Name : Sandra P M

Rollno: 34

Batch: MCA B

Date :22/08/22

Machine learning lab record

Experiment-1

Aim: Matrix operations (using vectorization) and transformation using python and SVD using Python.

import numpy as np

a = np.array([1, 2, 3])   # Create a rank 1 array

print("type: %s" %type(a))            # Prints "<class 'numpy.ndarray'>"

print("shape: %s" %a.shape)            # Prints "(3,)"

print(a[0], a[1], a[2])   # Prints "1 2 3"

a[0] = 5                  # Change an element of the array

print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array

print("\n shape of b:",b.shape)                     # Prints "(2, 3)"

print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"

a = np.zeros((2,2))   # Create an array of all zeros

print("All zeros matrix:\n  %s" %a)              # Prints "[[ 0.  0.]

                      #          [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones

print("\nAll ones matrix:\n  %s" %b)              # Prints "[[ 1.  1.]]"

d = np.eye(2)         # Create a 2x2 identity matrix

print("\n identity matrix: \n%s"%d)              # Prints "[[ 1.  0.]

                      #          [ 0.  1.]]"

e = np.random.random((2,2))  # Create an array filled with random values

print("\n random matrix: \n%s"%e)

Output

type: <class 'numpy.ndarray'>

shape: 3

1 2 3

[5 2 3]

shape of b: (2, 3)

1 2 4

All zeros matrix:

[[0. 0.]

[0. 0.]]

All ones matrix:

[[1. 1.]]

identity matrix:

[[1. 0.]

[0. 1.]]

random matrix:

[[0.34891702 0.76887592]

[0.19143942 0.1416376 ]]

2 import numpy as np

a=np.array([5, 6, 7])

print("type: %s" %type(a))

print("shape: %s" %a.shape)

print(a[0], a[1], a[2])

a[0] = 5

print(a)

a = np.zeros((2,2))

print("All zeros matrix:\n  %s" %a)

b = np.ones((1,2))

print("\nAll ones matrix:\n  %s" %b)

d = np.eye(2)

print("\n identity matrix: \n%s"%d)

e = np.random.random((2,2))

print("\n random matrix: \n%s"%e)

type: <class 'numpy.ndarray'>

shape: 3

5 6 7

[5 6 7]

All zeros matrix:

[[0. 0.]

[0. 0.]]

All ones matrix:

[[1. 1.]]

identity matrix:

[[1. 0.]

[0. 1.]]

random matrix:

[[0.2394039 0.35878291]

[0.67513405 0.86445199]]

Name : Sandra P M

Roll no: 34

Batch : MCA B

Date :29/08/22

Experiment-2

AIM: Programs using matplotlib / plotly / bokeh / seaborn for data visualisation.

