SIPNA SHIKSHAN PRASARAK MANDAL’S

SIPNA COLLEGE OF ENGINEERING AND TECHNOLOGY, AMRAVATI, M.S

Accredited by NBA, IAO and NAAC (With ‘A+’ Grade)

(An ISO 9001 : 2015 and 14001 : 2015 Certified Institute)

Certificate

Name of Department Computer Science And Engineering

Subject Big Data Analytics LAB

This is to certify that this practical record contains the bonafide practical works of

Mr. Jagrut Manish Thakare Roll No. 21BE0097

For the semester 6 (Sixth) during

Academic Year – 2023 – 24

Date - / /20 **Subject In - Charge**

Subject: BDA LAB Subject Code: 6KS08 Year / Sem: 3 / 6

Course Outcomes : On Completion of the Practicals, the student will be able to

1. ....
2. ....
3. ....
4. ....
5. ....
6. ....
7. ....

INDEX

Sr. No.	Name of Experiment	Date	Grade	Co. Mapping	Prof. In-charge Sign
1	Installation of Anaconda Jupiter along with Python.				
2	Design and implementation of Python basic Library Such as NumPy ,Pandas and Matplot etc.				
3	Write a program to implement K-Means algorithm. Use an appropriate data set for clustering.				
4	Write a program to implement Linear Regression.				
5	Write a program to implement Logistic Regression.				
6	Write a program to implement K-Nearest Neighbour algorithm to classify the object. Use an appropriate dataset for classification.				
7	Write a program to implement Time Series Analysis.				
8	To Study and implement various commands for MongoDB				

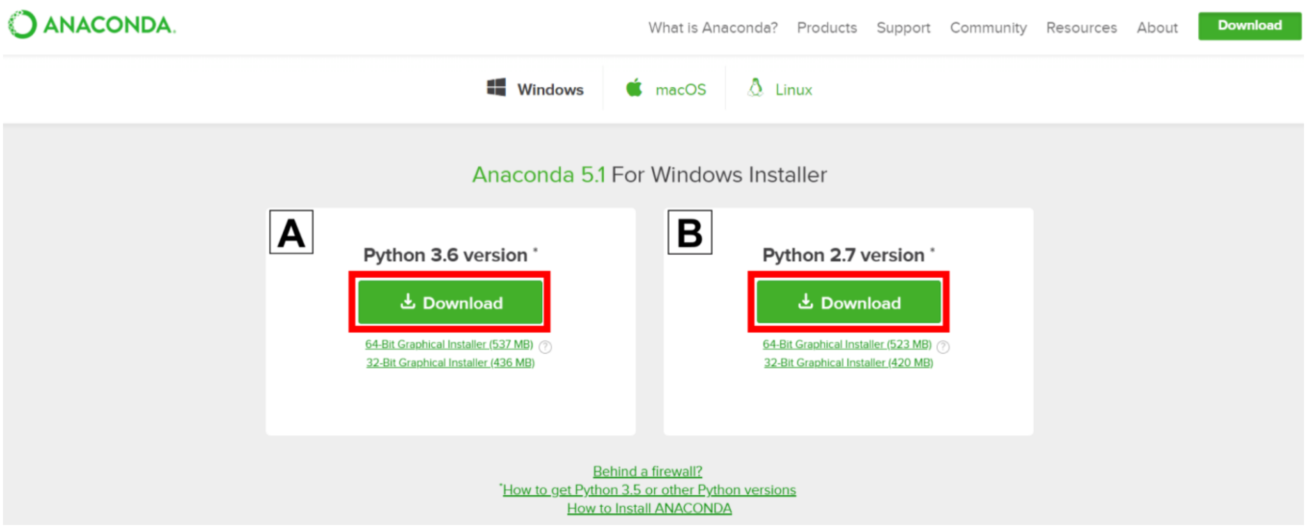
PRACTICAL 1

Aim : Installation of Anaconda Jupiter along with Python.

S/W Required : Python, Jupyter notebook, Anaconda.

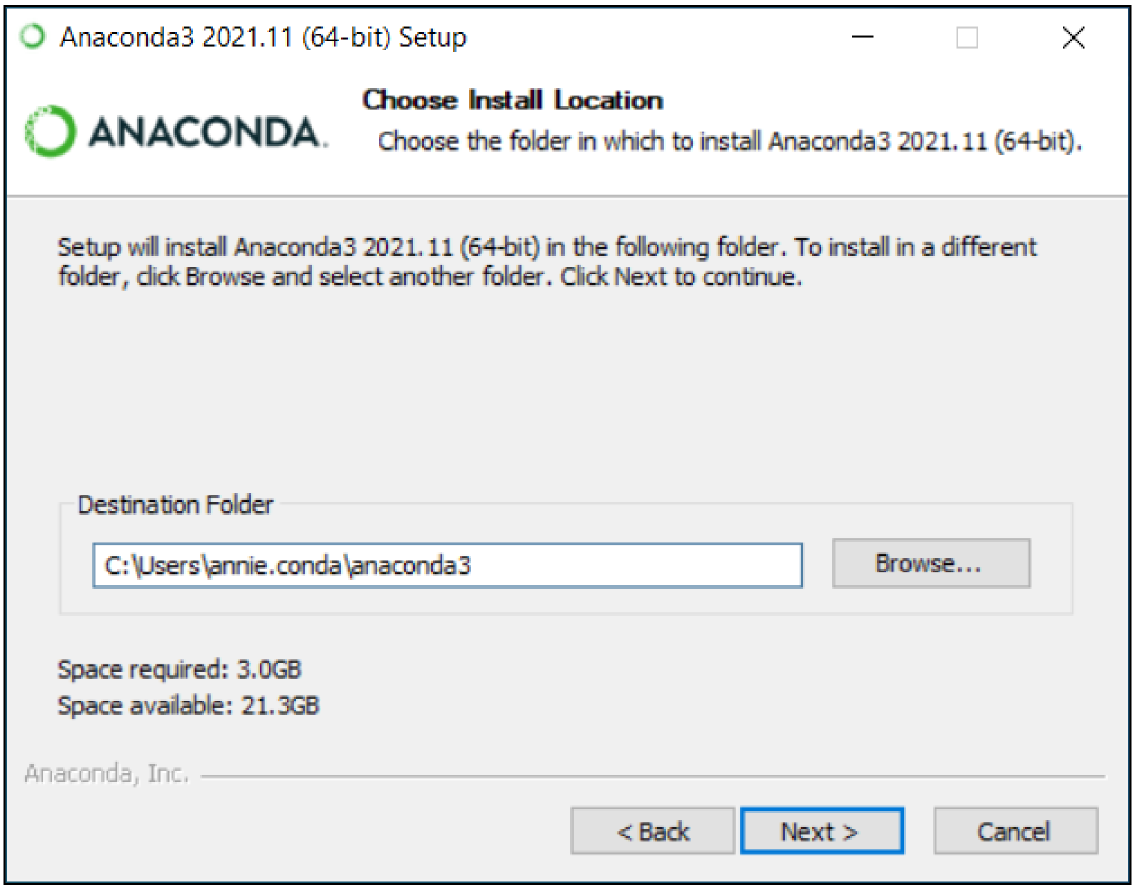
Installation Of Anaconda S/W:

1. Download the Anaconda installer.


ANACONDA.
What is Anaconda?
Products
Support
Community
Resources
About
Download
Windows
macOS
Linux
Anaconda 5.1 For Windows Installer
A
Python 3.6 version *
Download
64-Bit Graphical Installer (537 MB)
32-Bit Graphical Installer (436 MB)
B
Python 2.7 version *
Download
64-Bit Graphical Installer (523 MB)
32-Bit Graphical Installer (420 MB)
Behind a firewall?
*How to get Python 3.5 or other Python versions
How to Install ANACONDA

2. Go to your Downloads folder and double-click the installer to launch. To prevent permission errors, do not launch the installer from the Favorites folder.
3. Click **Next**.
4. Read the licensing terms and click **I Agree**.
5. It is recommended that you install for **Just Me**, which will install Anaconda Distribution to just the current user account. Only select an install for **All Users** if you need to install for all users' accounts on the computer (which requires Windows Administrator privileges).
6. Click **Next**

7. Select a destination folder to install Anaconda and click **Next**. Install Anaconda to a directory path that does not contain spaces or unicode character.


Anaconda3 2021.11 (64-bit) Setup

ANACONDA.

Choose Install Location

Choose the folder in which to install Anaconda3 2021.11 (64-bit).

Setup will install Anaconda3 2021.11 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

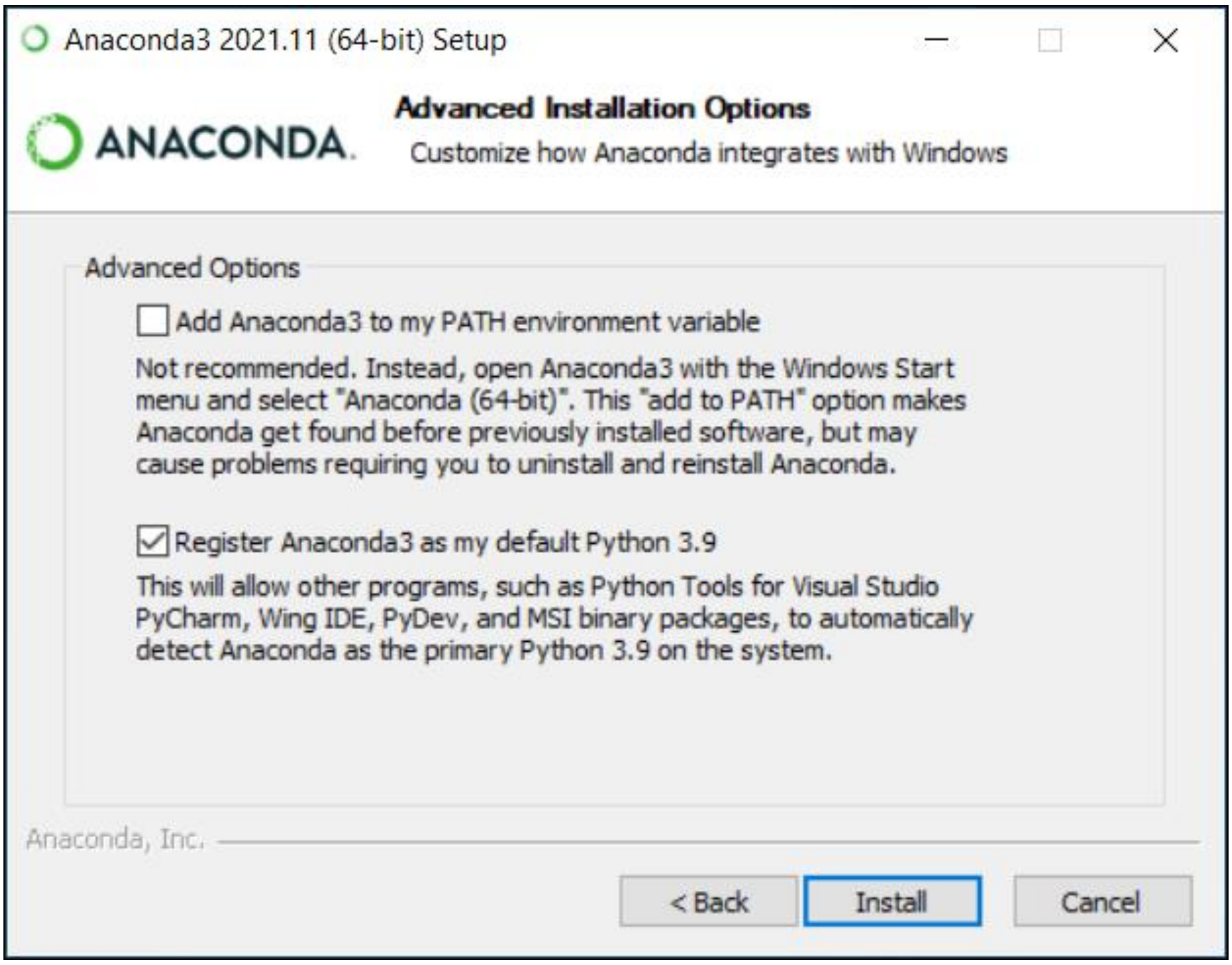
Destination Folder

C:\Users\annie.conda\anaconda3

Space required: 3.0GB
Space available: 21.3GB

Anaconda, Inc.

8. Choose whether to add Anaconda to your PATH environment variable or register Anaconda as your default Python. We **don't recommend** adding Anaconda to your PATH environment variable, since this can interfere with other software. Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, accept the default and leave this box checked. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.


Anaconda3 2021.11 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Anaconda3 to my PATH environment variable

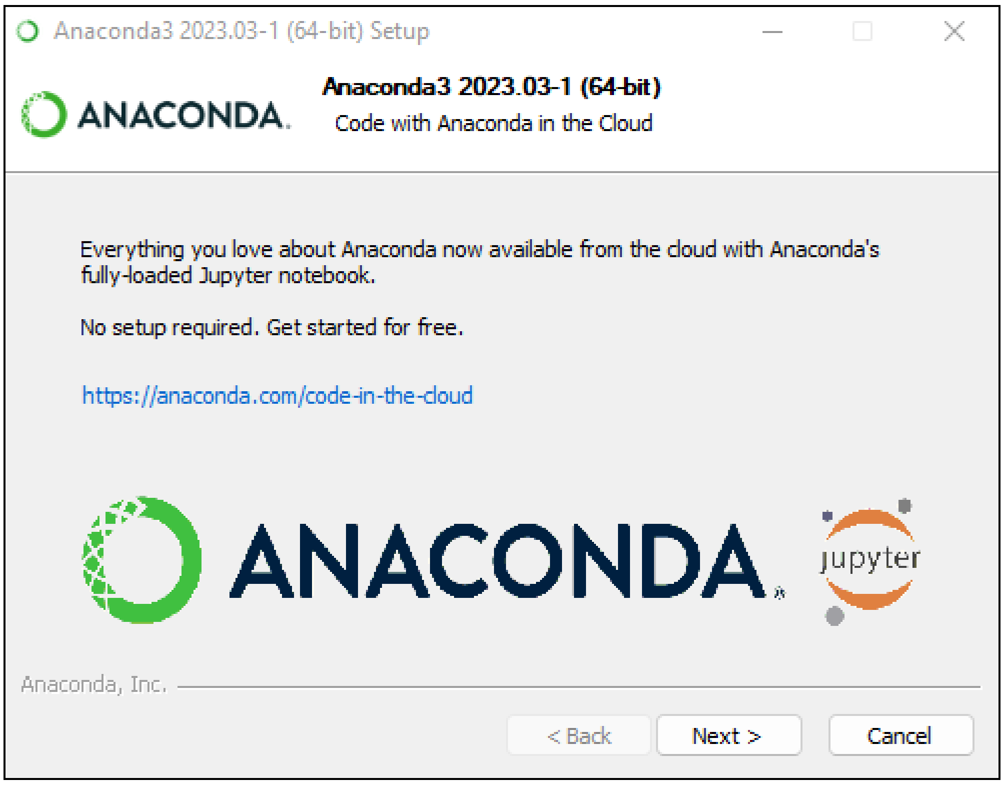
Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda3 as my default Python 3.9

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.9 on the system.

