Y = \beta_0 + \beta_1 X + E$$

- Y is the predicted value of the dependent variable (y) for any given value of the independent variable (x).
- B0 is the intercept, the predicted value of y when the x is 0.
- B1 is the regression coefficient – how much we expect y to change as x increases.
- X is the independent variable (the variable we expect is influencing y).
- e is the error of the estimate, or how much variation there is in our estimate of the regression coefficient.

Linear regression is a powerful tool for understanding and predicting the behaviour of a variable, however, it needs to meet a few conditions to be accurate and dependable solutions.

1. **Linearity:** The independent and dependent variables have a linear relationship with one another. This implies that changes in the dependent variable follow those in the independent variable(s) in a linear fashion.
2. **Independence:** The observations in the dataset are independent of each other. This means that the value of the dependent variable for one observation does not depend on the value of the dependent variable for another observation.
3. **Homoscedasticity:** Across all levels of the independent variable(s), the variance of the errors is constant. This indicates that the amount of the independent variable(s) has no impact on the variance of the errors.
4. **Normality:** The errors in the model are normally distributed.

5. **No multicollinearity:** There is no high correlation between the independent variables. This indicates that there is little or no correlation between the independent variables.

## CODE

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load Dataset
diabetes = datasets.load_diabetes()

# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y = True)

# Description of the dataset
print(diabetes['DESCR'])
print(diabetes.feature_names)
```

Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

**\*\*Data Set Characteristics:\*\***

- :Number of Instances: 442
- :Number of Attributes: First 10 columns are numeric predictive values
- :Target: Column 11 is a quantitative measure of disease progression one year after baseline
- :Attribute Information:
  - age age in years
  - sex
  - bmi body mass index
  - bp average blood pressure
  - s1 tc, total serum cholesterol
  - s2 ldl, low-density lipoproteins
  - s3 hdl, high-density lipoproteins
  - s4 tch, total cholesterol / HDL
  - s5 ltg, possibly log of serum triglycerides level
  - s6 glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the standard deviation times the square root of 'n\_samples' (i.e. the sum of squares of each column totals 1).

Source URL:  
<https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html>

For more information see:  
 Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with discussion), 407-499.  
 ([https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle\\_2002.pdf](https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf))

```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

---

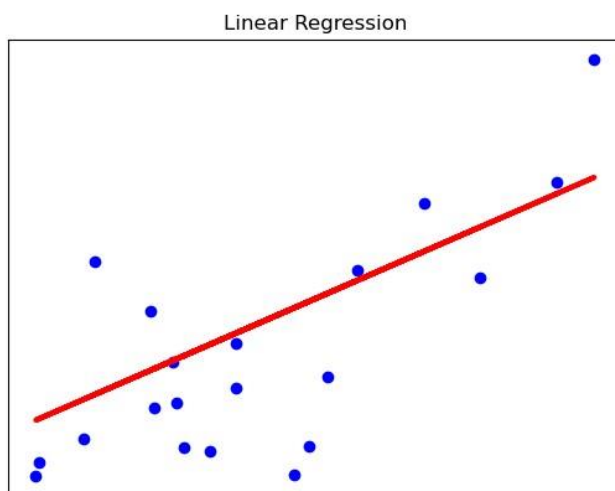
```
# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]

# Split the data into training and testing datasets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training and testing datasets
```

```
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
# Create linear regression object
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
# Make predictions using the testing sets
diabetes_y_pred = regr.predict(diabetes_X_test)
# The Coefficients
print('Coefficients : \n', regr.coef_)
# The mean squared error
print("Mean Squared Error: %.2f" %
      mean_squared_error(diabetes_y_test, diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" %
      r2_score(diabetes_y_test, diabetes_y_pred))
# plot the outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color='blue')
plt.plot(diabetes_X_test, diabetes_y_pred, color='red', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.title('Linear Regression')
plt.show()
print('Ayush PAtel - 53004230035')
```

## OUTPUT



Uttam - 53004230029

## Practical 4

### **Aim: Apply Multiple Regression on a dataset having a continuous independent variable.**

**Theory:** Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable.

Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

Assumptions for Multiple Linear Regression:

- A linear relationship should exist between the Target and predictor variables.
- The regression residuals must be normally distributed.
- MLR assumes little or no multicollinearity (correlation between the independent variable) in data.

Multiple linear regression (MLR) is used to determine a mathematical relationship among several random variables. In other terms, MLR examines how multiple independent variables are related to one dependent variable. Once each of the independent factors has been determined to predict the dependent variable, the information on the multiple variables can be used to create an accurate prediction on the level of effect they have on the outcome variable. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points.

Multiple linear regression formula:

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_n X_n + E$$

- Y = the predicted value of the dependent variable
- B<sub>0</sub> = the y-intercept (value of y when all other parameters are set to 0)
- B<sub>1</sub>X<sub>1</sub> = the regression coefficient (B<sub>1</sub>) of the first independent variable (X<sub>1</sub>) (a.k.a. the effect that increasing the value of the independent variable has on the predicted y value)
- ... = do the same for however many independent variables you are testing
- B<sub>n</sub>X<sub>n</sub> = the regression coefficient of the last independent variable
- e = model error (a.k.a. how much variation there is in our estimate of y)

A multiple regression considers the effect of more than one explanatory variable on some outcome of interest. It evaluates the relative effect of these explanatory, or independent, variables on the dependent variable when holding all the other variables in the model constant.