import plotly.express as px

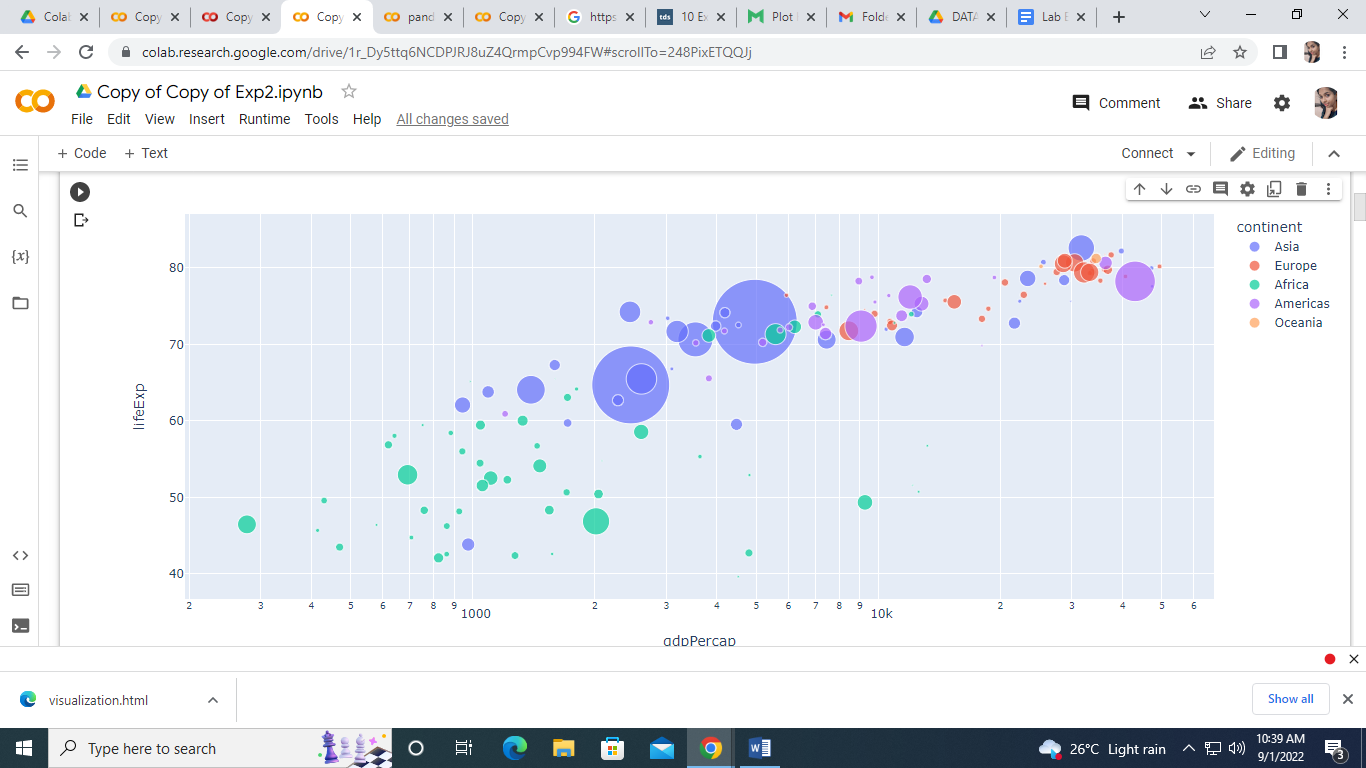
df = px.data.gapminder()

fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp",

           size="pop", color="continent",

                 hover\_name="country", log\_x=True, size\_max=60)

fig.show()