Anaconda, Inc.

9. Click **Install**. If you want to watch the packages Anaconda is installing, click Show Details.


Anaconda3 2023.03-1 (64-bit) Setup

Anaconda3 2023.03-1 (64-bit)

Code with Anaconda in the Cloud

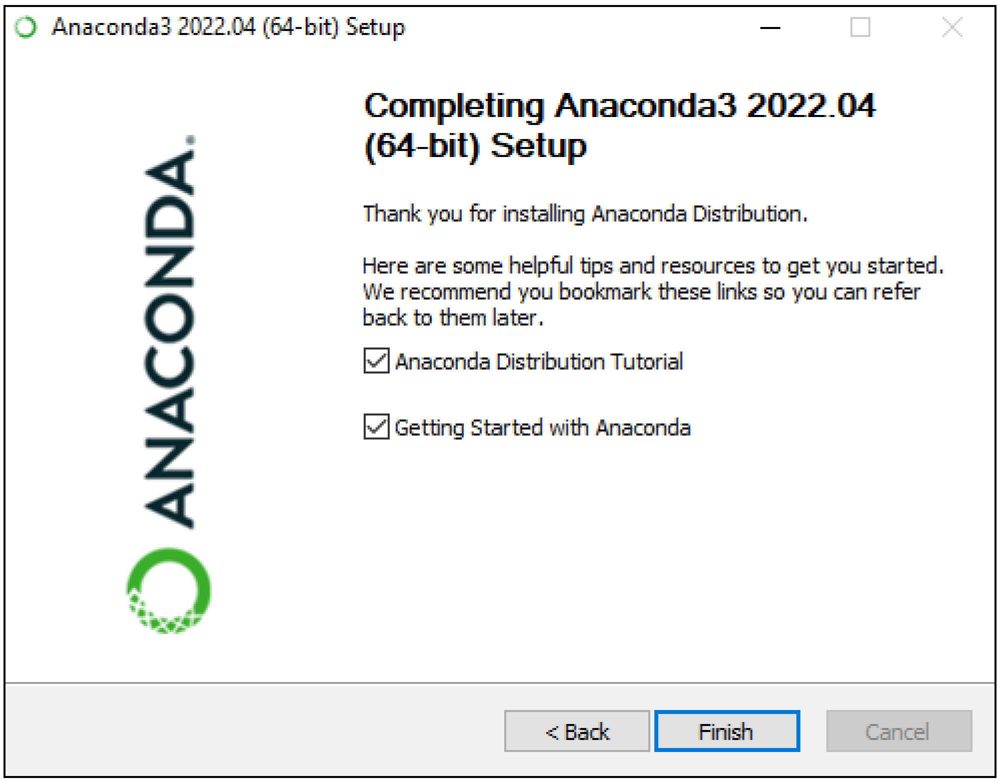
Everything you love about Anaconda now available from the cloud with Anaconda's fully-loaded Jupyter notebook.

No setup required. Get started for free.

https://anaconda.com/code-in-the-cloud

Anaconda, Inc.

10. Click **Next** Or click **Continue** to proceed.
11. After a successful installation you will see the “Thanks for installing Anaconda” dialog box:


Anaconda3 2022.04 (64-bit) Setup

Completing Anaconda3 2022.04 (64-bit) Setup

Thank you for installing Anaconda Distribution.

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

☑ Anaconda Distribution Tutorial

☑ Getting Started with Anaconda

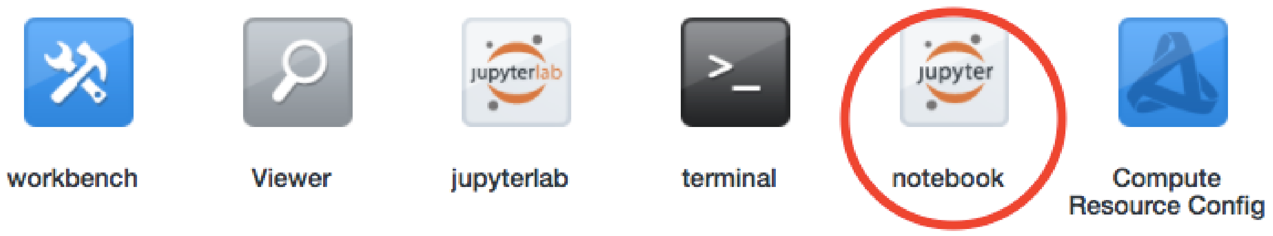
Using Jupyter Notebook

The Jupyter Notebook application allows you to create and edit documents that display the input and output of a Python or R language script. Once saved, you can share these files with others.

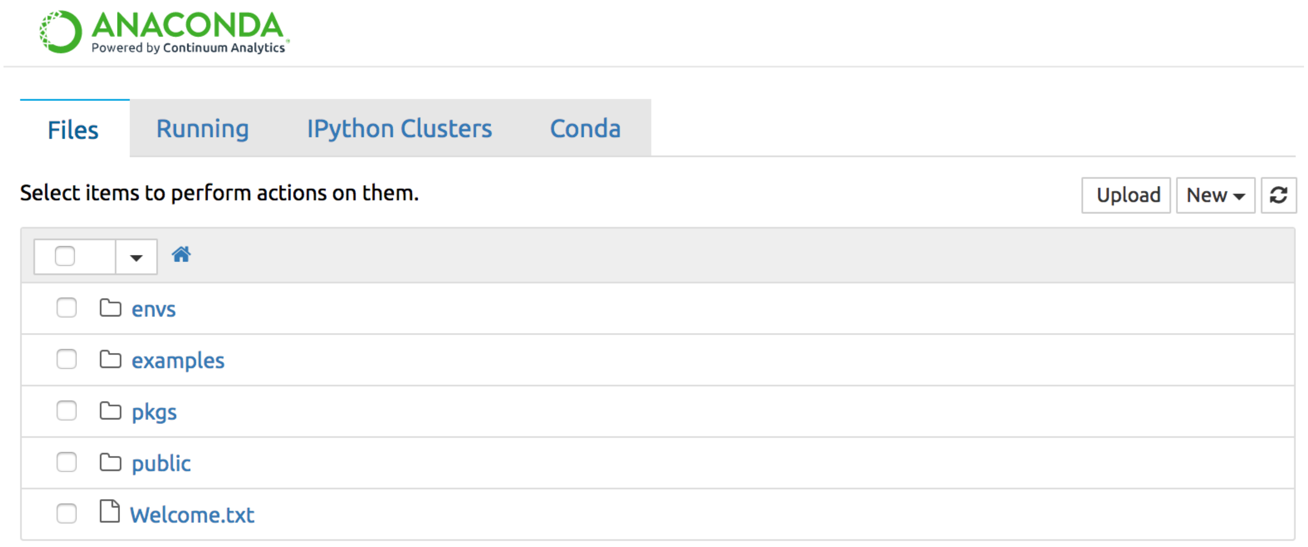
NOTE: Python and R language are included by default, but with customization, Notebook can run several other kernel environments.

Opening the Jupyter Notebook application

1. Log in to AEN.
2. Select the project you want to work on, or create a new project and open it.
3. On the project home page, click the Jupyter Notebook icon:



Jupyter Notebook opens in a new browser window:



Select items to perform actions on them.

envs
examples
pkgs
public
Welcome.txt

TIP: You can see the same File Manager in the Terminal, Workbench, and Viewer applications.

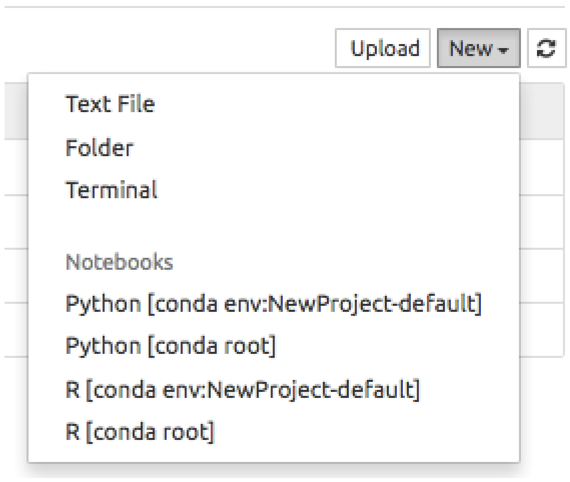
Using example notebooks

The Examples folder in Jupyter Notebook contains several types of Notebook examples created in Python—and one with R language—kernel environments.

Open any example notebook to experiment and see how it works.

Creating a new Jupyter Notebook

1. An the top right of the **Files** tab, click the New button.



2. Select the kernel environment to create your new notebook in.

   NOTE: Customizable Python and R Language kernel environments are automatically created for you during project creation.

   - Your project's default conda env kernels are a cloned copy of the root environment. You can customize them and install and delete additional packages.
   - Root environment is managed by your Administrator. You cannot make or save any changes to it.
   - You can switch between Python, R language and any other custom kernels in the notebook as you work in your notebook. For more information, see Using the Synchronize Environments extension.

   The new notebook is saved in the related project directory and displayed.

Conclusion :

Thus We have installed Anaconda, Jupyter, Python on Our Windows PC.

PRACTICAL 2

Aim : Design and implementation of Python basic Library Such as NumPy, Pandas and Matplotlib etc.

S/W Required : Python, VS Code, Jupyter notebook

Theory :

- **NumPy**: Numerical Computing with Python
    - NumPy, short for Numerical Python, is a cornerstone library for numerical computing in Python. It provides support for multidimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

    - Key features of NumPy include:
        - Multidimensional Arrays: NumPy's ndarray (n-dimensional array) is a powerful data structure for representing and manipulating numerical data.
        - Vectorized Operations: NumPy enables vectorized operations, allowing for efficient element-wise computations without the need for explicit looping.
        - Mathematical Functions: It offers a wide range of mathematical functions for array manipulation, linear algebra, Fourier analysis, and more.

- **Pandas**:
    - Data Analysis and Manipulation Pandas is a high-level data manipulation tool built on top of NumPy, designed to make data analysis tasks in Python simple and intuitive. It provides

data structures like DataFrame and Series, along with functionalities to manipulate and analyze structured data efficiently.

    - Key features of Pandas include:
        - DataFrame: Pandas' DataFrame is a two-dimensional labeled data structure capable of holding data of different types. It provides powerful indexing and slicing operations.
        - Data Manipulation: Pandas offers a plethora of functions for data manipulation tasks such as merging, reshaping, slicing, and grouping. Data Input/Output: It supports reading and writing data from various file formats including CSV, Excel, SQL databases, and more.

- **Matplotlib**: Data Visualization with Python Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a MATLAB-like interface for plotting 2D graphics, making it easy to generate a wide variety of plots and charts.

    - Key features of Matplotlib include:
        - Versatile Plotting: Matplotlib supports a wide range of plotting styles including line plots, scatter plots, bar plots, histograms, and more.
        - Customization: It offers extensive customization options to fine-tune the appearance of plots, including control over colors, labels, axes, and annotations.
        - Integration with Pandas: Matplotlib seamlessly integrates with Pandas, allowing for direct plotting of DataFrame and Series objects.

In [ ]:
```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

```
[1 2 3 4 5]
```

In [ ]:
```
a = np.array(42) # 0-D array with a single element
b = np.array([1, 2, 3, 4, 5]) # 1-D array
c = np.array([[1, 2, 3], [4, 5, 6]]) # 2-D array
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]]) # 3-D array
print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

```
0
1
2
3
```

In [ ]:
```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('5th element on 2nd row: ', arr[1, 4])
print('Last element from 2nd dim: ', arr[1, -1])
```

```
5th element on 2nd row:  10
Last element from 2nd dim:  10
```

In [ ]:
```
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5])
print(arr[4:])
print(arr[:4])
print(arr[-3:-1])
print(arr[1:5:2])
print(arr[::2])
```

```
[2 3 4 5]
[5 6 7]
[1 2 3 4]
[5 6]
[2 4]
[1 3 5 7]
```

In [ ]:
```
arr = np.array(['banana', 'cherry', 'apple'])
print(np.sort(arr))
```

```
['apple' 'banana' 'cherry']
```

In [ ]:
```
import pandas as pd
mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}
myvar = pd.DataFrame(mydataset)
print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

In [ ]:
```python
df = pd.read_csv('Iris.csv')
df.head()
```

Out[ ]:

	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

In [ ]:
```python
import pandas as pd
df = pd.read_json('data.json')
df.head()
```

Out[ ]:

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

In [ ]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Maxpulse  169 non-null    int64  
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 6.6 KB
```

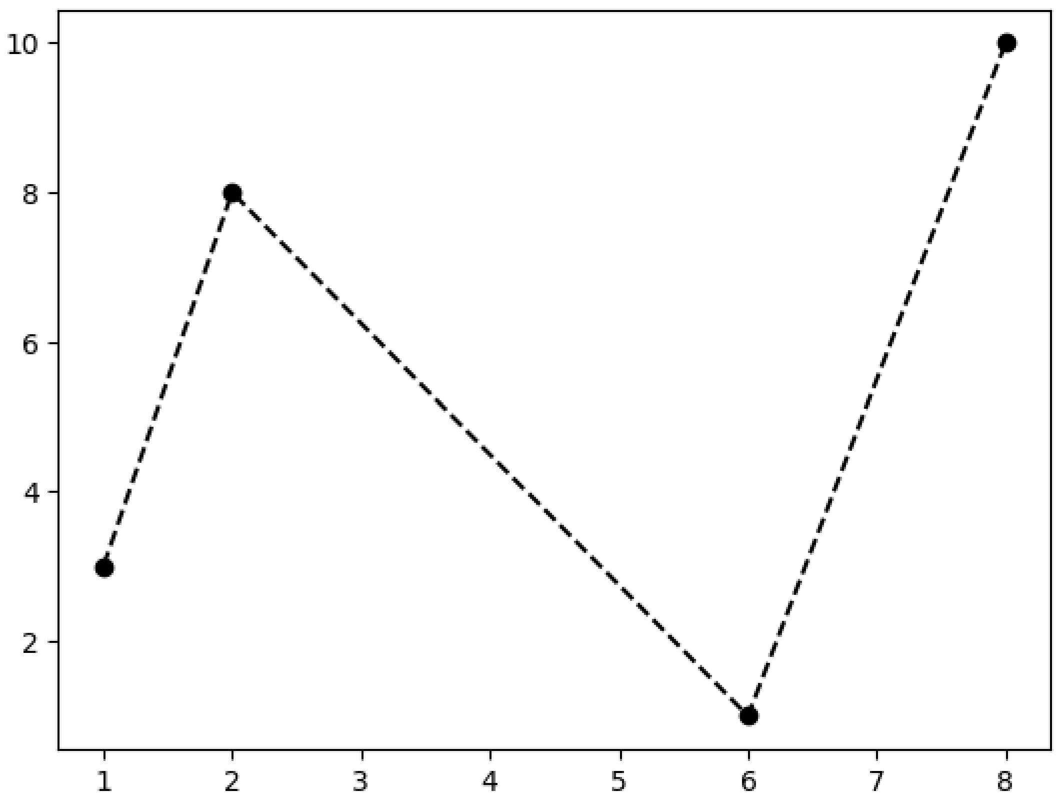
In [ ]:
```python
df.corr()
```

Out[ ]:

	Duration	Pulse	Maxpulse	Calories
Duration	1.000000	-0.155408	0.009403	0.922721
Pulse	-0.155408	1.000000	0.786535	0.025120
Maxpulse	0.009403	0.786535	1.000000	0.203814
Calories	0.922721	0.025120	0.203814	1.000000

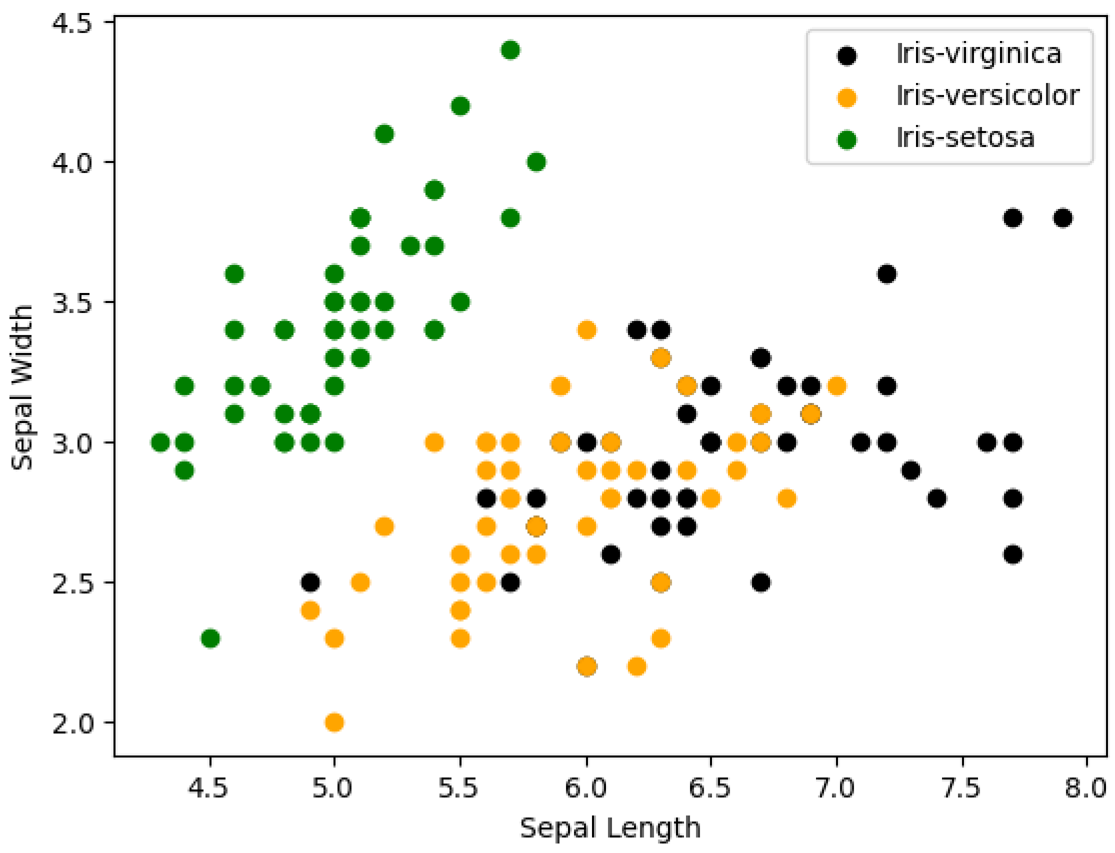
In [ ]:
```python
import matplotlib.pyplot as plt
xpoints = np.array([1, 2, 6, 8])
```

```
ypoints = np.array([3, 8, 1, 10])
plt.plot(xpoints, ypoints, marker='o', color='black', linestyle='dashed')
plt.show()
```



```
In [ ]: import matplotlib.pyplot as plt
        import seaborn as sns
        import pandas as pd

        colors = ['black', 'orange', 'green']
        species = ['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']
        for i in range(3):
            # filter data on each class
            x = df[df['species'] == species[i]]
            # plot the scatter plot
            plt.scatter(x['sepal_length'], x['sepal_width'], c = colors[i], label=specie
        plt.xlabel("Sepal Length")
        plt.ylabel("Sepal Width")
        plt.legend()
```

```
Out[ ]: <matplotlib.legend.Legend at 0x159c0c056a0>
```



Conclusion :

In conclusion, NumPy, Pandas, and Matplotlib serve as foundational pillars in the Python ecosystem for data science and visualization. Understanding the design principles and implementation details of these libraries is essential for mastering data analysis and visualization tasks in Python.

PRACTICAL 3

Aim : Write a program to implement K-Means algorithm. Use an appropriate data set for clustering.

S/W Required : Python, VS Code, Jupyter notebook

Theory :

K-means

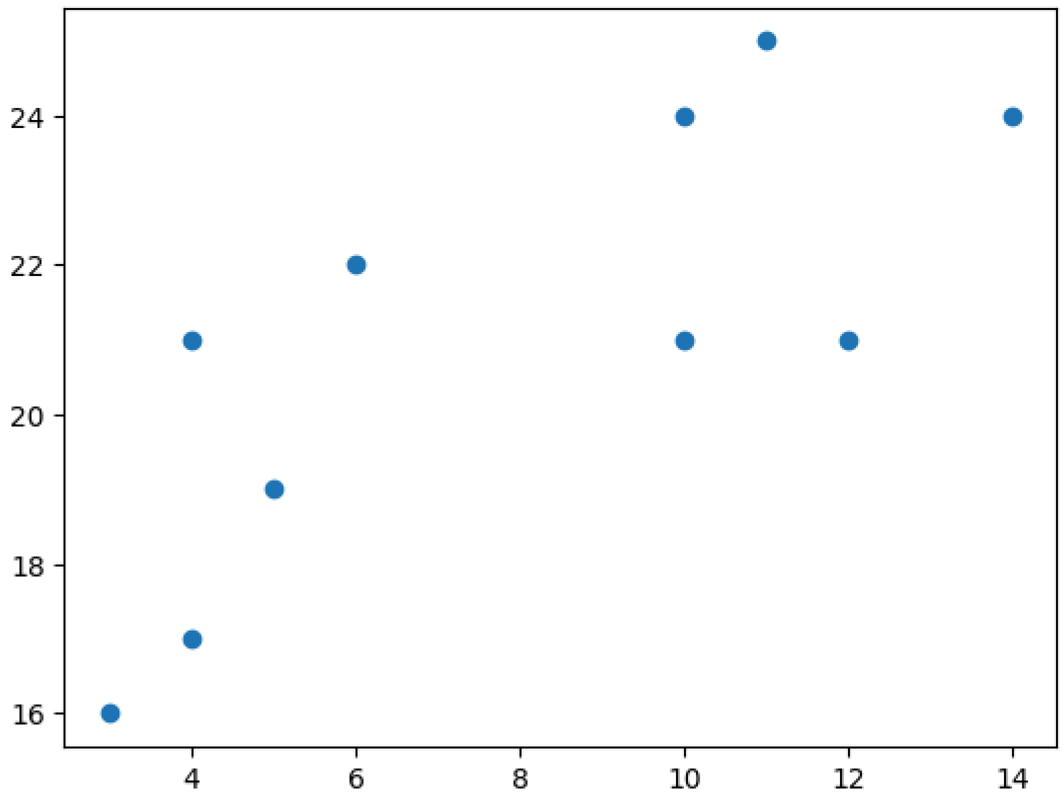
- K-means is an unsupervised learning method for clustering data points. The algorithm iteratively divides data points into K clusters by minimizing the variance in each cluster.
- Here, we will show you how to estimate the best value for K using the elbow method, then use K-means clustering to group the data points into clusters.

Algorithm

- First, each data point is randomly assigned to one of the K clusters.
- Then, we compute the centroid (functionally the centre) of each cluster, and reassign each data point to the cluster with the closest centroid.
- We repeat this process until the cluster assignments for each data point are no longer changing.
- K-means clustering requires us to select K, the number of clusters we want to group the data into. The elbow method lets us graph the inertia (a distance-based metric) and visualize the point at which it starts decreasing linearly. This point is referred to as the "elbow" and is a good estimate for the best value for K based on our data.

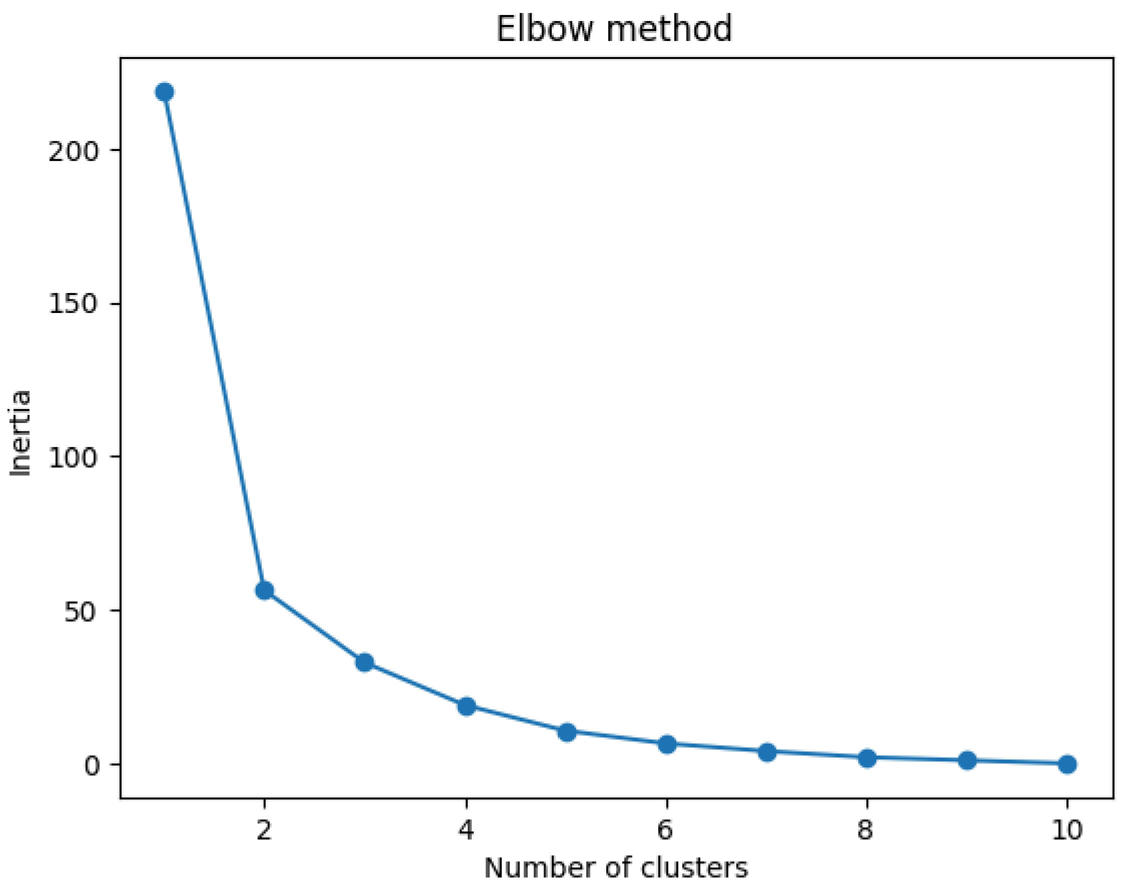
Start by visualizing some data points:

```
In [ ]: import matplotlib.pyplot as plt

        x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
        y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

        plt.scatter(x, y)
        plt.show()
```



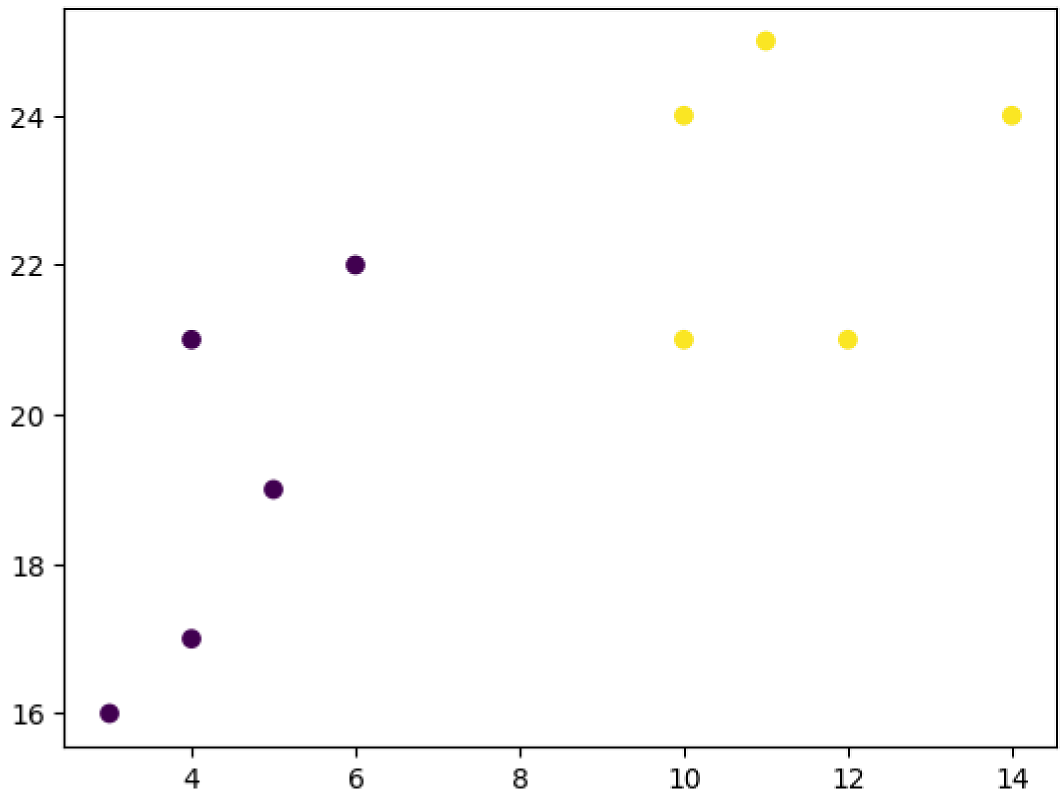
Now we utilize the elbow method to visualize the intertia for different values of K:

```
In [ ]: from sklearn.cluster import KMeans
        import warnings
        warnings.filterwarnings('ignore')

        data = list(zip(x, y))
        inertias = []

        for i in range(1,11):
            kmeans = KMeans(n_clusters=i)
            kmeans.fit(data)
            inertias.append(kmeans.inertia_)

        plt.plot(range(1,11), inertias, marker='o')
        plt.title('Elbow method')
        plt.xlabel('Number of clusters')
        plt.ylabel('Inertia')
        plt.show()
```