**CODE**

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

diabetes = datasets.load_diabetes()
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
# Description of the dataset
print(diabetes['DESCR'])
print(diabetes.feature_names)
diabetes_X = diabetes_X[:, np.newaxis, 0]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-30:]
diabetes_y_train = diabetes_y[:-30]
diabetes_y_test = diabetes_y[-30:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print('Age')
print("Coefficients: \n", regr.coef_)
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test,
diabetes_y_pred))
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test,
diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test, color="red")
plt.plot(diabetes_X_test, diabetes_y_pred, color="red", linewidth=2,
label='Age')
plt.xticks(())
plt.yticks(())
plt.title('Multiple Regression')
#plt.xlabel('Age')
plt.ylabel('Disease Progression')
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
```



```
print(diabetes.feature_names)
diabetes_X = diabetes_X[:, np.newaxis, 3]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-30:]
diabetes_y_train = diabetes_y[:-30]
diabetes_y_test = diabetes_y[-30:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
print('BP')
print("Coefficients: \n", regr.coef_)
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test,
diabetes_y_pred))
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test,
diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test, color="blue")
plt.plot(diabetes_X_test, diabetes_y_pred, color="blue", linewidth=2,
label='BP')
plt.xticks(())
plt.yticks(())
plt.title('Multiple Regression')
plt.ylabel('Disease Progression')
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
print(diabetes.feature_names)
diabetes_X = diabetes_X[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-30:]
diabetes_y_train = diabetes_y[:-30]
diabetes_y_test = diabetes_y[-30:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print('BMI')
print("Coefficients: \n", regr.coef_)
```

```

print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test,
diabetes_y_pred))
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test,
diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test, diabetes_y_pred, color="black", linewidth=2,
label='BMI')
plt.xticks(())
plt.yticks(())
plt.title('Multiple Regression')
plt.ylabel('Disease Progression')
plt.legend()
plt.show()
print('Ayush PAtel - 53004230035')

```

## OUTPUT

```

- sex
- bmi    body mass index
- bp     average blood pressure
- s1     tc, total serum cholesterol
- s2     ldl, low-density lipoproteins
- s3     hdl, high-density lipoproteins
- s4     tch, total cholesterol / HDL
- s5     ltg, possibly log of serum triglycerides level
- s6     glu, blood sugar level

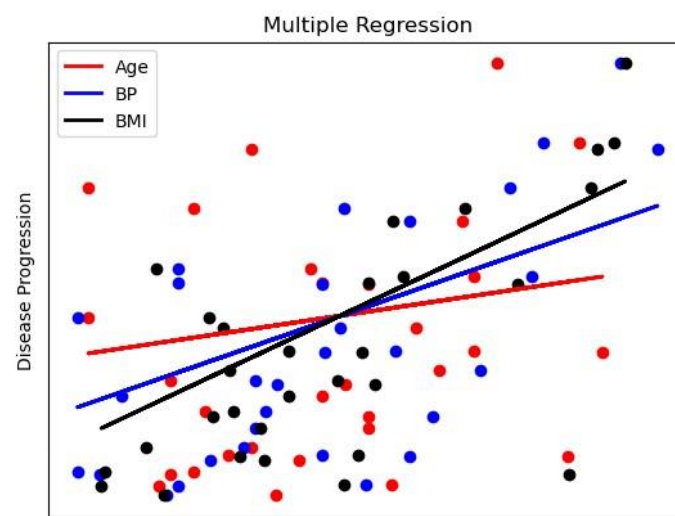
Note: Each of these 10 feature variables have been mean centered and scaled by the standard deviation times the square root of 'n_samples' (i.e. the sum
of squares of each column totals 1).

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with discussion), 407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle\_2002.pdf)

['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
Age
Coefficients:
[298.46194553]
Mean squared error: 5275.14
Coefficient of determination: -0.02
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
BP
Coefficients:
[693.36333298]
Mean squared error: 3429.00
Coefficient of determination: 0.33
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
BMI
Coefficients:
[941.43097333]
Mean squared error: 3035.06
Coefficient of determination: 0.41

```



Uttam - 53004230029

---

## Practical 5

### Aim: Build a Classification Model

**Theory:** Classification is defined as the process of recognition, understanding, and grouping of objects and ideas into preset categories i.e. “sub-populations.” With the help of these pre-categorized training datasets, classification in machine learning programs leverage a wide range of algorithms to classify future datasets into respective and relevant categories. Based on training data, the Classification algorithm is a Supervised Learning technique used to categorize new observations. In classification, a program uses the dataset or observations provided to learn how to categorize new observations into various classes or groups. For instance, 0 or 1, red or blue, yes or no, spam or not spam, etc. Targets, labels, or categories can all be used to describe classes. The Classification algorithm uses labelled input data because it is a supervised learning technique and comprises input and output information. A discrete output function (y) is transferred to an input variable in the classification process (x).

### Types of Classification Algorithms

1. Logistic Regression: – It is one of the linear models which can be used for classification. It uses the sigmoid function to calculate the probability of a certain event occurring. It is an ideal method for the classification of binary variables.
2. K-Nearest Neighbours (KNN): – It uses distance metrics like Euclidean distance, Manhattan distance, etc. to calculate the distance of one data point from every other data point. To classify the output, it takes a majority vote from k nearest neighbours of each data point.
3. Decision Trees: – It is a non-linear model that overcomes a few of the drawbacks of linear algorithms like Logistic regression. It builds the classification model in the form of a tree structure that includes nodes and leaves. This algorithm involves multiple if else statements which help in breaking down the structure into smaller structures and eventually providing the final outcome. It can be used for regression as well as classification problems.
4. Random Forest: – It is an ensemble learning method that involves multiple decision trees to predict the outcome of the target variable. Each decision tree provides its own outcome. In the case of the classification problem, it takes the majority vote of these multiple decision trees to classify the final outcome. In the case of the regression problem, it takes the average of the values predicted by the decision trees.
5. Naïve Bayes: – It is an algorithm that is based upon Bayes’ theorem. It assumes that any particular feature is independent of the inclusion of other features. i.e., They are not correlated to one another.
6. Support Vector Machine: – It represents the data points in multi-dimensional space. These data points are then segregated into classes with the help of hyperplanes. It plots an n-dimensional space for the n number of features in the dataset and then tries to create the hyperplanes such that it divides the data points with maximum margin.

**CODE**