import matplotlib.pyplot as plt

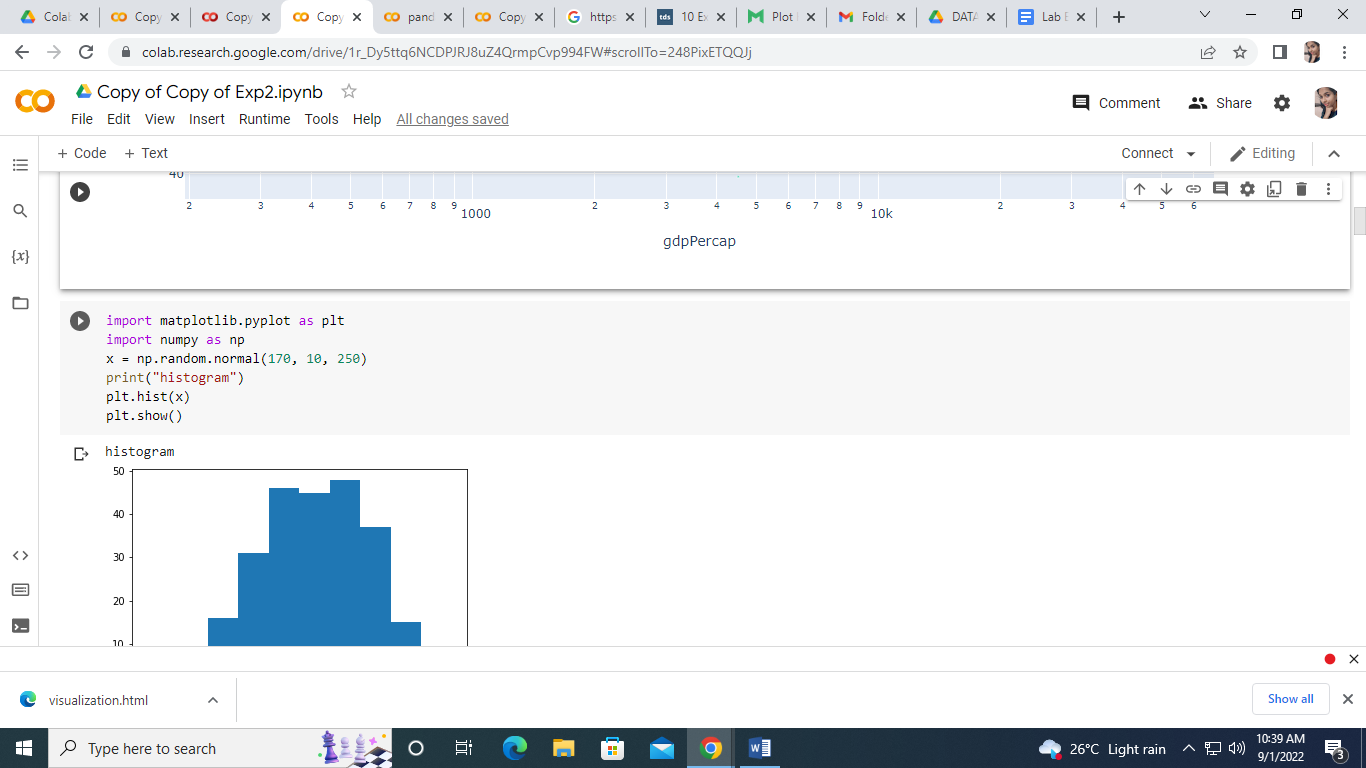
import numpy as np

x = np.random.normal(170, 10, 250)