The elbow method shows that 2 is a good value for K, so we retrain and visualize the result:

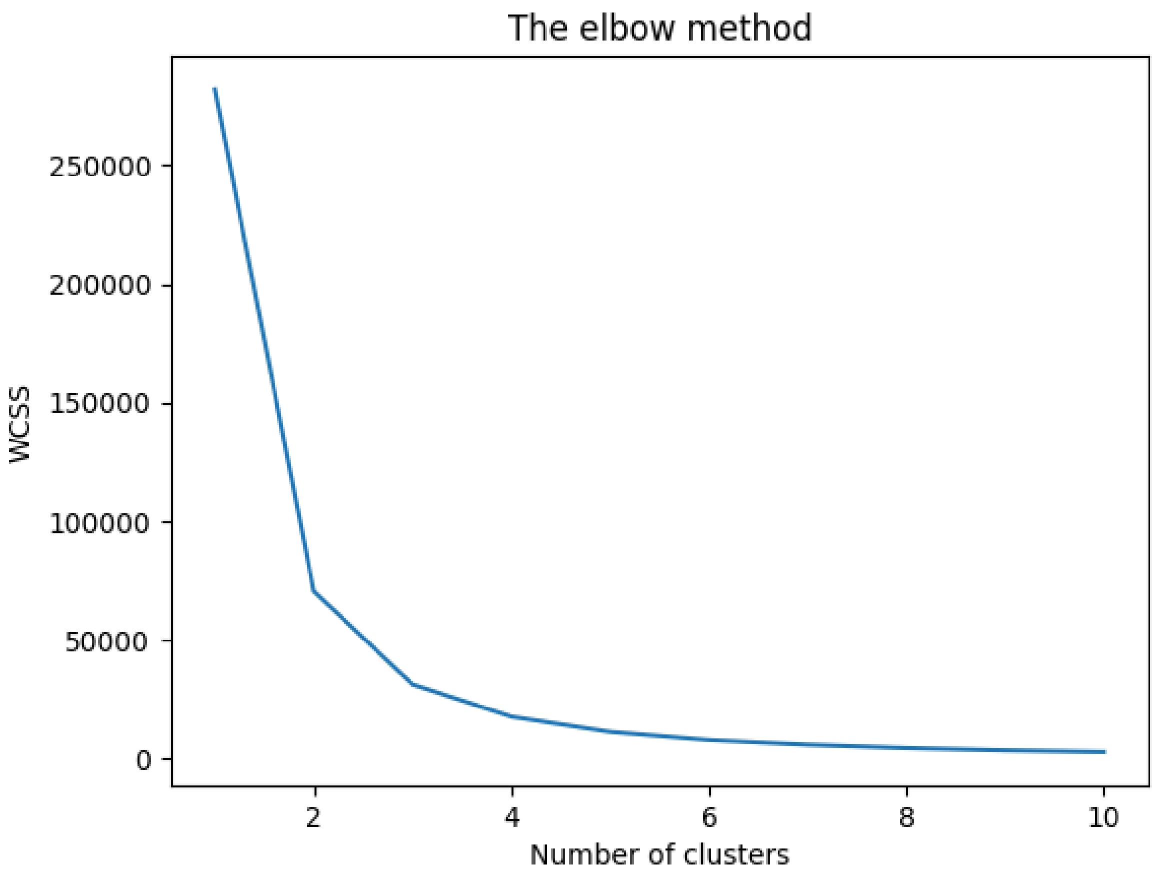
In [ ]:
```
kmeans = KMeans(n_clusters=2)
kmeans.fit(data)

plt.scatter(x, y, c=kmeans.labels_)
plt.show()
```



In [ ]:
```
#Finding the optimum number of clusters for k-means classification
from sklearn.cluster import KMeans
import pandas as pd
wcss = []

iris = pd.read_csv("iris.csv")
x = iris.iloc[:, [0, 1, 2, 3]].values

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init =
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

Using the elbow method to determine the optimal number of clusters for k-means clustering

In [ ]:
```
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```

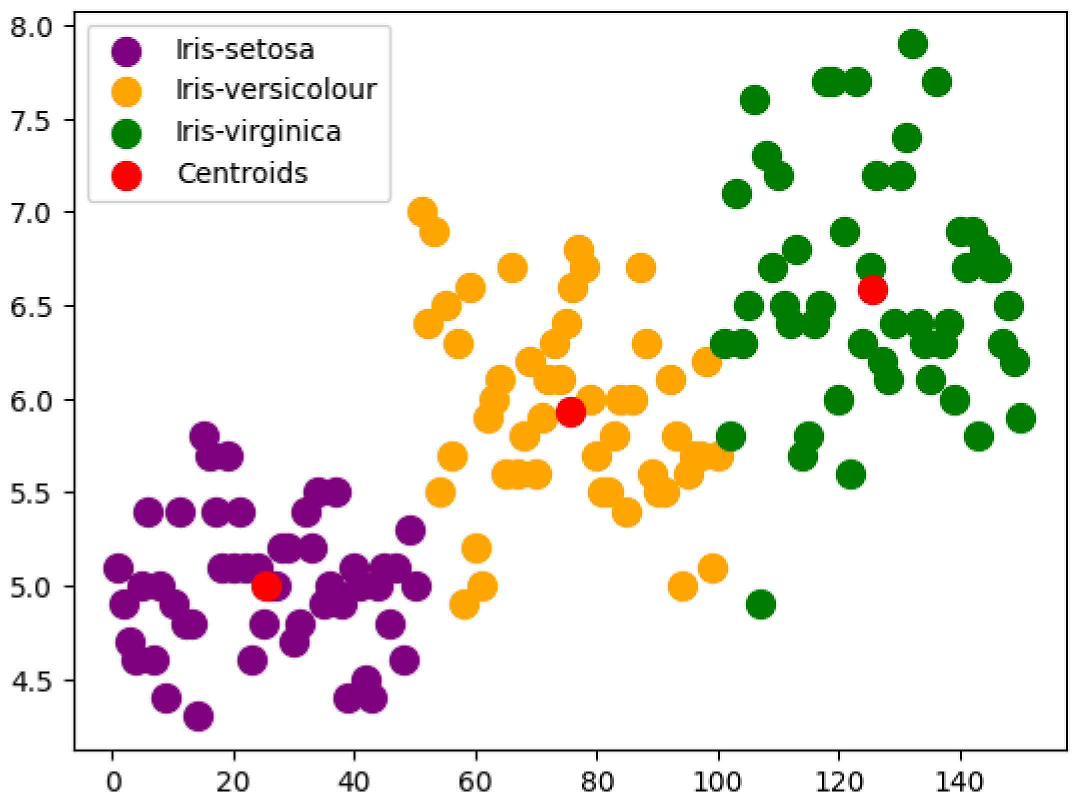


Implementing K-Means Clustering

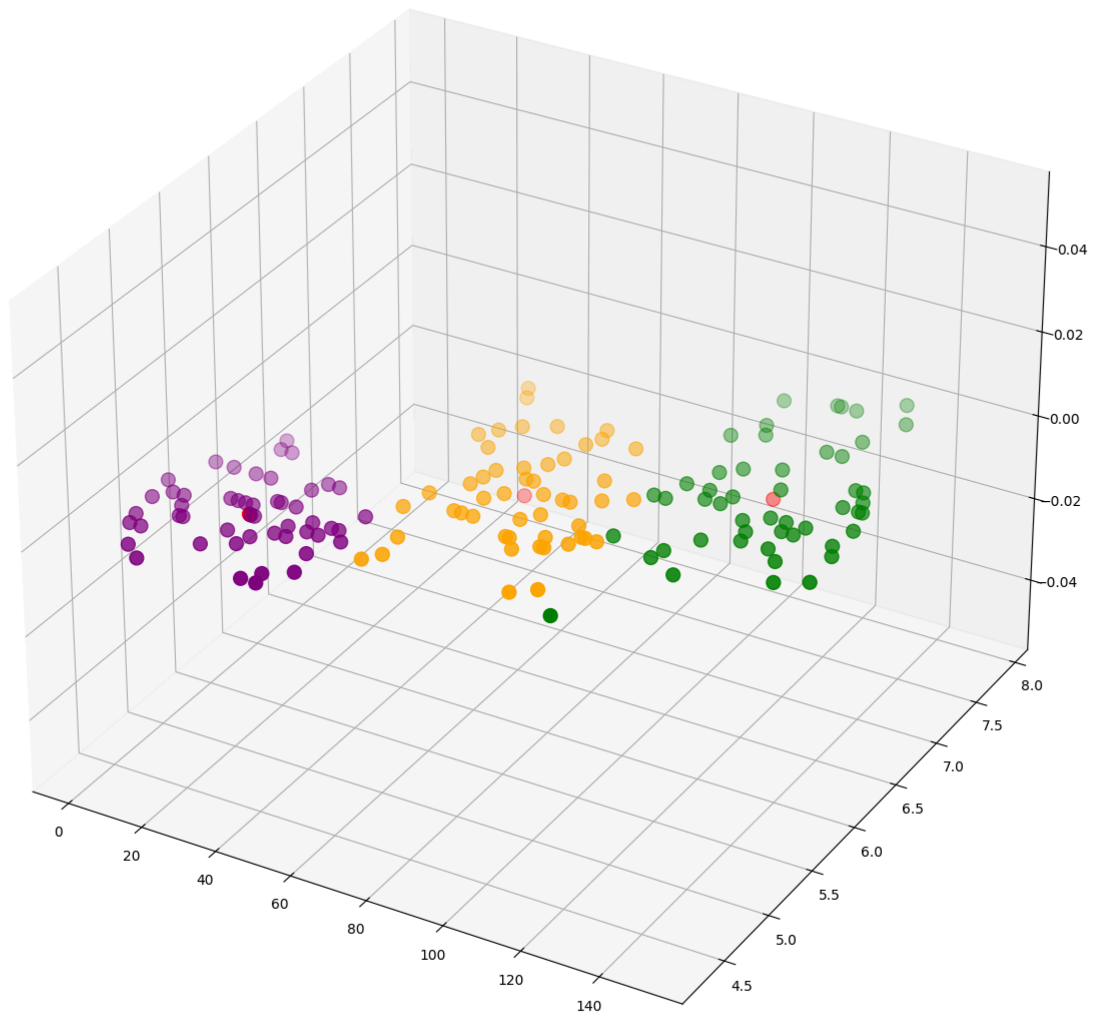
```
In [ ]: kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10,
        y_kmeans = kmeans.fit_predict(x)
```

```
In [ ]: #Visualising the clusters
        plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple', lab
        plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange', lab
        plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', labe

        #Plotting the centroids of the clusters
        plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100

        plt.legend()
```

```
Out[ ]: <matplotlib.legend.Legend at 0x214f72f68a0>
```



In [ ]:
```
# 3d scatterplot using matplotlib

fig = plt.figure(figsize = (15,15))
ax = fig.add_subplot(111, projection='3d')
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple', lab
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange', lab
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', labe

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100
plt.show()
```



Conclusion

Thus we have implemented K- means clustering algorithm and visualized the clusters.

PRACTICAL 4

Aim : Write a program to implement Linear Regression.

S/W Required : Python, VS Code, Jupyter notebook

Theory :

Linear regression is a statistical method used to model the relationship between two variables by fitting a linear equation to observed data. It is one of the simplest and most commonly used techniques in machine learning and statistics for predictive modelling.

Here is how it works:

1. **Variables**: In linear regression, you have two types of variables:
    a. **Independent Variable** (X): This is the variable whose values are manipulated or controlled by the researcher. It's also called the predictor variable.
    b. **Dependent Variable** (Y): This is the variable being studied and predicted. It is also called the response variable.

2. **Assumption of Linearity**: Linear regression assumes that there is a linear relationship between the independent variable(s) and the dependent variable. This means that changes in the independent variable(s) are associated with proportional changes in the dependent variable.

3. **Fitting the Line**: The goal of linear regression is to find the best-fitting straight line (or hyperplane, in the case of multiple independent variables) through the data points. This line is represented by the equation $y = mx + c$ where 'm' is the slope of the line and 'c' is the intercept (the point where the line crosses the Y-axis).

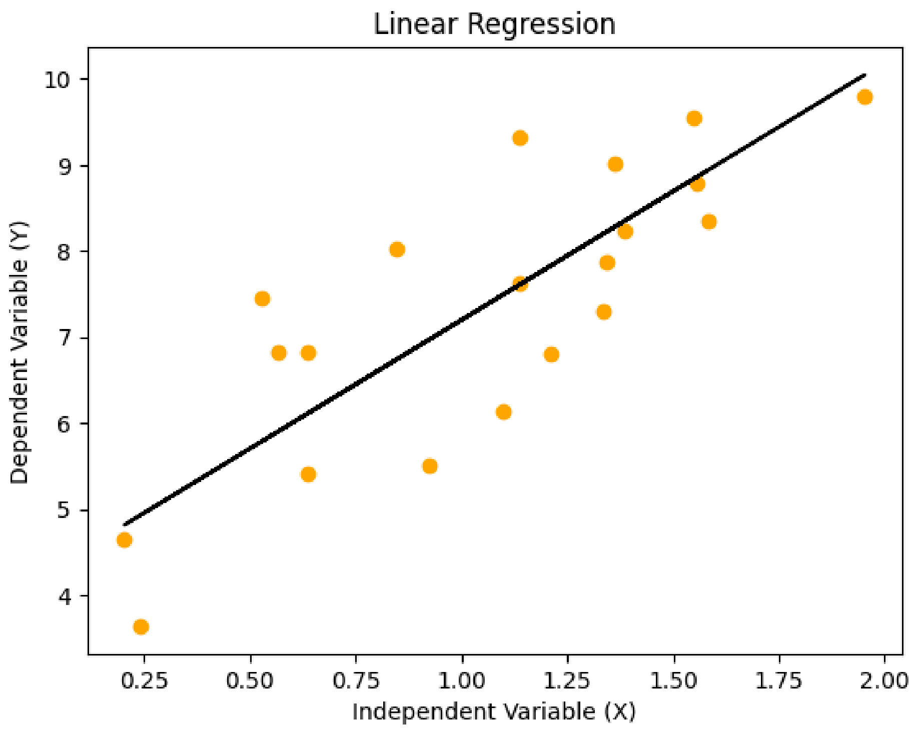
4. **Minimizing Errors**: The line is fitted to the data by minimizing the vertical distances (residuals) between the observed data points and the line. This is usually done by minimizing the sum of squared differences between the observed and predicted values of the dependent variable.