```
import pandas as pd

col_names
['pregnant','glucose','bp','skin','insulin','bmi','pedigree','age','label']
# Load Dataset

pima = pd.read_csv('diabetes.csv', header=None, names=col_names)

pima.head()
```

```
|:   pregnant  glucose  bp  skin  insulin  bmi  pedigree  age  label
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     137  40   35    168  43.1     2.288   33     1
```

```
# Split dataset in features and target variable
feature_cols = ['pregnant','insulin','bmi','age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
Y = pima.label # Target variable

# Split X and Y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test =
train_test_split(X,Y,test_size=0.25,random_state=16)

# import the class
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=16)

# fit the model with data
logreg.fit(X_train,Y_train)
Y_pred = logreg.predict(X_test)

# import the metrics class
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(Y_test,Y_pred)

cnf_matrix
```

```
array([[115, 10],
       [ 25, 42]], dtype=int64)
```

```

# Visualizing confusion matrix using HeatMap
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class_names = [0,1]

fig, ax = plt.subplots()

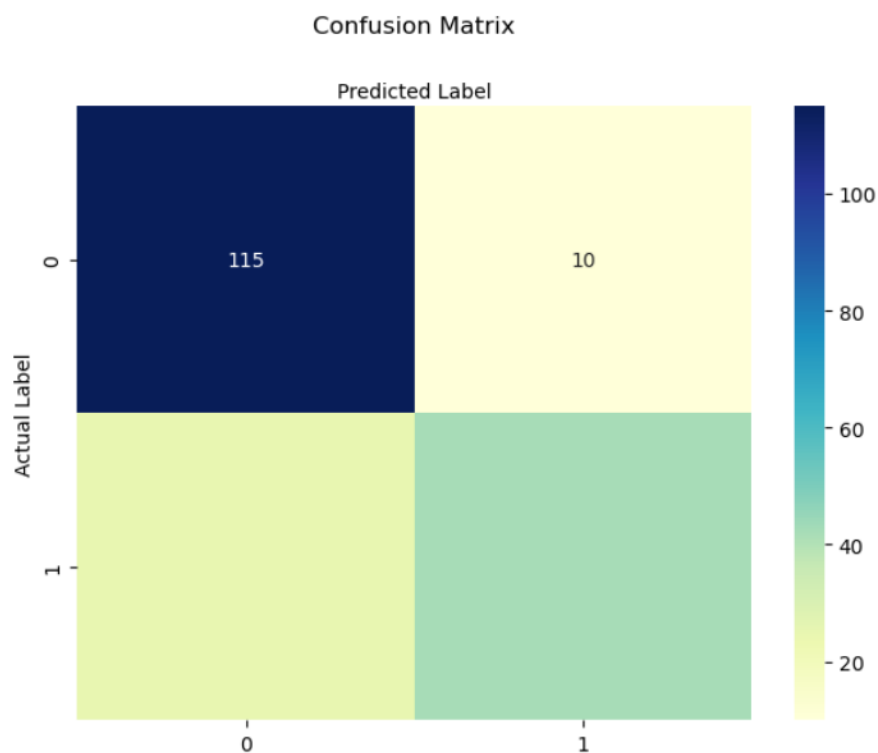
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

# create HeatMap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap='YlGnBu', fmt='g')
ax.xaxis.set_label_position('top')
plt.tight_layout()
plt.title('Confusion Matrix', y=1.1)
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')

```

## OUTPUT

Text(0.5, 427.9555555555555, 'Predicted Label')



```

from sklearn.metrics import classification_report
target_names = ['Without Diabetes', 'With Diabetes']

```

```
print('Classification Report:-')
print(classification_report(Y_test,Y_pred,target_names=target_names))
```

---

```
Classification Report:-
              precision    recall  f1-score   support

