MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

IT5612 - DATA ANALYTICS AND CLOUD COMPUTING LABORATORY

RECORD

REGISTER NUMBER: 2019506026

NAME: DEEPSHIKA RAGHURAMAN

SEMESTER: VI

DEPARTMENT OF INFORMATION TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHROMEPET, CHENNAI – 600 044

BONAFIDE CERTIFICATE

Certified that the bonafide record of the practical work done by

~~Mr~~./Ms. Deepshika Raghuraman Register Number 2019506026 of **Sixth** Semester, **B.Tech Information Technology** in the **IT5612 - Data Analytics and Cloud Computing Laboratory** during the academic period from **March 2022 to June 2022**

Submitted for Practical Examination held on _____

Date:

Course Instructor

S. Eliza Femi Sherley

Internal Examiner

External Examiner

TABLE OF CONTENTS

Date: 08.03.22, Tuesday **Exp. No.:** 01.a	**Study on Numpy, Scipy, Statsmodels and Pandas Packages**

1. Explore the functions in Numpy , SciPy , Statsmodels and pandas packages and prepare a study with an example for each function

Aim:

To study about the Python packages required to work with data analytics.

Theory:

a. Pandas

Pandas is an open-source library that is made mainly for working with relational and labelled data. It provides various data structures and operations for manipulating data.

Functions in Pandas

S. No	Description	Example
1.	read_csv( ) : This helps to read a csv [comma-separated-values] file into a pandas dataframe. It can also read files separated by delimiter	data_e = pd.read_csv (r'c:user_ds.csv')
2.	head( ) : The head(n) is used to return the first n rows of a dataset. By default, the function returns 5 rows of the dataframe	data_e.head(10)
3.	describe( ) : It is used to generate descriptive statistics of data in a pandas data frame or services	data_e.describe( )
4.	memory_usage( ) : It returns a pandas series having the memory usage of each column in a dataframe	data_e.memory_usage(deep = = true)
5.	loc[ ] : It helps to access a group of rows and columns in a dataset	data_e.loc[0:4, ['Name', 'Age', 'State']]

b. Numpy

Numpy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transformation and matrices.

Functions in Numpy

S. No	Description	Example
1.	numpy reshape : This function allows us to change the dimension of the array without hampering the data value	res = arr.reshape(3, 2)
2.	numpy.concatenate( ) : This function is used to join two arrays of same size, either in a row-wise or column-wise way	res = numpy.concatenate((arr1, arr2), axis =1)
3.	numpy.char.add( ) : It concatenates the data value of two arrays, merge them and represent a new array as a result	res = numpy.char.add([‘python’], [‘simple’])
4.	numpy.median( ) : It calculates the median of an ordered array	med = numpy.median(X)
5.	numpy.average( ) : It returns the average of all data values of the passed array	avg = numpy.average(X)

c. Scipy

Scipy contains a variety of sub packages which help to solve the most common issue related to scientific computation. It is built on top of the numpy library.

Functions in Scipy

S. No	Description	Example
1.	special.logsumexp(x) : log sum exponential computes the log of sum exponential input element	np.log(np.sum (np.exp(a)))
2.	linalg.det : linear algebra of scipy is an implementation of BLAS and ALAS LAPACK libraries. It accepts 2D arrays and gives a @D array	ar = np.array([4, 5], [5, 2]) linalg.det(ar)
3.	linalg.eig( ) : The most common problem in linear algebra is Eigen value and Eigen value	eg_val, eg_vec = linalg.eig(ar)
4.	Scipy.sparse : It is used for creating sparse matrix using multiple data structures	from scipy.sparse import csr_matrix ar = np.array([0, 0], [1, 1]) print(csr_matrix(ar).data)

d. Stats Model

It provides classes and functions for the estimation of many different statistical models for conducting statistical test and statistical data exploration.

Functions in Scipy

S. No	Description	Example
1.	get_rdataset : It is used to download any dataset we want	data = sm.datasets. get_rdataset("Guerry", "Hist").data
2.	add_constant(X) : It is used to add a constant column to input dataset	X = sm.addconstant(X)
3.	OLS(y, x).fit( ) : It is a type of linear square method for estimating unknown parameters in linear regression	res = sm.OLS(y, x).fit( )
4.	linear_rainbow( ) : The null hypothesis is the fit of the model using full sample. It is the same as using a central subset. Rainbow test has power against many different forms of non-linearity	print(sm.stats.linear_rainbow.—doc--)

Result:

Thus, a study on the Python packages used to work with data analysis has been made.

----- X -----

Date: 08.03.22, Tuesday **Exp. No.:** 01.b	**Read Data/Dataset**

1. Using Inbuilt packages – Read data from text file, Excel and the web

Aim:

To write python programs to read and display content from text file, csv file and web.

Codes and Output:

a. Reading from Text File

Code:

```
lines = [ ]
with open('sample.txt') as f:
    lines = f.readlines()
count = 0
for line in lines:
    count + = 1
    print(f 'line {count}: {line}')
```

Output:

```
line 1: Hello

line 2: Welcome to lab
```

b. Reading from CSV File

Code:

```
import csv
with open('mental-health-survey.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

Output:

```
['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history', 'treatment', 'work_interfere', 'no_empl
oyees', 'remote_work', 'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_help', 'anonymity', 'leave', 'me
ntal_health_consequence', 'phys_health_consequence', 'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_inter
view', 'mental_vs_physical', 'obs_consequence', 'comments']
['27-08-2014 11:29', '37', 'Female', 'United States', 'IL', 'NA', 'No', 'Yes', 'Often', 'Jun-25', 'No', 'Yes', 'Yes', 'Not su
re', 'No', 'Yes', 'Yes', 'Somewhat easy', 'No', 'No', 'Some of them', 'Yes', 'No', 'Maybe', 'Yes', 'No', 'NA']
['27-08-2014 11:29', '44', 'M', 'United States', 'IN', 'NA', 'No', 'No', 'Rarely', 'More than 1000', 'No', 'No', "Don't kno
w", 'No', "Don't know", "Don't know", "Don't know", "Don't know", 'Maybe', 'No', 'No', 'No', 'No', 'No', "Don't know", 'No',
'NA']
['27-08-2014 11:29', '32', 'Male', 'Canada', 'NA', 'NA', 'No', 'No', 'Rarely', 'Jun-25', 'No', 'Yes', 'No', 'No', 'No', 'No',
"Don't know", 'Somewhat difficult', 'No', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'NA']
['27-08-2014 11:29', '31', 'Male', 'United Kingdom', 'NA', 'NA', 'Yes', 'Yes', 'Often', '26-100', 'No', 'Yes', 'No', 'Yes',
'No', 'No', 'No', 'Somewhat difficult', 'Yes', 'Yes', 'Some of them', 'No', 'Maybe', 'Maybe', 'No', 'Yes', 'NA']
['27-08-2014 11:30', '31', 'Male', 'United States', 'TX', 'NA', 'No', 'No', 'Never', '100-500', 'Yes', 'Yes', 'Yes', 'No', "D
on't know", "Don't know", "Don't know", "Don't know", 'No', 'No', 'Some of them', 'Yes', 'Yes', 'Yes', "Don't know", 'No', 'N
A']
['27-08-2014 11:31', '33', 'Male', 'United States', 'TN', 'NA', 'Yes', 'No', 'Sometimes', 'Jun-25', 'No', 'Yes', 'Yes', 'Not
sure', 'No', "Don't know", "Don't know", "Don't know", 'No', 'No', 'Yes', 'Yes', 'No', 'Maybe', "Don't know", 'No', 'NA']
['27-08-2014 11:31', '35', 'Female', 'United States', 'MI', 'NA', 'Yes', 'Yes', 'Sometimes', '01-May', 'Yes', 'Yes', 'No', 'N
```

c. Reading from Web

Code:

```
import requests
from bs4 import BeautifulSoup
r = requests.get("https://www.mitindia.edu/en/")
soup = BeautifulSoup (r.content, 'html.parser')
lines = soup.find_all('p')
for line in lines:
    print(line.text)
```

Output:

```
In 1949,Shri.C.Rajam, gave the newly independent India-Madras Institute of Technology, so that MIT could establish the strong t
echnical base it needed to take its place in the world. It was the rare genius and daring of its founder that made MIT offer co
urses like Aeronautical Engineering, Automobile Engineering, Electronics Engineering and Instrument Technology for the first ti
me in our country. Now it also provides technical education in other engineering fields such as Rubber and Plastic Technology &
Production Technology. It was merged with Anna University in the year 1978. Sixty years hence, while it continues to be a pione
er in courses that it gave birth to, it is already renowned for producing the crème de la crème of the scientific community in
more nascent courses such as Computer Science and Information Technology MIT has produced great scientist like Dr.A.P.J.Abdul K
alam, versatile genius like Sujatha and many more. The broad-based education, coupled with practice-oriented training in their
speciality, has enabled the students of MIT to handle with skill and success a wide variety of technical problems. The Madras I
nstitute of Technology has developed into an important centre of engineering education and earned an excellent reputation both
in India and abroad.MIT had received many awards which includes an award for the Best Overall Performance, awarded by Indian So
ciety of Technical Education (ISTE) during the year 1999.
Copyright © 2016. All Rights Reserved. Commercial use and distribution of the contents of the website is not allowed without ex
press and prior written consent of Madras Institute of Technology.No part of this website may be reproduced without Prior Notic
e.
Designed And Maintained by WebTeam MitIndia
```

Result:

Thus, the Python programs to read and display content from text file, CSV file, and web have been written and verified.

----- X -----

Date: 15.03.22, Tuesday
Exp. No.: 02

Descriptive Analysis

1. Using Iris Dataset-Explore various commands for doing descriptive analytics

Aim:

To explore the various commands for performing descriptive data analysis on the Iris Dataset.

Codes and Output:

i. Import the necessary packages and dataset

```
import pandas as pd
df = pd.read_csv("Iris.csv")
```

ii. head( )

```
df.head( )
```

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

iii. Getting Information about the dataset

- Shape parameter

```
df.shape
```

Output:

```
df.shape
(150, 5)
```

- info( ) method

```
df.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

- describe( ) method

```
df.describe( )
```

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

iv. Checking Missing Values

- isnull( ) method

```
df.isnull().sum()
```

Output:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

v. Checking Duplicates

- drop_duplicates( )

```
data = df.drop_duplicates(subset ="Species",)
data
```

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
100	101	6.3	3.3	6.0	2.5	Iris-virginica

- Series.value_counts( )

```
df.value_counts("Species")
```

Output:

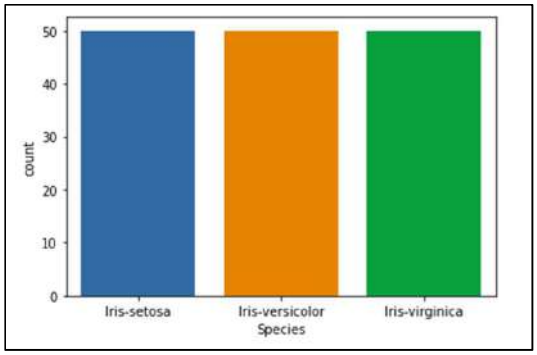
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

vi. Data Visualization

- Visualize Target Column using Countplot

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='Species', data=df, )
plt.show()
```

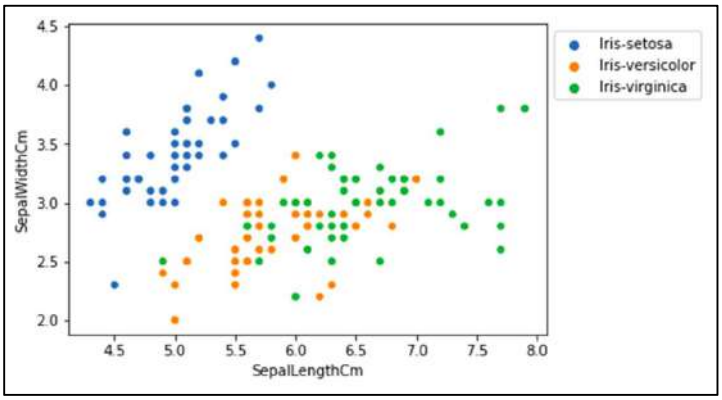
Output:



- Relation between Variables using Scatterplot

```
sns.scatterplot(x='SepalLengthCm', y='SepalWidthCm', hue='Species', data=df, )
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

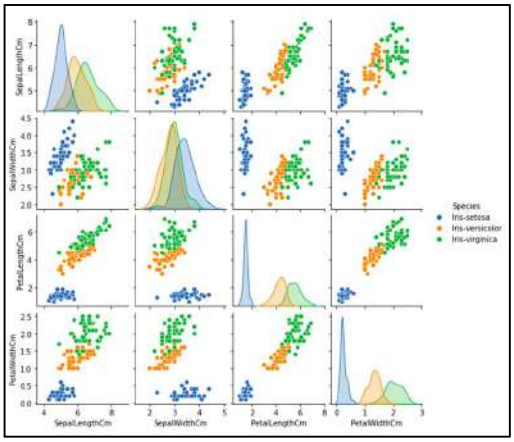
Output:



➢ Relationship of all Columns using Pairplot

```
sns.pairplot(df.drop(['Id'], axis = 1), hue='Species', height=2)
```

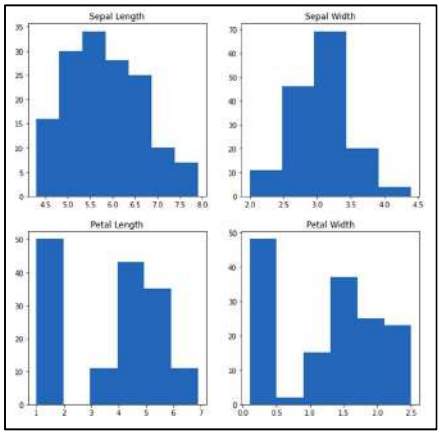
Output:



➢ Histogram

```
fig, axes = plt.subplots(2, 2, figsize=(10,10))
axes[0,0].set_title("Sepal Length")
axes[0,0].hist(df['SepalLengthCm'], bins=7)
axes[0,1].set_title("Sepal Width")
axes[0,1].hist(df['SepalWidthCm'], bins=5);
axes[1,0].set_title("Petal Length")
axes[1,0].hist(df['PetalLengthCm'], bins=6);
axes[1,1].set_title("Petal Width")
axes[1,1].hist(df['PetalWidthCm'], bins=6);
```

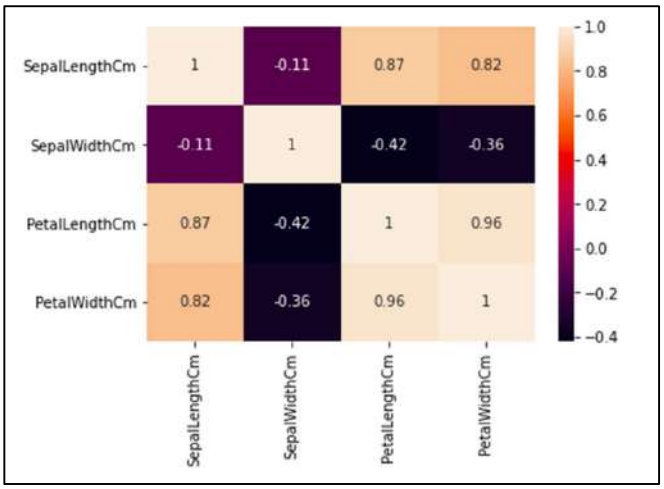
Output:



➢ Heatmap

```
sns.heatmap(df.corr(method='pearson').drop( ['Id'], axis=1).drop(['Id'], axis=0),
        annot = True);
plt.show()
```

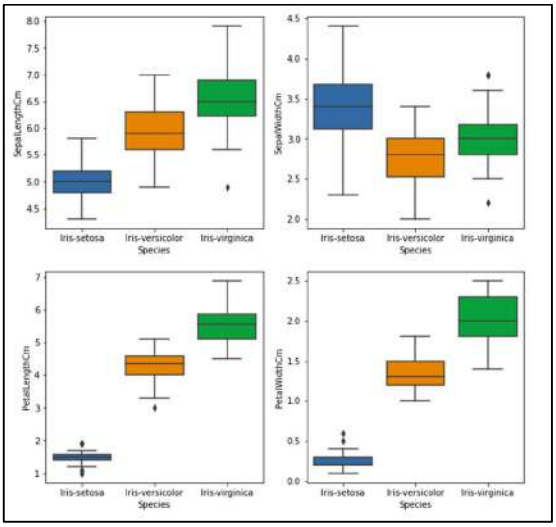
Output:



➢ Boxplots

```
def graph(y):
   sns.boxplot(x="Species", y=y, data=df)
plt.figure(figsize=(10,10))
plt.subplot(221)
graph('SepalLengthCm')
plt.subplot(222)
graph('SepalWidthCm')
plt.subplot(223)
graph('PetalLengthCm')
plt.subplot(224)
graph('PetalWidthCm')
plt.show()
```

Output:

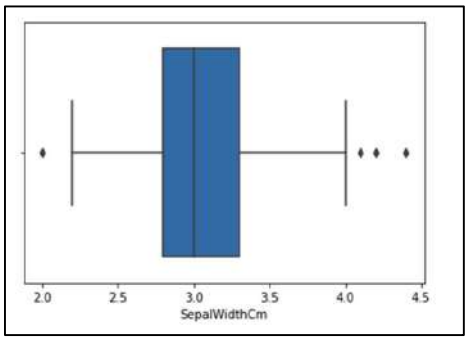


vii. Handling Outliers

- Boxplot for SepalWidthCm Column

```
sns.boxplot(x='SepalWidthCm', data=df)
```

Output:

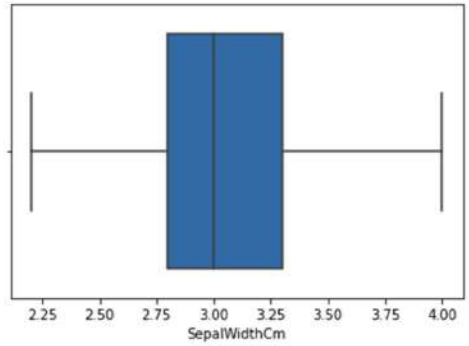


- Removing Outliers

```
import sklearn
from sklearn.datasets import load_boston
import pandas as pd
import seaborn as sns

df = pd.read_csv('Iris.csv')
Q1 = np.percentile(df['SepalWidthCm'], 25, interpolation = 'midpoint')
Q3 = np.percentile(df['SepalWidthCm'], 75, interpolation = 'midpoint')
IQR = Q3 - Q1
print("Old Shape: ", df.shape)
upper = np.where(df['SepalWidthCm'] >= (Q3+1.5*IQR))
lower = np.where(df['SepalWidthCm'] <= (Q1-1.5*IQR))
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)
print("New Shape: ", df.shape)
sns.boxplot(x='SepalWidthCm', data=df)
```

Output:

```
Old Shape:  (150, 6)
New Shape:  (146, 6)

<AxesSubplot:xlabel='SepalWidthCm'>
```



Result:

Thus, the various commands for performing descriptive data analysis on the Iris Dataset have been executed and verified.

2. Using Unique dataset assigned to you explore the commands in descriptive analytics

Aim:

To explore the various commands for performing descriptive data analysis on the given dataset.

Codes and Output:

i. Import the necessary packages and dataset

```
import pandas as pd
df = pd.read_csv('mental-health-survey.csv')
```

ii. head( )

```
df.head( )
```

Output:

Out[2]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequen
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25	...	Somewhat easy	
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	Ma
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25	...	Somewhat difficult	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	\
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	

5 rows × 27 columns

iii. Getting Information about the dataset

- Shape parameter

```
df.shape
```

Output:

```
In [71]: df.shape
Out[71]: (1256, 27)
```

- info( ) method

```
df.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1256 entries, 0 to 1255
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Timestamp                  1256 non-null   object
 1   Age                        1256 non-null   int64
 2   Gender                     1256 non-null   object
 3   Country                    1256 non-null   object
 4   state                      743 non-null    object
 5   self_employed              1238 non-null   object
 6   family_history             1256 non-null   object
 7   treatment                  1256 non-null   object
 8   work_interfere             992 non-null    object
 9   no_employees               1256 non-null   object
 10  remote_work                1256 non-null   object
 11  tech_company               1256 non-null   object
 12  benefits                   1256 non-null   object
 13  care_options               1256 non-null   object
 14  wellness_program           1256 non-null   object
 15  seek_help                  1256 non-null   object
 16  anonymity                  1256 non-null   object
 17  leave                      1256 non-null   object
 18  mental_health_consequence  1256 non-null   object
 19  phys_health_consequence    1256 non-null   object
 20  coworkers                  1256 non-null   object
 21  supervisor                 1256 non-null   object
 22  mental_health_interview    1256 non-null   object
 23  phys_health_interview      1256 non-null   object
 24  mental_vs_physical         1256 non-null   object
 25  obs_consequence            1256 non-null   object
 26  comments                   163 non-null    object
dtypes: int64(1), object(26)
memory usage: 265.1+ KB
```

- describe( ) method

```
df.describe( )
```

Output:

Out[18]:

	Age
count	1219.000000
mean	31.468417
std	6.276310
min	18.000000
25%	27.000000
50%	31.000000
75%	35.000000
max	49.000000

iv. Checking Missing Values

- isnull( ) method

```
df.isnull().sum()
```

Output:

```
Out[7]: Timestamp                      0
        Age                            0
        Gender                         0
        Country                        0
        state                        515
        self_employed                 18
        family_history                 0
        treatment                      0
        work_interfere               264
        no_employees                   0
        remote_work                    0
        tech_company                   0
        benefits                       0
        care_options                   0
        wellness_program               0
        seek_help                      0
        anonymity                      0
        leave                          0
        mental_health_consequence      0
        phys_health_consequence        0
        coworkers                      0
        supervisor                     0
        mental_health_interview        0
        phys_health_interview          0
        mental_vs_physical             0
        obs_consequence                0
        comments                    1095
        dtype: int64
```

v. Checking Duplicates

- drop_duplicates( )

```
data = df.drop_duplicates(subset ="Country",)
data
```

Output:

Out[18]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	menta
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25	...	Somewhat easy	
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25	...	Somewhat difficult	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	
11	2014-08-27 11:32:49	29	male	Bulgaria	NaN	NaN	No	No	Never	100-500	...	Don't know	
19	2014-08-27 11:35:08	36	Male	France	NaN	Yes	Yes	No	NaN	6-25	...	Somewhat easy	
37	2014-08-27 11:41:50	38	Male	Portugal	NaN	No	No	No	NaN	100-500	...	Somewhat easy	
43	2014-08-27 11:43:10	18	Male	Netherlands	NaN	No	No	No	Often	6-25	...	Somewhat difficult	

- Series.value_counts( )

```
df.value_counts("Country")
```

Output:

Out[21]:

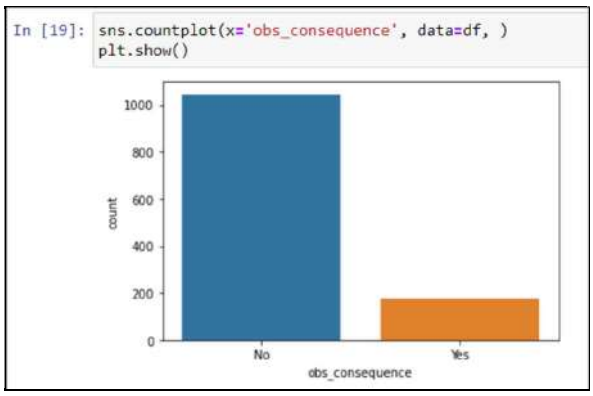
```
Country
United States              751
United Kingdom             185
Canada                      72
Germany                     45
Netherlands                 27
Ireland                     27
Australia                   21
France                      13
India                       10
New Zealand                  8
Italy                        7
Sweden                       7
Poland                       7
Switzerland                  7
South Africa                 6
Brazil                       6
Belgium                      6
Israel                       5
Bulgaria                     4
Singapore                    4
Russia                       3
Mexico                       3
Finland                      3
Austria                      3
Colombia                     2
Denmark                      2
Greece                       2
Portugal                     2
Croatia                      2
Spain                        1
Slovenia                     1
Uruguay                      1
Romania                      1
Thailand                     1
Japan                        1
Philippines                  1
Norway                       1
Nigeria                      1
Moldova                      1
Latvia                       1
Hungary                      1
Georgia                      1
Czech Republic               1
Costa Rica                   1
China                        1
Bosnia and Herzegovina       1
Bahamas, The                 1
Zimbabwe                     1
dtype: int64
```

vi. Data Visualization

➢ Visualize a Column using Countplot

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x=obs_consequence, data=df, )
plt.show()
```

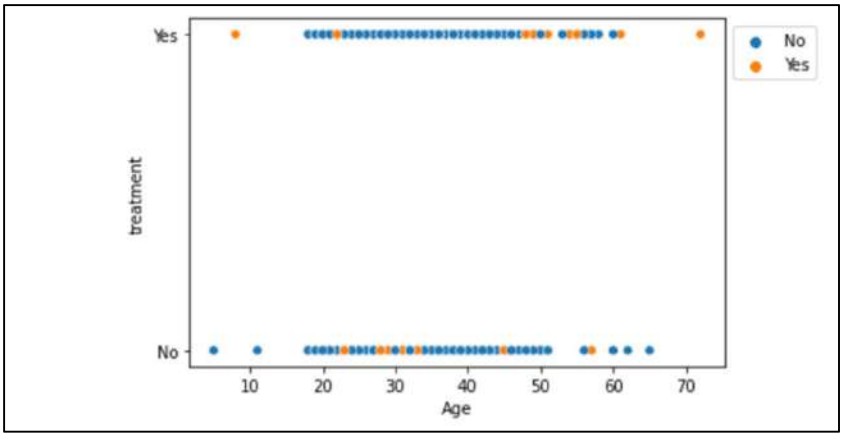
Output:



➢ Visualize Relation between Variables using Scatterplot

```
sns.scatterplot(x='Age', y='treatment', hue='obs_consequence', data=df, )
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

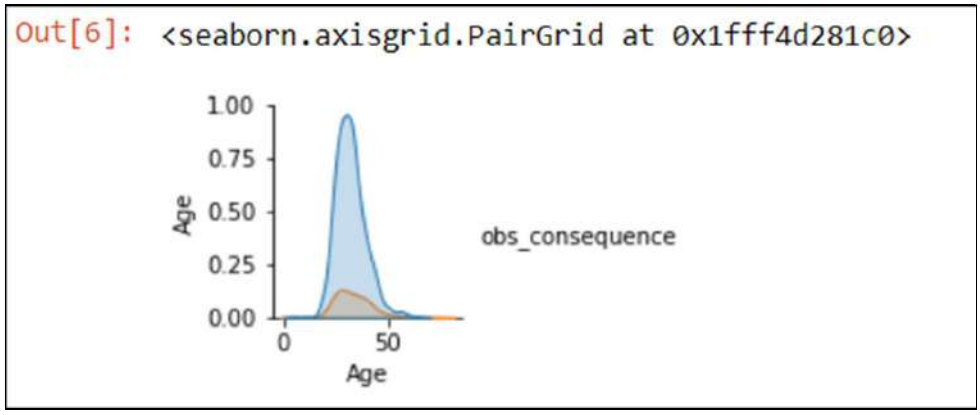
Output:



- Relationship of all Columns using Pairplot

```
sns.pairplot(df.drop(['Timestamp'], axis = 1), hue='obs_consequence', height=2)
```

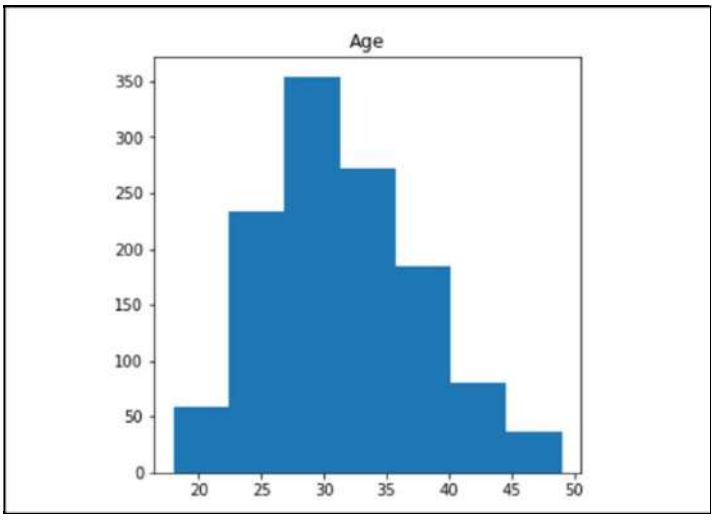
Output:



- Histogram

```
fig, axes = plt.subplots(2, 2, figsize=(10,10))
axes[0,0].set_title("Age")
axes[0,0].hist(df['Age'], bins=7)
```

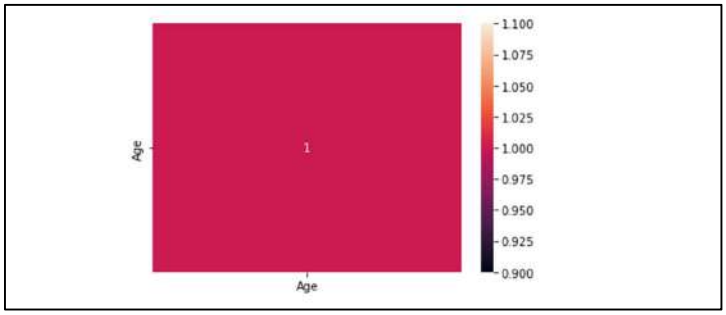
Output:



➢ Heatmap

```
sns.heatmap(df.corr(method='pearson'), annot = True);
plt.show()
```

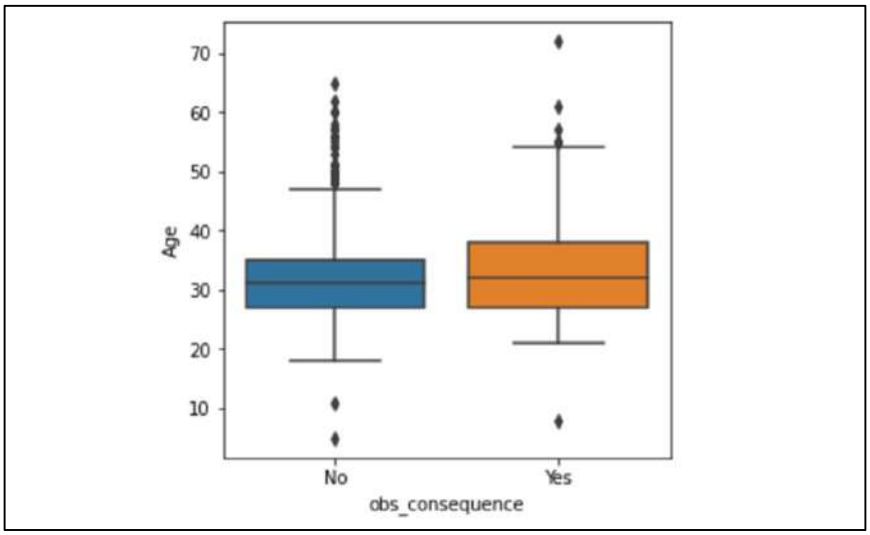
Output:



➢ Boxplots

```
def graph(y):
   sns.boxplot(x="obs_consequence", y=y, data=df)
plt.figure(figsize=(10,10))
plt.subplot(221)
graph('Age')
```

Output:

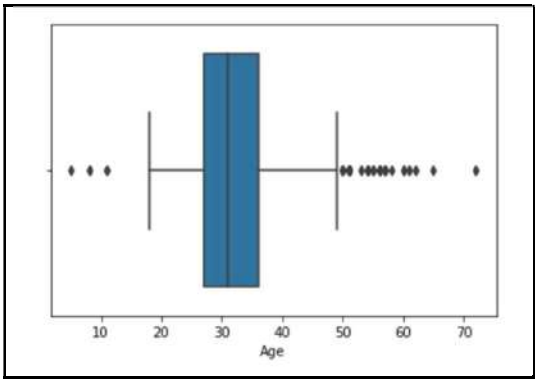


viii. Handling Outliers

- Boxplot for Age Column

```
sns.boxplot(x='Age', data=df)
```

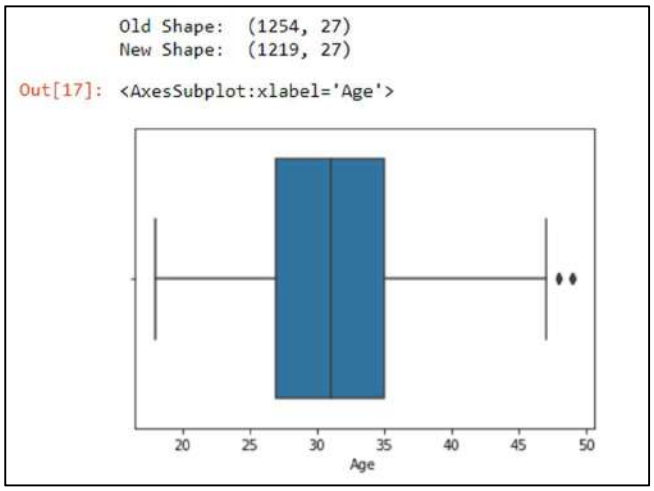
Output:



- Removing Outliers

```
import sklearn
from sklearn.datasets import load_boston
import pandas as pd
import seaborn as sns

Q1 = np.percentile(df['Age'], 25, interpolation = 'midpoint')
Q3 = np.percentile(df['Age'], 75, interpolation = 'midpoint')
IQR = Q3 - Q1
print("Old Shape: ", df.shape)
upper = np.where(df['Age'] >= (Q3+1.5*IQR))
lower = np.where(df['Age'] <= (Q1-1.5*IQR))
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)
print("New Shape: ", df.shape)
sns.boxplot(x='Age', data=df)
```

Output:



Result:

Thus, the various commands for performing descriptive data analysis on the given dataset have been executed and verified.