5. **Interpretation**: Once the line is fitted, you can use it to make predictions about the dependent variable based on new values of the independent variable. You can also interpret the slope and intercept of the line to understand the relationship between the variables.

6. **Evaluation**: Linear regression models are often evaluated using metrics like R-squared (which measures the proportion of the variance in the dependent variable that is predictable from the independent variable) and the root mean squared error (RMSE).
   Linear regression is widely used in various fields such as economics, finance, social sciences, and machine learning for tasks like predicting sales, analysing trends, and understanding relationships between variables.

```
In [ ]: # Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Generating synthetic dataset
np.random.seed(0)
X = 2 * np.random.rand(100, 1)  # Independent variable
Y = 4 + 3 * X + np.random.randn(100, 1)  # Dependent variable

# Splitting the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Creating a linear regression model
model = LinearRegression()

# Fitting the model to the training data
model.fit(X_train, Y_train)

# Making predictions on the testing data
Y_pred = model.predict(X_test)

# Evaluating the model
mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)

# Plotting the results
plt.scatter(X_test, Y_test, color='orange')
plt.plot(X_test, Y_pred, color='black')
plt.xlabel('Independent Variable (X)')
plt.ylabel('Dependent Variable (Y)')
plt.title('Linear Regression')
plt.show()
```

```
Mean Squared Error: 0.9177532469714291
R-squared: 0.6521157503858556
```



Model accuracy - 65%

Conclusion

Thus we have implemented Linear Regression using python

PRACTICAL 5

Aim : Write a program to implement Logistic Regression.

S/W Required : Python, VS Code, Jupyter notebook

Theory :

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

Sigmoid Function:

$$p = \frac{1}{1 + e^{-y}}$$

Apply Sigmoid function on linear regression:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_n X_n)}}$$

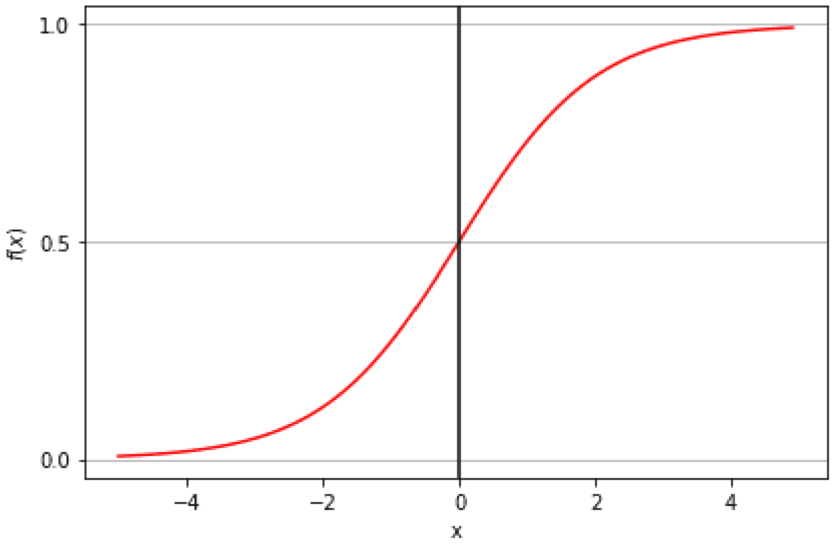
Properties of Logistic Regression:

- The dependent variable in logistic regression follows Bernoulli Distribution.
- Estimation is done through maximum likelihood.
- No R Square, Model fitness is calculated through Concordance, KS-Statistics.

Sigmoid Function

The sigmoid function, also called logistic function gives an 'S' shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO. The output cannot for example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that a patient will suffer from cancer.

$$f(x) = \frac{1}{1 + e^{-x}}$$



Types of Logistic Regression

Types of Logistic Regression:

- Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer, or No Cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.
- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

Model building in Scikit-learn

Let's build the diabetes prediction model.

Here, you are going to predict diabetes using the Logistic Regression Classifier.

Let's first load the required Pima Indian Diabetes dataset using the pandas' read CSV function. You can download data from the following link: **https://www.kaggle.com/uciml/pima-indians-diabetes-database**.

Diabetes Prediction Model

In [ ]:
```
#import pandas
import pandas as pd
# load dataset
pima = pd.read_csv("diabetes.csv")
pima.head()
```

Out[ ]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Selecting Features

Here, you need to divide the given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

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

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split the dataset by using the function train_test_split(). You need to pass 3 parameters: features, target, and test_set size. Additionally, you can use random_state to select records randomly.

In [ ]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)
```

Model Development and Prediction

First, import the Logistic Regression module and create a Logistic Regression classifier object using the LogisticRegression() function with random_state for reproducibility.

Then, fit your model on the train set using fit() and perform prediction on the test set using predict().

In [ ]:
```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression(random_state=16, max_iter=1000)

# fit the model with data
logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)
```

Model Evaluation using Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model. You can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions summed up class-wise. You have two classes 0 and 1. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions. In the output, 115 and 39 are actual predictions, and 30 and 8 are incorrect predictions.

In [ ]:
```
# import the metrics class
from sklearn import metrics
```

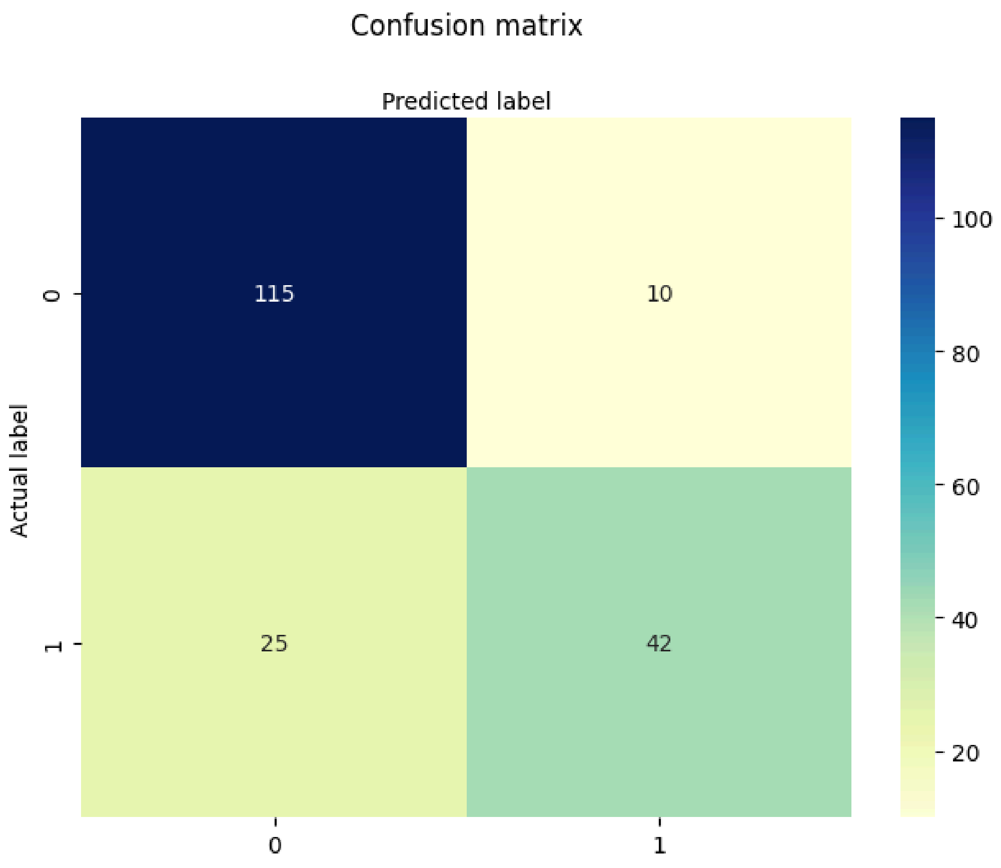
```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(cnf_matrix)
```

```
[[115  10]
 [ 25  42]]
```

Visualizing Confusion Matrix using Heatmap

Let's visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn.

In [ ]:
```
# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

plt.Text(0.5,257.44,'Predicted label')
```



Confusion Matrix Evaluation Metrics

Let's evaluate the model using classification_report for accuracy, precision, and recall.

classification rate of 80%

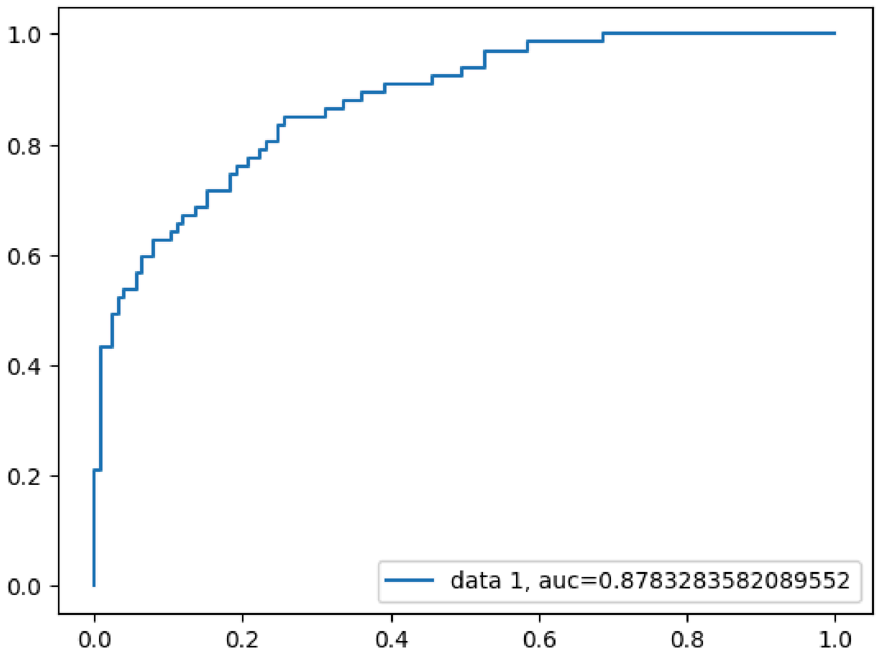
In [ ]:
```
from sklearn.metrics import classification_report
target_names = ['without diabetes', 'with diabetes']
print(classification_report(y_test, y_pred, target_names=target_names))
```

```
                  precision    recall  f1-score   support

without diabetes       0.82      0.92      0.87       125
   with diabetes       0.81      0.63      0.71        67

        accuracy                           0.82       192
       macro avg       0.81      0.77      0.79       192
    weighted avg       0.82      0.82      0.81       192
```

ROC Curve

Receiver Operating Characteristic(ROC) curve is a plot of the true positive rate against the false positive rate. It shows the tradeoff between sensitivity and specificity. AUC score for the case is **0.87**. AUC score 1 represents a perfect classifier, and 0.5 represents a worthless classifier.

```
In [ ]: y_pred_proba = logreg.predict_proba(X_test)[::,1]
        fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
        auc = metrics.roc_auc_score(y_test, y_pred_proba)
        plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
        plt.legend(loc=4)
        plt.show()
```



Conclusion :

Thus we have successfully created a model to predict diabetes with an accuracy of 80% and AUC score of 0.87 And studied Logistic regression

PRACTICAL 6

Aim : Write a program to implement K-Nearest Neighbour algorithm to classify the object. Use an appropriate dataset for classification.

S/W Required : Python, VS Code, Jupyter notebook

Theory :

The kNN algorithm can be considered a voting system, where the majority class label determines the class label of a new data point among its nearest ‘k’ (where k is an integer) neighbours in the feature space.

Imagine a small village with a few hundred residents, and you must decide which political party you should vote for. To do this, you might go to your nearest neighbours and ask which political party they support. If the majority of your’ k’ nearest neighbours support party A, then you would most likely also vote for party A.

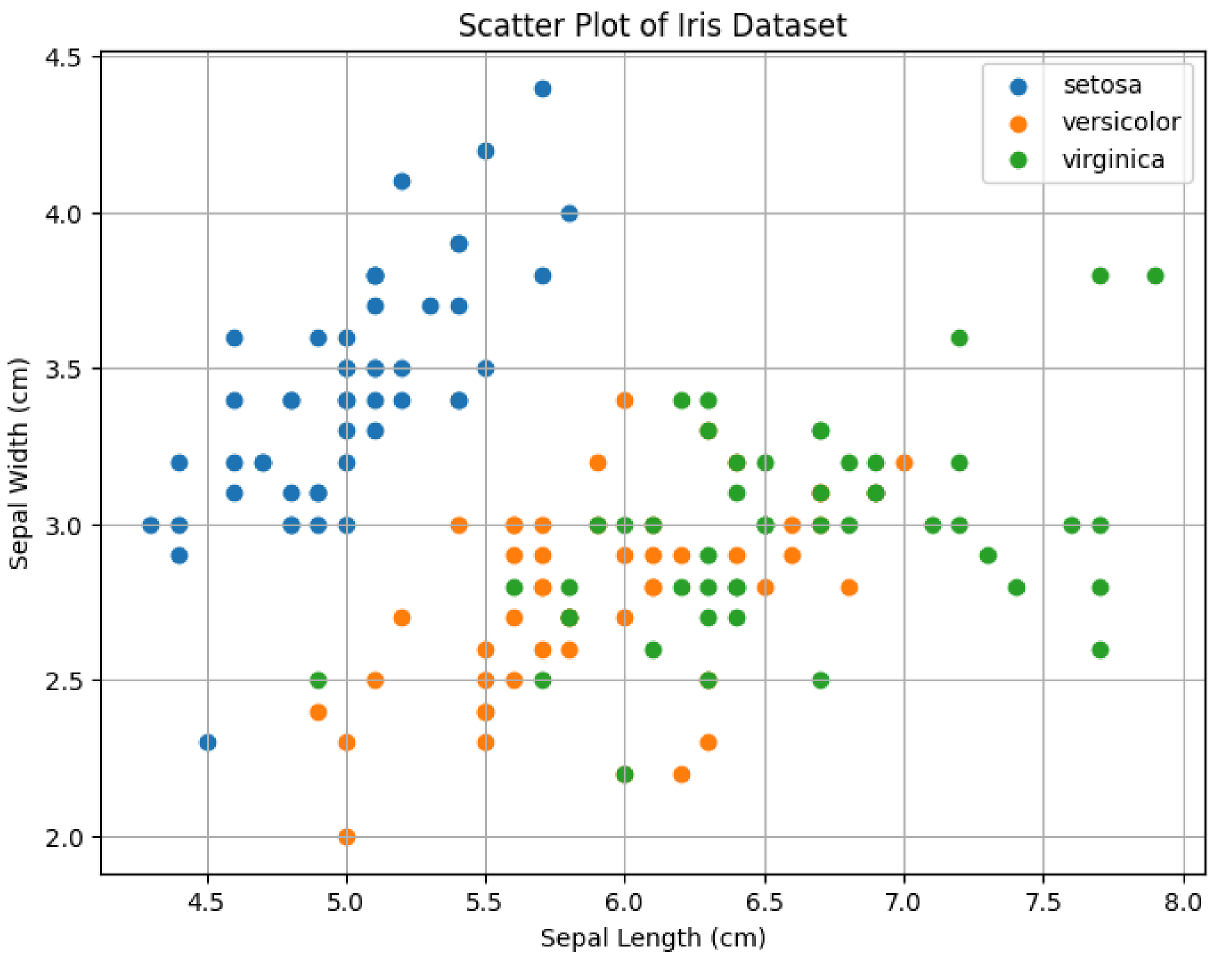
This is similar to how the kNN algorithm works, where the majority class label determines the class label of a new data point among its k nearest neighbours.

Let us take a deeper look with another example. Imagine you have data about fruit, specifically grapes and pears. You have a score for how round the fruit is and the diameter. You decide to plot these on a graph.

If someone hands you a new fruit, you could plot this on the graph too, then measure the distance to k (a number) nearest points to decide what fruit it is. In the example below, if we choose to measure three points, we can say the three nearest points are pears, so I am 100% sure this is a pear.

If we choose to measure the four nearest points, three are pears while one is a grape, so we would say we are 75% sure this is a pear.

In [ ]:
```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
X = iris.data[:, :2]  # We only take the first two features for visualization purposes
y = iris.target
target_names = iris.target_names