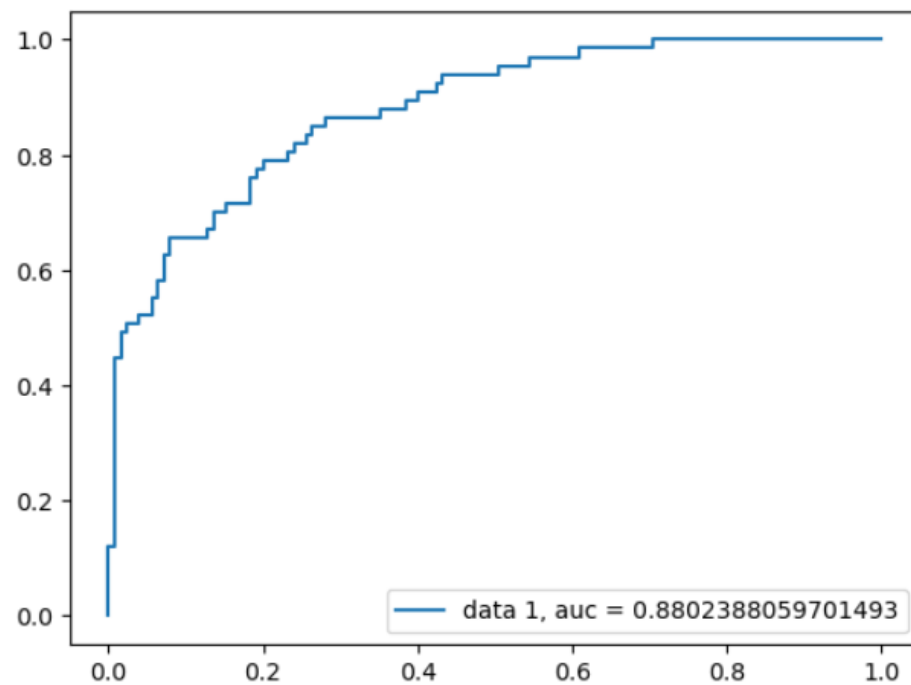
Without Diabetes      0.82      0.92      0.87       125
With Diabetes         0.81      0.63      0.71        67

 accuracy              0.82       192
 macro avg             0.81       192
 weighted avg          0.82       192
```

```
# ROC Curve
```

```
Y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(Y_test, Y_pred_proba)
auc = metrics.roc_auc_score(Y_test, Y_pred_proba)
plt.plot(fpr, tpr, label = 'data 1, auc = '+str(auc))
plt.legend(loc=4)
plt.show()
print('Ayush Patel - 53004230035')
```

## OUTPUT



Ayush Patel - 53004230035

## Practical 6

### Aim: Build a clustering model

**Theory:** Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behaviour, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method; hence no supervision is provided to the algorithm, and it deals with the unlabelled dataset.

### Types of Clustering algorithms:

1. K-Means Clustering: – It initializes a pre-defined number of k clusters and uses distance metrics to calculate the distance of each data point from the centroid of each cluster. It assigns the data points into one of the k clusters based on its distance.
2. Agglomerative Hierarchical Clustering (Bottom-Up Approach):– It considers each data point as a cluster and merges these data points on the basis of distance metric and the criterion which is used for linking these clusters.
3. Divisive Hierarchical Clustering (Top-Down Approach):– It initializes with all the data points as one cluster and splits these data points on the basis of distance metric and the criterion. Agglomerative and Divisive clustering can be represented as a dendrogram and the number of clusters to be selected by referring to the same.
4. DBSCAN (Density-based Spatial Clustering of Applications with Noise):– It is a density-based clustering method. Algorithms like K-Means work well on the clusters that are fairly separated and create clusters that are spherical in shape. DBSCAN is used when the data is in arbitrary shape and it is also less sensitive to the outliers. It groups the data points that have many neighboring data points within a certain radius.
5. OPTICS (Ordering Points to Identify Clustering Structure):– It is another type of densitybased clustering method and it is similar in process to DBSCAN except that it considers a few more parameters. But it is more computationally complex than DBSCAN. Also, it does not separate the data points into clusters, but it creates a reachability plot which can help in the interpretation of creating clusters.

### Code:

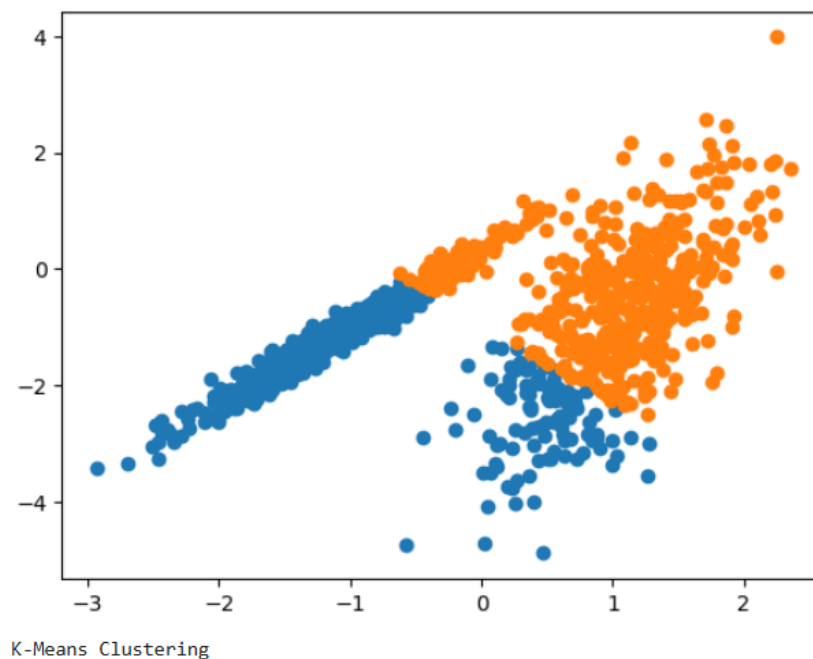
#### 1) K-Means Clustering

```
# k-means clustering
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import KMeans
from matplotlib import pyplot
```