3. From the Unique dataset picked by you from UCI/Kaggle perform the following

i. Find correlation between features and visualize it
ii. Perform dimensionality reduction
iii. Detect outliers
iv. Fill missing data
v. Data Partitioning
vi. Perform the following statistical test for the dataset
- Normality test
- Coorelation test
- Parametric statistiacl Hypothesis test
- Nonparametric statistical Hypothesis test

Aim:

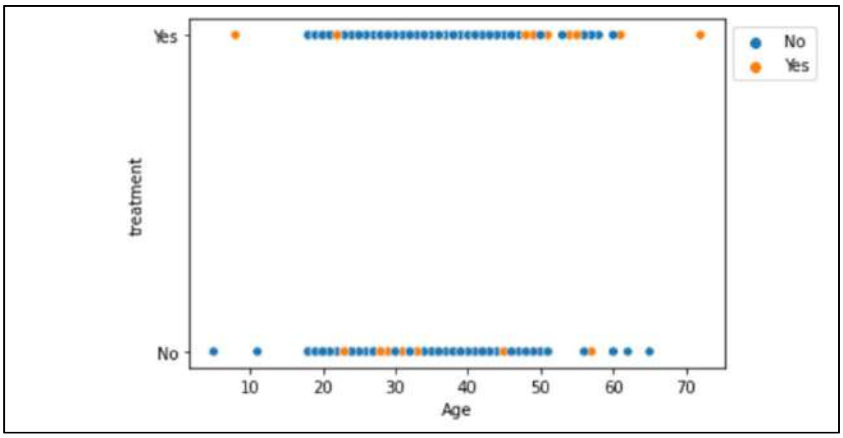
To perform the given analysis measures on the chosen dataset.

Codes and Output:

i. Find correlation between features and visualize it

```
sns.scatterplot(x='Age', y='treatment', hue='obs_consequence', data=df, )
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

Output:



ii. Perform dimensionality reduction

```
df.isnull().sum()/len(df)*100
```

```
Out[8]: Timestamp                    0.000000
        Age                          0.000000
        Gender                       0.000000
        Country                      0.000000
        state                       40.905481
        self_employed                1.429706
        family_history               0.000000
        treatment                    0.000000
        work_interfere              20.969023
        no_employees                 0.000000
        remote_work                  0.000000
        tech_company                 0.000000
        benefits                     0.000000
        care_options                 0.000000
        wellness_program             0.000000
        seek_help                    0.000000
        anonymity                    0.000000
        leave                        0.000000
        mental_health_consequence    0.000000
        phys_health_consequence      0.000000
        coworkers                    0.000000
        supervisor                   0.000000
        mental_health_interview      0.000000
        phys_health_interview        0.000000
        mental_vs_physical           0.000000
        obs_consequence              0.000000
        comments                    86.973789
        dtype: float64
```

Remove columns that have more than 80% values missing

```
a = df.isnull().sum()/len(df)*100
variables = df.columns
variable = [ ]
for i in range(0,26):
    if a[i]<=80:   #setting the threshold as 80%
        variable.append(variables[i])
c = [ ]
data = df
for i in variables:
    if i not in variable:
        c.append(i)
for i in c:
    data = df.drop([i], axis = 1)
data.info
```

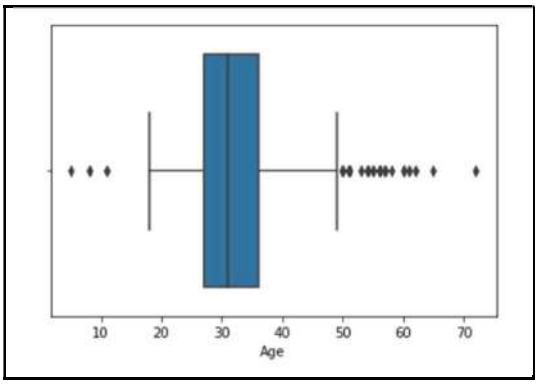
Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 26 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Timestamp                  1259 non-null   object
 1   Age                        1259 non-null   int64
 2   Gender                     1259 non-null   object
 3   Country                    1259 non-null   object
 4   state                      744 non-null    object
 5   self_employed              1241 non-null   object
 6   family_history             1259 non-null   object
 7   treatment                  1259 non-null   object
 8   work_interfere             995 non-null    object
 9   no_employees               1259 non-null   object
 10  remote_work                1259 non-null   object
 11  tech_company               1259 non-null   object
 12  benefits                   1259 non-null   object
 13  care_options               1259 non-null   object
 14  wellness_program           1259 non-null   object
 15  seek_help                  1259 non-null   object
 16  anonymity                  1259 non-null   object
 17  leave                      1259 non-null   object
 18  mental_health_consequence  1259 non-null   object
 19  phys_health_consequence    1259 non-null   object
 20  coworkers                  1259 non-null   object
 21  supervisor                 1259 non-null   object
 22  mental_health_interview    1259 non-null   object
 23  phys_health_interview      1259 non-null   object
 24  mental_vs_physical         1259 non-null   object
 25  obs_consequence            1259 non-null   object
dtypes: int64(1), object(25)
memory usage: 255.9+ KB
```

iii. Detect Outliers

```
sns.boxplot(x='Age', data=df)
```

Output:



iv. Fill Missing Data

```
data = df.fillna(0)
data.isnull( ).sum( )/len(data)*100
```

Output:

```
Out[6]: Timestamp                    0.0
        Age                          0.0
        Gender                       0.0
        Country                      0.0
        state                        0.0
        self_employed                0.0
        family_history               0.0
        treatment                    0.0
        work_interfere               0.0
        no_employees                 0.0
        remote_work                  0.0
        tech_company                 0.0
        benefits                     0.0
        care_options                 0.0
        wellness_program             0.0
        seek_help                    0.0
        anonymity                    0.0
        leave                        0.0
        mental_health_consequence    0.0
        phys_health_consequence      0.0
        coworkers                    0.0
        supervisor                   0.0
        mental_health_interview      0.0
        phys_health_interview        0.0
        mental_vs_physical           0.0
        obs_consequence              0.0
        comments                     0.0
        dtype: float64
```

v. Data Partitioning

```
out_arr = np.partition(data.columns, 3)
out_arr
```

Output:

```
Out[53]: array(['Age', 'Country', 'Gender', 'Timestamp', 'anonymity', 'benefits',
                'care_options', 'coworkers', 'family_history', 'leave',
                'mental_health_consequence', 'mental_health_interview',
                'mental_vs_physical', 'no_employees', 'obs_consequence',
                'phys_health_consequence', 'phys_health_interview', 'remote_work',
                'seek_help', 'self_employed', 'state', 'supervisor',
                'tech_company', 'treatment', 'wellness_program', 'work_interfere'],
               dtype=object)
```

vi. Statistical Tests

- Normality Test

```
#K-squared test
from scipy import stats
print(['AGE'])
a, b =  stats.normaltest(data[['Age']])
print(a, b)
alpha = 0.05
if b < alpha:
   print("Null hypotheses can be rejected")
else:
   print("Null hypotheses cannot be rejected")
```

Output:

```
['AGE']
[3578.76220511] [0.]
Null hypotheses can be rejected
```

- Correlation Test

```
#one-hot encoding data and performing correlation test
encData = pd.get_dummies(df, columns = ['obs_consequence'])
from scipy.stats import pearsonr
param1 = encData['Age']
param2 = encData['obs_consequence_Yes']
corr, p = pearsonr(param1, param2)
print('Pearsons correlation: %.3f, p = %.3f' % (corr,p))
```

Output:

```
Pearsons correlation: 0.066, p = 0.020
```

- Parametric statistical Hypothesis test

```
#T-Test
from scipy.stats import ttest_ind
param1 = encData['Age']
param2 = encData['obs_consequence_Yes']
stat, p = ttest_ind(param1, param2)
print(Statistics: %.3f, p = %.3f' % (stat,p))
```

Output:

```
Statistics: 152.872, p = 0.000
```

- Non-Parametric statistical Hypothesis test

```
#Mann-Whitney U Test
from scipy.stats import mannwhitneyu
param1 = encData['Age']
param2 = encData['obs_consequence_Yes']
stat, p = mannwhitneyu (param1, param2)
print(Statistics: %.3f, p = %.3f' % (stat,p))
```

Output:

```
Statistics: 0.000, p = 0.000
```

Result:

Thus, the given analysis measures have been performed on the given dataset and have been verified.

----- X -----

Date: 22.03.22, Tuesday
Exp. No.: 02

Univariate Analysis

1. Use the diabetes data set from UCI and Pima Indians Diabetes data set and perform the following: Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis

Aim:

To perform univariate analysis on the Pima Indians Diabetes dataset.

Codes and Output:

i. Frequency

```
import pandas as pd
df = pd.read_csv("diabetes.csv")
df['Outcome'].value_counts()
```

Output:

```
0    500
1    268
Name: Outcome, dtype: int64
```

ii. Mean

```
df['BloodPressure'].mean()
```

Output:

```
69.10546875
```

iii. Median

```
df['BloodPressure'].median()
```

Output:

```
72.0
```

iv. Mode

```
df['BloodPressure'].mode()
```

Output:

```
0    70
dtype: int64
```

v. Variance

```
df['BloodPressure'].var()
```

Output:

```
374.6472712271838
```

vi. Standard Deviation

```
df['BloodPressure'].std()
```

Output:

```
19.355807170644777
```

vii. Skewness

```
from scipy.stats import skew
import numpy as np
import pylab as p

y1 = df['Outcome']
print("Skewness = ", skew(y1))
```

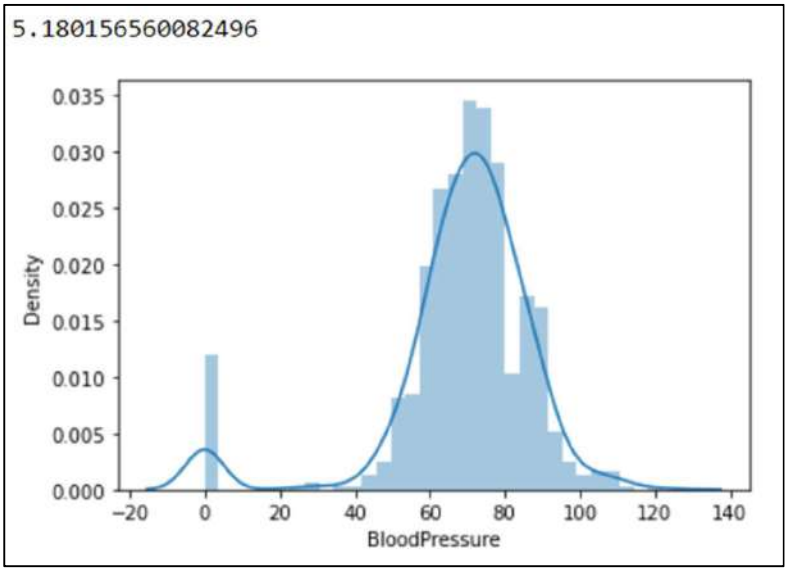
Output:

```
Skewness =  0.6337757030614577
```

viii. Kurtosis

```
import seaborn as sns
sns.distplot(df['BloodPressure'], hist=True, kde=True)
df['BloodPressure'].kurt()
```

Output:



Result:

Thus, univariate analysis on the Pima Indians Diabetes dataset has been performed and verified.

2. Use Unique dataset assigned to you and perform the following Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis

Aim:

To perform univariate analysis on the chosen dataset.

Codes and Output:

i. Frequency

```
import pandas as pd
df = pd.read_csv("mental-health-survey.csv")
df['obs_consequence'].value_counts()
```

Output:

```
No     1072
Yes     182
Name: obs_consequence, dtype: int64
```

ii. Mean

```
df['Age'].mean()
```

Output:

```
32.01913875598086
```

iii. Median

```
df['Age'].median()
```

Output:

```
31.0
```

iv. Mode

```
df['Age'].mode()
```

Output:

```
0    29
dtype: int64
```

v. Variance

```
df['Age'].var()
```

Output:

```
54.39069486820141
```

vi. Standard Deviation

```
df['Age'].std()
```

Output:

```
7.375004736825693
```

vii. Skewness

```
encD = pd.get_dummies(df, columns = [‘obs_consequence’])
from scipy.stats import skew
import numpy as np
import pylab as p
y1 = encD[‘obs_consequence_Yes’]
print("Skewness for data : ", skew(y1))
```

Output:

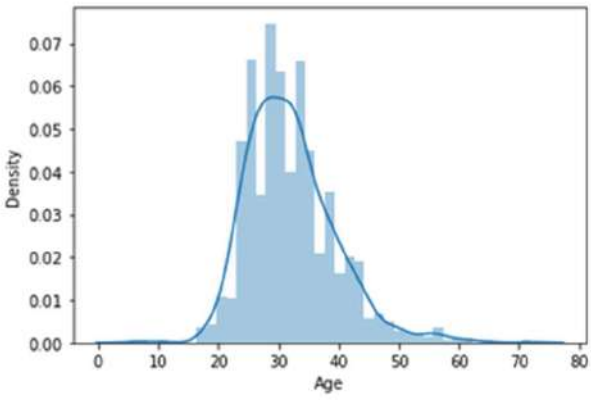
```
Skewness for data :  2.0149158814980037
```

viii. Kurtosis

```
import seaborn as sns
sns.distplot(encD['Age'], hist=True, kde=True)
encD['Age'].kurt()
```

Output:

```
1.9629042596291257
```



Result:

Thus, univariate analysis on the chosen dataset has been performed and verified.

3. From the Unique dataset picked by you from UCI/Kaggle perform the following:

- Univariate Linear Regression
- Summary statistics
- Frequency table
- Charts
- univariate analysis for numerical and categorical attributes
- Univariate Function Optimization
- Perform Feature Extraction with Univariate Statistical Tests and summarize the selected features

Aim:

To perform the various commands for univariate analysis on the given dataset.

Codes and Output:

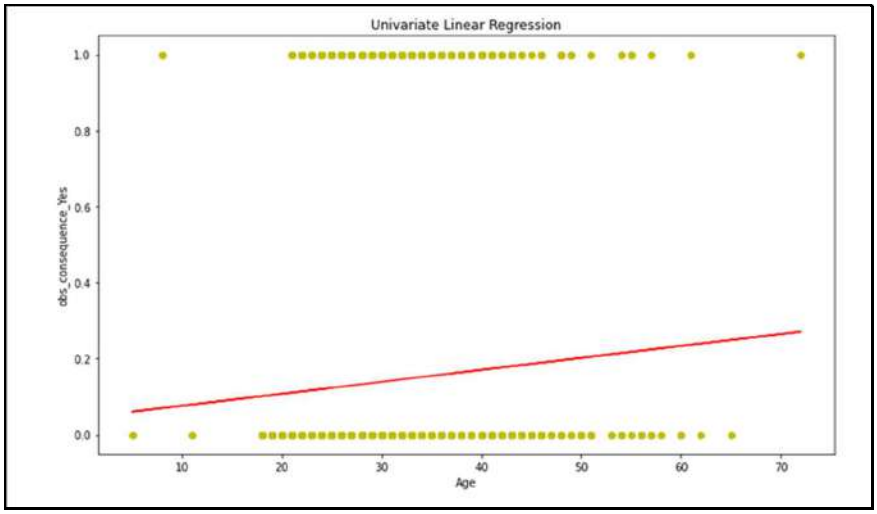
a. Univariate Linear Regression

```
x = encD['Age']
y = encD['obs_consequence_Yes']

def mean(vals):
    return sum(vals)/float(len(vals))

def variance(vals):
    val_mean = mean(vals)
    return sum([(x-val_mean)**2 for x in vals])

def covariance(x,x_mean,y,y_mean):
    cov = 0.0
    for i in range(len(x)):
        cov += (x[i]-x_mean) * (y[i]-y_mean)
    return cov

def getBetas(x,y):
    x_mean = mean(x)
    y_mean = mean(y)
    cov = covariance(x,x_mean,y,y_mean)
    x_var = variance(x)
    b1 = cov/x_var
    b0 = np.mean(y) - b1 * np.mean(x)
    return b0,b1

def predict(x,y):
    intercept,coeff = getBetas(x,y)
    preds = intercept+(coeff*x)
    return preds

intercept, coeff = getBetas(x,y)
print(f'Intercept: {intercept}\n\nCoefficient: {coeff}')
fcov_preds = predict(x,y)
```

```
#with plt.style.context('dark_background'):
plt.figure(figsize=(12,7))
plt.plot(x,y,'yo',x,fcov_preds,'r-',fillstyle='full')
plt.xlabel('Age')
plt.ylabel('obs_consequence_Yes')
plt.title('Univariate Linear Regression');
```

Output:

```
Intercept: 0.04435073577468868

Coefficient: 0.0031476433885868796
```



b. Summary Statistics

```
#Mean age based on whether patient gets treatment
df[ ['treatment', 'Age']].groupby('treatment').mean( )
```

Output:

treatment	Age
No	31.458937
Yes	32.568720

```
#Mean age based on treatment and obs_consequence
df.groupby(['treatment', 'obs_consequence'])['Age'].mean( )
```

Output:

```
treatment  obs_consequence
No         No                31.371681
           Yes               32.339286
Yes        No                32.317554
           Yes               33.579365
Name: Age, dtype: float64
```

c. Frequency Table

```
freq_table = pd.crosstab(df['treatment'], df['obs_consequence'])
freq_table
```

Output:

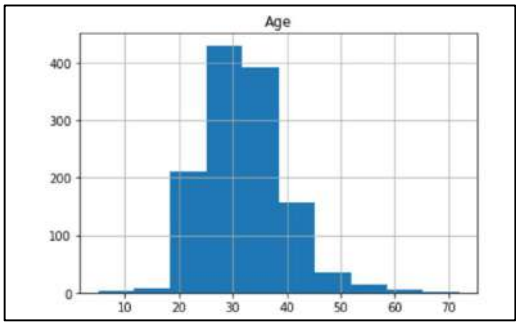
obs_consequence	No	Yes
treatment		
No	565	56
Yes	507	126

d. Charts

➢ Histogram

```
import matplotlib.pyplot as plt
dt = pd.DataFrame(df['Age'], columns = ['Age'])
dt.hist( )
plt.show( )
```

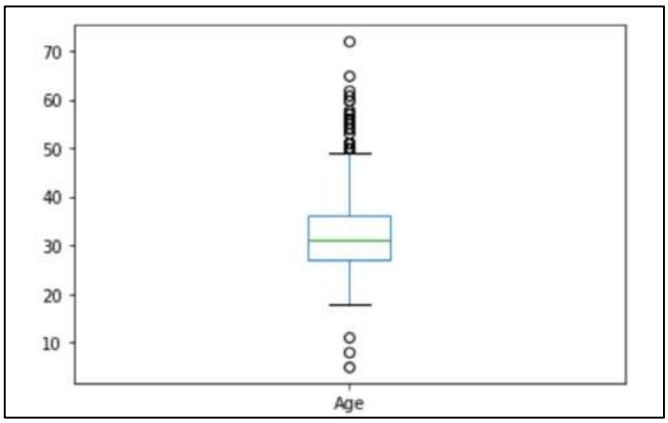
Output:



- Boxplot

```
df.plot.box( )
```

Output:

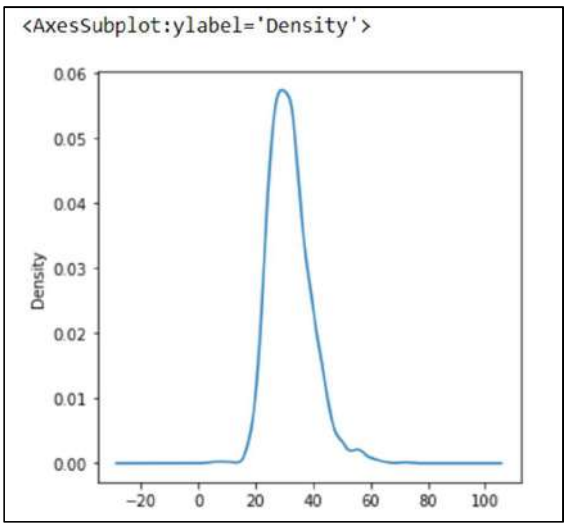


e. Univariate Analysis

- Numerical Attribute

```
plt.figure(figsize = (5, 5))
df['Age'].plot(kind = 'density')
```

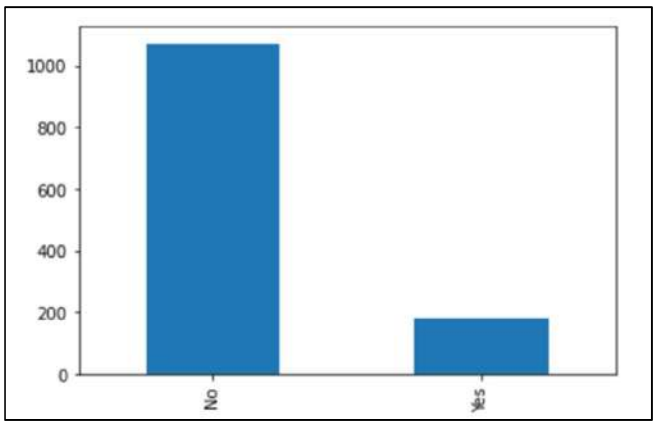
Output:



- Categorical Attribute

```
df['obs_consequence'].value_counts( ).plot( ).bar( )
```

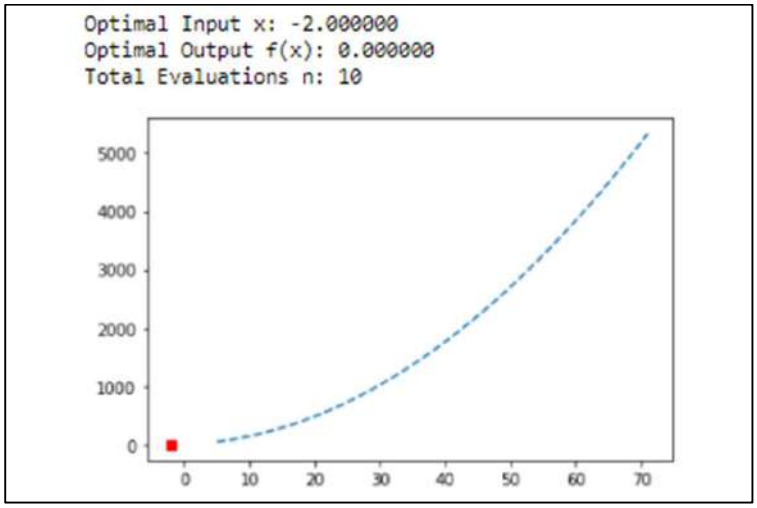
Output:



f. Univariate Function Optimization

```
from numpy import arange
from scipy.optimize import minimize_scalar
from matplotlib import pyplot

def objective(x):
        return (2.0 + x)**2.0
result = minimize_scalar(objective, method='brent')
opt_x, opt_y = result['x'], result['fun']
print('Optimal Input x: %.6f' % opt_x)
print('Optimal Output f(x): %.6f' % opt_y)
print('Total Evaluations n: %d' % result['nfev'])
r_min, r_max = df['Age'].min( ), df['Age'].max( )
inputs = arange(r_min, r_max, 1)
targets = [objective(x) for x in inputs]
pyplot.plot(inputs, targets, '--')
pyplot.plot([opt_x], [opt_y], 's', color='r')
pyplot.show()
```

Output:



Result:

Thus, the various commands for univariate analysis have been performed on the given dataset and have been verified.

----- X -----

Date: 29.03.22, Tuesday
Exp. No.: 02.b

Univariate Time Series Analysis

1. From the dataset you choose from UCI/Kaggle Repository, perform the following:

a. Create time series features from the dataset you imported
b. Plot feature importance
c. Forecast and Evaluate the data

Aim:

To perform univariate time series analysis on the given dataset.

Codes and Output:

- Import Packages and Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv("mental-health-survey.csv", parse_dates = [‘Timestamp’], index_col = [‘Timestamp’])
```

a. Time Series Features

- Mean in periods of 3 months

```
data = df.resample(‘3M’).mean( )
data
```

Output:

Timestamp	Age
2014-01-31	29.437500
2014-04-30	36.562500
2014-07-31	35.800000
2014-10-31	31.955420
2015-01-31	37.000000
2015-04-30	31.764706
2015-07-31	33.100000
2015-10-31	33.250000
2016-01-31	32.000000

- Mean in periods of 3 years

```
df.resample('3Y').mean( )
```

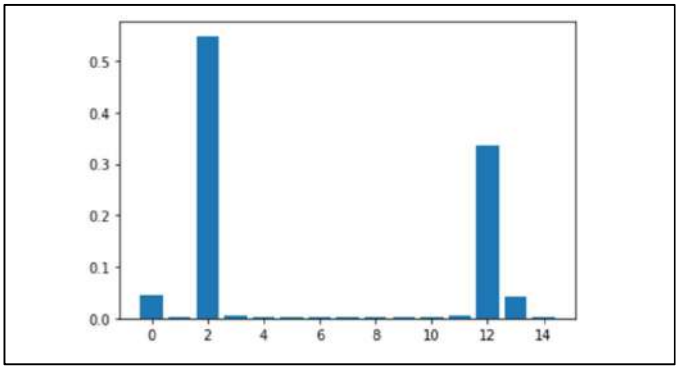
Output:

Timestamp	Age
2014-12-31	32.016878
2017-12-31	32.057971

b. Feature Importance

```
from sklearn.datasets import make_regression
from sklearn.tree import DecisionTreeRegressor
X, y = make_regression(n_samples=1000, n_features=15, n_informative=5, random_state=1)
model = DecisionTreeRegressor()
model.fit(X, y)
importance = model.feature_importances_
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

Output:



c. Forecast and Evaluate Data

```
pred = model.predict(X)
errors = abs(pred - y)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
```

Output:

```
Mean Absolute Error: 0.0 degrees.
```

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / y)# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

Output:

```
Accuracy: 100.0 %.
```

Result:

Thus, univariate time series analysis has been performed on the given dataset and has been verified.

----- X -----

Date: 29.03.22, Tuesday **Exp. No.:** 03	**OpenStack**

1. Understanding OpenStack deployment, its implementation and its applications - Study

Aim:

To understand OpenStack deployment, its implementation and its applications.

Theory:

OpenStack is a cloud operating system that controls large pools of compute, storage and networking resources throughout a data-centre, all managed and provisioned through APIs with common authentication mechanisms.

A dashboard is also available, giving administrators control while empowering their users to provide resources through a web interface. It began in 2010 as a joint project of Ra despace hosting and NASA. It was managed by the OpenStack Foundation, a non-profit entity.

Working of OpenStack

It is essentially a series of commands known as scripts. These scripts are bundled into packages called projects that relay tasks that create cloud environments. To create these environments, OpenStack relies on

- Virtualization that creates a layer of virtual resources abstracted from hardware
- A base as that carries out commands given by OpenStack scripts

OpenStack uses virtualized resources to build clouds. It doesn't execute commands, rather delays them to base OS.

Components of OpenStack

OpenStack's architecture is made up of numerous open source projects. These are used to set up OpenStack's undercloud and overcloud. Overcloud is used by cloud users and undercloud is used by system admins. Underclouds contain the core components system admins need to set up and manage end user's environments called overclouds.

There are 6 stable, core services that handle computing, networking, storage, identity and images. These make up the infrastructure of OpenStack that allows the rest of the projects to handle dashboarding, orchestration, etc.

The 6 components are:

- Nova: It is a full management tool that helps compute resource-handling, scheduling, creation, deletion, etc.

- Neutron: It connects the networks across other services
- Swift: It is a highly fault-tolerant object storage service that stores and retrieves unstructured data objects
- Cinder: Provides persistent block storage
- Keystone: Authenticates and authorises all services
- Glance: Stores and retrieves VM disc images from various locations

Deployment of OpenStack

It is mostly deployed as infrastructure-as-a-Service (IaaS) in both public and private clouds where virtual servers and other resources are made available to users. The software platform consists of interrelated components control diverse, multivendor hardware. Lifecycle management tools and packaging are used to help instances maintain the lifecycle of deployments. Frameworks for the above are Tripleo, OpenStack-helm, kolla-ansible, kayole, etc.

Challenges in Implementation

- Installation challenges
- Documentation
- Upgrading OpenStack
- Long Term Support

Deployment Models

- OpenStack-based public cloud
- On-premises distribution
- Hosted OpenStack Private Cloud
- Appliance based OpenStack

Applications of OpenStack

- Private clouds and Public clouds
- Network function virtualization
- Containers

Result:

Thus, OpenStack deployment, its implementation and its applications have been studied.

----- X -----

Date: 05.04.22, Tuesday **Exp. No.:** 03.a	**OpenStack Installation**

1. Implementation of infrastructure as Service using OpenStack

Aim:

To install OpenStack and implement Infrastructure-as-a-service using it.

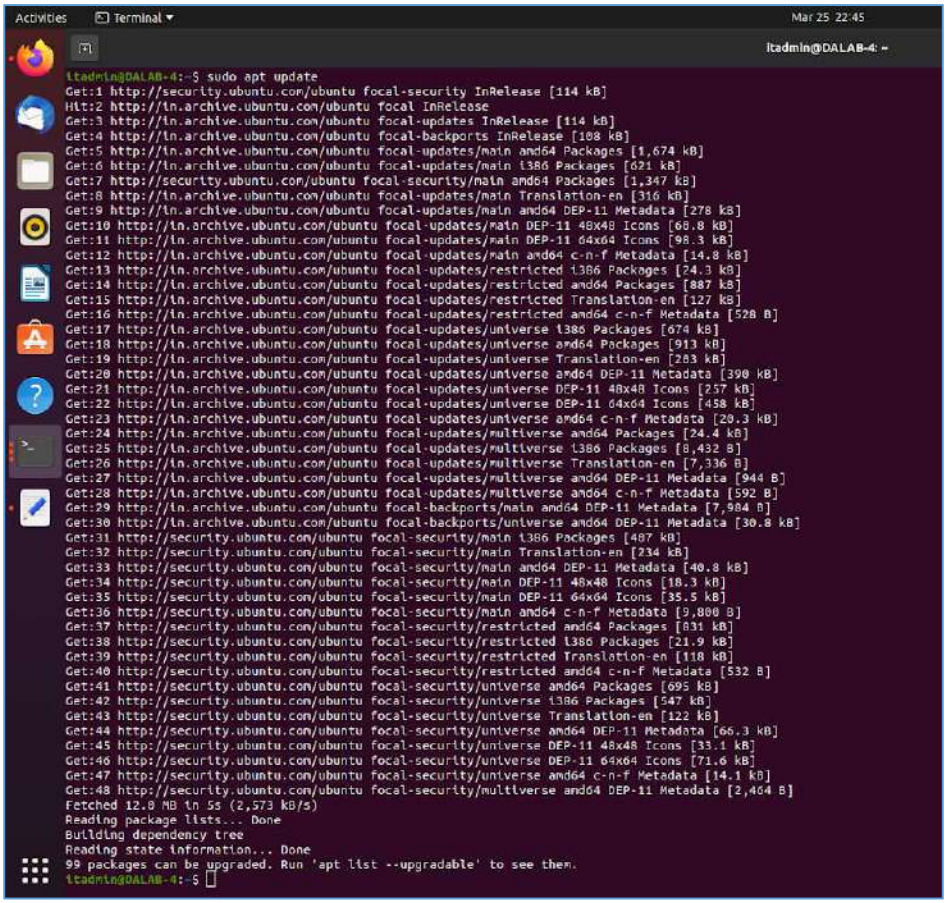
Procedure:

Installation of OpenStack in Ubuntu

1. Login to sudo ("superuser do") user
2. Open Terminal
3. In terminal, type:

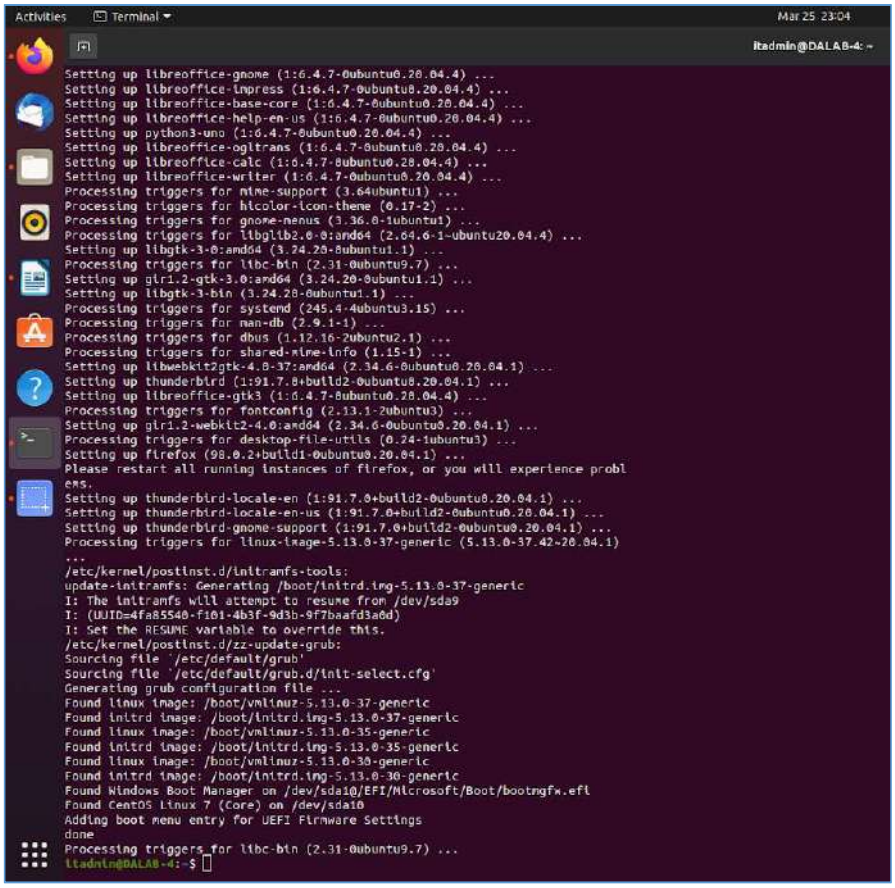
```
sudo apt – update
```

This downloads package information from all configured sources.