# Plot the data points
plt.figure(figsize=(8, 6))
for i, target_name in enumerate(target_names):
    plt.scatter(X[y == i, 0], X[y == i, 1], label=target_name)

# Set labels and title
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Scatter Plot of Iris Dataset')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create K-NN classifier
k = 3  # Number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)

# Train the classifier
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

```
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

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

Conclusion

Thus we have implemented K-NN Classifier from scratch and tested it on the Iris dataset. We have also implemented the K-NN Classifier using the scikit-learn library and compared the results of both the implementations. We have also seen that the accuracy of the scikit-learn implementation is higher than the accuracy of the implementation from scratch.

PRACTICAL 7

Aim : Write a program to implement Time Series Analysis.

S/W Required : Python, VS Code, Jupyter notebook

Theory :

Time series analysis is a statistical technique used to analyse and extract patterns from sequential data points collected or recorded over time. It is commonly applied in various fields such as finance, economics, engineering, environmental science, and many others where understanding and forecasting temporal patterns are essential.

The primary goals of time series analysis include:

1. **Descriptive Analysis**: Understanding the underlying structure, patterns, trends, and behaviours within the time series data.

2. **Forecasting**: Predicting future values or events based on past observations and patterns.

3. **Modelling**: Developing mathematical or statistical models that represent the underlying dynamics of the time series data.

4. **Anomaly Detection**: Identifying abnormal or unusual behaviours or events within the time series data.

5. **Signal Processing**: Filtering and smoothing noisy time series data to extract meaningful information.

Some common techniques and methods used in time series analysis include:

- **Moving Averages**: A simple technique to smooth out fluctuations and identify trends in time series data.
- **Exponential Smoothing**: A family of algorithms that assign exponentially decreasing weights to past observations to forecast future values.
- **Autoregressive Integrated Moving Average (ARIMA)**: A widely used method that models the next value in a time series based on its own past values, the past forecast errors, and possibly the past values of other series.
- **Seasonal Decomposition**: A method to decompose a time series into trend, seasonal, and residual components.
- **Spectral Analysis**: Analysing the frequency domain of a time series to identify periodic patterns and dominant frequencies.
- **Machine Learning Models**: Techniques such as Support Vector Machines (SVM), Random Forests, Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks can also be used for time series analysis and forecasting.

Time series analysis plays a crucial role in decision-making processes, risk management, resource allocation, and strategic planning in various domains.

In [ ]:
```
import pandas as pd
import numpy as np
import datetime

# Generate date range
start_date = datetime.datetime(2023, 8, 1)
end_date = datetime.datetime(2023, 12, 31)
date_range = pd.date_range(start=start_date, end=end_date)

# Generate synthetic time series data
np.random.seed(0)  # for reproducibility
data_values = np.random.normal(loc=100, scale=20, size=len(date_range))

# Create DataFrame
data = pd.DataFrame({'date': date_range, 'value': data_values})

# Save DataFrame to CSV
data.to_csv('time_series_data.csv', index=False)

print("CSV file 'time_series_data.csv' has been generated successfully.")
```

```
CSV file 'time_series_data.csv' has been generated successfully.
```

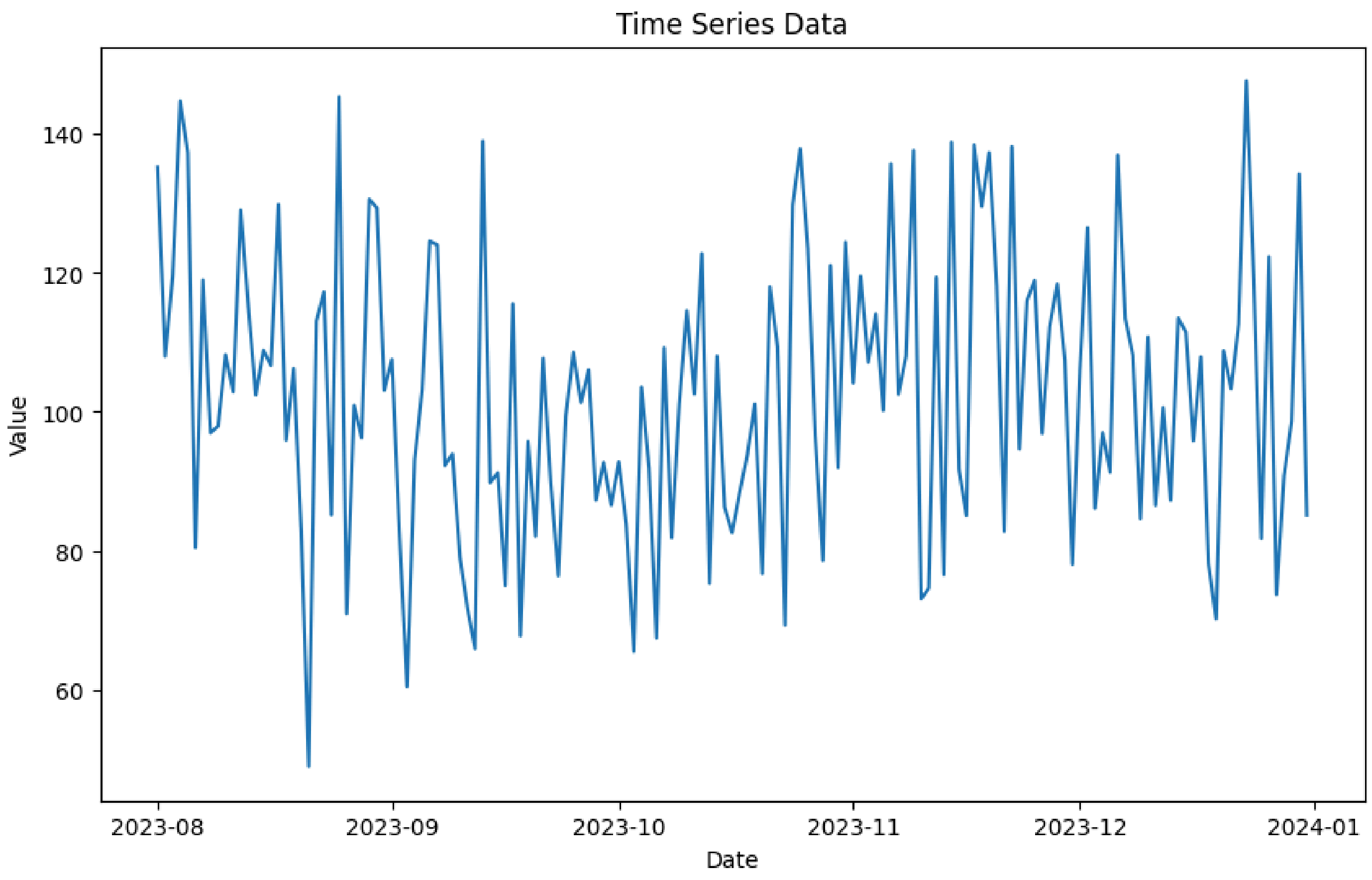
In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import warnings

warnings.filterwarnings('ignore')

# Step 1: Load the time series data
# Assuming you have a CSV file with a 'date' and 'value' column
data = pd.read_csv('time_series_data.csv')
data['date'] = pd.to_datetime(data['date'])
data.set_index('date', inplace=True)
```

In [ ]:
```
# Step 2: Explore the data
data.head()
```

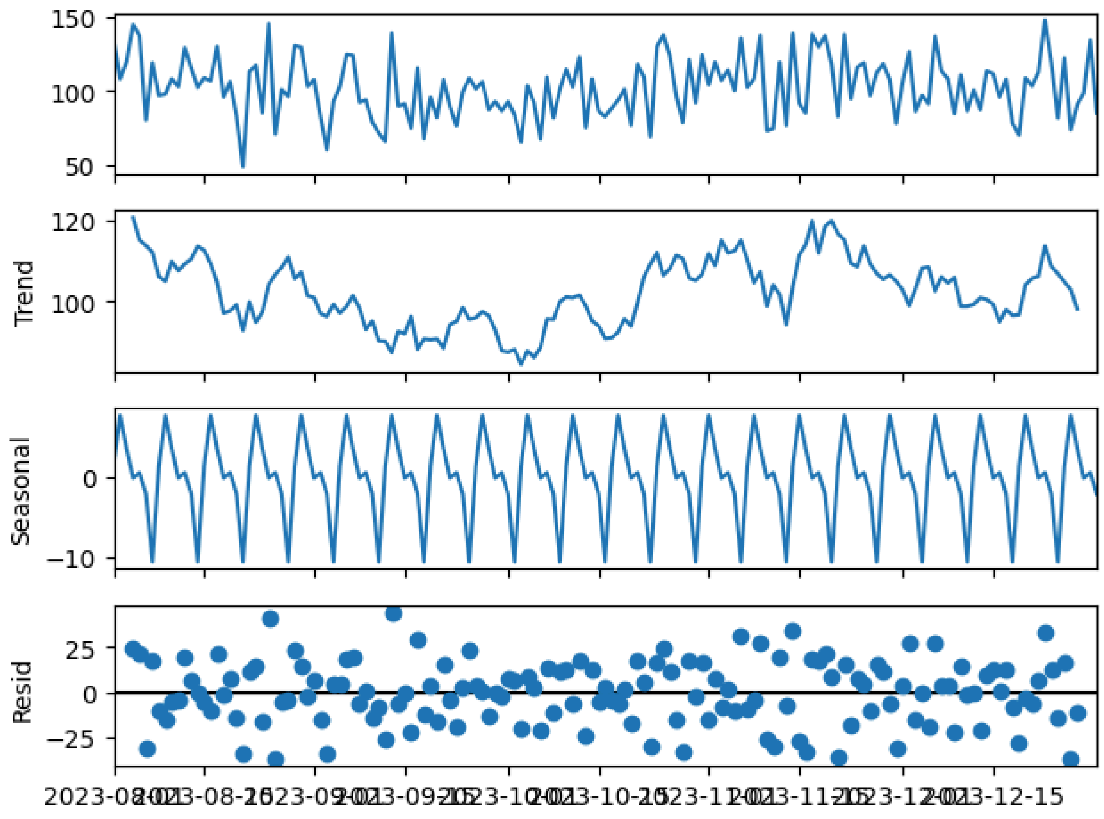
Out[ ]:

	value
date	
2023-08-01	135.281047
2023-08-02	108.003144
2023-08-03	119.574760
2023-08-04	144.817864
2023-08-05	137.351160

In [ ]:
```
# Step 3: Visualize the time series data
plt.figure(figsize=(10, 6))
plt.plot(data)
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.show()
```



```
In [ ]: # Step 4: Decompose the time series (optional)
        decomposition = sm.tsa.seasonal_decompose(data, model='additive')
        fig = decomposition.plot()
        plt.show()
```



```
In [ ]: # Step 5: Fit a time series model
        # Example: ARIMA model
```

```
model = sm.tsa.ARIMA(data, order=(1,1,1)) # Example order, you can adjust
results = model.fit()
```

In [ ]:
```
# Step 6: Evaluate the model
print(results.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  value   No. Observations:                  153
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -674.867
Date:                Sat, 06 Apr 2024   AIC                           1355.735
Time:                        14:52:44   BIC                           1364.806
Sample:                    08-01-2023   HQIC                          1359.420
                         - 12-31-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0649      0.085     -0.764      0.445      -0.232       0.102
ma.L1         -0.9172      0.038    -24.343      0.000      -0.991      -0.843
sigma2       415.3427     52.663      7.887      0.000     312.125     518.560
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 0.57
Prob(Q):                              0.97   Prob(JB):                         0.75
Heteroskedasticity (H):               0.85   Skew:                             0.02
Prob(H) (two-sided):                  0.55   Kurtosis:                         2.70
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [ ]:
```
# Step 7: Forecasting (optional)
forecast = results.forecast(steps=10) # Example forecast for the next 10 steps
print(forecast)
```