```
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2,
n_redundant=0, n_clusters_per_class=1, random_state=4)
# define the model
model = KMeans(n_clusters=2)
# fit the model
model.fit(X)
# assign a cluster to each example
yhat = model.predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
    # get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)
    # create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
print("K-Means Clustering")
```

## OUTPUT



## 2) Agglomerative Clustering

### Code:

```
# Agglomerative clustering
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import AgglomerativeClustering
from matplotlib import pyplot

# define dataset
X, _ = make_classification(n_samples=1000, n_features=2,
                           n_informative=2, n_redundant=0, n_clusters_per_class=1,
                           random_state=4)

# define the model
model = AgglomerativeClustering(n_clusters=2)

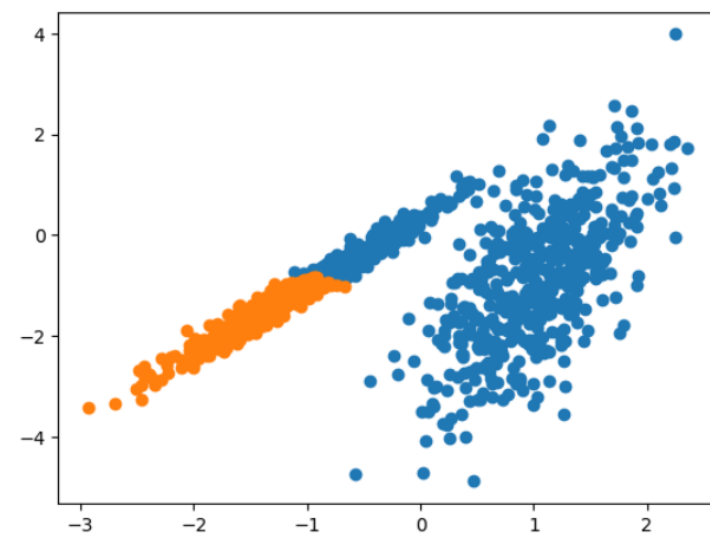
# fit model and predict clusters
yhat = model.fit_predict(X)

# retrieve unique clusters
clusters = unique(yhat)

# create scatter plot for samples from each cluster
for cluster in clusters:
    # get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)

    # create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])

# show the plot
pyplot.show()
print('Ayush Patel - 53004230035')
print('Agglomerative Clustering')
```

**OUTPUT**

Ayush Patel - 53004230035  
Agglomerative Clustering

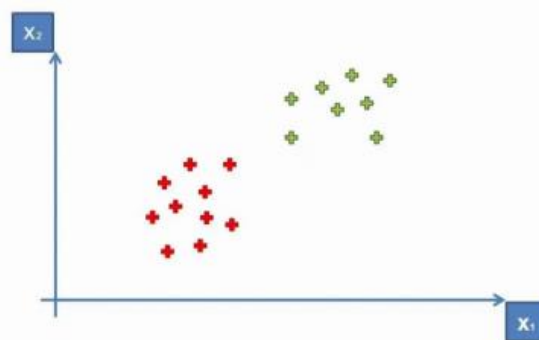
## Practical 7

### Aim: Implement SVM classification technique

**Theory:** SVM (Support Vector Machine) is a supervised machine learning algorithm. That's why training data is available to train the model. SVM uses a classification algorithm to classify a two-group problem. SVM focus on decision boundary and support vectors.

How SVM Works?

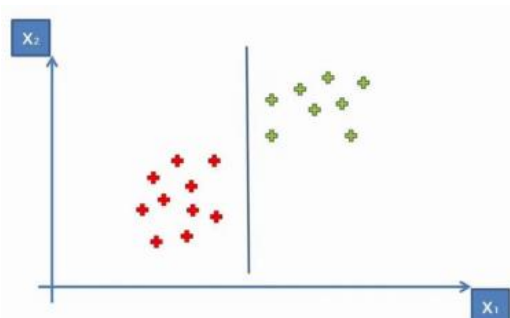
Here, we have two points in two-dimensional space, we have two columns  $x_1$  and  $x_2$ . And we have some observations such as red and green, which are already classified. This is linearly separable data.



But, now how do we derive a line that separates these points? This means a separation or decision boundary is very important for us when we add new points. So to classify new points, we need to create a boundary between two categories, and when in the future we will add new points and we want to classify them, then we know where they belong. Either in a Green Area or Red Area.

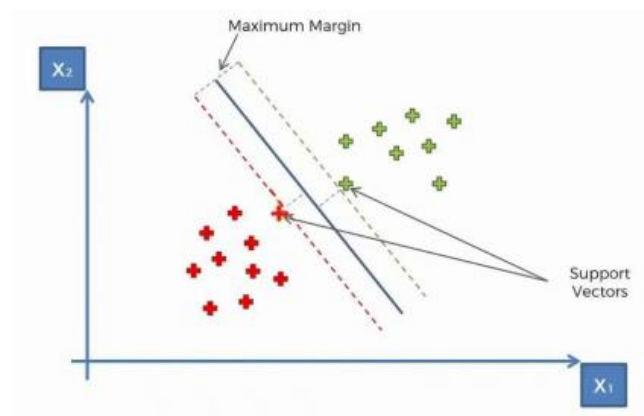
So how can we separate these points?

One way is to draw a vertical line between two areas, so anything on the left is Red and anything on the right is Green.



However, there is one more way, draw a horizontal line or diagonal line. You can create multiple diagonal lines, which achieve similar results to separate our points into two classes. But our main task is to find the optimal line or best decision boundary. And for this SVM is used. SVM finds the best decision boundary, which helps us to separate points into different spaces. SVM finds the best or optimal line through the

maximum margin, which means it has max distance and equidistance from both classes or spaces. The sum of these two classes has to be maximized to make this line the maximum margin.



These, two vectors are support vectors. In SVM, only support vectors are contributing. That's why these points or vectors are known as support vectors. Due to support vectors, this algorithm is called a Support Vector Algorithm (SVA).

In the picture, the line in the middle is a maximum margin hyperplane or classifier. In a two-dimensional plane, it looks like a line, but in a multi-dimensional, it is a hyperplane. That's how SVM works.

### CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Load Dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
dataset.head()
```

```
1]:      15624510   Male   19   19000   0
0    15810944   Male   35   20000   0
1    15668575  Female   26   43000   0
2    15603246  Female   27   57000   0
3    15804002   Male   19   76000   0
4    15728773   Male   27   58000   0
```

```
# Split Dataset into X and Y
X = dataset.iloc[:, [2,3]].values
Y = dataset.iloc[:, 4].values
```

```
# Split the X and Y dataset into Training set and Testing set
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
random_state=0)

# Perform feature scaling- feature scaling helps us to normalize the data
within a particular range
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fit SVM to the training set
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state=0)
classifier.fit(X_train, Y_train)
# Predict the test set results
Y_pred = classifier.predict(X_test)
# Make the confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cnf = confusion_matrix(Y_test, Y_pred)
print('Confusion Matrix:-')
print(cnf)
print('Accuracy Score:-')
accuracy_score(Y_test, Y_pred)
```