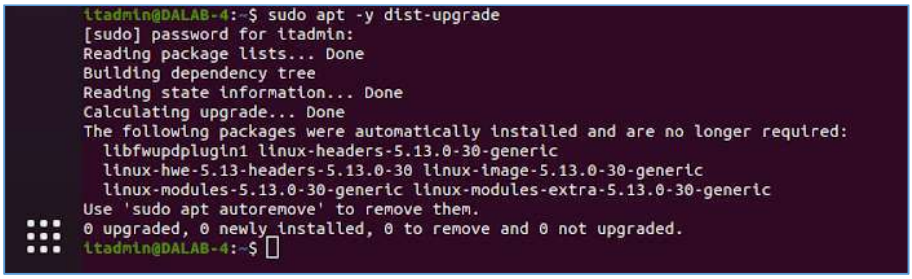
4. Upgrade everything in the system, all the packages, and the kernel to the latest versions as supported by the repositories:

```
sudo apt –y upgrade
```



5. Upgrade existing packages, installs new dependencies that are not in the system, and deletes those that are not needed:

```
sudo apt –y dist-upgrade
```



6. Add user called stack to run DevStack. It should be run as non-root user with sudo enabled:

```
sudo useradd -s /bin/bash -d /opt/stack -m stack
```

```
(base) student@DALAB-4:~/Desktop$ sudo useradd -s /bin/bash -d /opt/stack -m stack
[sudo] password for student:
student is not in the sudoers file.  This incident will be reported.
(base) student@DALAB-4:~/Desktop$ su - itadmin
Password:
itadmin@DALAB-4:~$ sudo useradd -s /bin/bash -d /opt/stack -m stack
[sudo] password for itadmin:
itadmin@DALAB-4:~$
```

7. User should have sudo privileges to make changes to the system:

```
echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
```

```
itadmin@DALAB-4:~$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
stack ALL=(ALL) NOPASSWD: ALL
itadmin@DALAB-4:~$
```

8. Log in to the stack once user is created:

```
sudo su - stack
```

```
itadmin@DALAB-4:~$ sudo su - stack
stack@DALAB-4:~$ sudo su -
```

9. Install git:

```
sudo apt -y install git
```

```
stack@DALAB-4:~$ sudo apt -y install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfwupdplugin1 linux-headers-5.13.0-30-generic
  linux-hwe-5.13-headers-5.13.0-30 linux-image-5.13.0-30-generic
  linux-modules-5.13.0-30-generic linux-modules-extra-5.13.0-30-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 5,465 kB of archives.
After this operation, 38.4 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu focal/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 git-man all 1:2.25.1-1ubuntu3.2 [884 kB]
9% [2 git-man 114 kB/884 kB 13%]
pdates/main amd64 git amd64 1:2.25.1-1ubuntu3.2 [4,554 kB]
Fetched 5,465 kB in 4min 3s (22.5 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 213052 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.25.1-1ubuntu3.2_all.deb ...
Unpacking git-man (1:2.25.1-1ubuntu3.2) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.25.1-1ubuntu3.2_amd64.deb ...
Unpacking git (1:2.25.1-1ubuntu3.2) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.25.1-1ubuntu3.2) ...
Setting up git (1:2.25.1-1ubuntu3.2) ...
Processing triggers for man-db (2.9.1-1) ...
```

10. Download devstack from its repository into system:

```
git clone https://git.openstack.org/openstack-dev/devstack
```

```
stack@DALAB-4:~$ git clone https://github.com/openstack-dev/devstack.git
Cloning into 'devstack'...
remote: Enumerating objects: 48499, done.
remote: Counting objects: 100% (1725/1725), done.
remote: Compressing objects: 100% (686/686), done.
remote: Total 48499 (delta 1177), reused 1444 (delta 1033), pack-reused 46774
Receiving objects: 100% (48499/48499), 15.48 MiB | 4.31 MiB/s, done.
Resolving deltas: 100% (33808/33808), done.
```

11. Download devstack setup configurations files for it. Need to navigate devstack folder by running:

```
cd devstack
nano local.conf
vi local.conf
```

```
stack@DALAB-4:~$ cd devstack
stack@DALAB-4:~/devstack$ nano local.conf
stack@DALAB-4:~/devstack$ vi local.conf
```

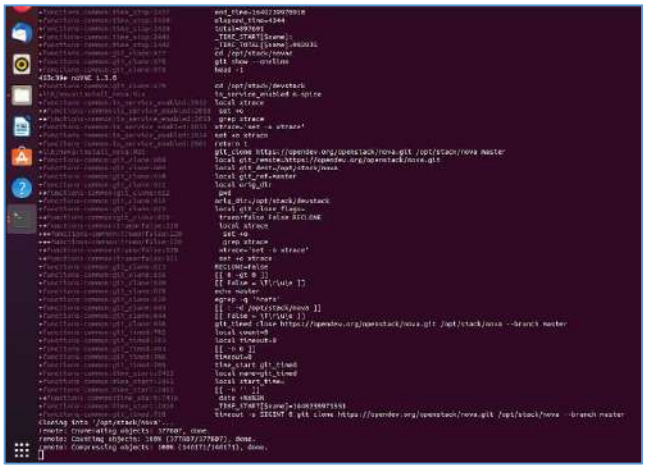
12. Add following inside local.conf:

```
[[local|localrc]]
# Password for KeyStone, Database, RabbitMQ and Service
ADMIN_PASSWORD=StrongAdminSecret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

After setting above, in local.conf, press Esc key and type :wq to write/save and quit

13. Run following to setup Openstack on system:

```
./stack.sh
```



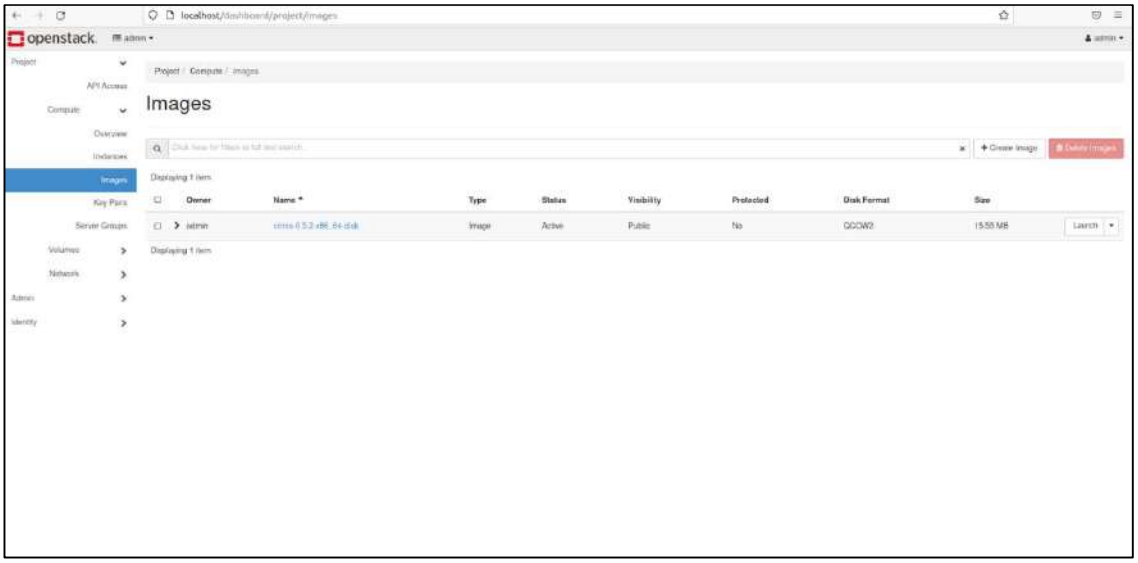
After installation, terminal:

```
=========================
DevStack Component Timing
 (times are in seconds)
=========================
wait_for_service      18
pip_install          280
apt-get              942
run_process           68
dbsync                66
git_timed           1034
apt-get-update         1
test_with_retry       51
async_wait          1564
osc                  283
-------------------------
Unaccounted time     816
=========================
Total runtime        5123

================
 Async summary
================
 Time spent in the background minus waits: 2015 sec
 Elapsed time: 5123 sec
 Time if we did everything serially: 7138 sec
 Speedup:  1.39332


This is your host IP address: 192.168.112.197
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.112.197/dashboard
Keystone is serving at http://192.168.112.197/identity/
The default users are: admin and demo
The password: StrongAdminSecret

Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html

DevStack Version: zed
Change: 0ed70e3f7687ffa62a8a4a38cdad14abdc8c7fa7 Merge "Update DEVSTACK_SERIES to zed" 2022-03-28 13:02:03 +0000
OS Version: Ubuntu 20.04 focal

2022-03-29 15:26:41.917 | stack.sh completed in 5123 seconds.
```

14. Browse URL on browser:

http://localhost/dashboard

15. Enter credentials. Log in as admin using username ‘admin’ and password as given in local.conf file:



Result:

Thus, OpenStack has been successfully installed.

----- X -----

Date: 09.04.22, Saturday **Exp. No.:** 04	**Creation of VMs in OpenStack**

1. Creation of VM in OpenStack and execution of simple application in OpenStack

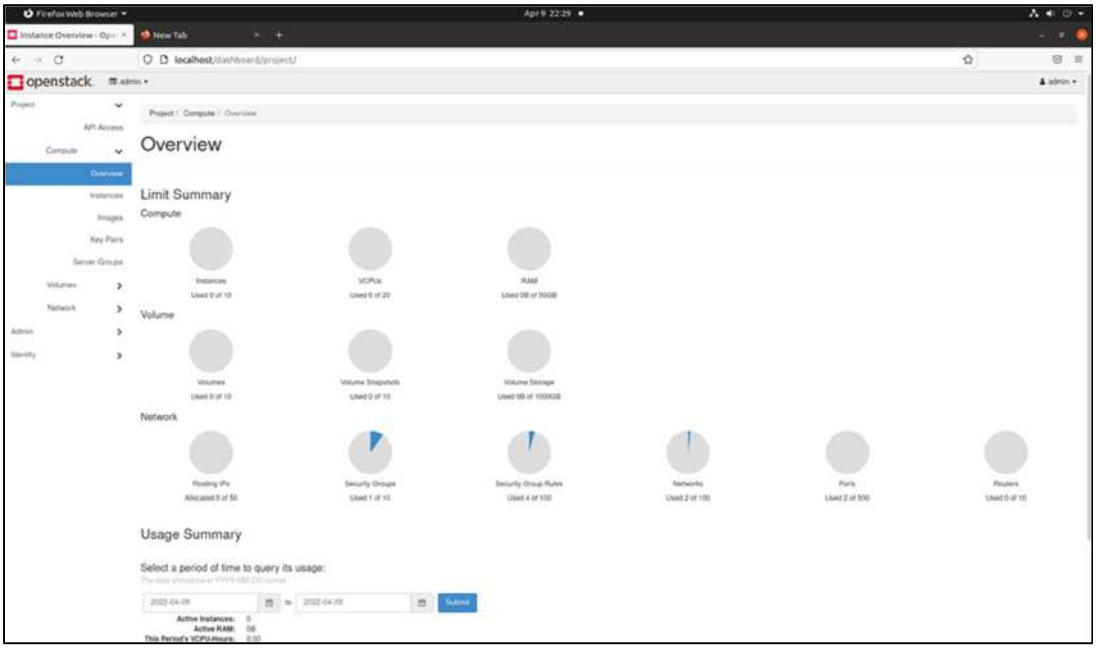
Aim:

To deploy a VM in OpenStack and execute a simple application on it.

Procedure:

Deploying VM in OpenStack

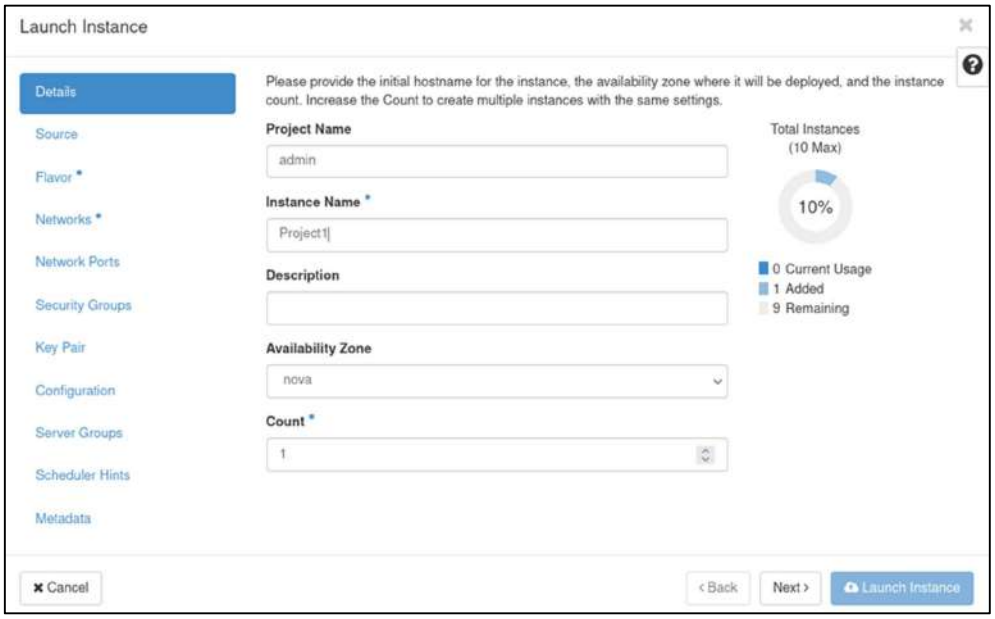
1. Login to OpenStack using credentials



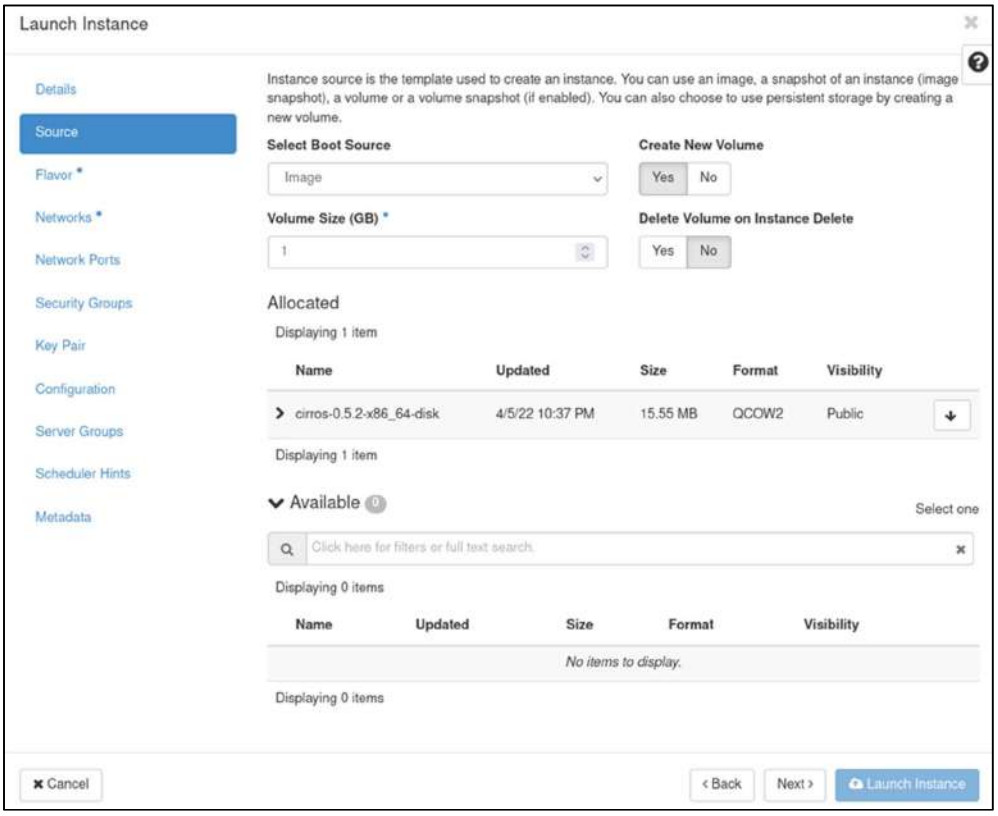
2. Launch an Existing Image Instance in OpenStack

Once launch option is selected, pop-up window is displayed.

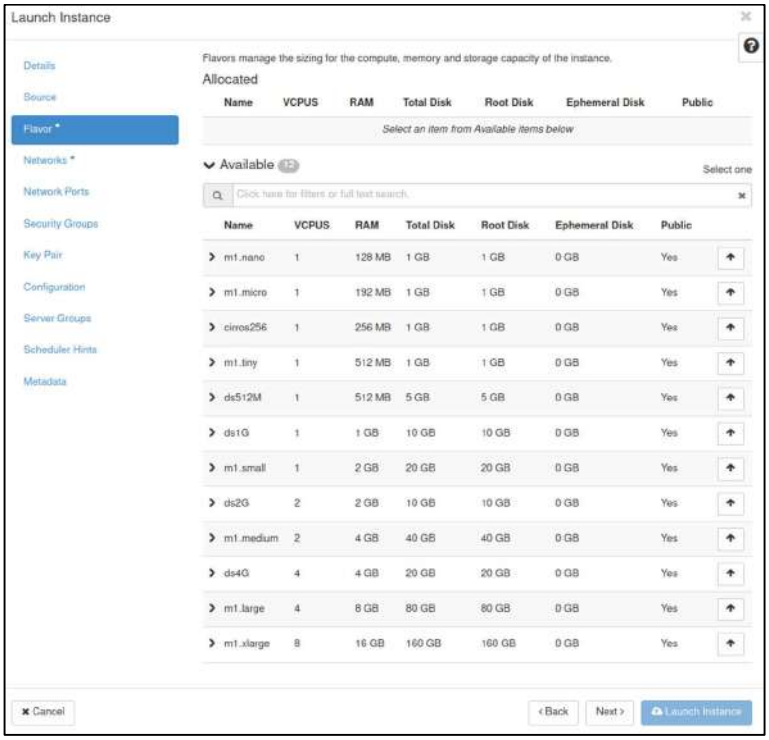
3. Give instance name of your choice. Leave the other fields as default values and click next



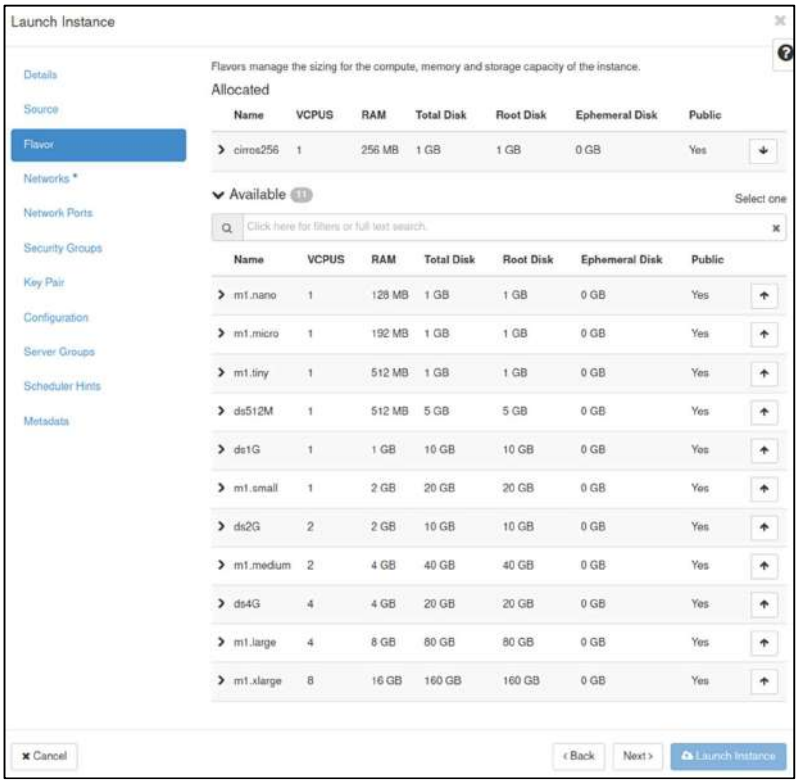
4. Source tab will be next screen in the launch instance window where storage is created and click next



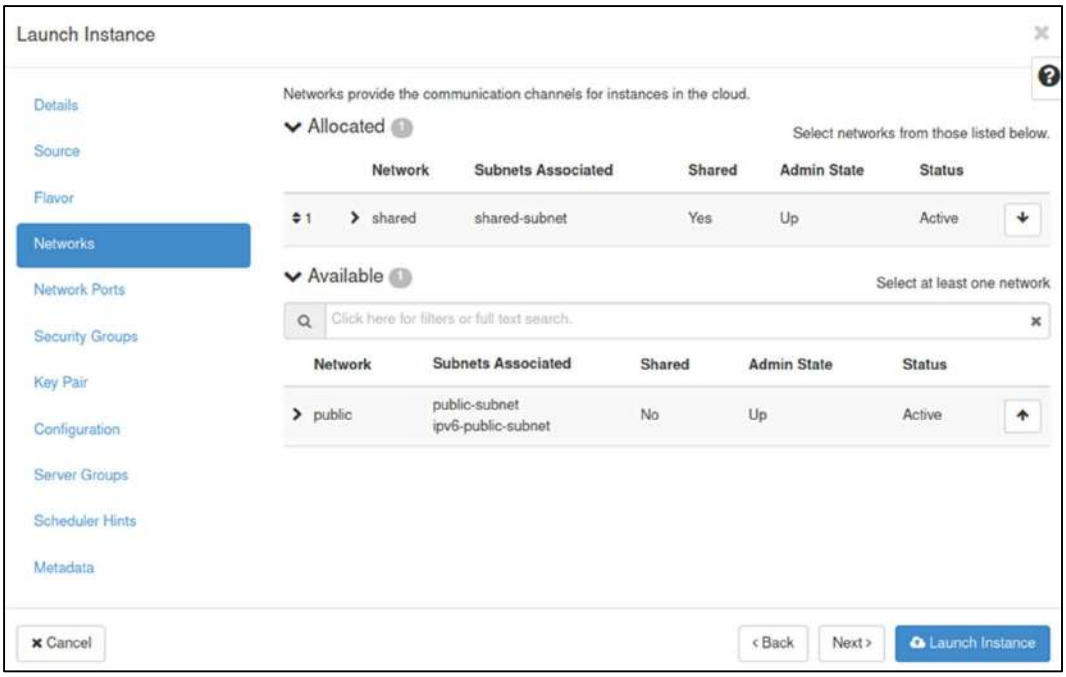
5. Allocate VM resources by adding a flavour best suitable for your needs and click on Next



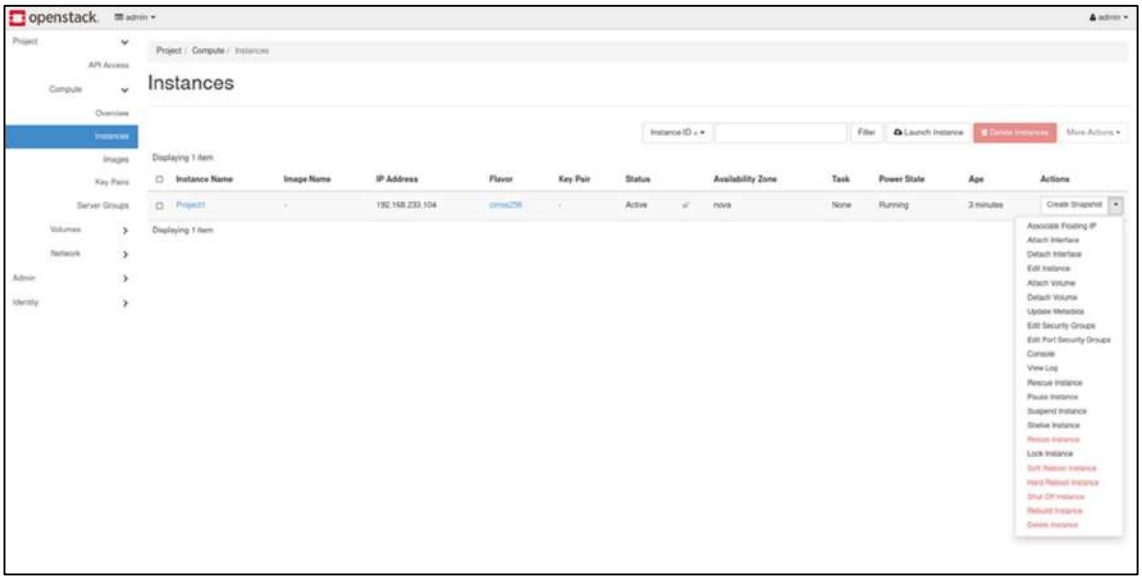
6. From available instance, add instance which already launched so that the instance moves under allocated



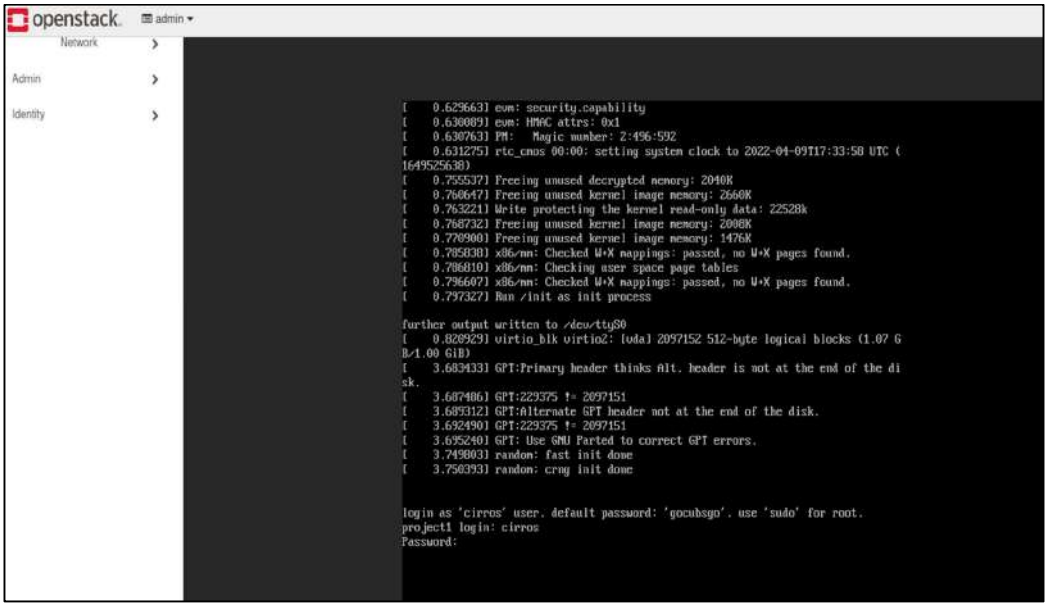
7. Finally, add shared network from available networks in OpenStack to your instance using upward arrow-mark button and hit on ‘Launch Instance’ to start the virtual machine



8. Once launch is clicked, select instance in right tab. The added instance is seen under it



9. Click ‘console’



10. Create a text file:

```
vi test.txt
```

11. In console type:

```
ls
```

12. Display text content using:

```
cat test.txt
```

Result:

Thus, VM has been deployed in OpenStack and a simple application has been executed.

----- X -----

Date: 12.04.22, Saturday **Exp. No.:** 05	**MongoDB Basics**

1. Study on MongoDB

Aim:

To study about the basics of MongoDB and its operations.

Theory:

MongoDB is a general-purpose document database designed for modern application development and for the cloud. Its scale-out architecture allows you to meet the increasing demand for your system by adding more nodes to share the load.

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Document

MongoDB stores data as JSON documents. The document data model maps naturally to objects in application code, making it simple for developers to learn and use. A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data. Documents can be nested to express hierarchical relationships and to store structures such as arrays.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

_id is a 12 bytes hexadecimal number which assures the uniqueness of every document. The first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

Sample Document

```
{
  '_id': ObjectId(7df78ad8902c),
  'Name': 'ABC',
  'Age': 20,
  'Reg_No': '201901a',
  'Subjects': [ 'Math', 'English'],
  'Details': {   'Father' : 'XY',
                 'Mother' : 'PQ'
             }
}
```

Operations

1. Use
   - To create and use a database. If the database exists, it returns existing database
   - Syntax:
     - use Database_name
2. Create
   - To create/insert a new document to a collection
   - Syntax:
     - db.collection_name.insertOne( )
     - db.collection_name.insertMany( )
3. Read
   - To retrieve documents from a collection
   - Syntax:
     - db.collection_name.find( )
4. Update
   - To modify existing documents in a collection
   - Syntax:
     - db.collection_name.updateOne( )
     - db.collection_name.updateMany( )
     - db.collection_name.replaceOne( )

5. Delete

- To remove documents from a collection
- Syntax:
  - db.collection_name.deleteOne( )
  - db.collection_name.deleteMany( )

Result:

Thus, a study on the basics of MongoDB and its operations has been made.

2. Pick any system as an example, create database and its corresponding tables. Also populate the tables with data. Apply insert /search operations

Aim:

To design a student management system using MongoDB and perform CRUD operations on it.

Code:

i. Create and Use Database 'StuDB'

```
use StuDb
db
```

Output:

```
> use StuDb
switched to db StuDb
> db
StuDb
```

ii. Insert

➢ Insert one document into collection

```
db.StuDb.insert( { "StuId" : 1, "Name" : "Ashwin", "Age" : 19, "Marks" : { "Math" : 100, "English" : 99 } } )
Db.StuDB.find( ).pretty( )
```

Output:

```
> db.StuDb.find().pretty()
{
        "_id" : ObjectId("6255a2667bb66dce7e8855ae"),
        "StuId" : 1,
        "Name" : "Ashwin",
        "Age" : 19,
        "Marks" : {
                "Math" : 100,
                "English" : 99
        }
}
```

- Insert multiple documents into collection

```
db.StuDb.insertMany( [ { "StuId" : 2, "Name" : "Charlie", "Age" : 20, "Marks" : { "Math" : 98, "English" : 100 } } , { "StuId" : 3, "Name" : "David", "Age" : 20, "Marks" : { "Math" : 89, "English" : 78 } }, { "StuId" : 4, "Name" : "Davis", "Age" : 20, "Marks" : { "Math" : 96, "English" : 95 } } ] )
```

Output:

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6255a46a7bb66dce7e8855af"),
                ObjectId("6255a46a7bb66dce7e8855b0"),
                ObjectId("6255a46a7bb66dce7e8855b1")
        ]
}
```

iii. Search

- Find document where 'StuId' is 3

```
db.StuDb.find( { 'StuID' : 3 } ).pretty( )
```

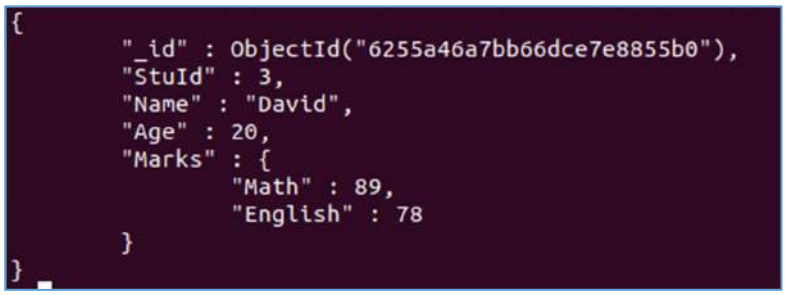
Output:

```
{
        "_id" : ObjectId("6255a46a7bb66dce7e8855b0"),
        "StuId" : 3,
        "Name" : "David",
        "Age" : 20,
        "Marks" : {
                "Math" : 89,
                "English" : 78
        }
}
```

- Find documents where marks in English is less than 80

```
db.StuDb.find( { 'Marks.English' : { $lt : 80 } } )
```

Output:



iv. View

```
db.StuDb.find( ).pretty( )
```

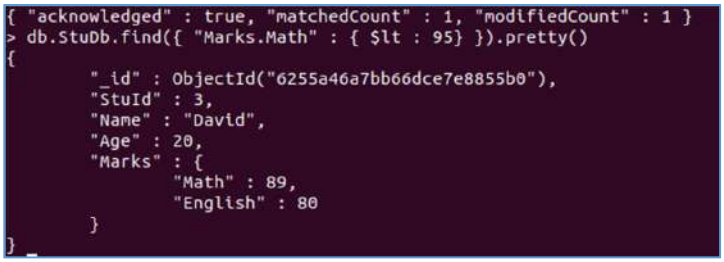
Output:

v. Update

Update the marks in English to 80 for all documents in collection where marks in math is less than 95.

```
db.StuDb.updateMany( { 'Marks.Math' : { $lt : 95 }}, { $set : { 'Marks.English' : 80 } } )
db.StuDb.find( { 'Marks.Math' : { $lt : 95 } } ).pretty( )
```

Output:



vi. Delete

Delete all documents that have marks in math less than 97.

```
db.StuDb.deleteMany( { 'Marks.Math' : { $lt : 97 } } )
db.StuDb.find( ).pretty( )
```

Output:

Result:

A student management system has been designed using MongoDB and CRUD operations have been performed on it successfully.

----- X -----

Date: 19.04.22, Tuesday
Exp. No.: 06

Bivariate Analysis

1. From the dataset you choose from UCI/Kaggle Repository, perform the following:

a. **Pearson Correlation Coefficients and Interpretation**
b. **Simple Linear Regression**
c. **Chi-square test**
d. **T-test**
e. **Analysis of Variance**
f. **Scatterplots**

Aim:

To perform bivariate analysis on the given dataset.