print("histogram")

plt.hist(x)

plt.show()



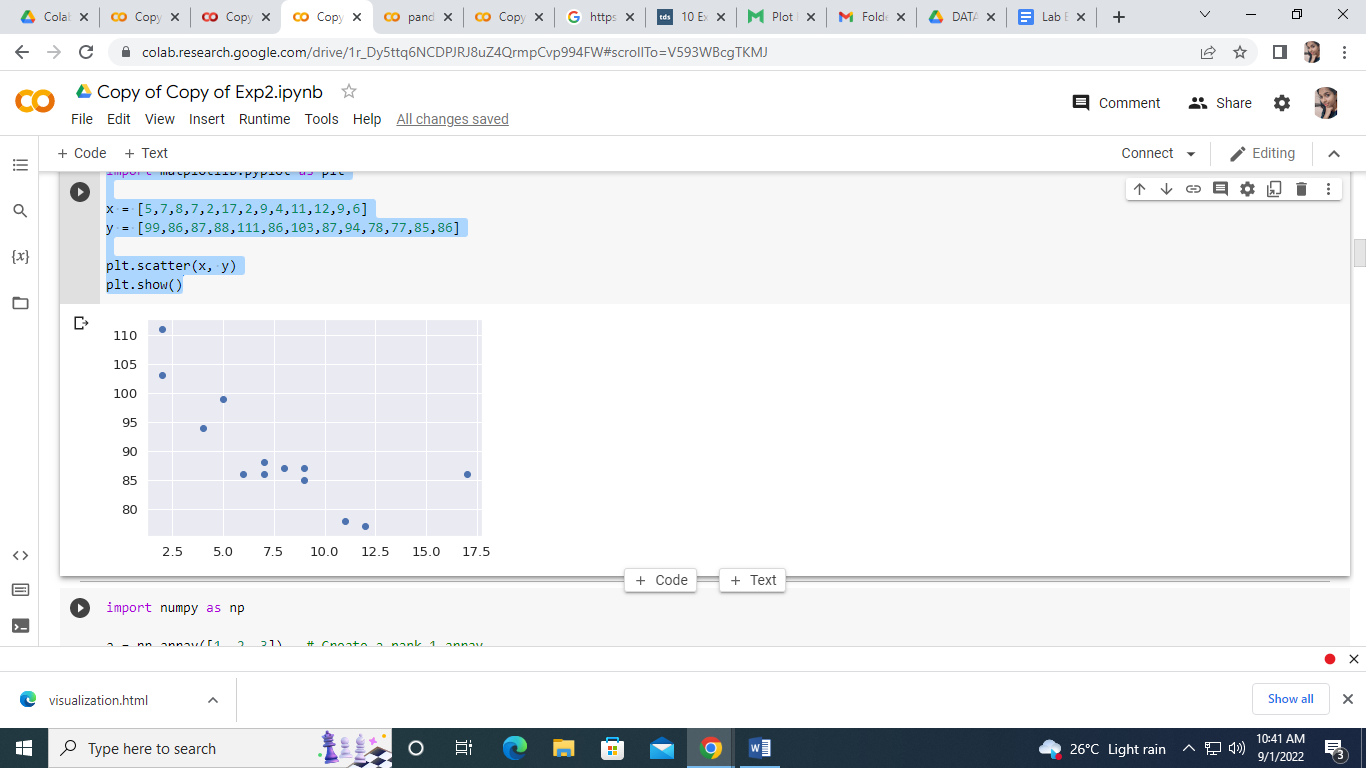
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)