```
2024-01-01    104.229809
2024-01-02    102.987960
2024-01-03    103.068597
2024-01-04    103.063361
2024-01-05    103.063701
2024-01-06    103.063679
2024-01-07    103.063681
2024-01-08    103.063680
2024-01-09    103.063680
2024-01-10    103.063680
Freq: D, Name: predicted_mean, dtype: float64
```

Conclusion

Thus we have implemented Time Series Analysis using python, matplotlib.

PRACTICAL 8

Aim : Installation of Hadoop in Linux operating system

S/W Required : Hadoop, Linux

Theory :

Installation of Hadoop: Hadoop software can be installed in three modes of operation:

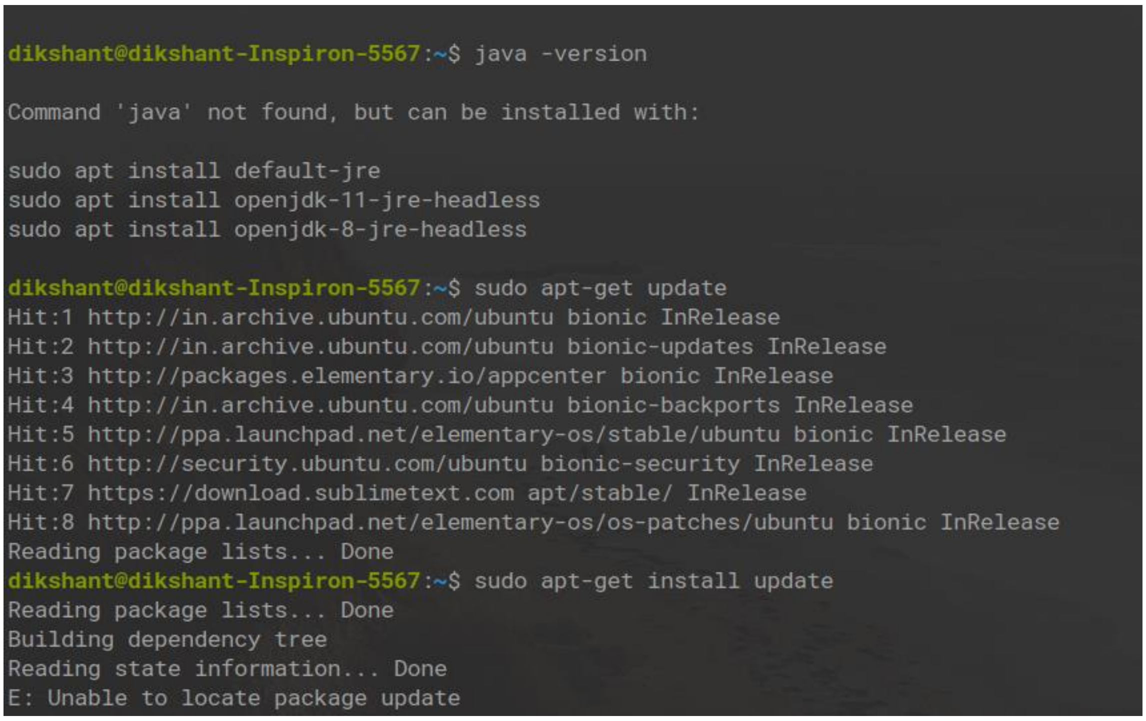
Stand Alone Mode: Hadoop is a distributed software and is designed to run on a commodity of machines. However, we can install it on a single node in stand-alone mode. In this mode, Hadoop software runs as a single monolithic java process. This mode is extremely useful for debugging purpose. You can first test run your Map-Reduce application in this mode on small data, before actually executing it on cluster with big data.

Pseudo Distributed Mode: In this mode also, Hadoop software is installed on a Single Node. Various daemons of Hadoop will run on the same machine as separate java processes. Hence all the daemons namely NameNode, DataNode, SecondaryNameNode, JobTracker, TaskTracker run on single machine.

Fully Distributed Mode: In Fully Distributed Mode, the daemons NameNode, JobTracker, SecondaryNameNode (Optional and can be run on a separate node) run on the Master Node. The daemons DataNode and TaskTracker run on the Slave Node.

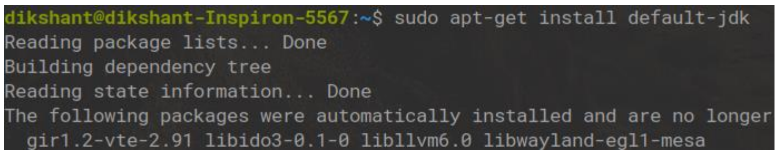
Hadoop Installation: Ubuntu Operating System in stand-alone mode
Steps for Installation

1. Run the commands $sudo apt-get update/ install



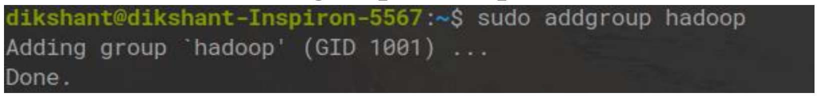
```
dikshant@dikshant-Inspiron-5567:~$ java -version

Command 'java' not found, but can be installed with:

sudo apt install default-jre
sudo apt install openjdk-11-jre-headless
sudo apt install openjdk-8-jre-headless

dikshant@dikshant-Inspiron-5567:~$ sudo apt-get update
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://packages.elementary.io/appcenter bionic InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:5 http://ppa.launchpad.net/elementary-os/stable/ubuntu bionic InRelease
Hit:6 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:7 https://download.sublimetext.com apt/stable/ InRelease
Hit:8 http://ppa.launchpad.net/elementary-os/os-patches/ubuntu bionic InRelease
Reading package lists... Done
dikshant@dikshant-Inspiron-5567:~$ sudo apt-get install update
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package update
```

2. Install jdk



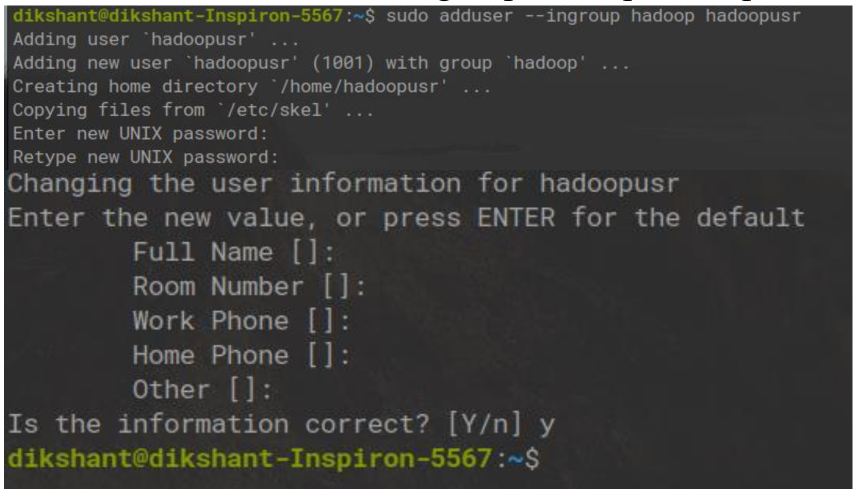
```
dikshant@dikshant-Inspiron-5567:~$ sudo apt-get install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer
  gir1.2-vte-2.91 libido3-0.1-0 libllvm6.0 libwayland-egl1-mesa
```

3. $ sudo addgroup hadoop



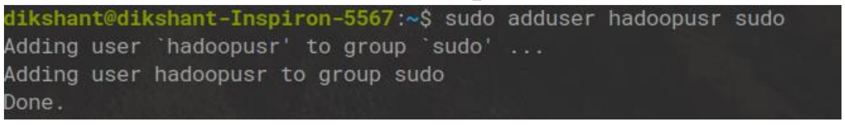
```
dikshant@dikshant-Inspiron-5567:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1001) ...
Done.
```

4. $ sudo adduser --ingroup hadoop hadoopusr



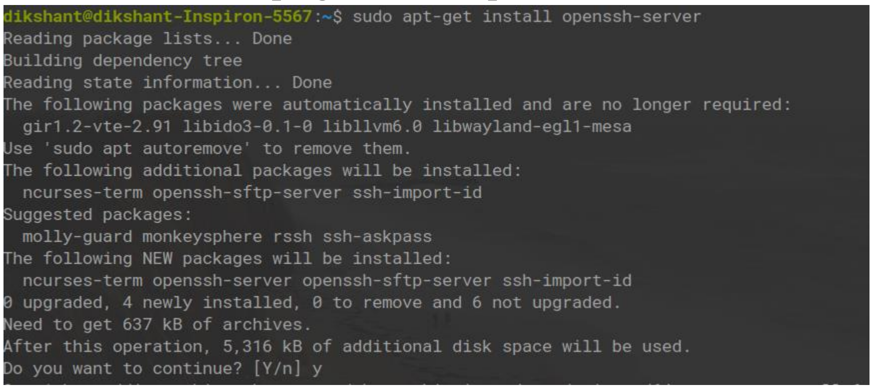
```
dikshant@dikshant-Inspiron-5567:~$ sudo adduser --ingroup hadoop hadoopusr
Adding user `hadoopusr' ...
Adding new user `hadoopusr' (1001) with group `hadoop' ...
Creating home directory `/home/hadoopusr' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
Changing the user information for hadoopusr
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
dikshant@dikshant-Inspiron-5567:~$
```

5. $ sudo adduser hadoopusr sudo



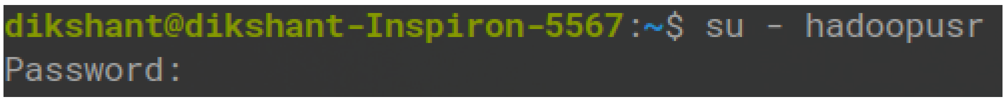
```
dikshant@dikshant-Inspiron-5567:~$ sudo adduser hadoopusr sudo
Adding user `hadoopusr' to group `sudo' ...
Adding user hadoopusr to group sudo
Done.
```

6. $ sudo apt-get install openssh-server



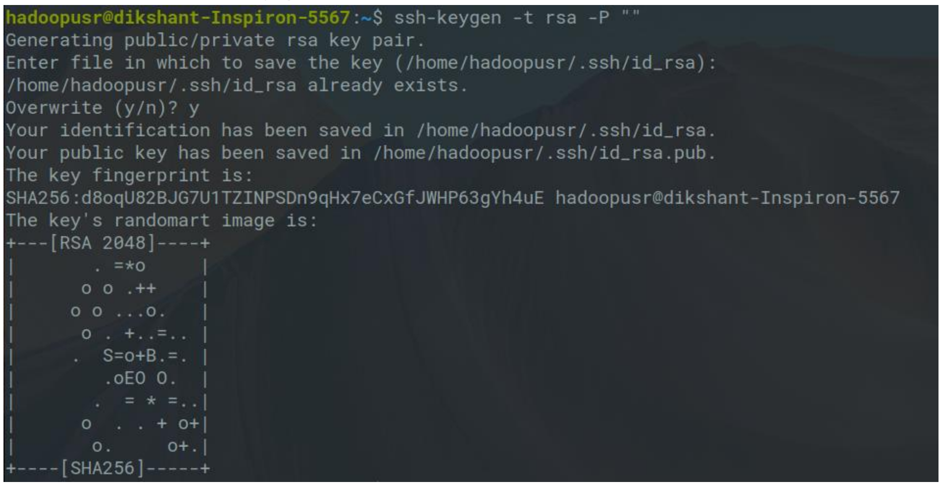
```
dikshant@dikshant-Inspiron-5567:~$ sudo apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-vte-2.91 libido3-0.1-0 libllvm6.0 libwayland-egl1-mesa
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere rssh ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 6 not upgraded.
Need to get 637 kB of archives.
After this operation, 5,316 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

7. $ su - hadoopusr



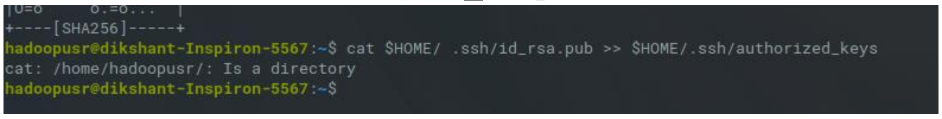
```
dikshant@dikshant-Inspiron-5567:~$ su - hadoopusr
Password:
```

8. $ssh-keygen -t rsa -P ""



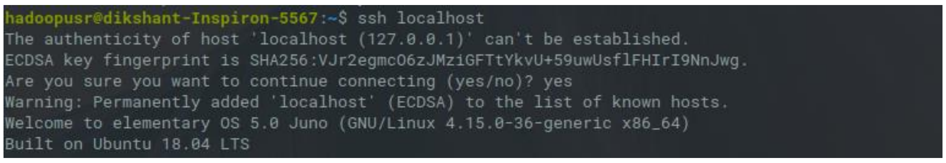
```
hadoopusr@dikshant-Inspiron-5567:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoopusr/.ssh/id_rsa):
/home/hadoopusr/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Your identification has been saved in /home/hadoopusr/.ssh/id_rsa.
Your public key has been saved in /home/hadoopusr/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:d8oqU82BJG7U1TZINPSDn9qHx7eCxGfJWHP63gYh4uE hadoopusr@dikshant-Inspiron-5567
The key's randomart image is:
+---[RSA 2048]----+
|        . =*o    |
|      o o .++    |
|     o o ...o.   |
|      o . +..=.. |
|     .  S=o+B.=. |
|         .oEO O. |
|         . = * =..|
|      o  . . + o+|
|       o.     o+.|
+----[SHA256]-----+
```

9. $cat $HOME/ .ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys



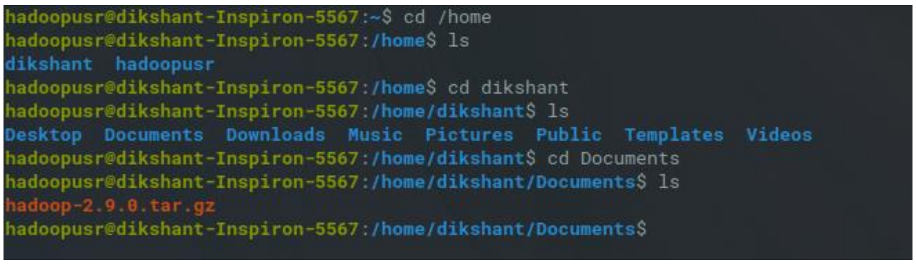
```
+----[SHA256]-----+
hadoopusr@dikshant-Inspiron-5567:~$ cat $HOME/ .ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
cat: /home/hadoopusr/: Is a directory
hadoopusr@dikshant-Inspiron-5567:~$
```

10. $ssh localhost



```
hadoopusr@dikshant-Inspiron-5567:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:VJr2egmcO6zJMziGFTtYkvU+59uwUsflFHIrI9NnJwg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to elementary OS 5.0 Juno (GNU/Linux 4.15.0-36-generic x86_64)
Built on Ubuntu 18.04 LTS
```

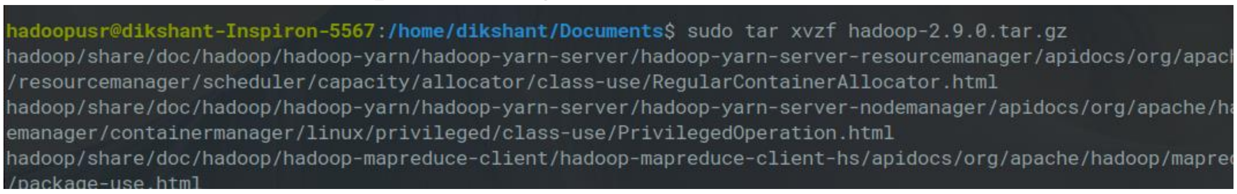
11. Now download the package that you will going to install . download it from Hadoop-2.9.0
12. Once you have download hadoop-2.9.0.tar.gz then place this tar file to your preferred location then extract it with below commands. In my case I moved it to the /Documents folder.



```
hadoopusr@dikshant-Inspiron-5567:~$ cd /home
hadoopusr@dikshant-Inspiron-5567:/home$ ls
dikshant  hadoopusr
hadoopusr@dikshant-Inspiron-5567:/home$ cd dikshant
hadoopusr@dikshant-Inspiron-5567:/home/dikshant$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
hadoopusr@dikshant-Inspiron-5567:/home/dikshant$ cd Documents
hadoopusr@dikshant-Inspiron-5567:/home/dikshant/Documents$ ls
hadoop-2.9.0.tar.gz
hadoopusr@dikshant-Inspiron-5567:/home/dikshant/Documents$
```