Codes and Output:

Import necessary packages and dataset

```
import pandas as pd
import numpy as np
df = pd.read_csv("mental-health-survey.csv")
encD = pd.get_dummies(df, columns = [ 'obs_consequence'] )
```

i. Pearson Correlation Coefficients and Interpretation

```
from scipy.stats import pearsonr
encD.corr( )
```

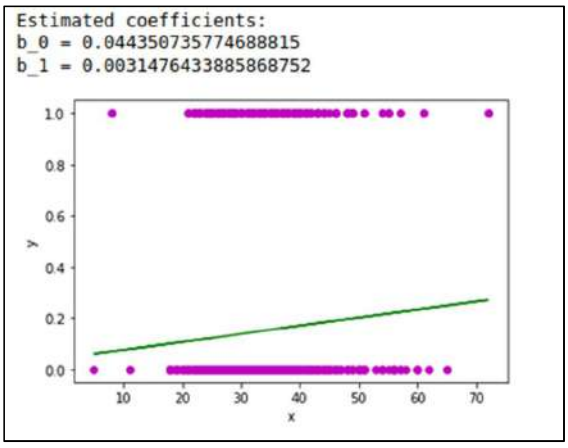
Output:

	Age	obs_consequence_No	obs_consequence_Yes
Age	1.000000	-0.065878	0.065878
obs_consequence_No	-0.065878	1.000000	-1.000000
obs_consequence_Yes	0.065878	-1.000000	1.000000

ii. Simple Linear Regression

```
import matplotlib.pyplot as plt
def estimate_coef(x, y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)
def plot_regression_line(x, y, b):
    plt.scatter(x, y, color = "m",
            marker = "o", s = 30)
    # predicted response vector
    y_pred = b[0] + b[1]*x
    plt.plot(x, y_pred, color = "g")
    plt.xlabel('x')
    plt.ylabel('y')
    plt.show( )
def main():
    x = encD[‘Age’]
    y = encD[‘obs_consequence_Yes’]
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {} \nb_1 = {}".format(b[0], b[1]))
    plot_regression_line(x, y, b)
 if __name__ == "__main__":
    main( )
```

Output:



iii. Chi-Squared Test

```
from scipy.stats import chi2_contingency
stat, p, dof, expected = chi2_contingency(encD[‘Age’])
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
   print('Dependent (reject H0)')
else:
   print('Independent (H0 holds true)')
```

Output:

```
p value is 1.0
Independent (H0 holds true)
```

iv. T-Test

```
from scipy.stats import ttest_ind
param1 = encD[‘Age’]
param2 = encD[‘obs_consequence_Yes’]
stat, p = ttest_ind(param1, param2)
print(‘Statistics: %.3f, p = %.3f’ % (stat,p))
```

Output:

```
Statistics: 152.872, p = 0.000
```

v. Analysis of Variance

```
import scipy.stats as stats
fvalue, pvalue = stats.f_oneway(encD[‘Age’], encD[‘obs_consequence_Yes’], encD
[‘obs_consequence_No’] )
print(“fvalue: ”, fvalue, “, pvalue: ”, pvalue)
```

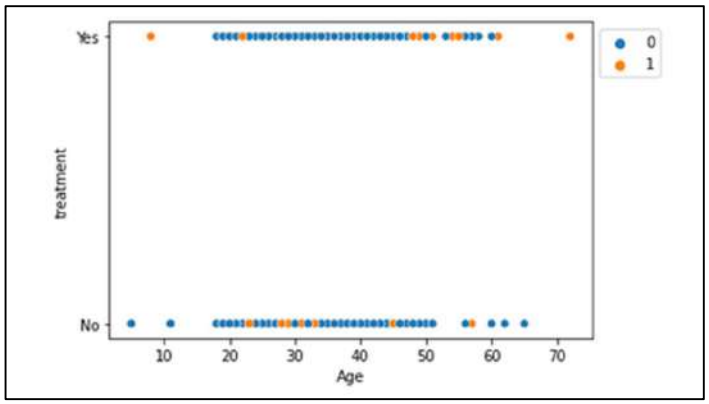
Output:

```
fvalue:  22809.108814167186 , pvalue:  0.0
```

vi. Scatterplots

```
import seaborn as sns
sns.scatterplot(x = 'Age', y = 'treatment', hue= 'obs_consequence_Yes', data = encD)
plt.legend(bbox_to_anchor = (1,1), loc = 2)
plt.show( )
```

Output:



Result:

Thus, bivariate analysis has been successfully carried out on the chosen dataset.

----- X -----

Date: 26.04.22, Tuesday
Exp. No.: 07

Logistic Regression

1. From the dataset you choose from UCI/Kaggle Repository, perform the following types of Logistic Regression:

a) Binary
b) Multi-class or Multinominal
c) Ordinal

Aim:

To perform binary, multinomial and ordinal logistic regression on the given dataset.

Codes and Output:

Import necessary packages and dataset

```
import pandas as pd
df = pd.read_csv("mental-health-survey.csv")
encD = pd.get_dummies(df, columns = [ 'obs_consequence'] )
data =  encD.drop(columns = ['obs_consequence_No'], axis = 1)
```

i. Binary Logistic Regression

```
#Encoding Attributes
cat_vars=['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(data[var], prefix=var)
    data1=data.join(cat_list)
    data=data1
cat_vars=['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
data_vars=data.columns.values.tolist()
to_keep=[i for i in data_vars if i not in cat_vars]
data_final=data[to_keep]
data_final.columns.values
```

Output:

```
array(['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed',
       'work_interfere', 'no_employees', 'remote_work', 'tech_company',
       'benefits', 'wellness_program', 'seek_help', 'anonymity', 'leave',
       'coworkers', 'supervisor', 'mental_health_interview',
       'phys_health_interview', 'mental_vs_physical', 'comments',
       'obs_consequence_Yes', 'family_history_No', 'family_history_Yes',
       'treatment_No', 'treatment_Yes', 'mental_health_consequence_Maybe',
       'mental_health_consequence_No', 'mental_health_consequence_Yes',
       'phys_health_consequence_Maybe', 'phys_health_consequence_No',
       'phys_health_consequence_Yes', 'care_options_No',
       'care_options_Not sure', 'care_options_Yes'], dtype=object)
```

```
#Split Input and Output Attributes
cols=['family_history_No', 'family_history_Yes',
      'treatment_No', 'treatment_Yes', 'mental_health_consequence_No',
      'mental_health_consequence_Yes',  'phys_health_consequence_Maybe',
      'phys_health_consequence_No',  'phys_health_consequence_Yes', 'care_options_No',
       'care_options_Yes', 'Age']
X=data[cols]
y=data['obs_consequence_Yes']
X.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1254 entries, 0 to 1253
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   family_history_No              1254 non-null   uint8
 1   family_history_Yes             1254 non-null   uint8
 2   treatment_No                   1254 non-null   uint8
 3   treatment_Yes                  1254 non-null   uint8
 4   mental_health_consequence_No   1254 non-null   uint8
 5   mental_health_consequence_Yes  1254 non-null   uint8
 6   phys_health_consequence_Maybe  1254 non-null   uint8
 7   phys_health_consequence_No     1254 non-null   uint8
 8   phys_health_consequence_Yes    1254 non-null   uint8
 9   care_options_No                1254 non-null   uint8
 10  care_options_Yes               1254 non-null   uint8
 11  Age                            1254 non-null   int64
dtypes: int64(1), uint8(11)
memory usage: 23.4 KB
```

```
#Implement Model
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

Output:

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 10
                                  Results: Logit
================================================================================================
Model:                          Logit                        Pseudo R-squared:           inf
Dependent Variable:             obs_consequence_Yes          AIC:                        inf
Date:                           2022-05-04 20:41             BIC:                        inf
No. Observations:               1254                         Log-Likelihood:             -inf
Df Model:                       9                            LL-Null:                    0.0000
Df Residuals:                   1244                         LLR p-value:                1.0000
Converged:                      1.0000                       Scale:                      1.0000
No. Iterations:                 10.0000
------------------------------------------------------------------------------------------------
                                Coef.    Std.Err.      z    P>|z|      [0.025         0.975]
------------------------------------------------------------------------------------------------
family_history_No               0.0419 7319116.7709  0.0000 1.0000 -14345205.2277 14345205.3115
family_history_Yes              0.3982 7315414.9545  0.0000 1.0000 -14337949.4446 14337950.2410
treatment_No                   -2.3305 7398153.1953 -0.0000 1.0000 -14500116.1454 14500111.4844
treatment_Yes                  -1.7421 7398994.6669 -0.0000 1.0000 -14501764.8110 14501761.3268
mental_health_consequence_No   -0.5840       0.2361 -2.4739 0.0134        -1.0466       -0.1213
mental_health_consequence_Yes   0.7339       0.2029  3.6175 0.0003         0.3363        1.1316
phys_health_consequence_Maybe  -0.5683 6115334.3154 -0.0000 1.0000 -11985835.5800 11985834.4434
phys_health_consequence_No     -0.8949 6115334.3154 -0.0000 1.0000 -11985835.9066 11985834.1168
phys_health_consequence_Yes    -0.3531 6115334.3154 -0.0000 1.0000 -11985835.3647 11985834.6586
care_options_No                 0.2281       0.2379  0.9591 0.3375        -0.2381        0.6944
care_options_Yes                0.5005       0.2358  2.1226 0.0338         0.0383        0.9627
Age                             0.0135       0.0110  1.2244 0.2208        -0.0081        0.0352
================================================================================================
```

```
#Choose Attributes with p<1
cols=['mental_health_consequence_No', 'mental_health_consequence_Yes',
      'care_options_No', 'care_options_Yes', 'Age']
X=data[cols]
y=data['obs_consequence_Yes']
#implement model and build model again
```

Output:

```
#Implement Model
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 7
                                  Results: Logit
================================================================================
Model:                          Logit                        Pseudo R-squared:     inf
Dependent Variable:             obs_consequence_Yes          AIC:                  inf
Date:                           2022-05-04 20:44             BIC:                  inf
No. Observations:               1254                         Log-Likelihood:       -inf
Df Model:                       4                            LL-Null:              0.0000
Df Residuals:                   1249                         LLR p-value:          1.0000
Converged:                      1.0000                       Scale:                1.0000
No. Iterations:                 7.0000
--------------------------------------------------------------------------------
                                Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
--------------------------------------------------------------------------------
mental_health_consequence_No   -1.0907   0.2114 -5.1598 0.0000 -1.5050 -0.6764
mental_health_consequence_Yes   0.6891   0.1802  3.8242 0.0001  0.3359  1.0422
care_options_No                -0.3161   0.2026 -1.5600 0.1188 -0.7132  0.0810
care_options_Yes                0.2681   0.2020  1.3275 0.1844 -0.1277  0.6640
Age                            -0.0493   0.0055 -8.9895 0.0000 -0.0600 -0.0385
================================================================================
```

```
#Build Logistic Regression model - binary
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'. format(logreg.score
(X_test, y_test)))
```

Output:

```
Accuracy of logistic regression classifier on test set: 0.87
```

ii. Multi-Class Linear Regression

```
#encoding attributes
encD = pd.get_dummies(df, columns = ['obs_consequence'])
data =  encD.drop(columns = ['obs_consequence_No'], axis = 1)
cat_vars=['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(data[var], prefix=var)
    data1=data.join(cat_list)
    data=data1
cat_vars=['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
data_vars=data.columns.values.tolist()
to_keep=[i for i in data_vars if i not in cat_vars]
data_final=data[to_keep]
data_final.columns.values
```

Output:

```
array(['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed',
       'work_interfere', 'no_employees', 'remote_work', 'tech_company',
       'benefits', 'wellness_program', 'seek_help', 'anonymity', 'leave',
       'coworkers', 'supervisor', 'mental_health_interview',
       'phys_health_interview', 'mental_vs_physical', 'comments',
       'obs_consequence_Yes', 'family_history_No', 'family_history_Yes',
       'treatment_No', 'treatment_Yes', 'mental_health_consequence_Maybe',
       'mental_health_consequence_No', 'mental_health_consequence_Yes',
       'phys_health_consequence_Maybe', 'phys_health_consequence_No',
       'phys_health_consequence_Yes', 'care_options_No',
       'care_options_Not sure', 'care_options_Yes'], dtype=object)
```

```
cols=['mental_health_consequence_No', 'mental_health_consequence_Yes',
      'care_options_No', 'care_options_Yes', 'Age']
X=data[cols]
y1=data['obs_consequence_Yes']
# evaluate multinomial logistic regression model
from numpy import mean
from numpy import std
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
# define dataset
# define the multinomial logistic regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
# define the model evaluation procedure
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate the model and collect the scores
n_scores = cross_val_score(model, X, y1, scoring='accuracy', cv=cv, n_jobs=-1)
# report the model performance
print('Mean Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

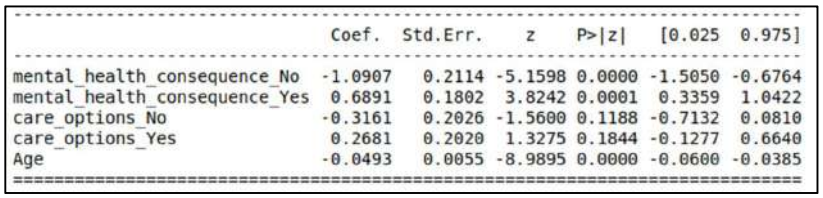
Output:

```
Mean Accuracy: 0.855 (0.003)
```

iii. Ordinal Logistic Regression

```
#get model summary and choose the attributes with p<1
cols=['mental_health_consequence_No', 'mental_health_consequence_Yes',
      'care_options_No', 'care_options_Yes', 'Age']
X=data[cols]
y=data['obs_consequence_Yes']
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

Output:



	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
mental_health_consequence_No	-1.0907	0.2114	-5.1598	0.0000	-1.5050	-0.6764
mental_health_consequence_Yes	0.6891	0.1802	3.8242	0.0001	0.3359	1.0422
care_options_No	-0.3161	0.2026	-1.5600	0.1188	-0.7132	0.0810
care_options_Yes	0.2681	0.2020	1.3275	0.1844	-0.1277	0.6640
Age	-0.0493	0.0055	-8.9895	0.0000	-0.0600	-0.0385

```
#build the model for ordinal logistic regression
from statsmodels.miscmodels.ordinal_model import OrderedModel
mod_prob = OrderedModel(y, X, distr='logit')
res_prob = mod_prob.fit(method='bfgs')
res  prob.summary()
```

Output:

```
Optimization terminated successfully.
         Current function value: 0.383282
         Iterations: 32
         Function evaluations: 36
         Gradient evaluations: 36
```

OrderedModel Results

Dep. Variable:	obs_consequence_Yes	**Log-Likelihood:**	-480.64
Model:	OrderedModel	**AIC:**	973.3
Method:	Maximum Likelihood	**BIC:**	1004.
Date:	Tue, 17 May 2022		
Time:	22:17:11		
No. Observations:	1254		
Df Residuals:	1248		
Df Model:	6		

	coef	std err	z	P>|z|	[0.025	0.975]
mental_health_consequence_No	-0.7673	0.223	-3.441	0.001	-1.204	-0.330
mental_health_consequence_Yes	0.8819	0.188	4.699	0.000	0.514	1.250
care_options_No	0.1768	0.234	0.754	0.451	-0.283	0.636
care_options_Yes	0.6434	0.229	2.810	0.005	0.195	1.092
Age	0.0157	0.011	1.448	0.147	-0.006	0.037
0/1	2.6536	0.411	6.454	0.000	1.848	3.459

Result:

Thus, binary, multinomial and ordinal logistic regression have been carried out on the given dataset successfully.

----- X -----

2. Build a Logistic Regression Model to predict values based on the dataset you have picked from the repository. Perform the following:

a) Data Understanding, Data cleaning, Exploratory Data Analysis-Relationship between target variable and other variables, Feature Engineering, Feature Normalization, Build and Train the model, Prediction, Perform Evaluation using standard metrics
b) Build Model by Optimizing the Hyperparameters, Evaluate the model

Aim:

To build a logistic regression model on the given dataset. Also, perform the given process for processing the data and building the model.

Codes and Output:

a) Data Understanding, Data cleaning, Exploratory Data Analysis-Relationship between target variable and other variables, Feature Engineering, Feature Normalization, Build and Train the model, Prediction, Perform Evaluation using standard metrics.

i. Data Understanding

```
df.shape
```

Output:

```
(1254, 27)
```

```
print(list(df.columns))
```

Output:

```
['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history', 'treatment', 'work_interfere', 'no_employees', 'remote_work', 'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_help', 'anonymity', 'leave', 'mental_health_consequence', 'phys_health_consequence', 'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_interview', 'mental_vs_physical', 'obs_consequence', 'comments']
```

```
df.head( )
```

Output:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequence
0	27-08-2014 11:29	37	Female	United States	IL	NaN	No	Yes	Often	Jun-25	...	Somewhat easy	No
1	27-08-2014 11:29	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	Maybe
2	27-08-2014 11:29	32	Male	Canada	NaN	NaN	No	No	Rarely	Jun-25	...	Somewhat difficult	No
3	27-08-2014 11:29	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	Yes
4	27-08-2014 11:30	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	No

5 rows × 27 columns

ii. Data Cleaning

```
#drop timestamp column
data = df.drop(columns = ['Timestamp'], axis = 1)
#drop rows where there is 'nan'
data['self_employed'].fillna('No', inplace = True)
data.shape
```

Output:

```
(1254, 26)
```

iii. Exploratory Data Analysis-Relationship between Target Variable and Other Variables

```
#Target class distribution
data['obs_consequence'].value_counts()
```

Output:

```
No     1072
Yes     182
Name: obs_consequence, dtype: int64
```

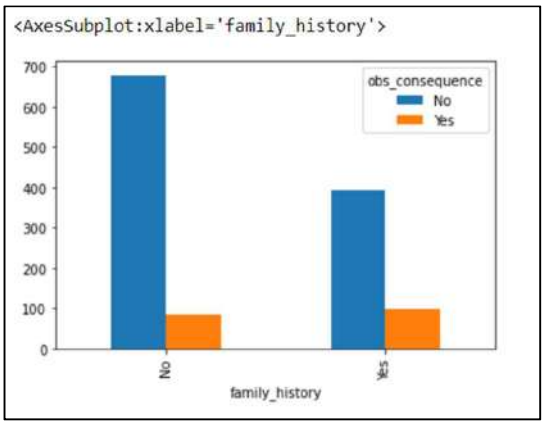
```
#Mean of data grouped by target class
data.groupby('obs_consequence').mean()
```

Output:

obs_consequence	Age
No	31.819030
Yes	33.197802

```
#visualize relationship between 'family_history' and target class
pd.crosstab(data.family_history, data.obs_consequence).plot(kind='bar')
```

Output:



iv. Feature Engineering

```
#Feature engineering - one hot encoding
encD = pd.get_dummies(data, columns = ['obs_consequence'])
print(list(encD.columns))
```

Output:

```
['Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history', 'treatment', 'work_interfere', 'no_employees', 'remote
_work', 'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_help', 'anonymity', 'leave', 'mental_health_conse
quence', 'phys_health_consequence', 'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_interview', 'mental_vs_p
hysical', 'comments', 'obs_consequence_No', 'obs_consequence_Yes']
```

```
#encode independent variables
cat_vars=['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(encD[var], prefix=var)
    data1=encD.join(cat_list)
    encD=data1
cat_vars=['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
data_vars=encD.columns.values.tolist()
to_keep=[i for i in data_vars if i not in cat_vars]
data_final=encD[to_keep]
data_final.columns.values
```

Output:

```
array(['Age', 'Gender', 'Country', 'state', 'self_employed',
       'work_interfere', 'no_employees', 'remote_work', 'tech_company',
       'benefits', 'wellness_program', 'seek_help', 'anonymity', 'leave',
       'coworkers', 'supervisor', 'mental_health_interview',
       'phys_health_interview', 'mental_vs_physical', 'comments',
       'obs_consequence_No', 'obs_consequence_Yes', 'family_history_No',
       'family_history_Yes', 'treatment_No', 'treatment_Yes',
       'mental_health_consequence_Maybe', 'mental_health_consequence_No',
       'mental_health_consequence_Yes', 'phys_health_consequence_Maybe',
       'phys_health_consequence_No', 'phys_health_consequence_Yes',
       'care_options_No', 'care_options_Not sure', 'care_options_Yes'],
      dtype=object)
```

v. Feature Normalization

```
#scaling
from sklearn.preprocessing import MinMaxScaler
encD[['Age']]= MinMaxScaler().fit_transform(encD[['Age']])
encD['Age']
```

Output:

```
0       0.477612
1       0.582090
2       0.402985
3       0.388060
4       0.388060
          ...
1249    0.313433
1250    0.402985
1251    0.432836
1252    0.611940
1253    0.298507
Name: Age, Length: 1254, dtype: float64
```

vi. Build and Train Model

```
cols=['family_history_No', 'family_history_Yes',
      'treatment_No', 'treatment_Yes',
      'mental_health_consequence_No', 'mental_health_consequence_Yes',
      'phys_health_consequence_Maybe', 'phys_health_consequence_No',
      'phys_health_consequence_Yes', 'care_options_No',
       'care_options_Yes', 'Age']
X=encD[cols]
y=encD['obs_consequence_Yes']
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

Output:

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 8
                                  Results: Logit
==================================================================================================
Model:                      Logit                    Pseudo R-squared:          inf
Dependent Variable:         obs_consequence_Yes      AIC:                       inf
Date:                       2022-05-04 23:43         BIC:                       inf
No. Observations:           1254                     Log-Likelihood:            -inf
Df Model:                   9                        LL-Null:                   0.0000
Df Residuals:               1244                     LLR p-value:               1.0000
Converged:                  1.0000                   Scale:                     1.0000
No. Iterations:             8.0000
--------------------------------------------------------------------------------------------------
                              Coef.    Std.Err.      z    P>|z|      [0.025         0.975]
--------------------------------------------------------------------------------------------------
family_history_No            -1.0609 8374331.5542 -0.0000 1.0000 -16413389.3017 16413387.1799
family_history_Yes           -0.7046 8366271.4967 -0.0000 1.0000 -16397591.5231 16397590.1138
treatment_No                 -1.1770         nan     nan    nan            nan           nan
treatment_Yes                -0.5886         nan     nan    nan            nan           nan
mental_health_consequence_No -0.5840      0.2361 -2.4739 0.0134        -1.0466       -0.1213
mental_health_consequence_Yes 0.7339      0.2029  3.6175 0.0003         0.3363        1.1316
phys_health_consequence_Maybe -0.5514         nan     nan    nan            nan           nan
phys_health_consequence_No   -0.8780         nan     nan    nan            nan           nan
phys_health_consequence_Yes  -0.3362         nan     nan    nan            nan           nan
care_options_No               0.2281      0.2379  0.9591 0.3375        -0.2381        0.6944
care_options_Yes              0.5005      0.2358  2.1226 0.0338         0.0383        0.9627
Age                           0.9056      0.7396  1.2244 0.2208        -0.5440        2.3553
==================================================================================================
```

```
#choose attributes with p<1
cols=['mental_health_consequence_No', 'mental_health_consequence_Yes',
      'care_options_No', 'care_options_Yes', 'Age']
X=encD[cols]
y=encD['obs_consequence_Yes']
```

```
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 7
                                Results: Logit
================================================================================
Model:                     Logit                  Pseudo R-squared:       inf
Dependent Variable:        obs_consequence_Yes    AIC:                    inf
Date:                      2022-05-04 23:54       BIC:                    inf
No. Observations:          1254                   Log-Likelihood:         -inf
Df Model:                  4                      LL-Null:                0.0000
Df Residuals:              1249                   LLR p-value:            1.0000
Converged:                 1.0000                 Scale:                  1.0000
No. Iterations:            7.0000
--------------------------------------------------------------------------------
                              Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
--------------------------------------------------------------------------------
mental_health_consequence_No  -1.1635   0.2089 -5.5681 0.0000 -1.5730 -0.7539
mental_health_consequence_Yes  0.6355   0.1784  3.5618 0.0004  0.2858  0.9852
care_options_No               -0.4262   0.1967 -2.1669 0.0302 -0.8118 -0.0407
care_options_Yes               0.1655   0.1964  0.8426 0.3994 -0.2195  0.5505
Age                           -3.5750   0.4134 -8.6489 0.0000 -4.3852 -2.7649
================================================================================
```

```
#Build Model
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Output:

```
LogisticRegression()
```

vii. Prediction

```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.3f}'. format(logreg.score
(X_test, y_test)))
```

Output:

```
Accuracy of logistic regression classifier on test set: 0.869
```

viii. Evaluation

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

Output:

```
[[218   0]
 [ 33   0]]
```

```
#Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Output:

	precision	recall	f1-score	support
0	0.87	1.00	0.93	218
1	0.00	0.00	0.00	33
accuracy			0.87	251
macro avg	0.43	0.50	0.46	251
weighted avg	0.75	0.87	0.81	251

b) Build Model by Optimizing the Hyperparameters, Evaluate the model

```
#Construct Random Forest Model with default parameters
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
#Default parameters
model = RandomForestClassifier(random_state= 101).fit(X_train,y_train)
predictionforest = model.predict(X_test)
print(confusion_matrix(y_test,predictionforest))
print(classification_report(y_test,predictionforest))
acc1 = accuracy_score(y_test,predictionforest)
```

Output:

```
[[196  22]
 [ 27   6]]
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       218
           1       0.21      0.18      0.20        33

    accuracy                           0.80       251
   macro avg       0.55      0.54      0.54       251
weighted avg       0.79      0.80      0.80       251
```

- Random Search for optimizing hyperparameters

```
import numpy as np
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
random_search = {'criterion': ['entropy', 'gini'],
        'max_depth': list(np.linspace(10, 1200, 10, dtype = int)) + [None],
        'max_features': ['auto', 'sqrt','log2', None],
        'min_samples_leaf': [4, 6, 8, 12],
        'min_samples_split': [5, 7, 10, 14],
        'n_estimators': list(np.linspace(151, 1200, 10, dtype = int))}
clf = RandomForestClassifier()
model = RandomizedSearchCV(estimator = clf, param_distributions = random_search,
n_iter = 50, cv = 4, verbose= 5, random_state= 40, n_jobs = -1)
model.fit(X_train,y_train)
```

Output:

```
Fitting 4 folds for each of 50 candidates, totalling 200 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:    7.4s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:   48.0s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:  2.4min finished

RandomizedSearchCV(cv=4, estimator=RandomForestClassifier(), n_iter=50,
                   n_jobs=-1,
                   param_distributions={'criterion': ['entropy', 'gini'],
                                        'max_depth': [10, 142, 274, 406, 538,
                                                      671, 803, 935, 1067, 1200,
                                                      None],
                                        'max_features': ['auto', 'sqrt', 'log2',
                                                         None],
                                        'min_samples_leaf': [4, 6, 8, 12],
                                        'min_samples_split': [5, 7, 10, 14],
                                        'n_estimators': [151, 267, 384, 500,
                                                         617, 733, 850, 966,
                                                         1083, 1200]},
                   random_state=40, verbose=5)
```

```
#Evaluating Optimized model
predictionforest = model.best_estimator_.predict(X_test)
print(confusion_matrix(y_test,predictionforest))
print(classification_report(y_test,predictionforest))
acc3 = accuracy_score(y_test,predictionforest)
```

Output:

```
[[218   0]
 [ 33   0]]
              precision    recall  f1-score   support

           0       0.87      1.00      0.93       218
           1       0.00      0.00      0.00        33

    accuracy                           0.87       251
   macro avg       0.43      0.50      0.46       251
weighted avg       0.75      0.87      0.81       251
```

Result:

Thus, the given process for processing the dataset has been carried out and a logistic regression model along with hyperparameter tuning has been built successfully.

----- X -----

Date: 26.04.22, Tuesday
Exp. No.: 08

Multiple Regression Analysis

Build a Multiple Linear Regression Model to predict values based on the dataset you have picked from repository. Perform the following:

a) Data Understanding, Data Preparation, Split Dataset, Feature Scaling, Build and Train the model, Residual Analysis, Prediction, Evaluate the model
b) Also Perform Recursive Feature Elimination, Build Model, Evaluate the model

Aim:

To build a multiple regression model on the given dataset. Also, perform the given process for processing the data and building the model.

Codes and Output:

a) Data Understanding, Data Preparation, Split Dataset, Feature Scaling, Build and Train the model, Residual Analysis, Prediction, Evaluate the model

i. Data Understanding

```
df.shape
```

Output:

```
(1254, 27)
```

```
print(list(df.columns))
```

Output:

```
['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history', 'treatment', 'work_interfere', 'no_employees', 'remote_work', 'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_help', 'anonymity', 'leave', 'mental_health_consequence', 'phys_health_consequence', 'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_interview', 'mental_vs_physical', 'obs_consequence', 'comments']
```

```
df.head( )
```

Output:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequence
0	27-08-2014 11:29	37	Female	United States	IL	NaN	No	Yes	Often	Jun-25	...	Somewhat easy	No
1	27-08-2014 11:29	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	Maybe
2	27-08-2014 11:29	32	Male	Canada	NaN	NaN	No	No	Rarely	Jun-25	...	Somewhat difficult	No
3	27-08-2014 11:29	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	Yes
4	27-08-2014 11:30	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	No

5 rows × 27 columns

ii. Data Preparation

```
encD = pd.get_dummies(data, columns = ['obs_consequence'])
encD.drop(columns = ['Timestamp', 'obs_consequence_No'], axis = -1, inplace=True)
print(encD.columns)
```

Output:

```
Index(['Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history',
       'treatment', 'work_interfere', 'no_employees', 'remote_work',
       'tech_company', 'benefits', 'care_options', 'wellness_program',
       'seek_help', 'anonymity', 'leave', 'mental_health_consequence',
       'phys_health_consequence', 'coworkers', 'supervisor',
       'mental_health_interview', 'phys_health_interview',
       'mental_vs_physical', 'comments', 'obs_consequence_Yes'],
      dtype='object')
```

```
cat_vars = ['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
for var in cat_vars:
    cat_list = 'var'+'_'+var
    cat_list = pd.get_dummies(encD[var], prefix=var)
    data1 = encD.join(cat_list)
    encD = data1
data_vars = encD.columns.values.tolist()
to_keep = [i for i in data_vars if i not in cat_vars]
data_final = encD[to_keep]
data_final.columns.values
```

Output:

```
array(['Age', 'Gender', 'Country', 'state', 'self_employed',
       'work_interfere', 'no_employees', 'remote_work', 'tech_company',
       'benefits', 'wellness_program', 'seek_help', 'anonymity', 'leave',
       'coworkers', 'supervisor', 'mental_health_interview',
       'phys_health_interview', 'mental_vs_physical', 'comments',
       'obs_consequence_Yes', 'family_history_No', 'family_history_Yes',
       'treatment_No', 'treatment_Yes', 'mental_health_consequence_Maybe',
       'mental_health_consequence_No', 'mental_health_consequence_Yes',
       'phys_health_consequence_Maybe', 'phys_health_consequence_No',
       'phys_health_consequence_Yes', 'care_options_No',
       'care_options_Not sure', 'care_options_Yes'], dtype=object)
```

iii. Split Dataset

```
cols=['family_history_No', 'family_history_Yes',
      'treatment_No', 'treatment_Yes',
      'mental_health_consequence_No', 'mental_health_consequence_Yes',
      'phys_health_consequence_Maybe', 'phys_health_consequence_No',
      'phys_health_consequence_Yes', 'care_options_No',
       'care_options_Yes', 'Age']
X=encD[cols]
y=encD['obs_consequence_Yes']
X.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1254 entries, 0 to 1253
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   family_history_No               1254 non-null   uint8
 1   family_history_Yes              1254 non-null   uint8
 2   treatment_No                    1254 non-null   uint8
 3   treatment_Yes                   1254 non-null   uint8
 4   mental_health_consequence_No    1254 non-null   uint8
 5   mental_health_consequence_Yes   1254 non-null   uint8
 6   phys_health_consequence_Maybe   1254 non-null   uint8
 7   phys_health_consequence_No      1254 non-null   uint8
 8   phys_health_consequence_Yes     1254 non-null   uint8
 9   care_options_No                 1254 non-null   uint8
 10  care_options_Yes                1254 non-null   uint8
 11  Age                             1254 non-null   int64
dtypes: int64(1), uint8(11)
memory usage: 23.4 KB
```

iv. Feature Scaling

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
X = min_max_scaler.fit_transform(X)
X
```

Output:

```
array([[1.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.47761194],
       [1.        , 0.        , 1.        , ..., 1.        , 0.        ,
        0.58208955],
       [1.        , 0.        , 1.        , ..., 1.        , 0.        ,
        0.40298507],
       ...,
       [0.        , 1.        , 0.        , ..., 0.        , 1.        ,
        0.43283582],
       [1.        , 0.        , 1.        , ..., 0.        , 1.        ,
        0.6119403 ],
       [0.        , 1.        , 0.        , ..., 0.        , 1.        ,
        0.29850746]])
```

v. Build and Train the Model

```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
from sklearn.linear_model import LinearRegression
linear_regression = LinearRegression()
linear_regression.fit(X_train, y_train)
```

Output:

```
(1003, 12)
```

```
1 from sklearn.linear_model import LinearRegression
2 linear_regression = LinearRegression()
3 linear_regression.fit(X_train, y_train)
```