plt.show()



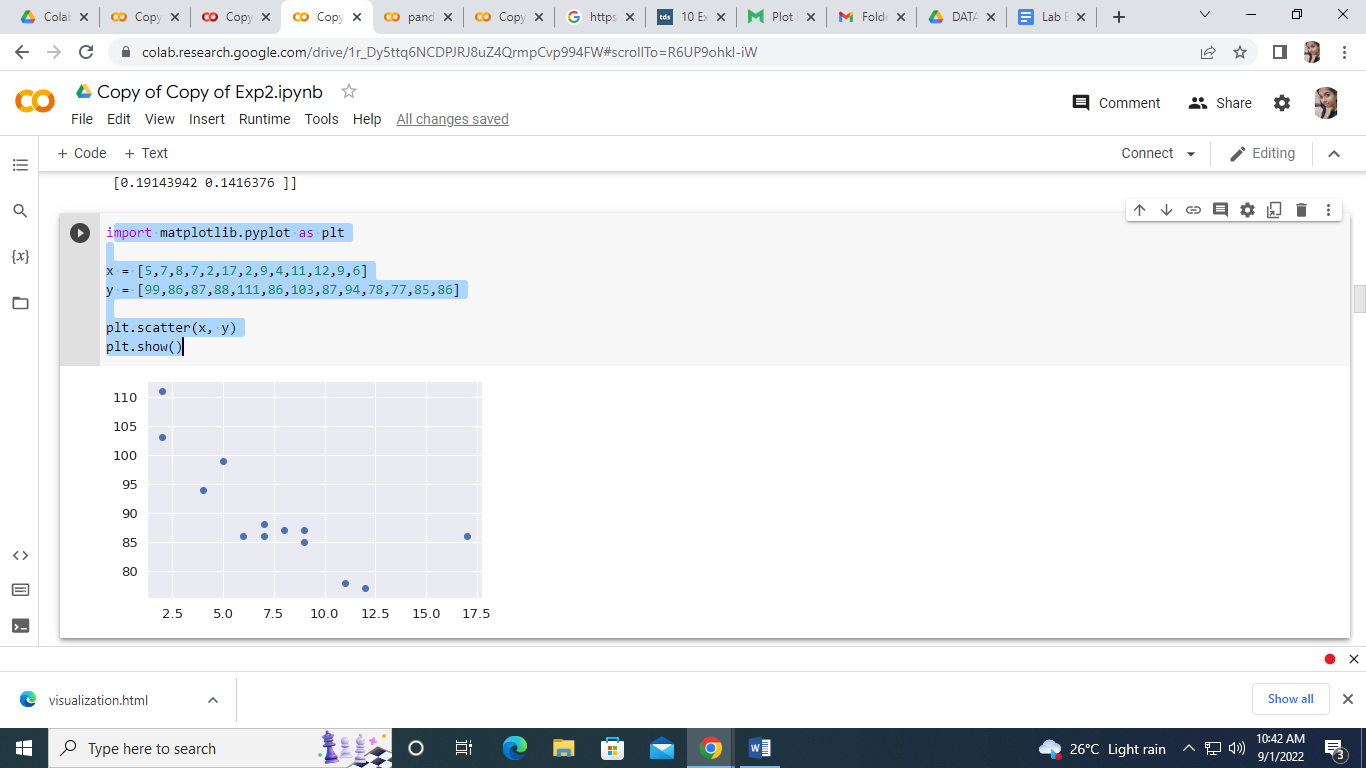
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)