---

```
Confusion Matrix:-
[[63  7]
 [ 1 29]]
Accuracy Score:-
0.92
```

```
# Visualise the test set results
from matplotlib.colors import ListedColormap
X_set, Y_set = X_test, Y_test
X1, X2 = np.meshgrid(np.arange(start=X_set[:,0].min()-1,
stop=X_set[:,0].max()+1,step=0.01),
```

```

        np.arange(start=X_set[:,1].min()-1,
stop=X_set[:,1].max()+1,step=0.01))

plt.contour(X1, X2,
classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape),
alpha=0.75, cmap=ListedColormap(('red','green')))

plt.xlim(X1.min(),X1.max())

plt.ylim(X2.min(),X2.max())

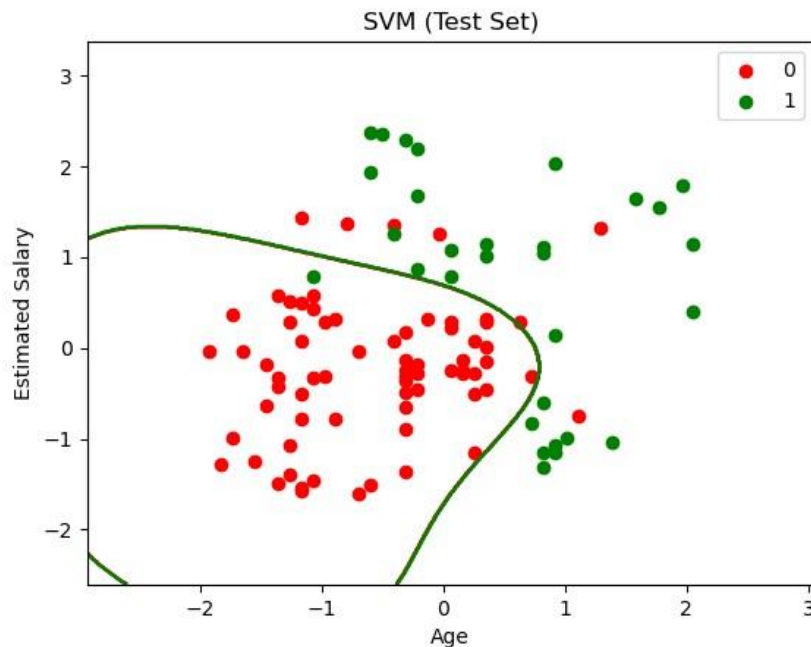
for i,j in enumerate(np.unique(Y_set)):

    plt.scatter(X_set[Y_set==j,0], X_set[Y_set==j,1],
c=ListedColormap(('red','green'))(i),label=j)

plt.title('SVM (Test Set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

```

## OUTPUT



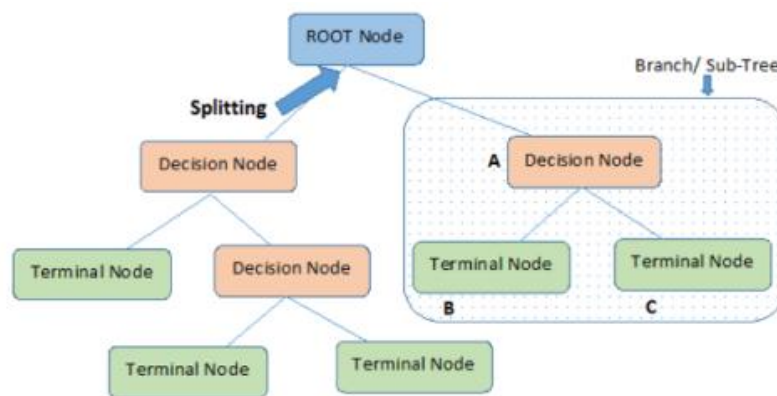
Uttam - 53004230029

## Practical 8

### Aim: Implement Decision Tree classification technique

**Theory:** A decision tree is a non-parametric supervised learning algorithm for classification and regression tasks. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. Decision trees are used for classification and regression tasks, providing easy-to-understand models.