```
LinearRegression()
```

vi. Residual Analysis

```
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols
s = 'obs_consequence_Yes ~ family_history_Yes + treatment_Yes +
mental_health_consequence_Yes + phys_health_consequence_Yes + care_options_Yes +
Age'
multi_model = ols(s, data=encD).fit()
print(multi_model.summary())
fig = plt.figure(figsize=(14, 8))
```

Output:

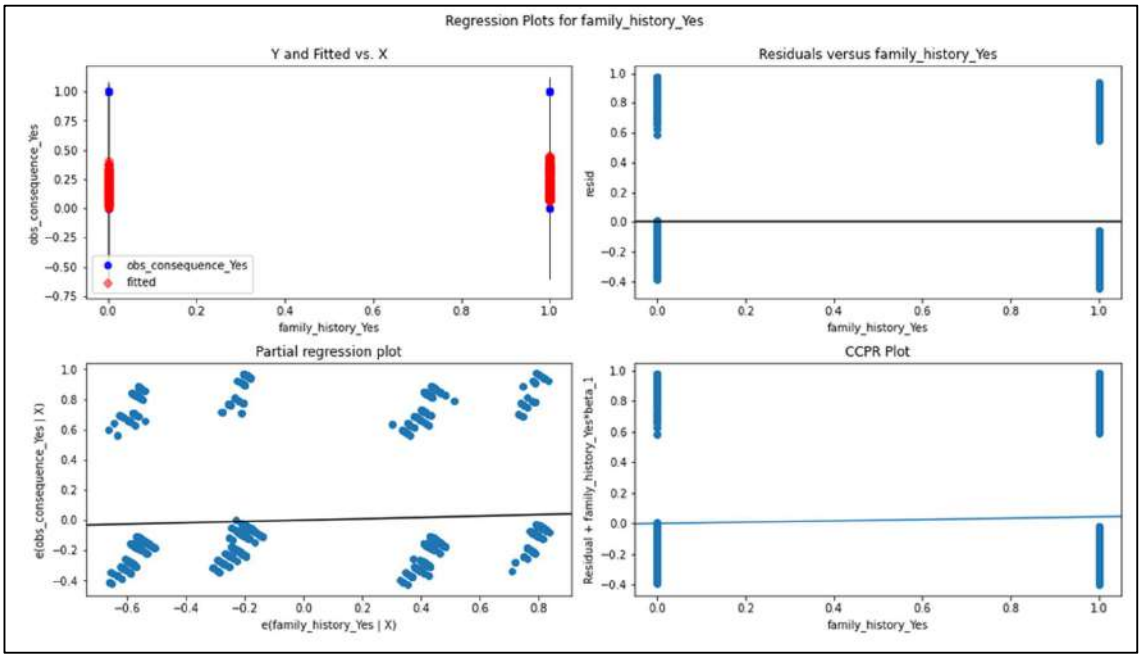
OLS Regression Results

Dep. Variable:	obs_consequence_Yes	R-squared:	0.075
Model:	OLS	Adj. R-squared:	0.070
Method:	Least Squares	F-statistic:	16.74
Date:	Tue, 24 May 2022	Prob (F-statistic):	1.20e-18
Time:	21:36:33	Log-Likelihood:	-422.29
No. Observations:	1254	AIC:	858.6
Df Residuals:	1247	BIC:	894.5
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0202	0.044	-0.463	0.643	-0.106	0.065
family_history_Yes	0.0446	0.021	2.095	0.036	0.003	0.086
treatment_Yes	0.0670	0.021	3.121	0.002	0.025	0.109
mental_health_consequence_Yes	0.1515	0.025	6.142	0.000	0.103	0.200
phys_health_consequence_Yes	0.0784	0.049	1.607	0.108	-0.017	0.174
care_options_Yes	0.0389	0.021	1.850	0.065	-0.002	0.080
Age	0.0019	0.001	1.470	0.142	-0.001	0.005

Omnibus:	392.850	Durbin-Watson:	1.855
Prob(Omnibus):	0.000	Jarque-Bera (JB):	871.082
Skew:	1.809	Prob(JB):	7.03e-190
Kurtosis:	4.891	Cond. No.	171.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
eval_env: 1



vii. Prediction

```
y_pred=linear_regression.predict(X_test)
y_pred
```

Output:

```
array([0.04161757, 0.1563285 , 0.17200591, 0.20281391, 0.07937323,
       0.04357725, 0.34190527, 0.03181919, 0.35774622, 0.2078019 ,
       0.16433076, 0.30201099, 0.19996319, 0.03377887, 0.1525727 ,
       0.15828818, 0.07529033, 0.02594016, 0.03965789, 0.16416721,
       0.25249727, 0.08508871, 0.03181919, 0.39447403, 0.28581177,
       0.15649205, 0.0694113 , 0.12069607, 0.03181919, 0.19604384,
       0.14669367, 0.14261077, 0.17592527, 0.16416721, 0.07725001,
       0.04945627, 0.1027258 , 0.16433076, 0.21760028, 0.1265751 ,
       0.03573854, 0.04945627, 0.3136055 , 0.10628688, 0.2797631 ,
       0.12069607, 0.08508871, 0.26086709, 0.1265751 , 0.03769822,
       0.03965789, 0.09880644, 0.35136115, 0.16825011, 0.02202081,
       0.04553692, 0.15240915, 0.04161757, 0.15061302, 0.19176622,
       0.03965789, 0.13049445, 0.12069607, 0.21368092, 0.03769822,
       0.3214442 , 0.13460243, 0.03573854, 0.17341877, 0.17412914,
       0.02594016, 0.14261077, 0.04945627, 0.03377887, 0.12480404,
       0.04357725, 0.17608881, 0.11481704, 0.08900806, 0.27013436,
       0.21172125, 0.03181919, 0.12265575, 0.19996319, 0.09292741,
       0.08508871, 0.0964885 , 0.14653012, 0.12461542, 0.04161757,
       0.40317704, 0.26229565, 0.19408416, 0.03573854, 0.1344138 ,
       0.17592527, 0.03181919, 0.30380712, 0.15436883, 0.1484898 ,
       0.15453238, 0.02789984, 0.03181919, 0.21509378, 0.22739866,
       0.02398048, 0.15828818, 0.22327498, 0.17733812, 0.10716696,
       0.27993274, 0.11677672, 0.16237108, 0.10697834, 0.0694113 ,
       0.06353227, 0.18125748, 0.19604384, 0.14261077, 0.21172125,
       0.22097281, 0.06317401, 0.20584222, 0.04357725, 0.19800352,
       0.18000817, 0.16612689, 0.2489551 , 0.0553353 , 0.04945627,
       0.09309096, 0.03181919, 0.04553692, 0.16629043, 0.21368092,
       0.21172125, 0.25641662, 0.1563285 , 0.01810146, 0.15436883,
       0.06317401, 0.02398048, 0.2423657 , 0.08508871, 0.02594016,
       0.21065262, 0.22097281, 0.17984462, 0.36542138, 0.02789984,
       0.19800352, 0.04553692, 0.13049445, 0.27228265, 0.15845173,
       0.08704839, 0.19016481, 0.2386349 , 0.355623  , 0.03573854,
       0.1406511 , 0.28617003, 0.05961292, 0.21955995, 0.27013436,
       0.07545388, 0.20976157, 0.04945627, 0.09505064, 0.25820666,
       0.10484902, 0.11873639, 0.03965789, 0.06317401, 0.10288934,
       0.15044948, 0.03573854, 0.38553995, 0.17788494, 0.1604114 ,
       0.02789984, 0.18588719, 0.12069607, 0.02985951, 0.17396559,
       0.12853477, 0.02398048, 0.09096774, 0.39643371, 0.11500566,
       0.19105586, 0.1484898 , 0.15649205, 0.15061302, 0.04553692,
       0.04945627, 0.18768332, 0.12069607, 0.04161757, 0.06121433,
       0.24394037, 0.19996319, 0.16237108, 0.33372407, 0.12052643,
       0.03377887, 0.21528234, 0.03573854, 0.12480404, 0.34940148,
       0.10893801, 0.07920968, 0.19408416, 0.04553692, 0.0474966 ,
       0.36899824, 0.16237108, 0.1265751 , 0.11481704, 0.34582462,
       0.23060764, 0.14457045, 0.03377887, 0.20584222, 0.1525727 ,
       0.1287234 , 0.16220753, 0.02398048, 0.11677672, 0.05781679,
       0.0474966 , 0.19016481, 0.14653012, 0.20548396, 0.09096774,
       0.22864797, 0.20584222, 0.10554977, 0.27405371, 0.0474966 ,
       0.21368092, 0.29025293, 0.18909618, 0.10501866, 0.12284437,
       0.08329258, 0.07333065, 0.2806274 , 0.08508871, 0.02398048,
       0.20584222])
```

viii. Evaluation

```
c = 0
for i in y_pred:
    if i>0.2:
        y_pred[c] = 1
    else:
        y_pred[c] = 0
    c+=1
y_pred
```

Output:

```
array([0., 0., 0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0.,
       0., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
       0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
       1., 1., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 1.,
       1., 0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0.,
       1., 0., 0., 0., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 0., 0., 1.,
       0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1.,
       0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1.,
       0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 1., 0.,
       1., 0., 1., 1., 0., 0., 0., 0., 0., 1., 0., 0., 1.])
```

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))
```

Output:

```
[[173  45]
 [ 17  16]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

Output:

	precision	recall	f1-score	support
0	0.91	0.79	0.85	218
1	0.26	0.48	0.34	33
accuracy			0.75	251
macro avg	0.59	0.64	0.59	251
weighted avg	0.83	0.75	0.78	251

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
def performance_metric(y_pred, y_test, linear_regression,X_test):
    print("The mean absolute error is", mean_absolute_error(y_test, y_pred))
    print("The mean squared error is", mean_squared_error(y_test, y_pred))
    print("The accuracy is", linear_regression.score(X_test, y_test))
performance_metric(y_pred, y_test, linear_regression, X_test)
```

Output:

```
The mean absolute error is 0.24701195219123506
The mean squared error is 0.24701195219123506
The accuracy is 0.0801683334100507
```

b) Also Perform Recursive Feature Elimination, Build Model, Evaluate the model

i. Recursive Feature Elimination

```
from numpy import mean
from numpy import std
from sklearn.datasets import make_regression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.feature_selection import RFE
from sklearn.tree import DecisionTreeRegressor
from sklearn.pipeline import Pipeline
rfe = RFE(estimator=LinearRegression(), n_features_to_select=5)
model = LinearRegression()
pipeline = Pipeline(steps=[('s',rfe),('m',model)])
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(pipeline, X, y, scoring='neg_mean_absolute_error', cv=cv,
n_jobs=-1, error_score='raise')
print('MAE: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

Output:

```
MAE: -0.231 (0.016)
```

ii. Build Model

```
estimator = LinearRegression()
selector = RFE(estimator, step=1)
selector = selector.fit(X, y)
X_RFE = X[X.columns[selector.support_]]
linear_regression.fit(X_RFE, y)
```

Output:

```
LinearRegression()
```

iii. Evaluation

```
from sklearn.model_selection import cross_val_score
def cross_validation(X, y):
    score = cross_val_score(linear_regression, X, y,cv=5, scoring="r2")
    print('R_squared Mean Score:',score.mean())
    print(score)
```

Output:

```
R_squared Mean Score: 0.052132621601214325
[ 0.08583778  0.02389953  0.07808371  0.07452545 -0.00168337]
R_squared Mean Score: 0.050419123099075105
[0.07314841 0.01634848 0.0761681  0.084291   0.00213963]
```

```
import statsmodels.api as sm
result = sm.OLS(y_train, X_train).fit()
print(result.summary())
```

Output:

OLS Regression Results

Dep. Variable:	obs_consequence_Yes	R-squared (uncentered):	0.209
Model:	OLS	Adj. R-squared (uncentered):	0.204
Method:	Least Squares	F-statistic:	43.97
Date:	Tue, 24 May 2022	Prob (F-statistic):	8.27e-48
Time:	22:14:48	Log-Likelihood:	-349.19
No. Observations:	1003	AIC:	710.4
Df Residuals:	997	BIC:	739.9
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
family_history_Yes	0.0383	0.024	1.579	0.115	-0.009	0.086
treatment_Yes	0.0783	0.024	3.207	0.001	0.030	0.126
mental_health_consequence_Yes	0.1448	0.027	5.282	0.000	0.091	0.199
phys_health_consequence_Yes	0.0383	0.055	0.692	0.489	-0.070	0.147
care_options_Yes	0.0436	0.024	1.840	0.066	-0.003	0.090
Age	0.0014	0.001	2.556	0.011	0.000	0.002

Omnibus:	306.934	Durbin-Watson:	2.157
Prob(Omnibus):	0.000	Jarque-Bera (JB):	656.717
Skew:	1.783	Prob(JB):	2.49e-143
Kurtosis:	4.732	Cond. No.	170.

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Result:

Thus, the given process for processing the dataset has been carried out and a multiple regression model along with recursive feature elimination has been built successfully.

----- X -----

Date: 17.05.22, Tuesday **Exp. No.:** 09	**Classification of Naïve Bayes**

1. Build a model to perform Naïve Bayes classifier with following specifications:

a. Exploratory data analysis

b. Identify Numerical and Categorical variables

c. Identify the missing values

d. Apply feature engineering and apply feature scaling

e. Train the model

f. Test the model and display the results

g. Check accuracy score

h. Check for overfitting and underfitting

i. Summarize the performance of model using confusion matrix

j. Evaluate the model performance with classification report

k. Analyze the model performance visually.

Aim:

To build models to perform Gaussian and Multinomial Naïve Bayes classification for the given dataset.

Codes and Output:

```
import pandas as pd
data = pd.read_csv("mental-health-survey.csv")
```

i. Exploratory Data Analysis

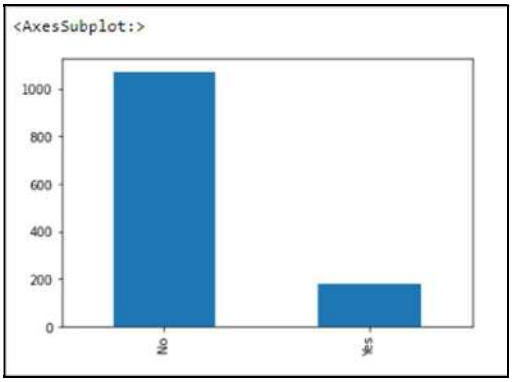
```
data.shape
```

Output:

```
import pandas as pd
data = pd.read_csv("mental-health-survey.csv")

#Exploratry data analysis
data.shape

(1254, 27)
```

```
import seaborn as sns
import matplotlib.pyplot as plt
data['obs_consequence'].value_counts().plot.bar()
```

Output:



ii. Identify Numerical and Categorical Data

```
import numpy as np
numeric_data = data.select_dtypes(include=[np.number])
categorical_data = data.select_dtypes(exclude=[np.number])
print("Number of numerical variables: ", numeric_data.shape[1])
print("Numerical attributes: ", numeric_data.columns)
```

Output:

```
Number of numerical variables:  1
Numerical attributes:  Index(['Age'], dtype='object')
```

```
print("Number of categorical variables: ", categorical_data.shape[1])
print("Categorical attributes: ")
print(categorical_data.columns)
```

Output:

```
Number of categorical variables:  26
Categorical attributes:
Index(['Timestamp', 'Gender', 'Country', 'state', 'self_employed',
       'family_history', 'treatment', 'work_interfere', 'no_employees',
       'remote_work', 'tech_company', 'benefits', 'care_options',
       'wellness_program', 'seek_help', 'anonymity', 'leave',
       'mental_health_consequence', 'phys_health_consequence', 'coworkers',
       'supervisor', 'mental_health_interview', 'phys_health_interview',
       'mental_vs_physical', 'obs_consequence', 'comments'],
      dtype='object')
```

iii. Identify the Missing Values

```
data.isnull().sum()
```

Output:

```
Timestamp                      0
Age                            0
Gender                         0
Country                        0
state                        513
self_employed                 18
family_history                 0
treatment                      0
work_interfere               263
no_employees                   0
remote_work                    0
tech_company                   0
benefits                       0
care_options                   0
wellness_program               0
seek_help                      0
anonymity                      0
leave                          0
mental_health_consequence      0
phys_health_consequence        0
coworkers                      0
supervisor                     0
mental_health_interview        0
phys_health_interview          0
mental_vs_physical             0
obs_consequence                0
comments                    1091
dtype: int64
```

iv. Apply Feature Engineering and Feature Scaling

```
#Feature Engineering
encD = pd.get_dummies(data, columns = ['obs_consequence'])
encD.columns
```

Output:

```
Index(['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed',
       'family_history', 'treatment', 'work_interfere', 'no_employees',
       'remote_work', 'tech_company', 'benefits', 'care_options',
       'wellness_program', 'seek_help', 'anonymity', 'leave',
       'mental_health_consequence', 'phys_health_consequence', 'coworkers',
       'supervisor', 'mental_health_interview', 'phys_health_interview',
       'mental_vs_physical', 'comments', 'obs_consequence_No',
       'obs_consequence_Yes'],
      dtype='object')
```

```
cat_vars=['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(encD[var], prefix=var)
    data1=encD.join(cat_list)
    encD=data1
data_vars=encD.columns.values.tolist()
to_keep=[i for i in data_vars if i not in cat_vars]
data_final=encD[to_keep]
data_final.columns.values
```

Output:

```
array(['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed',
       'work_interfere', 'no_employees', 'remote_work', 'tech_company',
       'benefits', 'wellness_program', 'seek_help', 'anonymity', 'leave',
       'coworkers', 'supervisor', 'mental_health_interview',
       'phys_health_interview', 'mental_vs_physical', 'comments',
       'obs_consequence_No', 'obs_consequence_Yes', 'family_history_No',
       'family_history_Yes', 'treatment_No', 'treatment_Yes',
       'mental_health_consequence_Maybe', 'mental_health_consequence_No',
       'mental_health_consequence_Yes', 'phys_health_consequence_Maybe',
       'phys_health_consequence_No', 'phys_health_consequence_Yes',
       'care_options_No', 'care_options_Not sure', 'care_options_Yes'],
      dtype=object)
```

```
#Feature Scaling
import math
convert_dict = {'Age': float}
encD = encD.astype(convert_dict)
mAge = encD['Age'].mean()
print("Mean age: ", mAge)
#Sum of squares of differences of age and mean age
sum = 0
c = 0
for i in encD['Age']:
    encD['Age'][c] = abs(i - mAge)
    sum = sum + encD['Age'][c]**2
    c = c+1
#Compute the normalization factor, NFactor, as standard deviate of age
NFactor = math.sqrt(sum/encD.shape[0])
print("Normalization Factor: ", NFactor)
```

```
#Bring 'Age' values within range
c = 0
for i in encD['Age']:
    if i >= NFactor:
        encD['Age'][c] = encD['Age'][c] % NFactor
    c+=1
#Perform normalization
c = 0
for i in encD['Age']:
    encD['Age'][c] = encD['Age'][c] / NFactor
    c+=1
```

Output:

```
Mean age:  32.01913875598086

<ipython-input-164-8a970939e9ee>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
rsus-a-copy
  encD['Age'][c] = abs(i - mAge)

Normalization Factor:  7.3720635583490735
```

```
#Normalized/Scaled 'Age' feature
encD['Age']
```

Output:

```
0       0.675640
1       0.625171
2       0.002596
3       0.138243
4       0.138243
          ...
1249    0.816479
1250    0.002596
1251    0.268698
1252    0.896465
1253    0.952127
Name: Age, Length: 1254, dtype: float64
```

v. Train the Model

```
#Split into input attributes and target
cols=['family_history_Yes', 'treatment_Yes','mental_health_consequence_Yes',
'phys_health_consequence_Yes',
    'care_options_Yes', 'Age']
X=encD[cols]
y=encD['obs_consequence_Yes']
X.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1254 entries, 0 to 1253
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   family_history_Yes            1254 non-null   uint8
 1   treatment_Yes                 1254 non-null   uint8
 2   mental_health_consequence_Yes 1254 non-null   uint8
 3   phys_health_consequence_Yes   1254 non-null   uint8
 4   care_options_Yes              1254 non-null   uint8
 5   Age                           1254 non-null   float64
dtypes: float64(1), uint8(5)
memory usage: 16.0 KB
```

```
#Split into train 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=0.2, random_state=40)
```

Gaussian Naive Bayes

Train the Model

```
#Train the model
from sklearn.naive_bayes import GaussianNB
gNBmodel = GaussianNB()
gNBmodel.fit(X_train,y_train)
```

Output:

```
#Train the model
from sklearn.naive_bayes import GaussianNB
gNBmodel = GaussianNB()
gNBmodel.fit(X_train,y_train)

GaussianNB()
```

vi. Test Model and Display Results

```
y_pred = gNBmodel.predict(X_test)
y_pred
```

Output:

```
array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0], dtype=uint8)
```

vii. Accuracy Score

```
print('Accuracy of Gaussian Naive Bayes on test set: {:.2f}'.format
(gNBmodel.score(X_test, y_test)))
```

Output:

```
#Accuracy score
print('Accuracy of Gaussian Naive Bayes on test set: {:.2f}'.format(gNBmodel.score(X_test, y_test)))
```

```
Accuracy of Gaussian Naive Bayes on test set: 0.87
```

viii. Check for Overfitting and Underfitting

```
from sklearn.metrics import mean_absolute_error
y_train_pred = gNBmodel.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_pred)
print(mae_train)
print(mae_test)
```

Output:

```
31.835493519441673
24.41832669322709
```

```
if mae_train < mae_test:
   print("Overfitting is present")
else:
   print("Underfitting is present")
```

Output:

```
Underfitting is present
```

ix. Summarise Performance with Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Output:

```
array([[209,   9],
       [ 24,   9]], dtype=int64)
```

x. Evaluate the Model with Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Output:

```
              precision    recall  f1-score   support

           0       0.90      0.96      0.93       218
           1       0.50      0.27      0.35        33

    accuracy                           0.87       251
   macro avg       0.70      0.62      0.64       251
weighted avg       0.84      0.87      0.85       251
```

xi. Analyse model performance visually

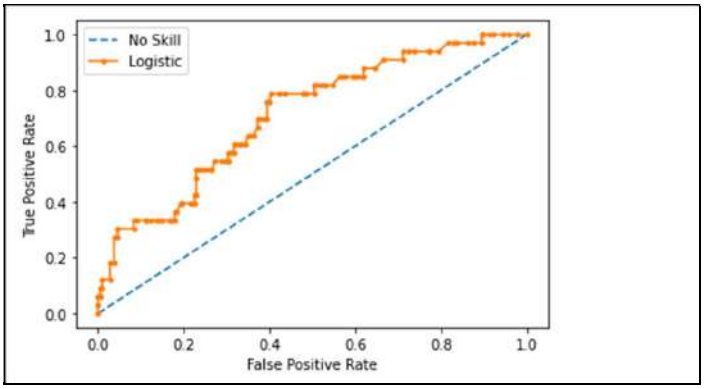
```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
ns_probs = [0 for _ in range(len(y_test))]
lr_probs = gNBmodel.predict_proba(X_test)
lr_probs = lr_probs[:, 1]
ns_auc = roc_auc_score(y_test, ns_probs)
lr_auc = roc_auc_score(y_test, lr_probs)
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
```

Output:

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.712
```

```
ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.show()
```

Output:



Multinomial Naive Bayes

Train the Model

```
#Train the model
from sklearn.naive_bayes import MultinomialNB
mNBmodel = MultinomialNB()
mNBmodel.fit(X_train,y_train)
```

Output:

```
MultinomialNB()
```

vi. Test Model and Display the Result

```
y_pred = mNBmodel.predict(X_test)
y_pred
```

Output:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=uint8)
```

vii. Accuracy Score

```
print('Accuracy of Multinomial Naive Bayes on test set:
{:.2f}'.format(mNBmodel.score(X_test, y_test)))
```

Output:

```
#Accuracy score
print('Accuracy of Multinomial Naive Bayes on test set: {:.2f}'.format(mNBmodel.score(X_test, y_test)))

Accuracy of Multinomial Naive Bayes on test set: 0.87
```

viii. Check for Overfitting and Underfitting

```
from sklearn.metrics import mean_absolute_error
y_train_pred = mNBmodel.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_pred)
print(mae_train)
print(mae_test)
```

Output:

```
37.88135593220339
33.52589641434263
```

```
if mae_train < mae_test:
    print("Overfitting is present")
else:
    print("Underfitting is present")
```

Output:

```
Underfitting is present
```

ix. Summarise Performance with Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Output:

```
array([[218,   0],
       [ 33,   0]], dtype=int64)
```

x. Evaluate the Model with Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Output:

	precision	recall	f1-score	support
0	0.87	1.00	0.93	218
1	0.00	0.00	0.00	33
accuracy			0.87	251
macro avg	0.43	0.50	0.46	251
weighted avg	0.75	0.87	0.81	251

xi. Analyse model performance visually

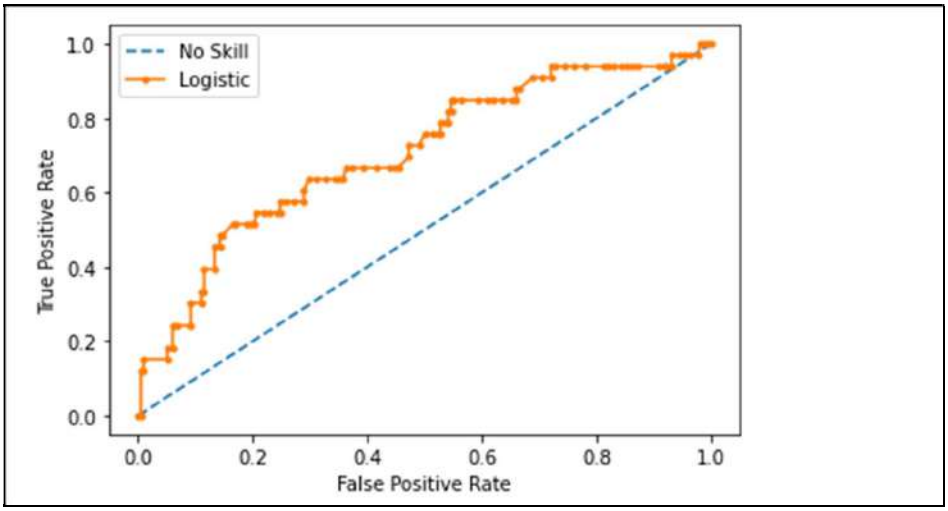
```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
ns_probs = [0 for _ in range(len(y_test))]
lr_probs = mNBmodel.predict_proba(X_test)
lr_probs = lr_probs[:, 1]
ns_auc = roc_auc_score(y_test, ns_probs)
lr_auc = roc_auc_score(y_test, lr_probs)
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
```

Output:

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.706
```

```
ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.show()
```

Output:



Result:

Thus, the models to perform Gaussian and Multinomial Naïve Bayes classification for the given dataset have been built successfully.

2. Build a model to perform Gaussian Naïve Bayes classifier for the IRIS Dataset:

Aim:

To build model to perform Gaussian Naïve Bayes classification for the IRIS dataset.

Codes and Output:

```
import pandas as pd
data = pd.read_csv("Iris.csv")
```

i. Exploratory Data Analysis

```
data.shape
```

Output:

```
(150, 5)
```

```
data.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   sepallength  150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   class        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

ii. Identify Numerical and Categorical Data

```
import numpy as np
numeric_data = data.select_dtypes(include=[np.number])
categorical_data = data.select_dtypes(exclude=[np.number])
print("Number of numerical variables: ", numeric_data.shape[1])
print("Numerical attributes: ", numeric_data.columns)
```

Output:

```
Number of numerical variables:  4
Numerical attributes:  Index(['sepallength', 'sepalwidth', 'petallength', 'petalwidth'], dtype='object')
```

```
print("Number of categorical variables: ", categorical_data.shape[1])
print("Categorical attributes: ")
print(categorical_data.columns)
```

Output:

```
Number of categorical variables:  1
Categorical attributes:
Index(['class'], dtype='object')
```

iii. Identify the Missing Values

```
data.isnull().sum()
```

Output:

```
#Identify the missing values
data.isnull().sum()

sepallength    0
sepalwidth     0
petallength    0
petalwidth     0
class          0
dtype: int64
```

iv. Apply Feature Engineering and Feature Scaling

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
data['class']= label_encoder.fit_transform(df['class'])
#Feature scaling - min max scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled = scaler.fit_transform(data)
encD = data
encD.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   sepallength  150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   class        150 non-null    int32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

v. Train the Model

```
#Split into input attributes and target
cols=['sepallength', 'sepalwidth', 'petallength', 'petalwidth']
X=encD[cols]
y=encD['class']
#Split into train 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=0.2, random_state=40)
#Train the model
from sklearn.naive_bayes import GaussianNB
gNBmodel = GaussianNB()
gNBmodel.fit(X_train,y_train)
```

Output:

```
GaussianNB()
```

vi. Test Model and Display Results

```
y_pred = gNBmodel.predict(X_test)
y_pred
```

Output:

```
array([0, 1, 2, 2, 1, 2, 1, 1, 1, 0, 1, 0, 0, 2, 1, 2, 2, 2, 1, 1, 2, 2,
       1, 0, 1, 0, 0, 2, 0, 1])
```

vii. Accuracy Score

```
print('Accuracy of Gaussian Naive Bayes on test set: {:.2f}'.format
(gNBmodel.score(X_test, y_test)))
```

Output:

```
#Accuracy score
print('Accuracy of Gaussian Naive Bayes on test set: {:.2f}'.format(gNBmodel.score(X_test, y_test)))

Accuracy of Gaussian Naive Bayes on test set: 1.00
```

viii. Check for Overfitting and Underfitting

```
from sklearn.metrics import mean_absolute_error
y_train_pred = gNBmodel.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_pred)
print(mae_train)
print(mae_test)
```

Output:

```
0.041666666666666664
0.0
```

```
if mae_train < mae_test:
    print("Overfitting is present")
else:
    print("Underfitting is present")
```

Output:

```
Neither underfit or overfit
```

ix. Summarise Performance with Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Output:

```
array([[ 8,  0,  0],
       [ 0, 12,  0],
       [ 0,  0, 10]], dtype=int64)
```

x. Evaluate the Model with Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Output:

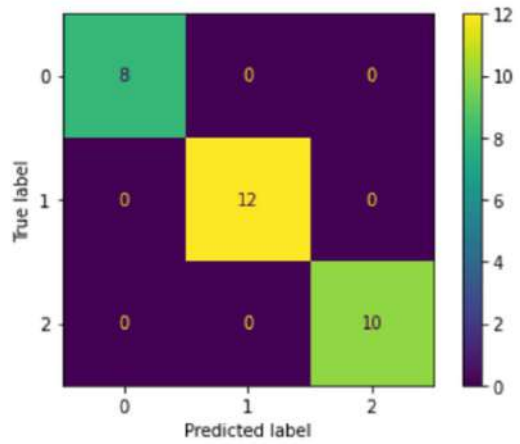
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00        12
           2       1.00      1.00      1.00        10

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

xi. Analyse model performance visually

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=gNBmodel.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=gNBmodel.classes_)
disp.plot()
```

Output:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x23c51b02160>
```



Result:

Thus, the model to perform Gaussian Naïve Bayes classification for the IRIS dataset has been built successfully.

3. Build a model to perform Gaussian Naïve Bayes classifier for the Diabetes Dataset:

Aim:

To build model to perform Gaussian Naïve Bayes classification for the Diabetes dataset.

Codes and Output:

```
import pandas as pd
data = pd.read_csv("diabetes.csv")
```

i. Exploratory Data Analysis

```
data.shape
```

Output:

```
(768, 9)
```

```
data.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

ii. Identify Numerical and Categorical Data

```
import numpy as np
numeric_data = data.select_dtypes(include=[np.number])
categorical_data = data.select_dtypes(exclude=[np.number])
print("Number of numerical variables: ", numeric_data.shape[1])
print("Numerical attributes: ", numeric_data.columns)
```

Output:

```
Number of numerical variables:  9
Numerical attributes:  Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
print("Number of categorical variables: ", categorical_data.shape[1])
print("Categorical attributes: ")
print(categorical_data.columns)
```

Output:

```
Number of categorical variables:  0
Categorical attributes:
Index([], dtype='object')
```

iii. Identify the Missing Values

```
data.isnull().sum()
```

Output:

```
#missing values
data.isnull().sum()

Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

iv. Apply Feature Engineering and Feature Scaling

```
from sklearn import preprocessing
#Feature scaling - min max scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled = scaler.fit_transform(data)
encD = data
encD.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

v. Train the Model

```
#Split into input attributes and target
cols=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
X=encD[cols]
y=encD['Outcome']
#Split into train 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=0.2, random_state=40)
#Train the model
from sklearn.naive_bayes import GaussianNB
gNBmodel = GaussianNB()
gNBmodel.fit(X_train,y_train)
```

Output:

```
GaussianNB()
```

vi. Test Model and Display Results

```
y_pred = gNBmodel.predict(X_test)
y_pred
```

Output:

```
array([1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0],
      dtype=int64)
```

vii. Accuracy Score

```
print('Accuracy of Gaussian Naive Bayes on test set: {:.2f}'.format
(gNBmodel.score(X_test, y_test)))
```

Output:

```
#Accuracy
print('Accuracy of Gaussian Naive Bayes on test set: {:.2f}'.format (gNBmodel.score(X_test, y_test)))

Accuracy of Gaussian Naive Bayes on test set: 0.71
```

viii. Check for Overfitting and Underfitting

```
from sklearn.metrics import mean_absolute_error
y_train_pred = gNBmodel.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_pred)
print(mae_train)
print(mae_test)
```

Output:

```
0.22312703583061888
0.2857142857142857
```

```
if mae_train < mae_test:
    print("Overfitting is present")
else:
    print("Underfitting is present")
```

Output:

```
Overfitting is present
```

ix. Summarise Performance with Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Output:

```
array([[77, 18],
       [26, 33]], dtype=int64)
```

x. Evaluate the Model with Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Output:

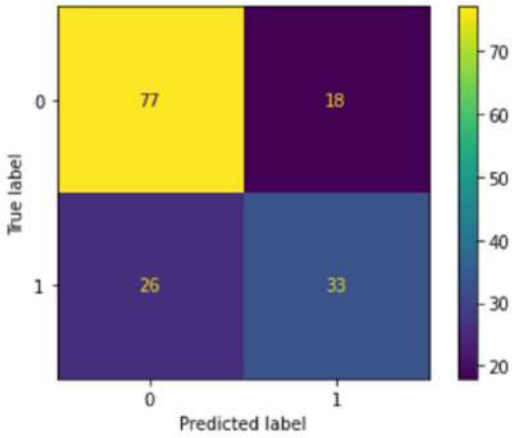
```
              precision    recall  f1-score   support

           0       0.75      0.81      0.78        95
           1       0.65      0.56      0.60        59

    accuracy                           0.71       154
   macro avg       0.70      0.68      0.69       154
weighted avg       0.71      0.71      0.71       154
```

xi. Analyse model performance visually

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=gNBmodel.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=gNBmodel.classes_)
disp.plot()
```

Output:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2051855e400>
```



Result:

Thus, the model to perform Gaussian Naïve Bayes classification for the Diabetes dataset has been built successfully.