13. Extraction

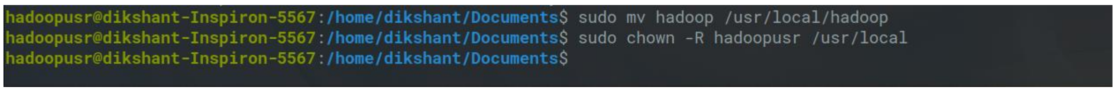
$ sudo tar xvzf hadoop-2.9.0.tar.gz



```
hadoopusr@dikshant-Inspiron-5567:/home/dikshant/Documents$ sudo tar xvzf hadoop-2.9.0.tar.gz
hadoop/share/doc/hadoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/apidocs/org/apach
/resourcemanager/scheduler/capacity/allocator/class-use/RegularContainerAllocator.html
hadoop/share/doc/hadoop/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-nodemanager/apidocs/org/apache/ha
emanager/containermanager/linux/privileged/class-use/PrivilegedOperation.html
hadoop/share/doc/hadoop/hadoop-mapreduce-client/hadoop-mapreduce-client-hs/apidocs/org/apache/hadoop/mapred
/package-use.html
```

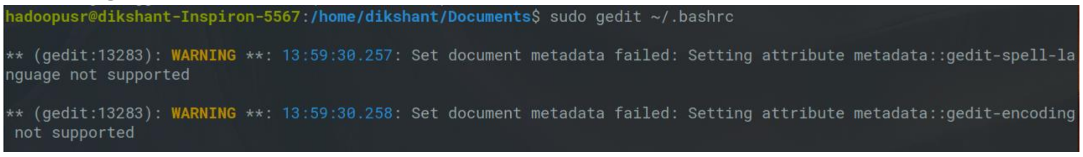
14.$ sudo mv hadoop /usr/local/hadoop

$ sudo chown -R hadoopusr /usr/local



```
hadoopusr@dikshant-Inspiron-5567:/home/dikshant/Documents$ sudo mv hadoop /usr/local/hadoop
hadoopusr@dikshant-Inspiron-5567:/home/dikshant/Documents$ sudo chown -R hadoopusr /usr/local
hadoopusr@dikshant-Inspiron-5567:/home/dikshant/Documents$
```

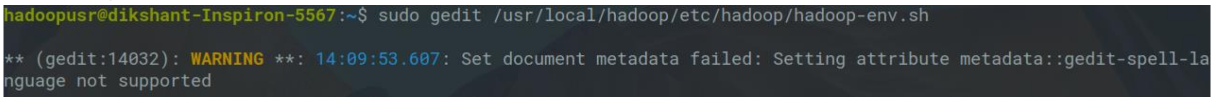
15.$sudo gedit ~/.bashrc



```
hadoopusr@dikshant-Inspiron-5567:/home/dikshant/Documents$ sudo gedit ~/.bashrc

** (gedit:13283): WARNING **: 13:59:30.257: Set document metadata failed: Setting attribute metadata::gedit-spell-la
nguage not supported

** (gedit:13283): WARNING **: 13:59:30.258: Set document metadata failed: Setting attribute metadata::gedit-encoding
 not supported
```

Then a ./bashrc file is open then copy the below command inside this file (change java version according to your PC java version like it might be java-8-openjdk-amd64).

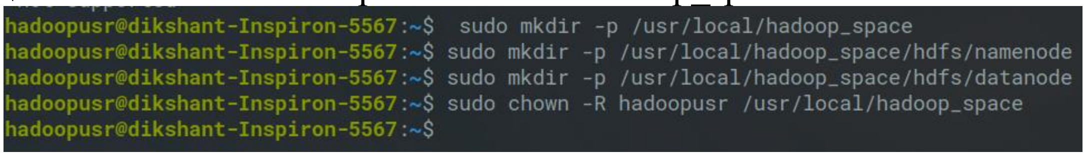
16.$sudo gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh



```
hadoopusr@dikshant-Inspiron-5567:~$ sudo gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh

** (gedit:14032): WARNING **: 14:09:53.607: Set document metadata failed: Setting attribute metadata::gedit-spell-la
nguage not supported
```

17.Now we have successfully configured all the files. So now it is time to check our installation. As we know that in Hadoop architecture we have name node and other blocks so we need to make one directory i.e. hadoop_space. Inside this directory we make another directory i.e. hdfs and namenode and datanode. The command to make directory is given below:

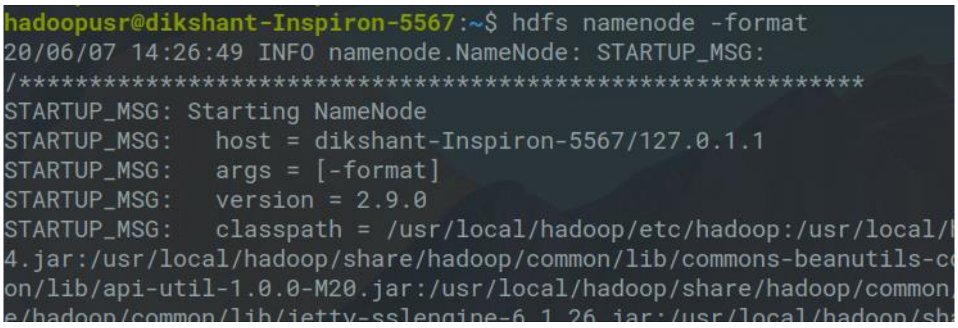
$sudo mkdir -p /usr/local/hadoop_space
$sudo mkdir -p /usr/local/hadoop_space/hdfs/namenode
$sudo mkdir -p /usr/local/hadoop_space/hdfs/datanode
$ sudo chown -R hadoopusr /usr/local/hadoop_space



```
hadoopusr@dikshant-Inspiron-5567:~$  sudo mkdir -p /usr/local/hadoop_space
hadoopusr@dikshant-Inspiron-5567:~$ sudo mkdir -p /usr/local/hadoop_space/hdfs/namenode
hadoopusr@dikshant-Inspiron-5567:~$ sudo mkdir -p /usr/local/hadoop_space/hdfs/datanode
hadoopusr@dikshant-Inspiron-5567:~$ sudo chown -R hadoopusr /usr/local/hadoop_space
hadoopusr@dikshant-Inspiron-5567:~$
```

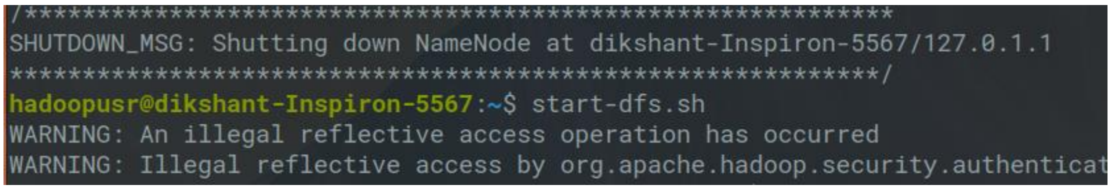
18.Running Hadoop

19. $ hdfs namenode -format



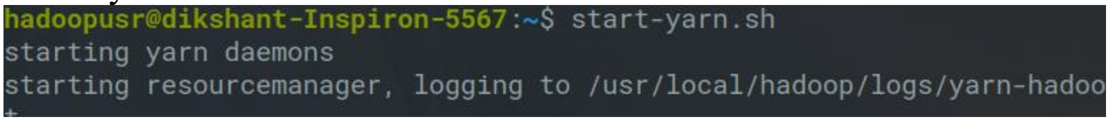
```
hadoopusr@dikshant-Inspiron-5567:~$ hdfs namenode -format
20/06/07 14:26:49 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = dikshant-Inspiron-5567/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.9.0
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop:/usr/local/
4.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-beanutils-c
on/lib/api-util-1.0.0-M20.jar:/usr/local/hadoop/share/hadoop/common
e/hadoop/common/lib/jetty-sslengine-6.1.26.jar:/usr/local/hadoop/sh
```

20. Now we need to start the DFS i.e. Distributed File System.
    $ start-dfs.sh



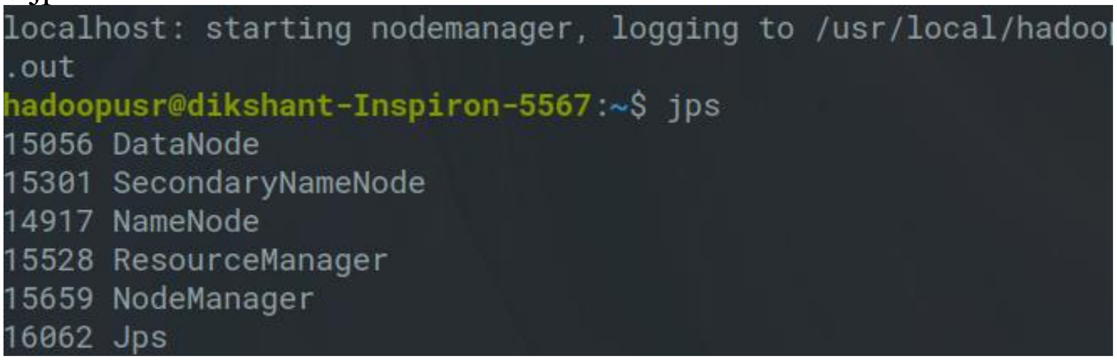
```
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at dikshant-Inspiron-5567/127.0.1.1
************************************************************/
hadoopusr@dikshant-Inspiron-5567:~$ start-dfs.sh
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authenticat
```

21.now the last thing you need to start is yarn
    $ start-yarn.sh



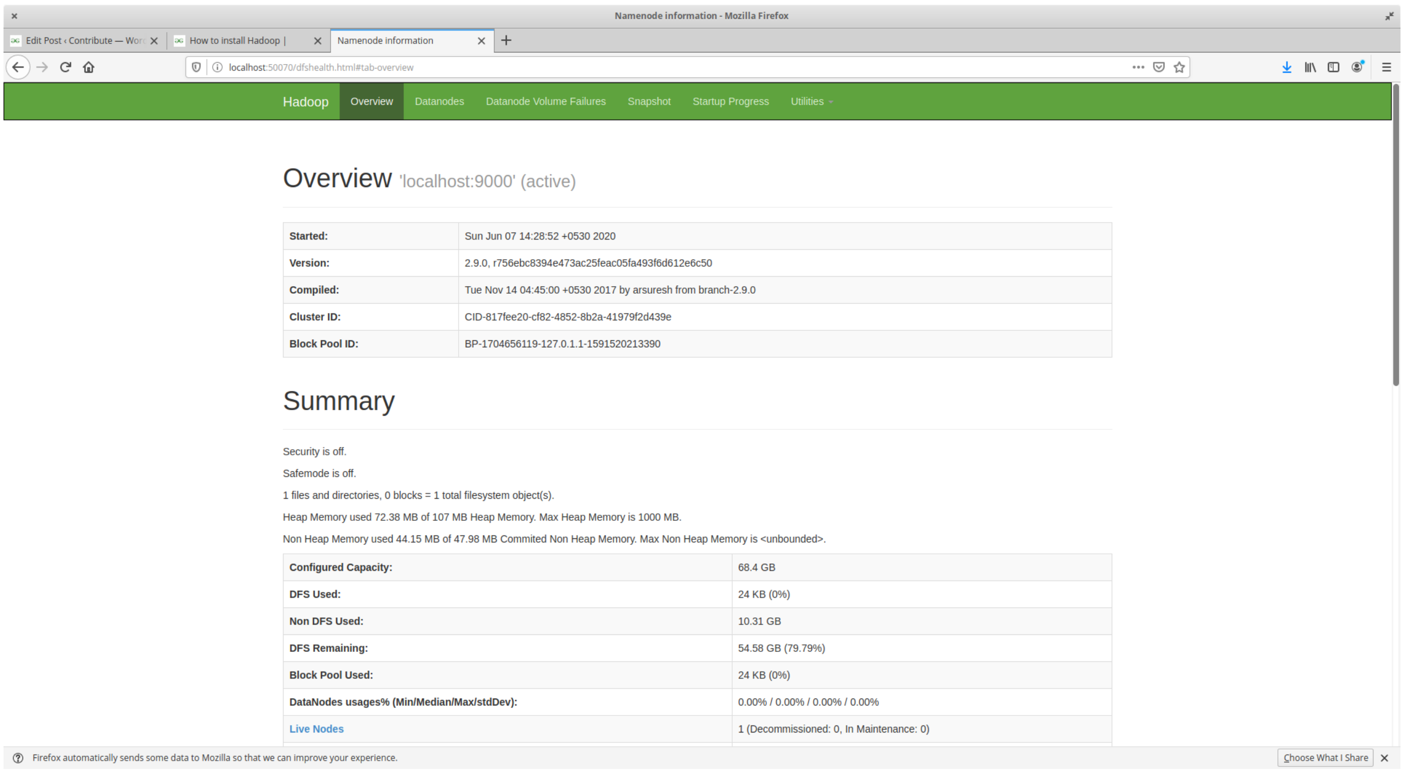
```
hadoopusr@dikshant-Inspiron-5567:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoo
```

22.Now use the following command:
    $ jps



```
localhost: starting nodemanager, logging to /usr/local/hadoo
.out
hadoopusr@dikshant-Inspiron-5567:~$ jps
15056 DataNode
15301 SecondaryNameNode
14917 NameNode
15528 ResourceManager
15659 NodeManager
16062 Jps
```

23.You have successfully installed hadoop on your system. Now to check all you cluster information you can use localhost:50070 in your browser. The Interface will look like as:


Namenode information - Mozilla Firefox
Edit Post ‹ Contribute — Wor
How to install Hadoop |
Namenode information
localhost:50070/dfshealth.html#tab-overview
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Firefox automatically sends some data to Mozilla so that we can improve your experience.
Choose What I Share

Overview 'localhost:9000' (active)

Started:	Sun Jun 07 14:28:52 +0530 2020
Version:	2.9.0, r756ebc8394e473ac25feac05fa493f6d612e6c50
Compiled:	Tue Nov 14 04:45:00 +0530 2017 by arsuresh from branch-2.9.0
Cluster ID:	CID-817fee20-cf82-4852-8b2a-41979f2d439e
Block Pool ID:	BP-1704656119-127.0.1.1-1591520213390

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 72.38 MB of 107 MB Heap Memory. Max Heap Memory is 1000 MB.

Non Heap Memory used 44.15 MB of 47.98 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	68.4 GB
DFS Used:	24 KB (0%)
Non DFS Used:	10.31 GB
DFS Remaining:	54.58 GB (79.79%)
Block Pool Used:	24 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)

Conclusion: Thus we have studied Hadoop Installation