A decision tree is a hierarchical model used in decision support that depicts decisions and their potential outcomes, incorporating chance events, resource expenses, and utility. This algorithmic model utilizes conditional control statements and is non-parametric, supervised learning, useful for both classification and regression tasks. The tree structure is comprised of a root node, branches, internal nodes, and leaf nodes, forming a hierarchical, tree-like structure.



### CODE

```
import pandas as pd

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi',
'pedigree', 'age', 'label']

pima = pd.read_csv('diabetes.csv',header=None, names=col_names)

pima.head()
```



```
1]:
```

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
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	137	40	35	168	43.1	2.288	33	1

```
# Split dataset into features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age', 'glucose', 'bp',
'pedigree']
X = pima[feature_cols]
Y = pima.label

# Split dataset into training set and testing set
X_train, X_test, Y_train, Y_test =
train_test_split(X,Y,test_size=0.3,random_state=1)

# 70% training 30% testing

# Create Decision Tree Classifier Object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifier
clf = clf.fit(X_train,Y_train)

#Predict the response for test dataset
Y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print('Accuracy:- ',metrics.accuracy_score(Y_test,Y_pred))
print('Ayush Patel - 53004230035')
```

Accuracy:- 0.670995670995671

Ayush Patel - 53004230035

```

# Visualizing Decision Trees
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
import os

os.environ["PATH"] += os.pathsep +
r'C:\Users\DELL\anaconda3\Library\bin\graphviz'

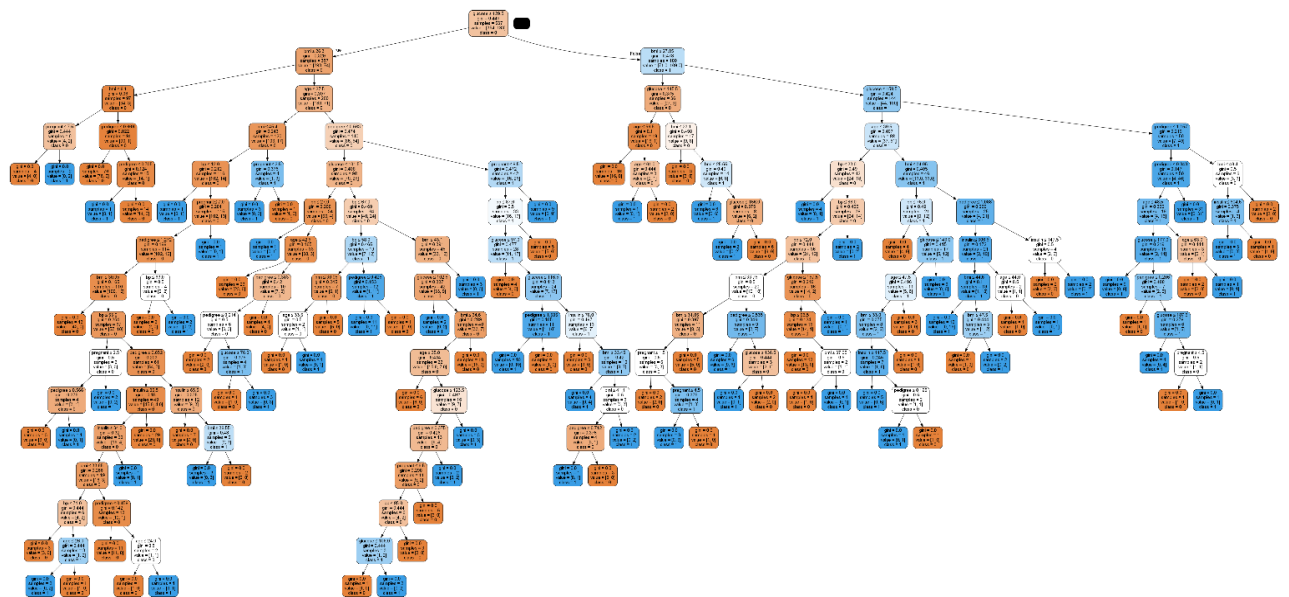
dot_data = StringIO()

export_graphviz(clf, out_file=dot_data, filled=True, rounded=True,
special_characters=True,

                 feature_names=feature_cols, class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())

```

## OUTPUT



## Practical 9

### AIM: Naïve Bayes Implementation

#### CODE

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

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting classifier to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
# Visualising the Training set results

from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train

X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                    np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
            alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Naive Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
print('Nishi Jain-53004230036')
plt.show()