----- X -----

Date: 17.05.22, Tuesday **Exp. No.:** 10	**Visualization Tools in Python**

1. Explore the following plotting/visualization tools in python:

Matplotlib, diagram, mayai, seaborn, VisPy, PyQtGraph, Chartify

Aim:

To study about the visualization tools in python that are used for data analytics.

Theory:

a. Matplotlib

Matplotlib is a visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

Functions in Matplotlib

S. No	Function	Description
1.	matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)	Make a bar plot for two attributes
2.	matplotlib.pyplot.boxplot(x, notch=None)	Make a box and whisker plot
3.	hexbin( ) : matplotlib.pyplot.hexbin(x, y, C=None, gridsize=100, bins=None, xscale='linear', yscale='linear')	Make a 2D hexagonal binning plot of points x, y
4.	matplotlib.pyplot.hist(x, bins=None, range=None, density=None, weights=None)	Plot a histogram
5.	matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None)	Plot a pie chart
6.	matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)	Plot y versus x as lines and/or markers
7.	matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None)	A scatter plot of y vs x with varying marker size and/or color

b. Diagram

Diagrams lets you draw the cloud system architecture in Python code. It was made for prototyping a new system architecture design without any design tools. You can also describe or visualize the existing system architecture as well.

Functions in Diagram

S. No	Function	Description
1.	Diagram("Diagram Name")	Create a diagram context with Diagram class
2.	ELB("Node Name")	To perform elastic load balance
3.	Route53("dns")	To establish routing connection
4.	Cluster("Cluster Name")	To create a cluster context
5.	RDS("Node Type")	To create a relational database service instance
6.	SQS("event queue")	To create a simple queue service
7.	Edge(color="*", style="*")	To create an edge between two nodes

c. MayaVi

It is a free, easy to use scientific data visualizer. It is written in Python and uses the amazing Visualization Toolkit (VTK) for the graphics. It provides a GUI written using Tkinter. MayaVi is free and distributed and it is also cross platform.

Functions in MayaVi

S. No	Function	Description
1.	barchart(*args, **kwargs)	Plots vertical glyphs (like bars) scaled vertical, to do histogram-like plots.
2.	contour3d(*args, **kwargs)	Plots iso-surfaces for a 3D volume of data supplied as arguments
3.	contour_surf(*args, **kwargs)	Plots the contours of a surface using grid-spaced data for elevation supplied as a 2D array
4.	flow(*args, **kwargs)	Creates a trajectory of particles following the flow of a vector field
5.	imshow(*args, **kwargs)	View a 2D array as an image
6.	mesh(*args, **kwargs)	Plots a surface using grid-spaced data supplied as 2D arrays
7.	plot3d(*args, **kwargs)	Draws lines between points

d. Seaborn

Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

Functions in Seaborn

S. No	Function	Description
1.	seaborn.distplot(x, bins = *, kde = *)	To plot approximate probability density across the y-axis
2.	pie( )	To plot a pie chart
3.	barh( )	To plot a bar graph
4.	scatter(x, y)	To plot a scatter plot
5.	jointplot(x, y, kind = 'reg')	Creates a regression line between 2 numerical parameters in the jointplot(scatterplot) to help visualize their linear relationships
6.	pairplot(x)	To plot multiple scatter plots at a time
7.	heatmap( )	To generate a heatmap

e. VisPy

VisPy is a library designed for use by data scientists, and it's intended to make creating complex, interactive visualizations as quickly as possible. VisPy takes advantage of the extra processing power granted by GPUs. This makes the rendering of large datasets faster than other libraries. VisPy graphs and charts can be scaled up, making visualizations out of thousands or even millions of points of data.

Functions in VisPy

S. No	Function	Description
1.	vispy.color.BaseColormap(colors=None)	Class representing a colormap
2.	texture_lut()	Return a texture2D object for LUT after its value is set. Can be None
3.	vispy.geometry.MeshData( )	Class for storing and operating on 3D mesh data
4.	vispy.geometry.curves.curve3_bezier(p1, p2, p3)	Generate the vertices for a quadratic Bezier curve
5.	vispy.io.imread(filename, format=None)	Read image data from disk
6.	vispy.io.imsave(filename, im, format=None)	Save image data to disk
7.	vispy.plot.fig.Fig( )	Create a figure window

f. PyQtGraph

PyQtGraph is a pure-python graphics and GUI library built on PyQt / PySide and numpy. It is intended for use in mathematics / scientific / engineering applications. Despite being written entirely in python, the library is very fast due to its heavy leverage of NumPy for number crunching and Qt's GraphicsView framework for fast display.

Functions in PyQtGraph

S. No	Function	Description
1.	pyqtgraph.plot( )	Create a new plot window showing data
2.	PlotWidget.plot( )	Add a new set of data to an existing plot widget
3.	GraphicsLayout.addPlot( )	Add a new plot to a grid of plots
4.	pyqtgraph.image( )	To display 2D or 3D data
5.	pyqtgraph.opengl.GLViewWidget()	To create a widget to display 3D objects
6.	pyqtgraph.opengl.GLGridItem()	To create a grid
7.	mkPen( )	To build a pen with desired configurations

g. Chartify

Chartify is an open-source data visualization library from Spotify that makes it easy for data analysts to create charts and graphs. Chartify is built on top of Bokeh, which is a very popular data visualization library.

Functions in Chartify

S. No	Function	Description
1.	chartify.Chart( )	create a basic chart
2.	chartify.Chart(x_axis_type='datetime')	To set the datatype of the x-axis
3.	set_title( )	To set the title of the chart
4.	set_subtitle( )	To set the subtitle of the chart
5.	axes.set_xaxis_label( )	To set the label of the x-axis
6.	plot.scatter(dataframe, x_col, y_col, color_col)	To draw a scatter plot
7.	plot.histogram(data_frame, values_column)	To draw a histogram

Result:

Thus, a study on the visualization tools in Python used for data analysis has been made.

----- X -----

Date: 24.05.22, Tuesday **Exp. No.:** 11	**Hadoop Installation**

1. Study on Hadoop Architecture and its components

Aim:

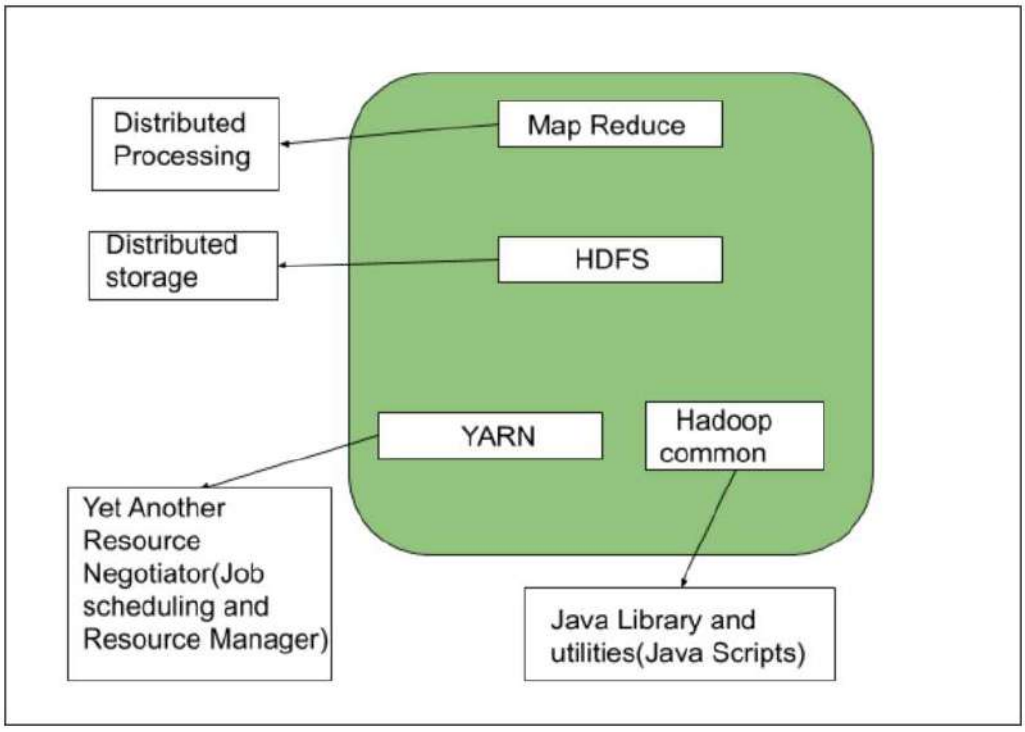
To study about the architecture of Hadoop and its components.

Theory:

Hadoop is a framework written in Java that utilizes a large cluster of commodity hardware to maintain and store big size data. Hadoop works on MapReduce Programming Algorithm that was introduced by Google.

The Hadoop Architecture Mainly consists of 4 components:

i. MapReduce
ii. HDFS(Hadoop distributed File System)
iii. YARN(Yet Another Resource Framework)
iv. Common Utilities or Hadoop Common



Components of Hadoop Architecture

1. MapReduce

MapReduce is an Algorithm or a data structure that is based on the YARN framework. The major feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster which Makes Hadoop fast. MapReduce has 2 main tasks which are divided phase-wise. In first phase, Map is utilized and in next phase Reduce is utilized.

The Input is provided to the Map( ) function and then its output is used as an input to the Reduce( ) function to receive the final output. The Input is a set of Data. The Map() function breaks data blocks into tuples that are key-value pairs. These key-value pairs are now sent as input to the Reduce( ). The Reduce( ) function then combines the broken key-value pairs based on its key value and form set of tuples. Some operation like sorting, summation, etc. is performed which is then sent to the final Output Node which generates the final output.

2. HDFS

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave. Both NameNode and DataNode are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the NameNode and DataNode software.

The NameNode manages the file system namespace by executing an operation like the opening, renaming and closing the files. The HDFS cluster contains several DataNodes each of which contains multiple data blocks that store data. The DataNodes read and write requests from the file system's clients.

3. YARN

YARN is a Framework on which MapReduce works. YARN performs 2 operations that are Job scheduling and Resource Management. The Purpose of Job scheduler is to divide a big task into small jobs so that each job can be assigned to various slaves in a Hadoop cluster and processing can be maximized. Job Scheduler also keeps track of which job is important, which job has more priority, dependencies between the jobs and all the other information like job timing, etc. And the use of resource manager is to manage all the resources that are made available for running a Hadoop cluster.

4. Hadoop Common

Hadoop common or Common utilities are nothing but Java library and Java files or the java scripts that are needed for all the other components present in a Hadoop cluster. These utilities are used by HDFS, YARN, and MapReduce for running the cluster. Hadoop Common

verifies that Hardware failure in a Hadoop cluster is common so it needs to be solved automatically in software by Hadoop Framework.

Result:

Thus, the architecture of Hadoop and its components have been studied.

2. Install and configure Hadoop in its two operating modes

Pseudo Distributed and Fully Distributed

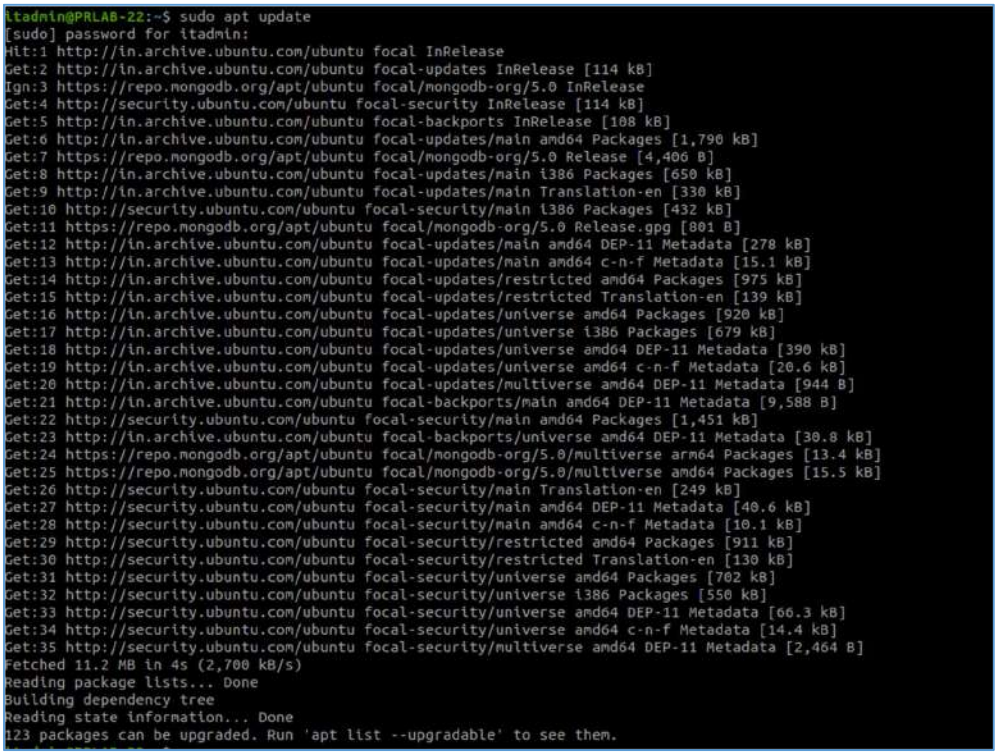
Aim:

To install Hadoop in its two operating modes.

Procedure:

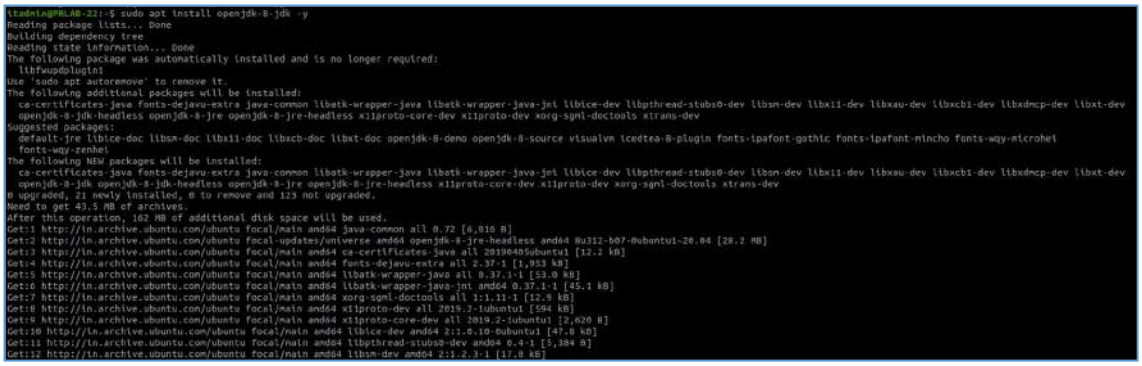
1. Update system before installation:

```
sudo apt – update
```



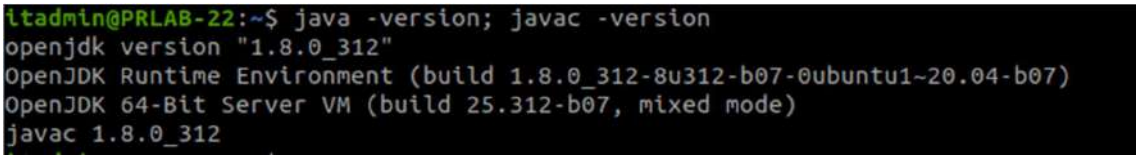
2. Install OpenJDK

```
sudo apt install openjdk-11-jdk
```



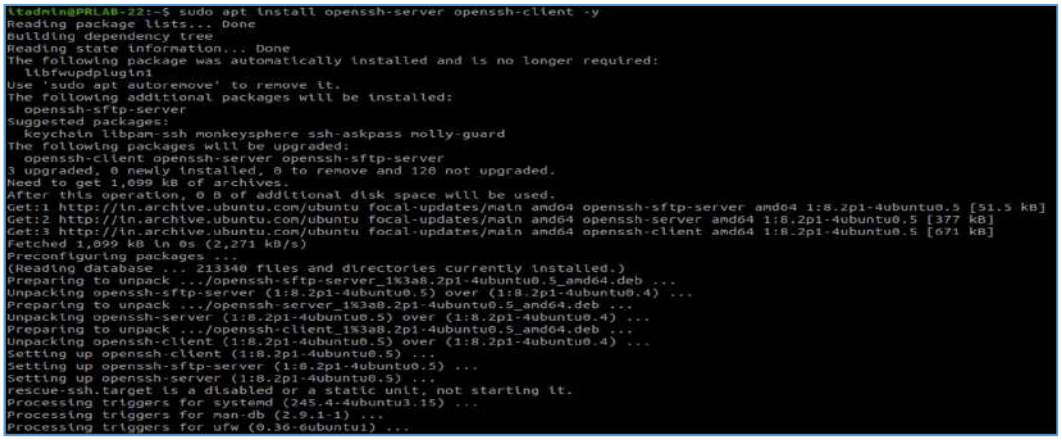
3. Verify intalled Java version

```
java -version; javac -version
```



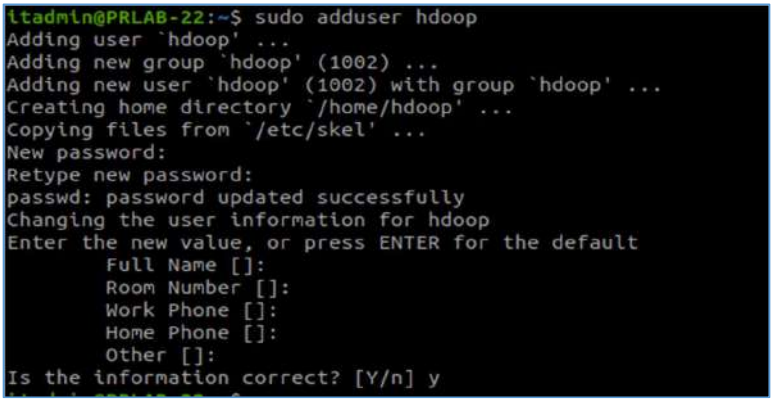
4. Install OpenSSH Server

```
sudo apt install openssh-server openssh-client -y
```



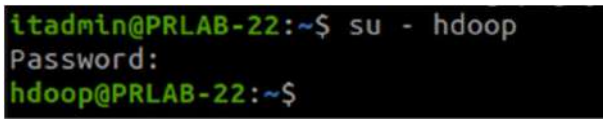
5. Create a Hadoop user

```
sudo adduser hdoop
```



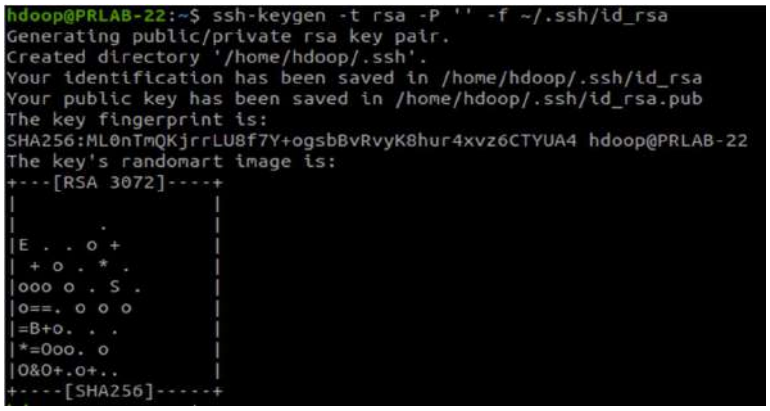
6. Switch to Hadoop user

```
su - hdoop
```



7. Generate an SSH key pair and define the location is is to be stored in

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```



8. Use the cat command to store the public key as authorized_keys in the ssh directory

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

9. Set the permissions for your user with the chmod command

```
chmod 0600 ~/.ssh/authorized_keys
```

```
hdoop@PRLAB-22:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
hdoop@PRLAB-22:~$ chmod 0600 ~/.ssh/authorized_keys
```

10. Verify everything is set up correctly by using the hdoop user to SSH to localhost

```
ssh localhost
```

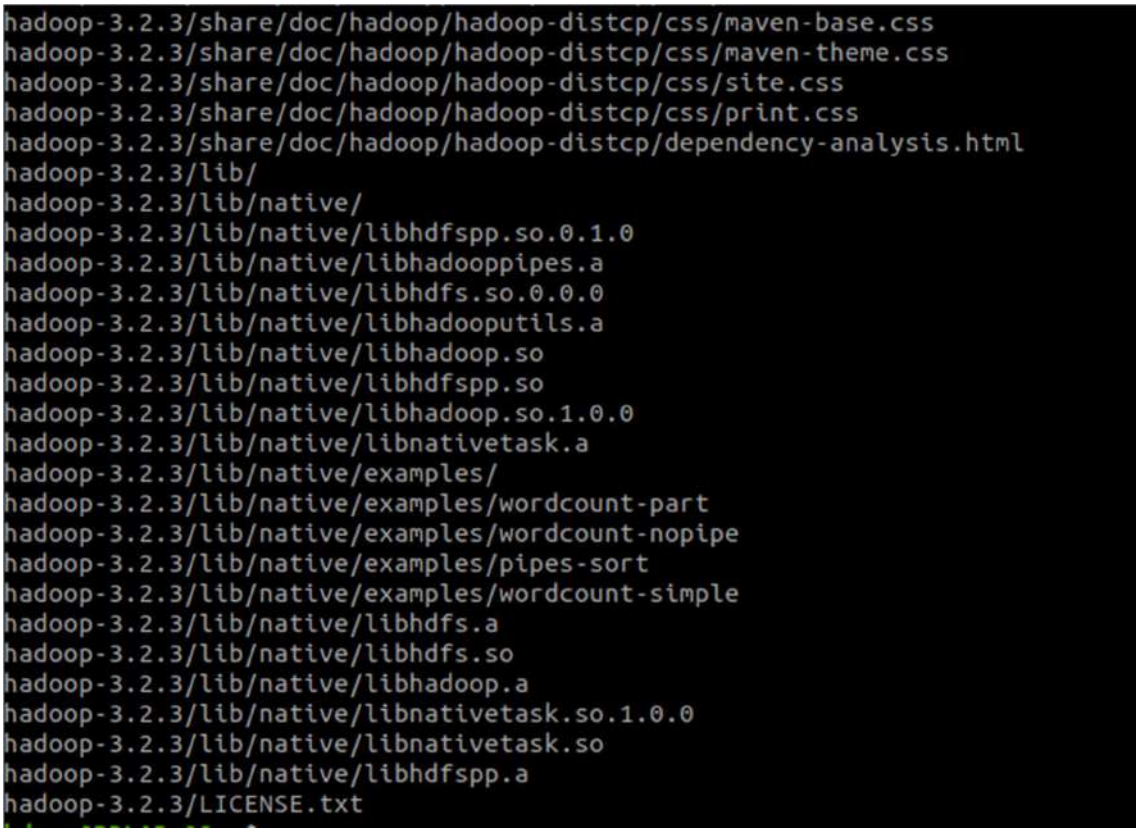
```
hdoop@PRLAB-22:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:SaTAp+60cPT8NCkg14kfRqJ8pzcBqlN5mV2T5rgLflE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-39-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

118 updates can be applied immediately.
67 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

11. Visit the official Apache Hadoop project page, and select the version of Hadoop you want to implement. Click the link to download locally in a machine or install the Hadoop framework in the system using wget command

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz
```

Extract the files to initiate the Hadoop installation.

```
tar xvzf hadoop-3.2.3.tar.gz
```



12. Rename the extracted directory by executing the command

```
mv hadoop-3.2.3 hadoop
```

13. Configure Java environment variables for setting up Hadoop

```
dirname $(dirname $(readlink -f $(which java)))
```

```
hdoop@PRLAB-22:~$ dirname $(dirname $(readlink -f $(which java)))
/usr/lib/jvm/java-8-openjdk-amd64/jre
```

14. Open the "~/.bashrc" file in "vi" text editor. Define the Hadoop environment variables.

```
vi ~/.bashrc
```

```
#Hadoop Related Options
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hdoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

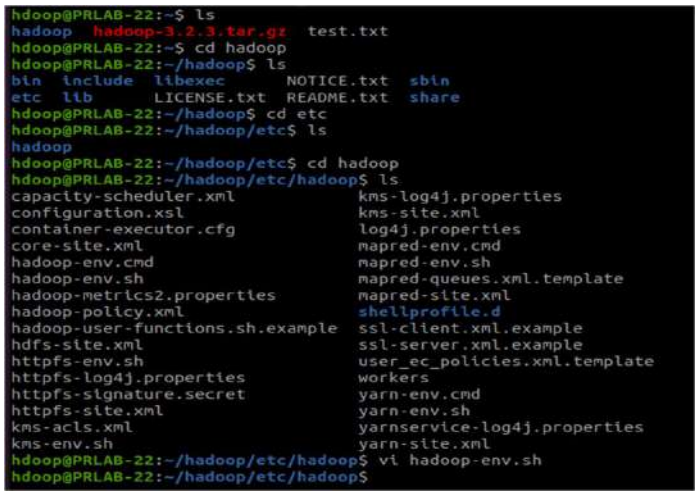
15. Apply the changes to the current running environment

```
source ~/.bashrc
```

```
hdoop@PRLAB-22:~$ vi ~/.bashrc
hdoop@PRLAB-22:~$ source ~/.bashrc
hdoop@PRLAB-22:~$
```

16. Add the full path to the OpenJDK installation on your system

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```



17. Configure Apache Hadoop on the Ubuntu system. For this, the step is to create two directories: datanode and namenode, inside the home directory of Hadoop.

```
mkdir -p ~/dfsdata/namenode
mkdir -p ~/dfsdata/datanode
```

18. Specify the URL for your NameNode, and the temporary directory Hadoop uses for the map and reduce process. Go to specific directory and open core-site.xml

```
hdoop@PRLAB-22:~/hadoop/etc/hadoop$ vi core-site.xml
```

Add the following configuration to override the default values for the temporary directory and add your HDFS URL to replace the default local file system setting

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/hdoop/tmpdata</value>
</property>
<property>

  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
```

19. Configure the file by defining the NameNode and DataNode storage directories in hdfs-site.xml

```
hdoop@PRLAB-22:~/hadoop/etc/hadoop$ vi hdfs-site.xml
```

Add the following configuration to the file and, if needed, adjust the NameNode and DataNode directories to your custom locations.

```
<configuration>

<property>
  <name>dfs.data.dir</name>
  <value>/home/hdoop/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hdoop/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property></configuration>
```

20. Use the following command to access the mapred-site.xml file and define MapReduce values:

```
hdoop@PRLAB-22:~/hadoop/etc/hadoop$ vi mapred-site.xml
```

Add the following configuration to change the default MapReduce framework name value to yarn

```
<configuration>

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

21. Edit yarn-site.xml File

Open the yarn-site.xml file in a text editor:
nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

Append the following configuration to the file

```
hdoop@PRLAB-22:~/hadoop/etc/hadoop$ vi yarn-site.xml
```

22. Format HDFS NameNode

```
hdfs namenode -format
```

23. Start Hadoop Cluster

```
./start-dfs.sh
```

```
hdoop@PRLAB-22:~$ cd hadoop
hdoop@PRLAB-22:~/hadoop$ ls
bin  etc  include  lib  libexec  LICENSE.txt  logs  NOTICE.txt  README.txt  sbin  share
hdoop@PRLAB-22:~/hadoop$ cd sbin
hdoop@PRLAB-22:~/hadoop/sbin$ ./start-dfs.sh
```

```
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [PRLAB-22]
```

24. Once the namenode, datanodes, and secondary namenode are up and running, start the YARN resource

```
./start-yarn.sh
```

```
hdoop@PRLAB-22:~/hadoop/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

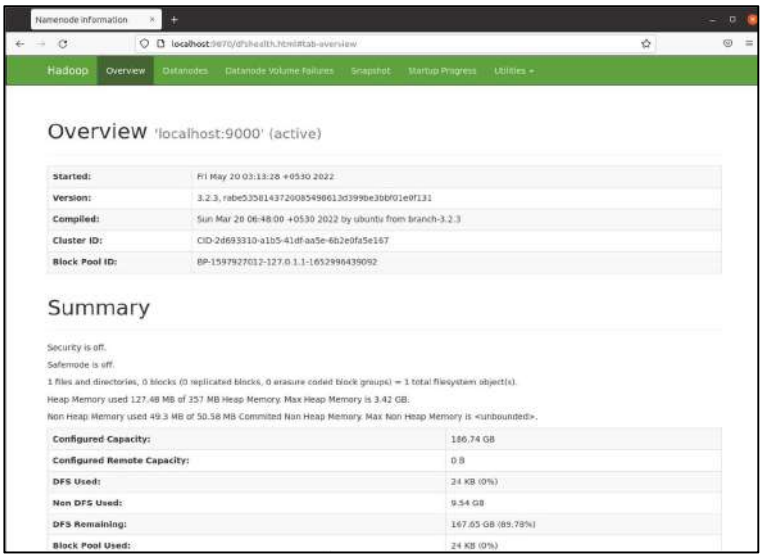
25. Check if all the daemons are active and running as Java processes

```
jps
```

```
hdoop@PRLAB-22:~/hadoop/sbin$ jps
36208 NodeManager
36627 Jps
35547 DataNode
36059 ResourceManager
35372 NameNode
35789 SecondaryNameNode
```

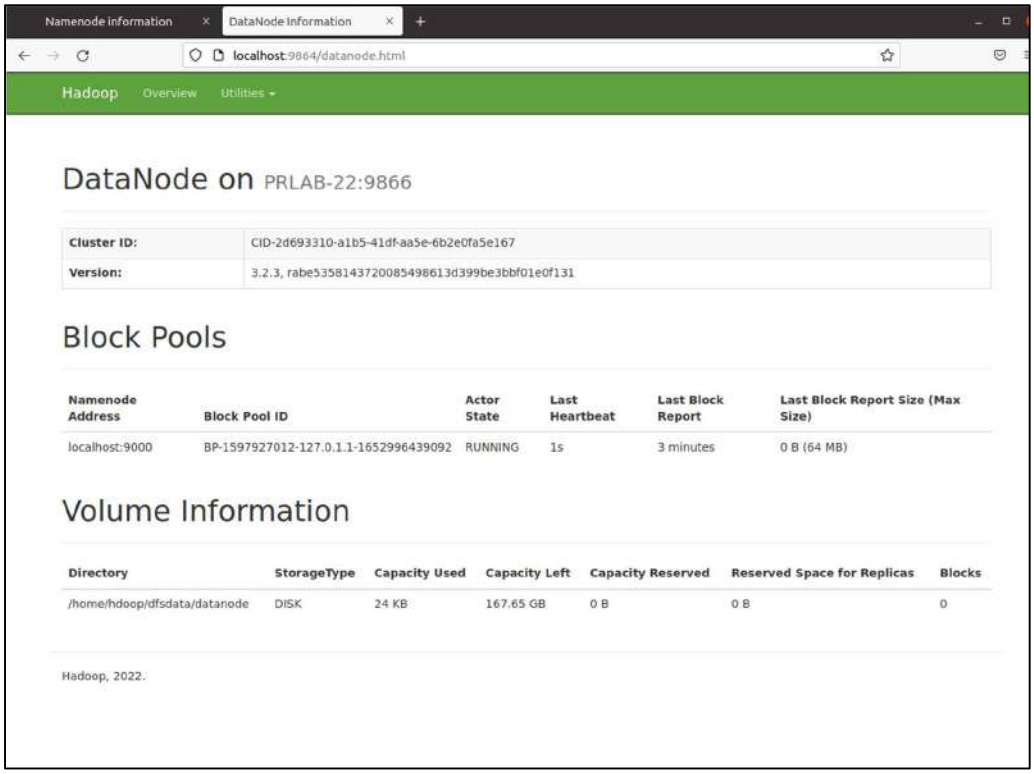
26. Access Hadoop UI from Browser

```
http://localhost:9870
```



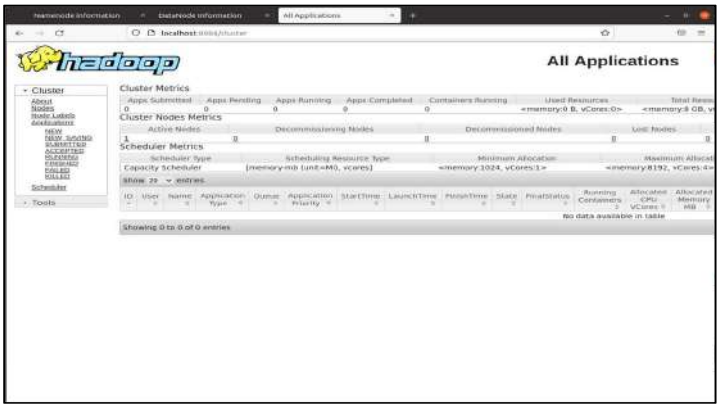
27. The default port 9864 is used to access individual DataNodes directly from browser

http://localhost:9864



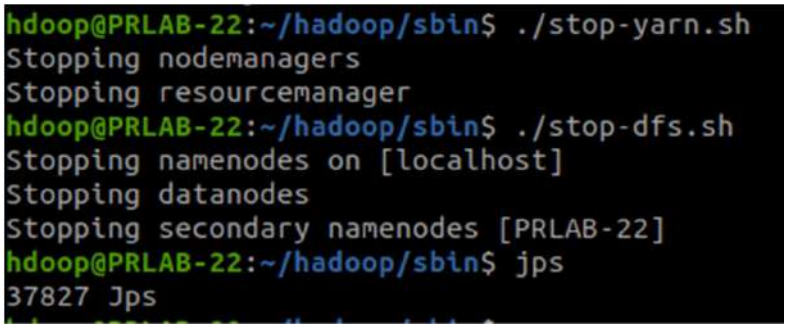
28. The YARN Resource Manager is accessible on port 8088. The Resource Manager is an invaluable tool that allows to monitor all running processes in Hadoop cluster

http://localhost:8088



29. Stop the NameNode, DataNode and YARN

```
./~stop-yarn.sh
```



Result:

Thus, Hadoop has been successfully installed.