# Visualising the Test set results

from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test

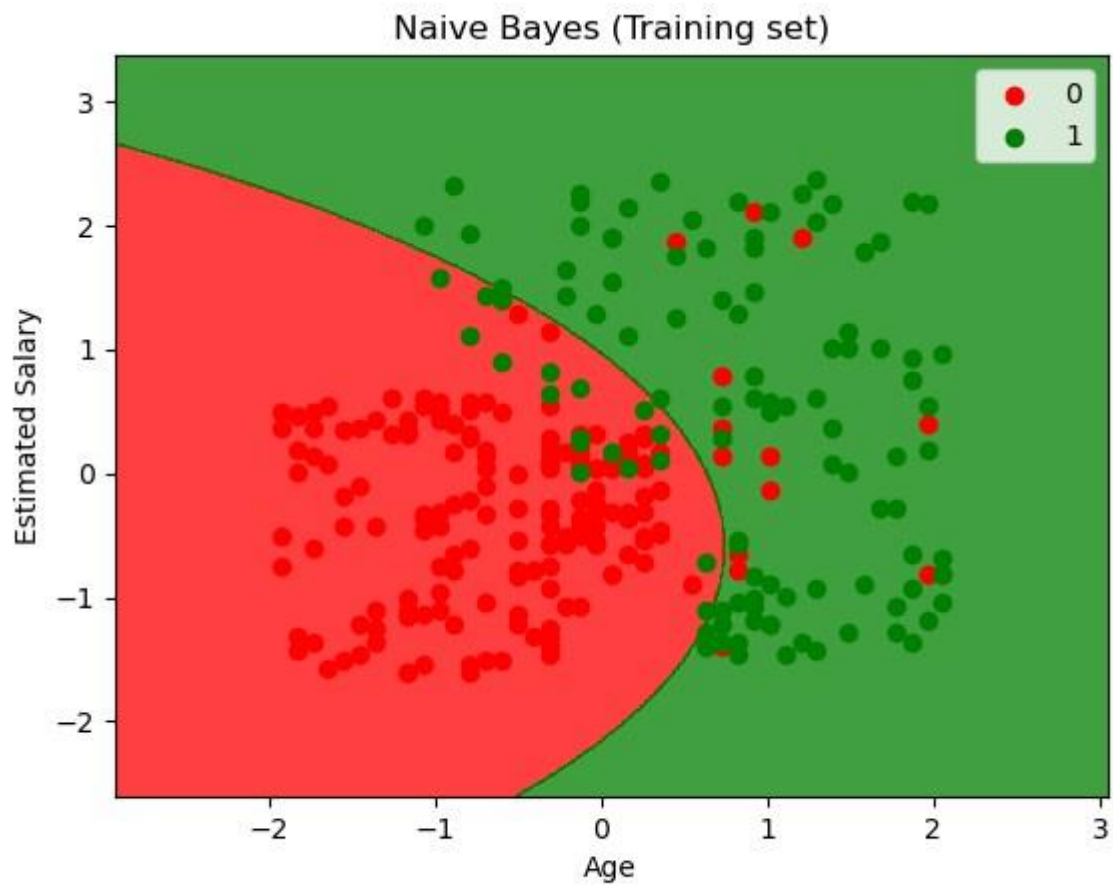
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                    np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
            alpha = 0.75, cmap = ListedColormap(('red', 'green')))

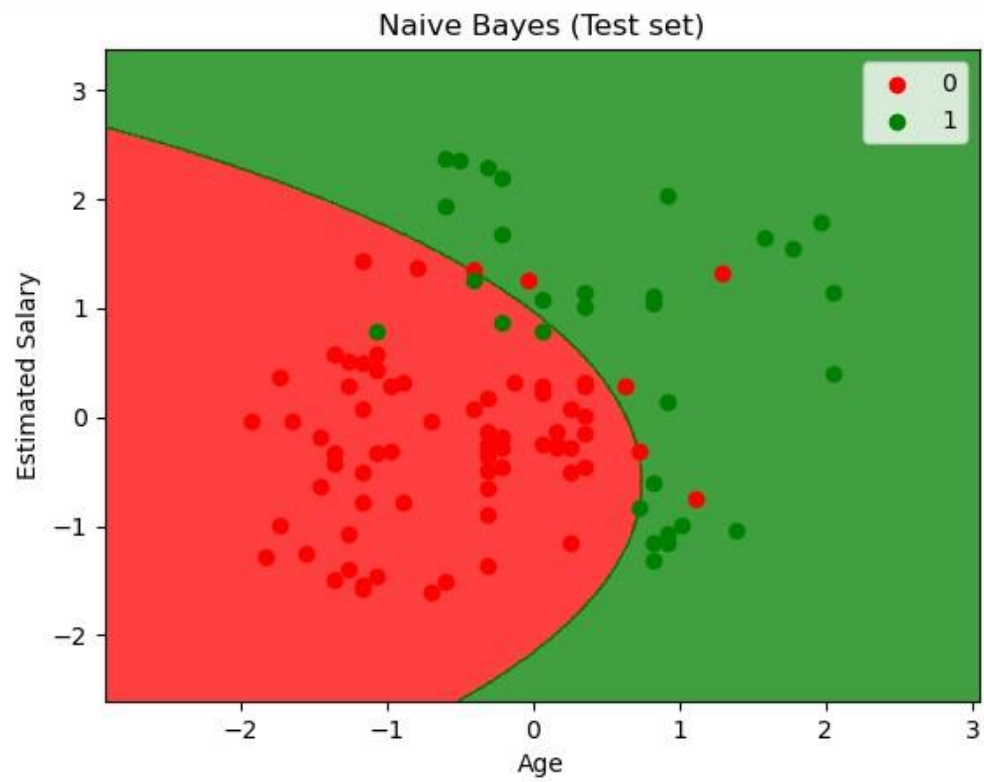
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
```

```
plt.title('Naive Bayes (Test set)')  
plt.xlabel('Age')  
plt.ylabel('Estimated Salary')  
plt.legend()  
plt.show()
```

## OUTPUT



Uttam - 53004230029



Uttam - 53004230029