----- X -----

Date: 24.05.22, Tuesday
Exp. No.: 12

File Management in Hadoop

1. Implement the following task in Hadoop:

a. Adding Directories.
b. Adding Files.
c. Retrieving files and its content.
d. Deleting files.

Aim:

To implement file management in the Hadoop File System of Hadoop.

Codes and Output:

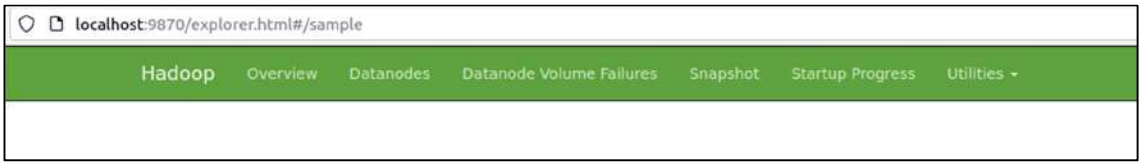
a) Adding Directories

```
hadoop fs -mkdir /sample
```

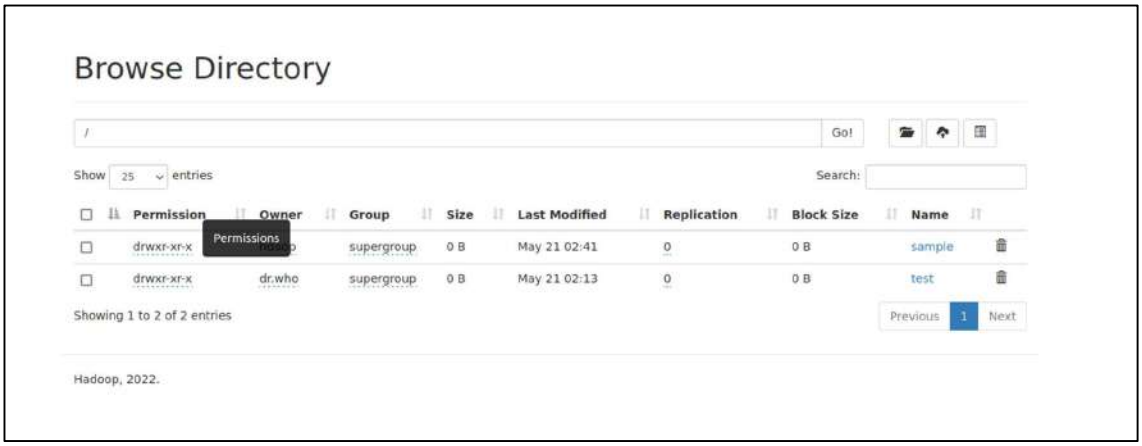
```
hdoop@PRLAB-22:~$ hadoop fs -mkdir /sample
```

Login to the web browser

```
http://localhost:9870
```



Under Utilities → click (Browse the file system option)



b) Adding Files

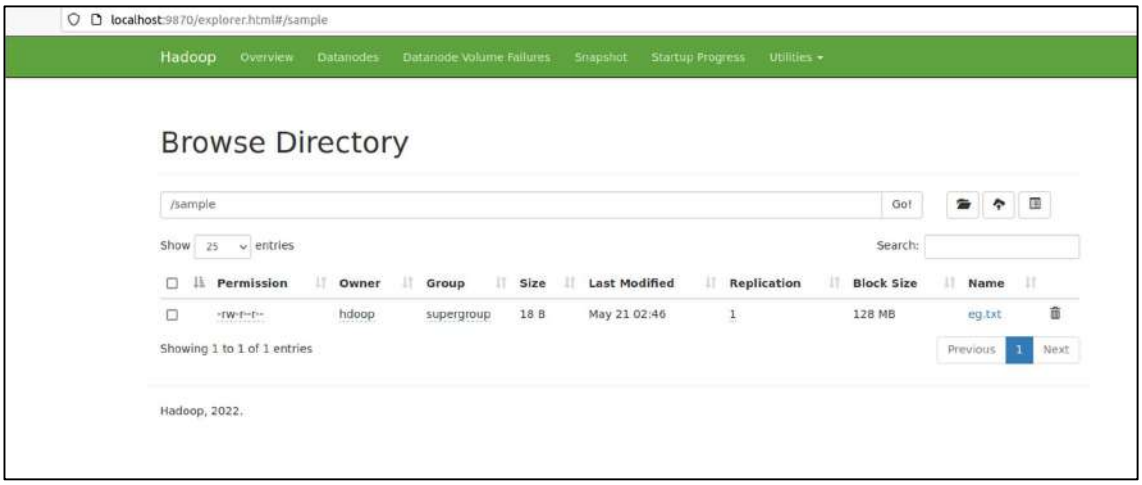
Create a new file in the terminal

```
vi eg.txt
```

```
hdoop@PRLAB-22:~$ vi eg.txt
```

Type contents and save it and exit out of the file. Move the file to the newly created directory.

```
hadoop fs -put eg.txt /sample
```



c) Retrieve File Contents

```
hadoop fs -cat /sample/eg.txt
```

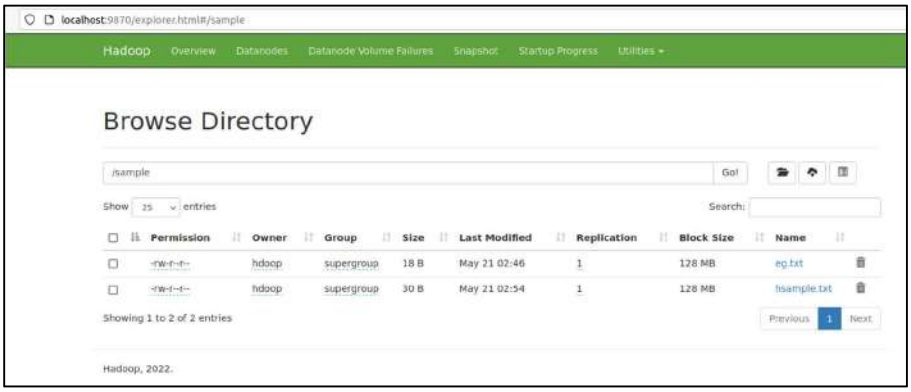
```
hdoop@PRLAB-22:~$ hadoop fs -cat /sample/eg.txt
welcome to hadoop
```

d) Delete File

Create a new file and move to the directory ‘sample’

```
vi hsample.txt
hadoop fs -put hsample.txt /sample
hadoop fs -cat /sample/hsample.txt
```

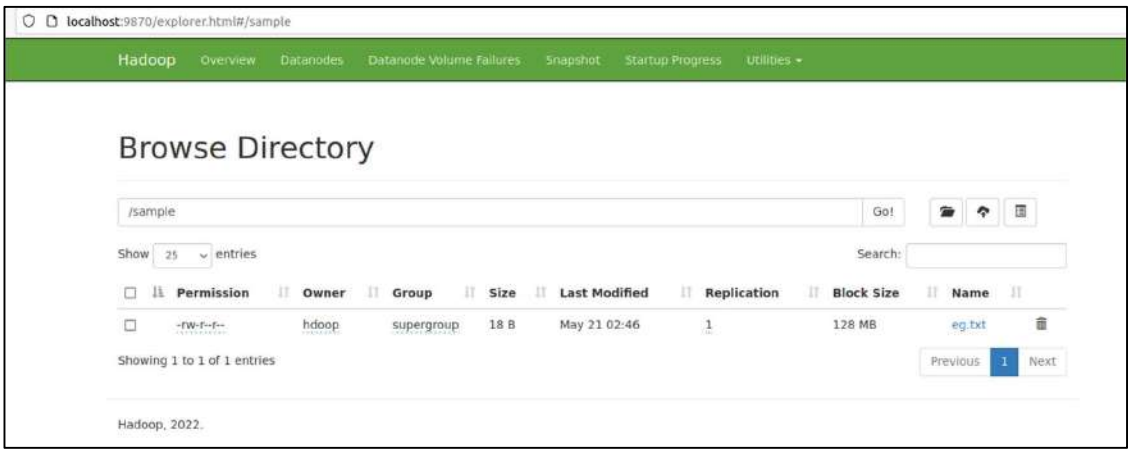
```
hdoop@PRLAB-22:~$ vi hsample.txt
hdoop@PRLAB-22:~$ hadoop fs -put hsample.txt /sample
hdoop@PRLAB-22:~$ hadoop fs -cat /sample/hsample.txt
Newly created file for hadoop
```



hadoop fs -rm /sample/hsample.txt

```
hdoop@PRLAB-22:~$ hadoop fs -rm /sample/hsample.txt
Deleted /sample/hsample.txt
```

After deletion



Result:

Thus, file management in HDFS of Hadoop has been successfully carried out.

----- X -----

Date: 31.05.22, Tuesday **Exp. No.:** 13	**MongoDB Data Replication**

1. Pick any management system of your choice and perform data replication in MongoDB

Aim:

To perform data replication in MongoDB.

Codes and Output:

1. Creating a node in port 27021

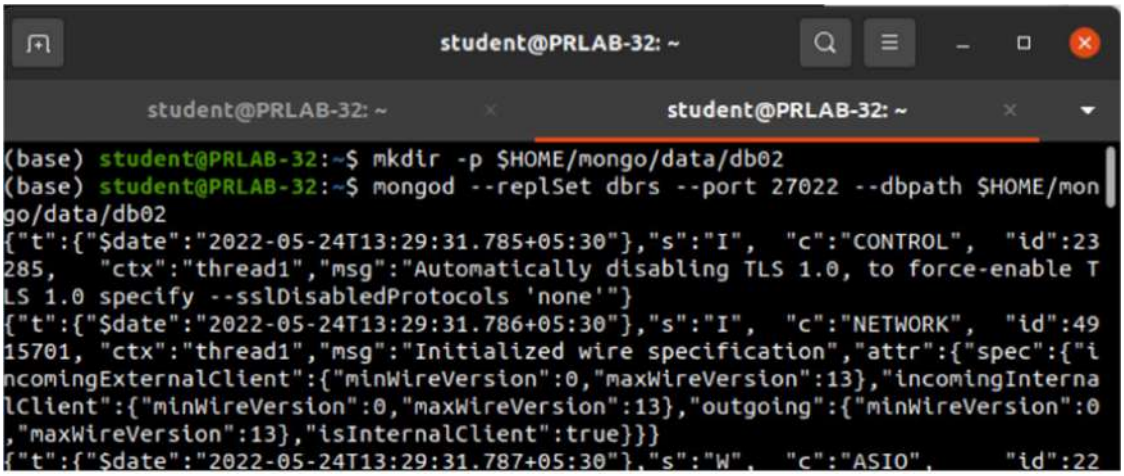
```
mkdir -p $HOME/mongo/data/db01
mongod --replSet dbrs --port 27021 --dbpath $HOME/mongo/data/db01
```

2. Open another terminal. Create node in a different port, 27022

```
mkdir -p $HOME/mongo/data/db02
mongod --replSet dbrs --port 27022 --dbpath $HOME/mongo/data/db02
```

3. In third terminal, create a third node

```
mkdir -p $HOME/mongo/data/db03
mongod --replSet dbrs --port 27023 --dbpath $HOME/mongo/data/db03
```



4. Connect to any one Mongo Daemon

```
mongo --port 27021
```


```
rsconf = {
 _id: "dbrs",
 members: [
 {
 _id: 0,
 host: "127.0.0.1:27021",
 priority: 3
 },
 {
 _id: 1,
 host: "127.0.0.1:27022",
 priority: 1,
 },
 {
 _id: 2,
 host: "127.0.0.1:27023",
 priority: 2
 }
 ]
};
```

```
db.getMongo().setReadPref('secondary')
```

```
        ]
}
> rs.initiate(rsconf);
{ "ok" : 1 }
```

Result:

Thus, data replication in MongoDB has been successfully carried out.

----- X -----

Date: 31.05.22, Tuesday
Exp. No.: 14

Classification - SVM

1. Build a model to perform SVM classifier with following specifications:

a. Exploratory data analysis
b. Explore missing values in variables
c. Handle outliers
d. Check the distribution of variables
e. Declare feature vector and target variable
f. Split data into separate training and test set
g. Perform Feature Scaling
h. Run SVM with default Hyperparameters
i. Run SVM with RBF kernel
j. Run SVM with Linear kernel
k. Run SVM with polynomial kernel
l. Run SVM with sigmoid kernel
m. Compare the train-set and test-set accuracy
n. Check for overfitting and underfitting
o. Summarize the performance of model using confusion matrix
p. Evaluate the model performance with classification report
q. Analyze the model performance visually

Aim:

To build models to perform classification using SVM classifier for the given dataset.

Codes and Output:

```
import pandas as pd
data = pd.read_csv("mental-health-survey.csv")
```

i. Exploratory Data Analysis

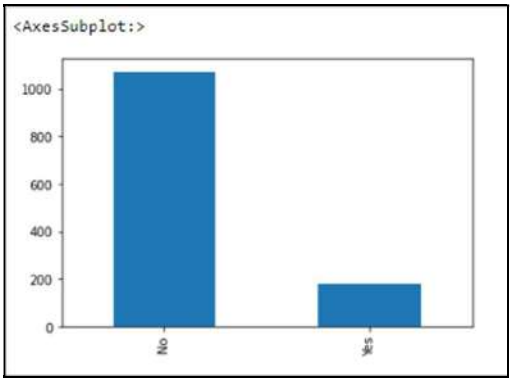
```
data.shape
```

Output:

```
(1254, 27)
```

```
import seaborn as sns
import matplotlib.pyplot as plt
data['obs_consequence'].value_counts().plot.bar()
```

Output:



ii. Explore Missing Values in Variables

```
data.isnull().sum()
```

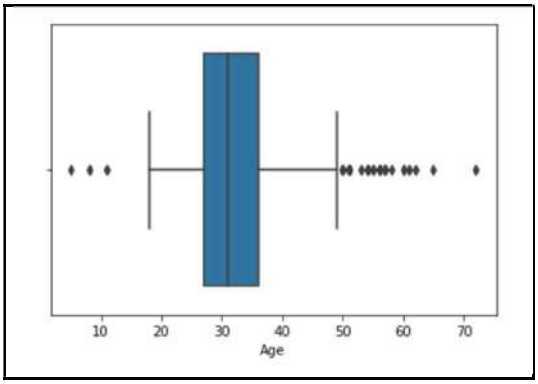
Output:

```
Timestamp                      0
Age                            0
Gender                         0
Country                        0
state                        513
self_employed                 18
family_history                 0
treatment                      0
work_interfere               263
no_employees                   0
remote_work                    0
tech_company                   0
benefits                       0
care_options                   0
wellness_program               0
seek_help                      0
anonymity                      0
leave                          0
mental_health_consequence      0
phys_health_consequence        0
coworkers                      0
supervisor                     0
mental_health_interview        0
phys_health_interview          0
mental_vs_physical             0
obs_consequence                0
comments                    1091
dtype: int64
```

iii. Handle Outliers

```
sns.boxplot(x='Age', data=df)
```

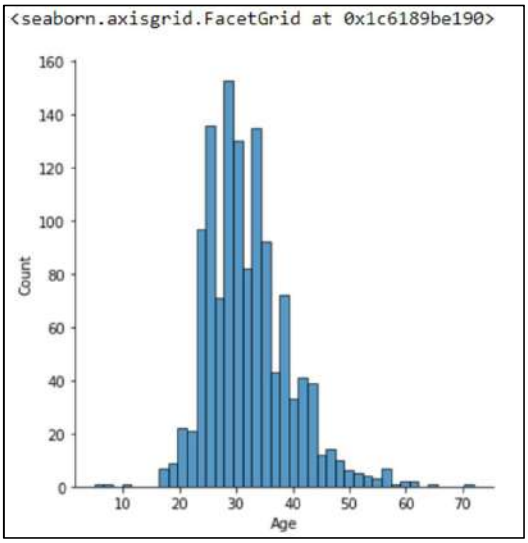
Output:



iv. Check the distribution of variable

```
import seaborn as sns
sns.displot(data, x="Age")
```

Output:



v. Declare Feature Vector and Target Variable

```
encD = pd.get_dummies(data, columns = ['obs_consequence'])
cat_vars=['family_history', 'treatment', 'mental_health_consequence',
'phys_health_consequence', 'care_options']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(encD[var], prefix=var)
    data1=encD.join(cat_list)
    encD=data1
data_vars=encD.columns.values.tolist()
to_keep=[i for i in data_vars if i not in cat_vars]
data_final=encD[to_keep]
data_final.columns.values
cols=['family_history_Yes', 'treatment_Yes','mental_health_consequence_Yes',
'phys_health_consequence_Yes',
     'care_options_Yes', 'Age']
X=encD[cols]
y=encD['obs_consequence_Yes']
X.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1254 entries, 0 to 1253
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   family_history_Yes            1254 non-null   uint8
 1   treatment_Yes                 1254 non-null   uint8
 2   mental_health_consequence_Yes 1254 non-null   uint8
 3   phys_health_consequence_Yes   1254 non-null   uint8
 4   care_options_Yes              1254 non-null   uint8
 5   Age                           1254 non-null   int64
dtypes: int64(1), uint8(5)
memory usage: 16.0 KB
```

vi. Split data into separate training 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=0.2, random_state=40)
```

vii. Perform Feature Scaling

```
#feature scaling
import math
convert_dict = {'Age': float}
X_train = X_train.astype(convert_dict)
mAge = X_train['Age'].mean()
print("Mean age: ", mAge)
#Sum of squares of differences of age and mean age
sum = 0
c = 0
for i in X_train['Age']:
    X_train['Age'][c] = abs(i - mAge)
    sum = sum + X_train['Age'][c]**2
    c = c+1
#Compute the normalization factor, NFactor, as standard deviate of age
NFactor = math.sqrt(sum/X_train.shape[0])
print("Normalization Factor: ", NFactor)
#Bring 'Age' values within range
c = 0
for i in X_train['Age']:
    if i >= NFactor:
        X_train['Age'][c] = X_train['Age'][c] % NFactor
    c+=1
#Perform normalization
c = 0
for i in X_train['Age']:
    X_train['Age'][c] = X_train['Age'][c] / NFactor
    c+=1
```

```
print("Mean age: ", mAge)
print("Normalization Factor: ", NFactor)
```

Output:

```
print("Mean age: ", mAge)
print("Normalization Factor: ", NFactor)
```

```
Mean age:  31.822532402791627
Normalization Factor:  4.7441482147100125
```

viii. Run SVM with default Hyperparameter

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc=SVC()
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with default hyperparameters: {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with default hyperparameters: 0.8685
```

ix. Run SVM with RBF Kernel

```
svc=SVC(C=100.0)
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with rbf kernel and C=100.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with rbf kernel and C=100.0 : 0.8685
```

x. Run SVM with Linear Kernel

```
linear_svc=SVC(kernel='linear', C=1.0)
linear_svc.fit(X_train,y_train)
y_pred_test=linear_svc.predict(X_test)
print('Model accuracy score with linear kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred_test)))
```

Output:

```
Model accuracy score with linear kernel and C=1.0 : 0.8685
```

xi. Run SVM with Polynomial Kernel

```
poly_svc=SVC(kernel='poly', C=1.0)
poly_svc.fit(X_train,y_train)
y_pred=poly_svc.predict(X_test)
print('Model accuracy score with polynomial kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with polynomial kernel and C=1.0 : 0.8685
```

xii. Run SVM with Sigmoid Kernel

```
#Sigmoidal
sigmoid_svc=SVC(kernel='sigmoid', C=1.0)
sigmoid_svc.fit(X_train,y_train)
y_pred=sigmoid_svc.predict(X_test)
print('Model accuracy score with sigmoid kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with sigmoid kernel and C=1.0 : 0.8048
```

xiii. Compare the Train-Set and Test-Set Accuracy

```
from sklearn.metrics import mean_absolute_error
y_train_pred = svc.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_pred)
print(mae_train)
print(mae_test)
```

Output:

```
37.88135593220339
33.52589641434263
```

xiv. Check for Overfitting and Underfitting

```
if mae_train < mae_test:
    print("Overfitting is present")
else:
    print("Underfitting is present")
```

Output:

```
Underfitting is present
```

xv. Summarize Performance using Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Output:

```
array([[218,   0],
       [ 33,   0]], dtype=int64)
```

xvi. Evaluate Model Performance with Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

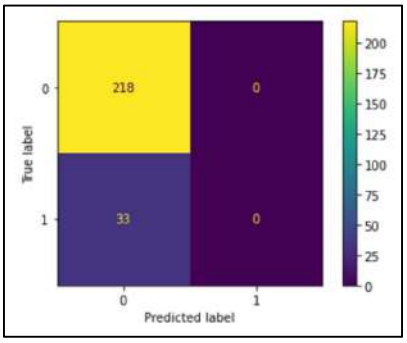
Output:

	precision	recall	f1-score	support
0	0.87	1.00	0.93	218
1	0.00	0.00	0.00	33
accuracy			0.87	251
macro avg	0.43	0.50	0.46	251
weighted avg	0.75	0.87	0.81	251

xvii. Analyse Performance Visually

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=svc.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svc.classes_)
disp.plot()
```

Output:



Result:

Thus, the models to perform classification using SVM classifier have been built successfully.

2. Build a model to perform SVM classification for the IRIS Dataset:

Aim:

To build model to perform SVM classification for the IRIS dataset.

Codes and Output:

```
import pandas as pd
data = pd.read_csv("Iris.csv")
```

i. Exploratory Data Analysis

```
data.shape
```

Output:

```
(150, 5)
```

```
data.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   sepallength  150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   class        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

ii. Explore the Missing Values in Variables

```
data.isnull().sum()
```

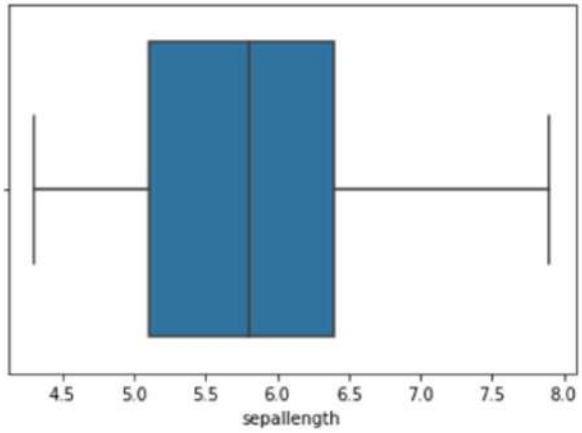
Output:

```
#Identify the missing values
data.isnull().sum()

sepallength    0
sepalwidth     0
petallength    0
petalwidth     0
class          0
dtype: int64
```

iii. Handle Outliers

```
sns.boxplot(x='sepallength', data=data)
```

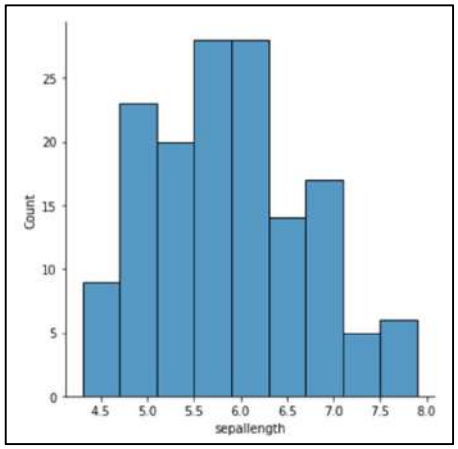
Output:



iv. Check the distribution of variable

```
import seaborn as sns
sns.displot(data, x="sepallength")
```

Output:



v. Declare Feature Vector and Target Variable

```
cols=['sepallength', 'sepalwidth', 'petallength', 'petalwidth']
X=data[cols]
y=data['class']
X.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   sepallength  150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

vi. Split data into separate training 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=0.2, random_state=40)
```

vii. Perform Feature Scaling

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled = scaler.fit_transform(X_train)
X_train.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 120 entries, 62 to 70
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   sepallength  120 non-null    float64
 1   sepalwidth   120 non-null    float64
 2   petallength  120 non-null    float64
 3   petalwidth   120 non-null    float64
dtypes: float64(4)
memory usage: 4.7 KB
```

```
scaled = scaler.fit_transform(X_test)
X_test.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30 entries, 38 to 81
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   sepallength  30 non-null     float64
 1   sepalwidth   30 non-null     float64
 2   petallength  30 non-null     float64
 3   petalwidth   30 non-null     float64
dtypes: float64(4)
memory usage: 1.2 KB
```

viii. Run SVM with default Hyperparameter

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc=SVC()
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with default hyperparameters: {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with default hyperparameters: 1.0000
```

ix. Run SVM with RBF Kernel

```
svc=SVC(C=100.0)
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with rbf kernel and C=100.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with rbf kernel and C=100.0 : 1.0000
```

x. Run SVM with Linear Kernel

```
linear_svc=SVC(kernel='linear', C=1.0)
linear_svc.fit(X_train,y_train)
y_pred_test=linear_svc.predict(X_test)
print('Model accuracy score with linear kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred_test)))
```

Output:

```
Model accuracy score with linear kernel and C=1.0 : 1.0000
```

xi. Run SVM with Polynomial Kernel

```
poly_svc=SVC(kernel='poly', C=1.0)
poly_svc.fit(X_train,y_train)
y_pred=poly_svc.predict(X_test)
print('Model accuracy score with polynomial kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with polynomial kernel and C=1.0 : 1.0000
```

xii. Run SVM with Sigmoid Kernel

```
#Sigmoid
sigmoid_svc=SVC(kernel='sigmoid', C=1.0)
sigmoid_svc.fit(X_train,y_train)
y_pred=sigmoid_svc.predict(X_test)
print('Model accuracy score with sigmoid kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with sigmoid kernel and C=1.0 : 0.2667
```

xiii. Compare the Train-Set and Test-Set Accuracy

```
y_train_pred = svc.predict(X_train)
print('Model accuracy score for train-set: {0:0.4f}'. format(accuracy_score(y_train,
y_train_pred)))
```

Output:

```
Model accuracy score for train-set: 0.9750
```

xiv. Summarize Performance using Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Output:

```
array([[ 8,  0,  0],
       [ 0, 12,  0],
       [ 0,  0, 10]], dtype=int64)
```

xv. Evaluate Model Performance with Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

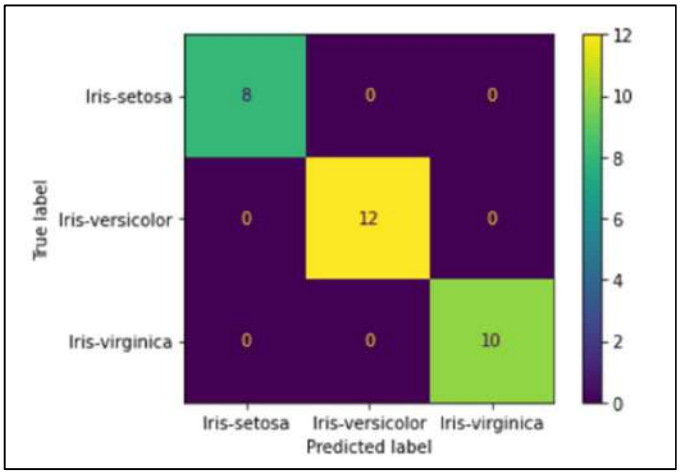
Output:

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       1.00      1.00      1.00        12
 Iris-virginica       1.00      1.00      1.00        10

      accuracy                           1.00        30
     macro avg       1.00      1.00      1.00        30
  weighted avg       1.00      1.00      1.00        30
```

xvi. Analyse Performance Visually

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=svc.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svc.classes_)
disp.plot()
```

Output:



Result:

Thus, the models to perform classification of IRIS dataset using SVM classifier have been built successfully.

3. Build a model to perform SVM classification for the Diabetes Dataset:

Aim:

To build model to perform SVM classification for the Diabetes dataset.

Codes and Output:

```
import pandas as pd
data = pd.read_csv("diabetes.csv")
```

i. Exploratory Data Analysis

```
data.shape
```

Output:

```
(768, 9)
```

```
data.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

ii. Explore the Missing Values in Variables

```
data.isnull().sum()
```

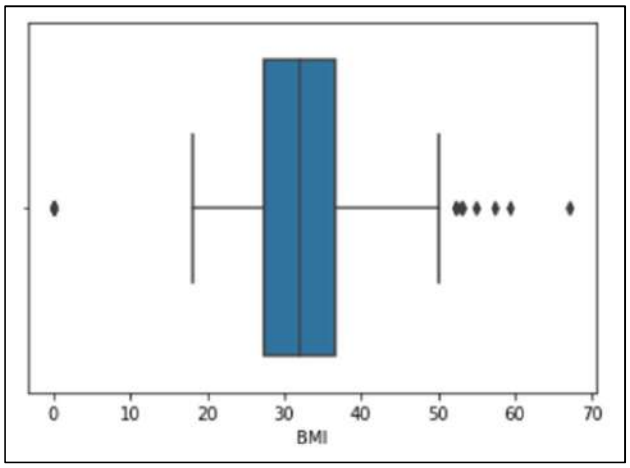
Output:

```
#missing values
data.isnull().sum()

Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

iii. Handle Outliers

```
import seaborn as sns
sns.boxplot(x='BMI', data=data)
```

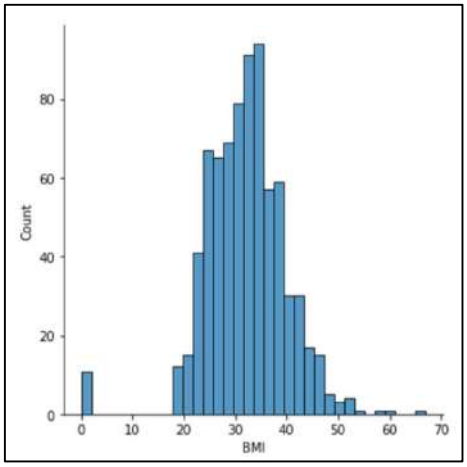
Output:



iv. Check the distribution of variable

```
sns.displot(data, x="BMI")
```

Output:



v. Declare Feature Vector and Target Variable

```
cols=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
X=encD[cols]
y=encD['Outcome']
X.info( )
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
dtypes: float64(2), int64(6)
memory usage: 48.1 KB
```

vi. Split data into separate training 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=0.2, random_state=40)
```

vii. Perform Feature Scaling

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled = scaler.fit_transform(X_train)
X_train.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 614 entries, 223 to 326
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               614 non-null    int64
 1   Glucose                   614 non-null    int64
 2   BloodPressure             614 non-null    int64
 3   SkinThickness             614 non-null    int64
 4   Insulin                   614 non-null    int64
 5   BMI                       614 non-null    float64
 6   DiabetesPedigreeFunction  614 non-null    float64
 7   Age                       614 non-null    int64
dtypes: float64(2), int64(6)
memory usage: 43.2 KB
```

```
scaled = scaler.fit_transform(X_test)
X_test.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 154 entries, 370 to 556
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               154 non-null    int64
 1   Glucose                   154 non-null    int64
 2   BloodPressure             154 non-null    int64
 3   SkinThickness             154 non-null    int64
 4   Insulin                   154 non-null    int64
 5   BMI                       154 non-null    float64
 6   DiabetesPedigreeFunction  154 non-null    float64
 7   Age                       154 non-null    int64
dtypes: float64(2), int64(6)
memory usage: 10.8 KB
```

viii. Run SVM with default Hyperparameter

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc=SVC()
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with default hyperparameters: {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with default hyperparameters: 0.7857
```

ix. Run SVM with RBF Kernel

```
svc=SVC(C=100.0)
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with rbf kernel and C=100.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with rbf kernel and C=100.0 : 0.7403
```

x. Run SVM with Linear Kernel

```
linear_svc=SVC(kernel='linear', C=1.0)
linear_svc.fit(X_train,y_train)
y_pred_test=linear_svc.predict(X_test)
print('Model accuracy score with linear kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred_test)))
```

Output:

```
Model accuracy score with linear kernel and C=1.0 : 0.7662
```

xi. Run SVM with Polynomial Kernel

```
poly_svc=SVC(kernel='poly', C=1.0)
poly_svc.fit(X_train,y_train)
y_pred=poly_svc.predict(X_test)
print('Model accuracy score with polynomial kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with polynomial kernel and C=1.0 : 0.7597
```

xii. Run SVM with Sigmoid Kernel

```
#Sigmoid
sigmoid_svc=SVC(kernel='sigmoid', C=1.0)
sigmoid_svc.fit(X_train,y_train)
y_pred=sigmoid_svc.predict(X_test)
print('Model accuracy score with sigmoid kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

Output:

```
Model accuracy score with sigmoid kernel and C=1.0 : 0.4481
```

xiii. Compare the Train-Set and Test-Set Accuracy

```
y_train_pred = svc.predict(X_train)
print('Model accuracy score for train-set: {0:0.4f}'. format(accuracy_score(y_train,
y_train_pred)))
```

Output:

```
Model accuracy score for train-set: 0.8127
```

xiv. Summarize Performance using Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Output:

```
array([[64, 31],
       [54,  5]], dtype=int64)
```

xv. Evaluate Model Performance with Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

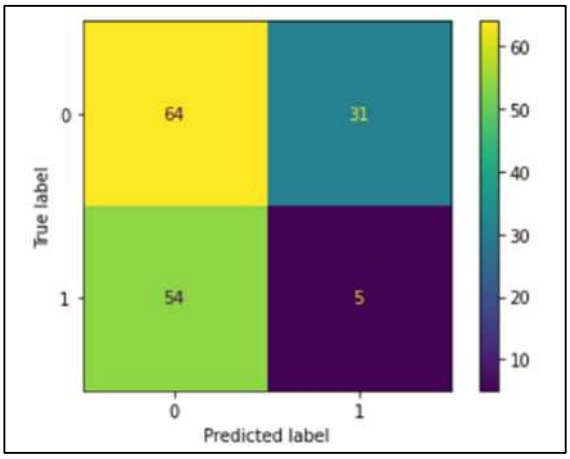
Output:

	precision	recall	f1-score	support
0	0.54	0.67	0.60	95
1	0.14	0.08	0.11	59
accuracy			0.45	154
macro avg	0.34	0.38	0.35	154
weighted avg	0.39	0.45	0.41	154

xvi. Analyse Performance Visually

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=svc.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svc.classes_)
disp.plot()
```

Output:



Result:

Thus, the models to perform classification of diabetes dataset using SVM classifier have been built successfully.

----- X -----

Date: 07.06.22, Tuesday **Exp. No.:** 15	**Hive Installation**

1. Study on Hive Architecture and its working

Aim:

To study about the architecture of Hive and its working.

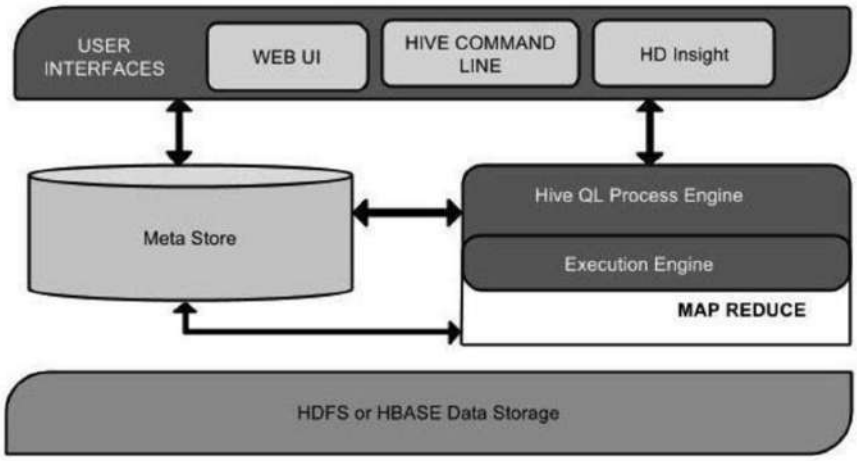
Theory:

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analysing easy. Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).

Features of Hive

- It stores schema in a database and processed data into HDFS
- It is designed for OLAP
- It provides SQL type language for querying called HiveQL or HQL
- It is familiar, fast, scalable, and extensible

Architecture of Hive



i. User Interface : Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).

ii. Meta Store: Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
iii. HiveQL Process Engine: HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
iv. Execution Engine: The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
v. HDFS or HBASE: Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Working of Hive

The following are the steps that are followed in the interaction between Hive and Hadoop

1. Execute Query - The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2. Get plan - The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query
3. Get Metadata - The compiler sends metadata request to Metastore (any database)
4. Send Metadata - Metastore sends metadata as a response to the compiler
5. Send Plan - The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete
6. Execute Plan - The driver sends the execute plan to the execution engine
7. Execute Job - Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job
   Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8. Fetch Result - The execution engine receives the results from Data nodes
9. Send Results - The execution engine sends those resultant values to the driver. The driver sends the results to Hive Interfaces

Result:

Thus, the architecture of Hive and its working have been studied.

2. Installation of Hive

Aim:

To install Hive.

Procedure:

1. Download Apache HIVE latest release

Visit https://hive.apache.org/downloads.html

2. In terminal, switch to the user where Hadoop is installed

```
su – hdoop
wget https://dlcdn.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

3. Extract the files to initiate the Hive installation

```
tar xvzf apache-hive-3.1.2-bin.tar.gz
```

```
hdoop@PRLAB-22:~$ tar xvzf apache-hive-3.1.2-bin.tar.gz
apache-hive-3.1.2-bin/LICENSE
apache-hive-3.1.2-bin/NOTICE
apache-hive-3.1.2-bin/RELEASE_NOTES.txt
apache-hive-3.1.2-bin/binary-package-licenses/asm-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.google.protobuf-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.ibm.icu.icu4j-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.sun.jersey-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.thoughtworks.paranamer-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/javax.transaction.transaction-api-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/javolution-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/jline-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/NOTICE
apache-hive-3.1.2-bin/binary-package-licenses/org.abego.treelayout.core-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.antlr-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.antlr.antlr4-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.antlr.stringtemplate-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.codehaus.janino-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.jamon.jamon-runtime-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.jruby-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.mozilla.rhino-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.slf4j-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/sqlline-LICENSE
apache-hive-3.1.2-bin/examples/files/2000_cols_data.csv
apache-hive-3.1.2-bin/examples/files/3col_data.txt
apache-hive-3.1.2-bin/examples/files/4col_data.txt
```

4. Configure Hive Environment Variables (bashrc)

```
vi .bashrc
```

Append the following Hive environment variables to the .bashrc file:

```
export HIVE_HOME= "/home/hdoop/apache-hive-3.1.2-bin"
export PATH=$PATH:$HIVE_HOME/bin
```

```
hdoop@PRLAB-22: ~
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
#Hadoop Related Options
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hdoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
#Hive Related Options
export HIVE_HOME="/home/hdoop/apache-hive-3.1.2-bin"
export PATH=$PATH:$HIVE_HOME/bin
```

5. Apply the changes to the current environment

```
source ~/.bashrc
```

```
hdoop@PRLAB-22:~$ source ~/.bashrc
```

6. Edit hive-config.sh file

```
vi $HIVE_HOME/bin/hive-config.sh
```

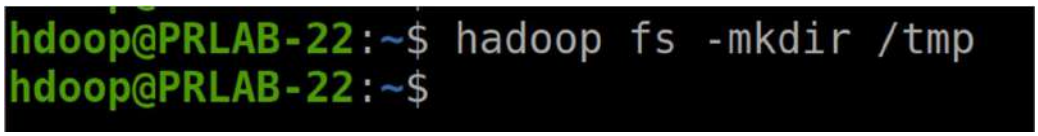
```
hdoop@PRLAB-22: ~/apache-hive-3.1.2-bin/bin
        HIVE_CONF_DIR=$confdir
        ;;
    --auxpath)
        shift
        HIVE_AUX_JARS_PATH=$1
        shift
        ;;
    *)
        break;
        ;;
  esac
done


# Allow alternate conf dir location.
HIVE_CONF_DIR="${HIVE_CONF_DIR:-$HIVE_HOME/conf}"

export HIVE_CONF_DIR=$HIVE_CONF_DIR
export HADOOP_HOME=/home/hdoop/hadoop
export HIVE_AUX_JARS_PATH=$HIVE_AUX_JARS_PATH
```

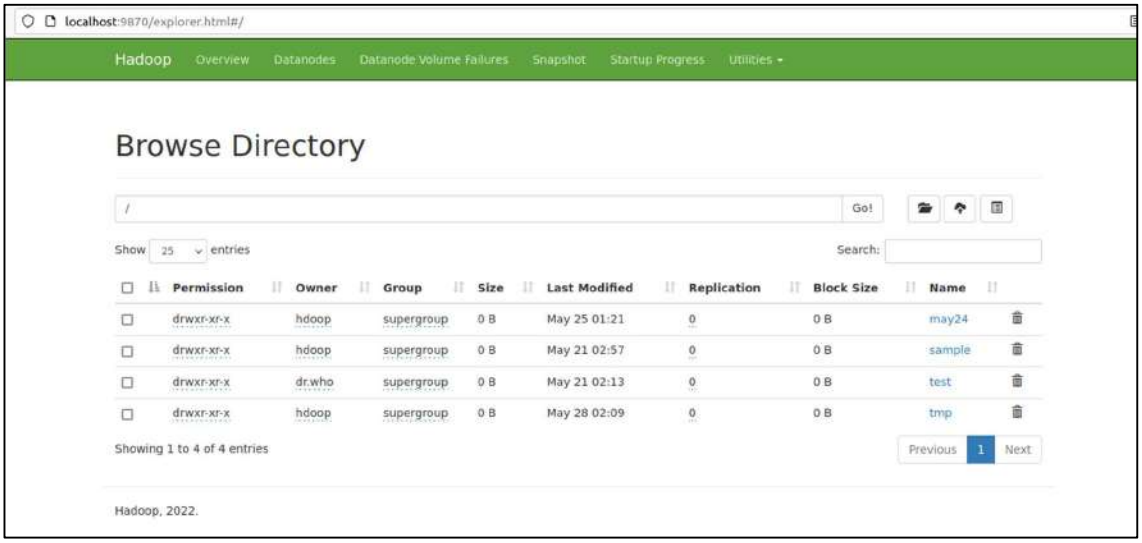
Start hadoop file system and yarn using the commands which were given during hadoop installation procedure.

```
itadmin@PRLAB-22:~$ su - hdoop
Password:
hdoop@PRLAB-22:~$ cd hadoop
hdoop@PRLAB-22:~/hadoop$ cd sbin
hdoop@PRLAB-22:~/hadoop/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [PRLAB-22]
hdoop@PRLAB-22:~/hadoop/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hdoop@PRLAB-22:~/hadoop/sbin$ jps
37620 NodeManager
38374 Jps
36616 DataNode
36856 SecondaryNameNode
37467 ResourceManager
36060 NameNode
```

7. Create two separate directories to store data in the HDFS layer

```
hadoop fs -mkdir /tmp
```



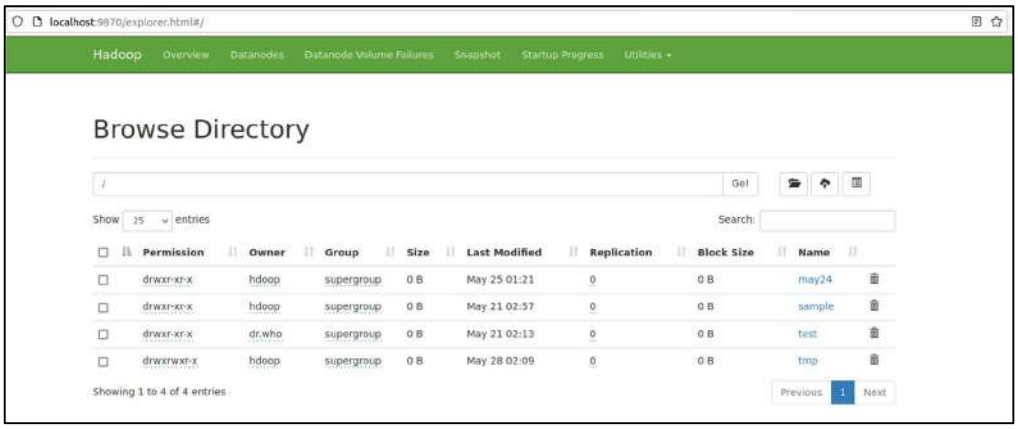


Add write and execute permissions to tmp group members

```
hadoop fs -chmod g+w /tmp
```

Check if the permissions were added correctly

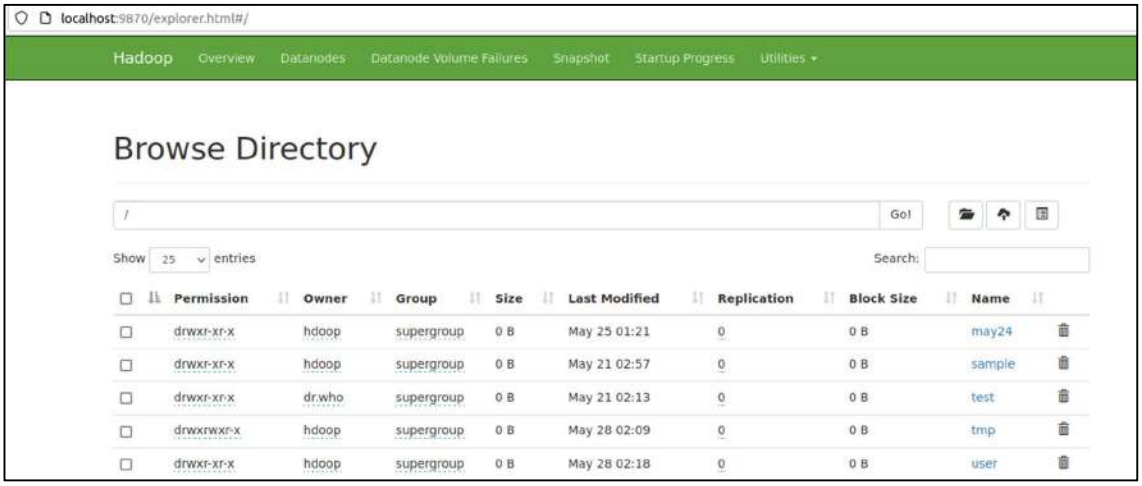
```
hadoop fs -ls /
```

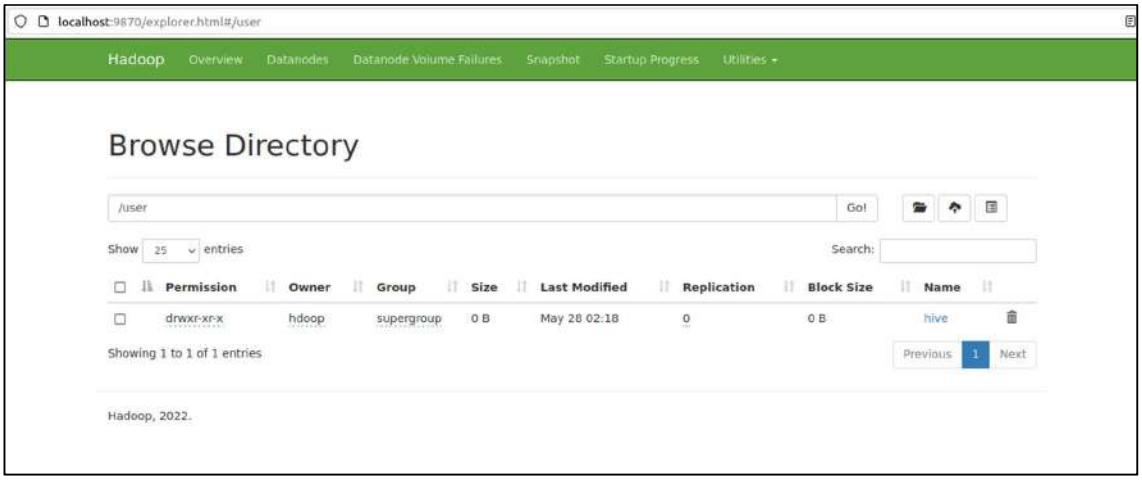


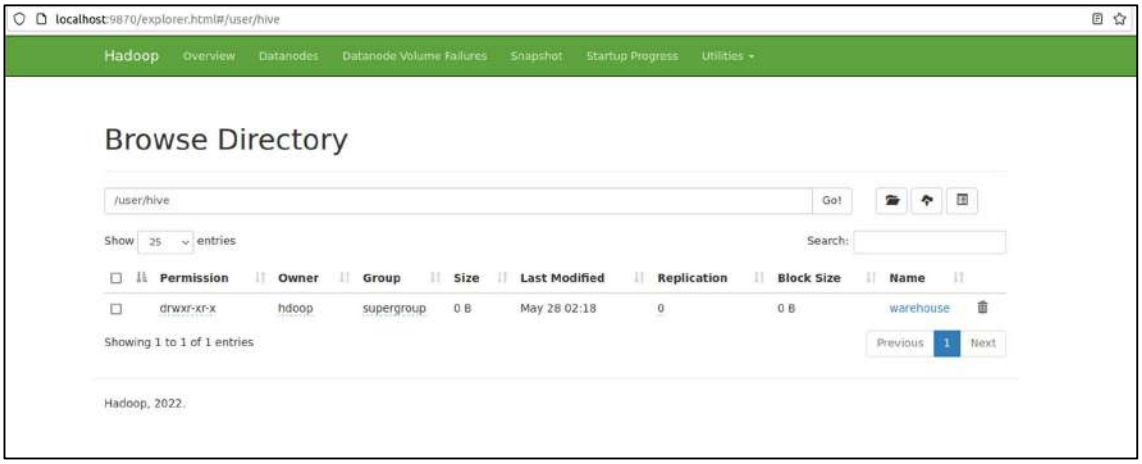
```
hdoop@PRLAB-22:~$ hadoop fs -ls /
Found 4 items
drwxr-xr-x   - hdoop  supergroup          0 2022-05-25 01:21 /may24
drwxr-xr-x   - hdoop  supergroup          0 2022-05-21 02:57 /sample
drwxr-xr-x   - dr.who supergroup          0 2022-05-21 02:13 /test
drwxrwxr-x   - hdoop  supergroup          0 2022-05-28 02:09 /tmp
hdoop@PRLAB-22:~$
```

Create warehouse Directory. Create the warehouse directory within the /user/hive/ parent directory

```
hadoop fs -mkdir -p /user/hive/warehouse
```





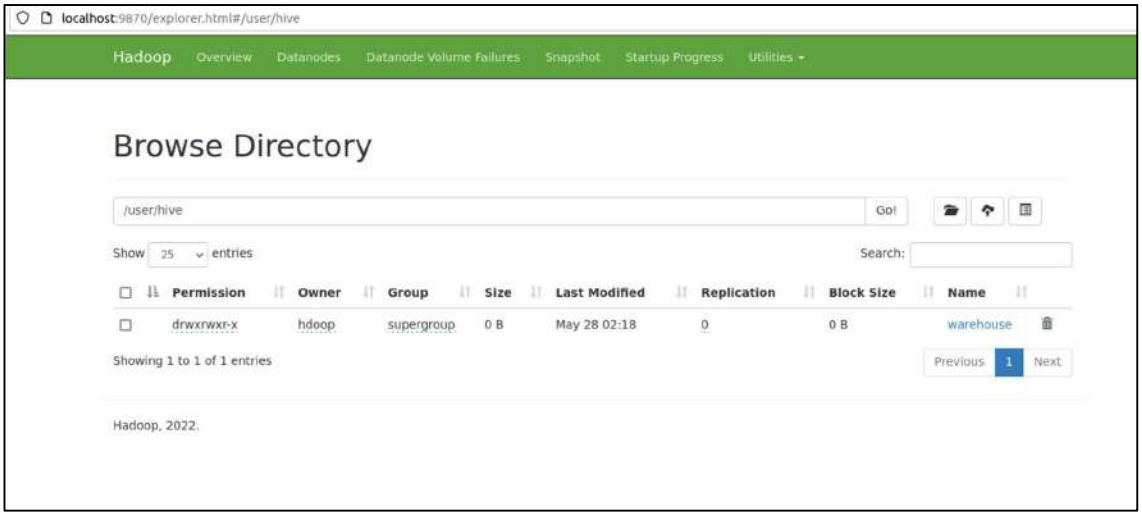


Add write and execute permissions to warehouse group members

```
hadoop fs -chmod g+w /user/hive/warehouse
```

```
hdoop@PRLAB-22:~$ hadoop fs -chmod g+w /user/hive/warehouse
hdoop@PRLAB-22:~$
```

Check if the permissions were added correctly

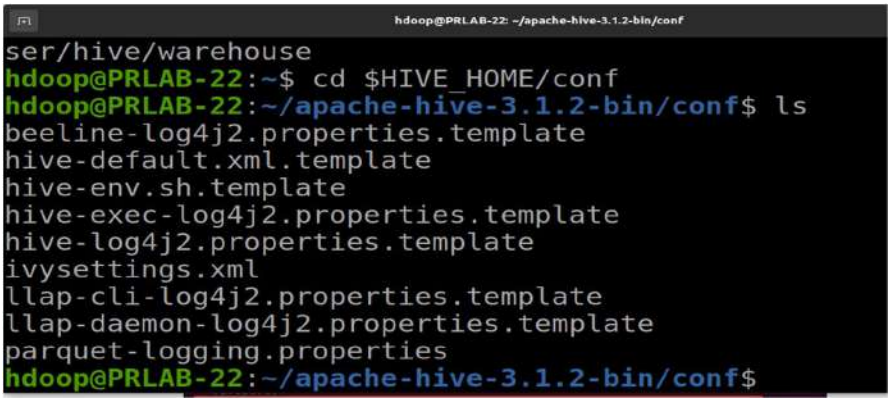



```
hdoop@PRLAB-22:~$ hadoop fs -ls /user/hive
Found 1 items
drwxrwxr-x   - hdoop supergroup          0 2022-05-28 02:18 /user/hive/warehouse
hdoop@PRLAB-22:~$
```

8. Configure hive-site.xml File

```
cd $HIVE_HOME/conf
```

List the files contained in the folder using the ls command



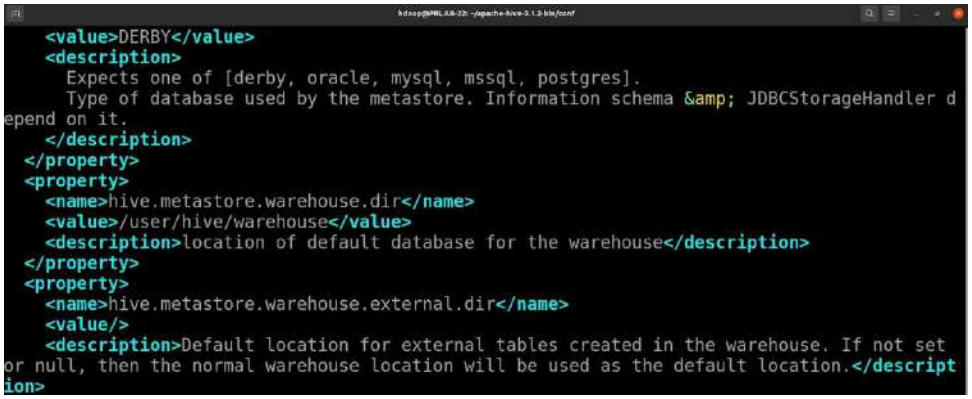
Use the hive-default.xml.template to create the hive-site.xml file:

```
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/conf$ cp hive-default.xml.template hive-site.xml
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/conf$
```

Access the hive-site.xml file using the text editor of your choice

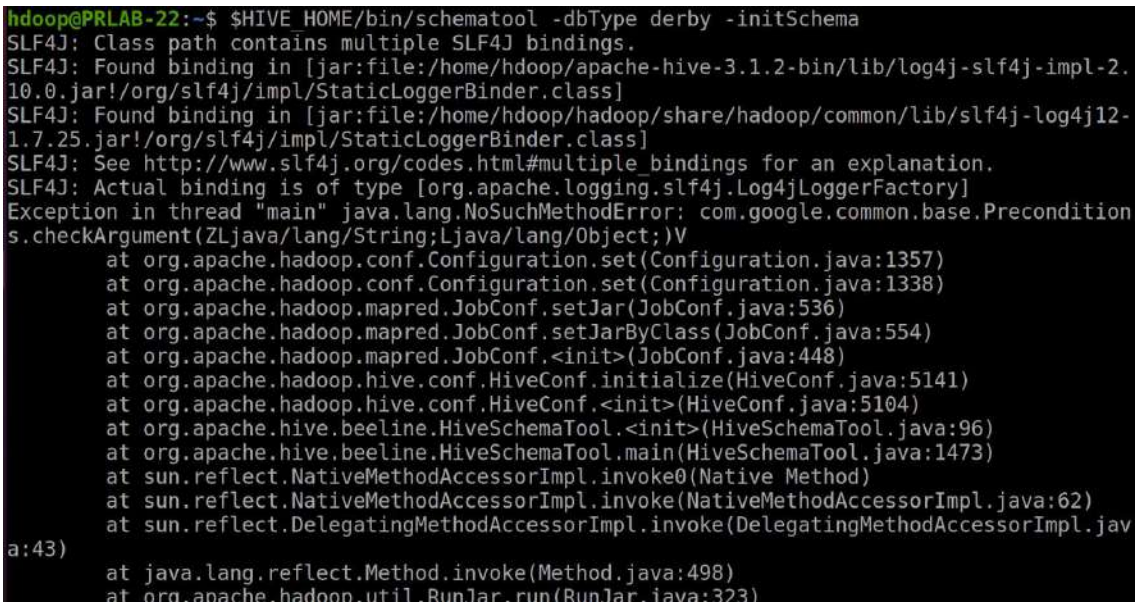
```
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/conf$ vi hive-site.xml
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/conf$
```

Using Hive in a stand-alone mode rather than in a real-life Apache Hadoop cluster is a safe option



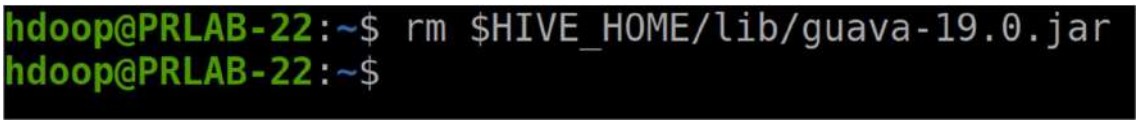
9. Initiate Database

```
$HIVE_HOME/bin/schematool -dbType derby -initSchema
```



10. Remove the existing guava file from the Hive lib directory

```
rm $HIVE_HOME/lib/guava-19.0.jar
```



11. Use the schematool command once again to initiate the Derby database

```
$HIVE_HOME/bin/schematool -dbType derby -initSchema
```

```
hdoop@PRLAB-22:~$ $HIVE_HOME/bin/schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hdoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.
10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hdoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-
1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to 3.1.0
Initialization script hive-schema-3.1.0.derby.sql
```

```
Initialization script completed
schemaTool completed
hdoop@PRLAB-22:~$
hdoop@PRLAB-22:~$
hdoop@PRLAB-22:~$
hdoop@PRLAB-22:~$
```

12. Launch Hive Client Shell on Ubuntu

cd $HIVE_HOME/bin/hive

```
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hdoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/sl
f4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hdoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 981ad83d-79b6-4856-89f7-9a1c51cef3f2

Logging initialized using configuration in jar:file:/home/hdoop/apache-hive-3.1.2-bin/lib/hive-common-3.1.2
.jar!/hive-log4j2.properties Async: true
Exception in thread "main" java.lang.IllegalArgumentException: java.net.URISyntaxException: Relative path i
n absolute URI: ${system:java.io.tmpdir%7D/$%7Bsystem:user.name%7D
        at org.apache.hadoop.fs.Path.initialize(Path.java:263)
        at org.apache.hadoop.fs.Path.<init>(Path.java:221)
        at org.apache.hadoop.hive.ql.session.SessionState.createSessionDirs(SessionState.java:710)
        at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:627)
        at org.apache.hadoop.hive.ql.session.SessionState.beginStart(SessionState.java:591)
        at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:747)
        at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:683)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:323)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:236)
```

```
Caused by: java.net.URISyntaxException: Relative path in absolute URI: ${system:java.io.tmpdir%7D/$%7Bsyste
m:user.name%7D
        at java.net.URI.checkPath(URI.java:1823)
        at java.net.URI.<init>(URI.java:745)
        at org.apache.hadoop.fs.Path.initialize(Path.java:260)
        ... 12 more
```

13. Error may occur when hive-shell started before metastore_db service. To avoid this just delete or move your metastore_db and try the below command

```
$ mv metastore_db metastore_db.tmp
```

14. Once again to initiate the Derby database

```
$ schematool -dbType derby -initSchema
```

Result:

Thus, Hive has been successfully installed.

----- X -----

Date: 07.06.22, Tuesday **Exp. No.:** 16	**Hive CRUD Operations**

1. Implement any schema definition in HIVE and perform CRUD operation

Aim:

To implement a schema definition in Hive and perform CRUD operations.

Codes and Output:

a) Table Creation

```
Create table studentnew (id int, name string, location string) cluster by (location) into 3 buckets row format delimited fields terminated by ‘,’ lines terminated by ‘\n’ stored as orc TABLEPROPERTIES (‘transactional’ = ‘true’);
```

Output:

```
hive> Set hive.support.concurrency = true;
hive> Set hive.enforce.bucketing = true;
hive> set hive.exec.dynamic.partition.mode = nonstrict;
hive> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.initiator.on = true;
hive> set hive.compactor.worker.threads =1;
hive> create table studentnew  (id int, name string, location string) clustered by (location) into 3 bucket
s row format delimited fields terminated by ',' lines terminated by '\n' stored as orc TBLPROPERTIES ('tran
sactional'='true');
OK
Time taken: 0.807 seconds
```

b) Insertion of Records

```
INSERT INTO TABLE studentnew VALUES (101,’abc’,’GST Road’),
  (102,’A’,’Guindy’),  (103,’PRABU’,’henry road’),  (104,’KUMAR’,’gandhi road’);
```

Output:

```
hive> insert into table studentnew values(101,'abc','GST Road');
Query ID = hdoop_20220531030502_54f3eff2-c5fa-452d-85e0-c728e6b2a704
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1653941890266_0001, Tracking URL = http://PRLAB-22:8088/proxy/application_1653941890266_
0001/
Kill Command = /home/hdoop/hadoop/bin/mapred job  -kill job_1653941890266_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 3
2022-05-31 03:05:13,825 Stage-1 map = 0%,  reduce = 0%
2022-05-31 03:05:16,916 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.7 sec
2022-05-31 03:05:22,021 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 4.5 sec
2022-05-31 03:05:24,050 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.36 sec
MapReduce Total cumulative CPU time: 6 seconds 360 msec
Ended Job = job_1653941890266_0001
Loading data to table default.studentnew
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
```

c) Retrieve of Records

```
select * from studentnew
```

Output:

```
hive> select * from studentnew;
OK
101     abc     GST Road
102     prabhu  guindy
103     gandhi  radha nagar chrompet
Time taken: 0.333 seconds, Fetched: 3 row(s)
```

d) Updation of Records

```
Update studentnew SET id = 113 WHERE id = 103;
```

Output:

```
hive> Update studentnew SET id = 113 WHERE id = 103;
Query ID = hdoop_20220531031127_9ec9ac2d-39f9-4bd5-b809-5c4ea6ceb5d2
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1653941890266_0007, Tracking URL = http://PRLAB-22:8088/proxy/application_1653941890266_0007/
Kill Command = /home/hdoop/hadoop/bin/mapred job  -kill job_1653941890266_0007
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 3
2022-05-31 03:11:33,595 Stage-1 map = 0%,  reduce = 0%
2022-05-31 03:11:38,685 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.12 sec
2022-05-31 03:11:42,751 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 9.56 sec
2022-05-31 03:11:43,770 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 12.97 sec
MapReduce Total cumulative CPU time: 12 seconds 970 msec
Ended Job = job_1653941890266_0007
Loading data to table default.studentnew
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3  Reduce: 3   Cumulative CPU: 12.97 sec   HDFS Read: 43124 HDFS Write: 1801 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 970 msec
OK
Time taken: 18.984 seconds
```

```
hive> select * from studentnew;
OK
101     abc     GST Road
102     prabhu  guindy
113     gandhi  radha nagar chrompet
Time taken: 0.16 seconds, Fetched: 3 row(s)
```

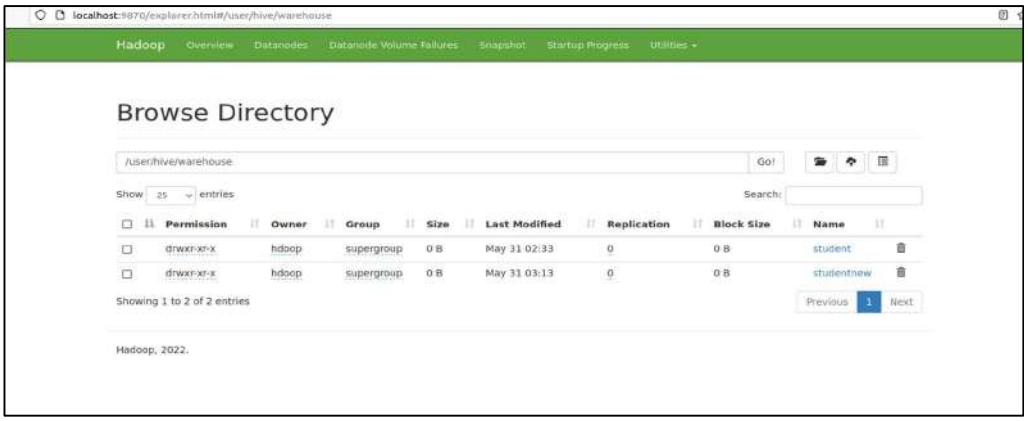
e) Deletion of Records

```
delete from studentnew where id=102;
```

Output:

```
hive> delete from studentnew where id=102;
Query ID = hdoop_20220531031339_a531bcab-2da2-4826-81af-a8d4f44dc0f4
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1653941890266_0008, Tracking URL = http://PRLAB-22:8088/proxy/application_1653941890266_0008/
Kill Command = /home/hdoop/hadoop/bin/mapred job  -kill job_1653941890266_0008
Hadoop job information for Stage-1: number of mappers: 4; number of reducers: 3
2022-05-31 03:13:45,465 Stage-1 map = 0%,  reduce = 0%
2022-05-31 03:13:51,592 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 11.5 sec
2022-05-31 03:13:55,652 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 13.28 sec
2022-05-31 03:13:57,690 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 16.71 sec
MapReduce Total cumulative CPU time: 16 seconds 710 msec
Ended Job = job_1653941890266_0008
Loading data to table default.studentnew
MapReduce Jobs Launched:
Stage-Stage-1: Map: 4  Reduce: 3   Cumulative CPU: 16.71 sec   HDFS Read: 51452 HDFS Write: 896 SUCCESS
Total MapReduce CPU Time Spent: 16 seconds 710 msec
OK
Time taken: 19.13 seconds
hive> select * from studentnew;
OK
101     abc     GST Road
113     gandhi  radha nagar chrompet
Time taken: 0.168 seconds, Fetched: 2 row(s)
```

View the table created in Browser-UI



Result:

Thus, a schema definition has been created in Hive and CRUD operations have been performed.

----- X -----