plt.show()



Experiment-3

Name : Sandra P M

Roll no :34

Batch : MCA B

Date :

AIM: Programs to handle data using pandas.

1.import pandas as pd

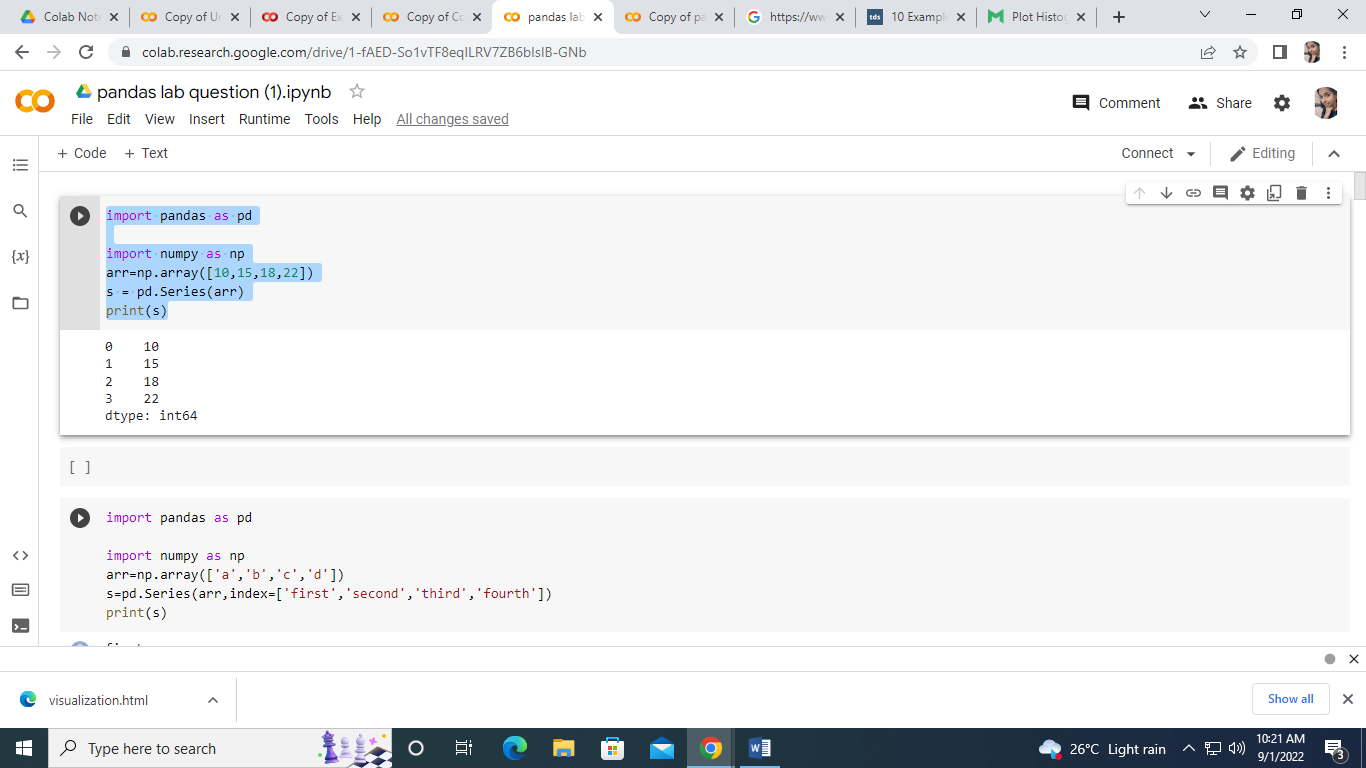
import numpy as np

arr=np.array([10,15,18,22])

s = pd.Series(arr)

print(s)

output



2.import pandas as pd

import numpy as np

arr=np.array(['a','b','c','d'])

s=pd.Series(arr,index=['first','second','third','fourth'])

print(s)

output

first a

second b

third c

fourth d

dtype: object

3) import pandas as pd

s=pd.Series(50,index=[0,1,2,3,4])

print(s)

output

0 50

1 50

2 50

3 50

4 50

dtype: int64

import pandas as pd

d={'Name':'sina','Age':23,'Rollno':29}

s=pd.Series(d)

print(s)

output

Name sina

Age 23

Rollno 29

dtype: object

4.import pandas as pd

s=pd.Series([1,2,3,4,5])

print('To Multiply all values in a series by 2')

print('---------------------------------------------')

print(s\*2)

print('To Find Square of all values in a series')

print('---------------------------------------------')

print(s\*\*2)

print('to print all values that are greater than 2')

print('---------------------------------------------')

print(s[s>2])

output

To Multiply all values in a series by 2

---------------------------------------------

0 2

1 4

2 6

3 8

4 10

dtype: int64

To Find Square of all values in a series

---------------------------------------------

0 1

1 4

2 9

3 16

4 25

dtype: int64

to print all values that are greater than 2

---------------------------------------------

2 3

3 4

4 5

dtype: int64

5.import pandas as pd

s1=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])

s2=pd.Series([10,20,30,40,50],index=['a','b','c','d','e'])

s3=pd.Series([5,14,23,32],index=['a','b','c','d'])

print('To add series1 and series2')

print('---------------------------------------------')

print(s1+s2)

print('To add series2 and series3')

print('---------------------------------------------')

print(s2+s3)

print('To add series2 and series3 and filled non matching index with 0')

print('---------------------------------------------')

print(s2.add(s3,fill\_value=0))

output

To add series1 and series2

---------------------------------------------

a 11

b 22

c 33

d 44

e 55

dtype: int64

To add series2 and series3

---------------------------------------------

a 15.0

b 34.0

c 53.0

d 72.0

e NaN

dtype: float64

To add series2 and series3 and filled non matching index with 0

---------------------------------------------

a 15.0

b 34.0

c 53.0

d 72.0

e 50.0

dtype: float64

[ ]

6 . import pandas as pd  
 import numpy as np  
 arr=np.array([10,15,18,22,55,77,42,48,97])  
 s=pd.Series(arr)  
 print(s.head())  
 print(s.head(3))

0 10

1 15

2 18

3 22

4 55

dtype: int64

0 10

1 15

2 18

dtype: int64

[ ]

7.import pandas as pd  
 import numpy as np  
 arr=np.array([10,15,18,22,55,77,42,48,97])  
 s=pd.Series(arr)  
 print(s.tail())  
 print(s.tail(4))

4 55

5 77

6 42

7 48

8 97

dtype: int64

5 77

6 42

7 48

8 97

dtype: int64

[ ]

8.import pandas as pd  
 import numpy as np  
 arr=np.array([10,15,18,22,55,77,42,48,97])  
 s=pd.Series(arr)  
 print(s.tail())  
 print(s.tail(4))

4 55

5 77

6 42

7 48

8 97

dtype: int64

5 77

6 42

7 48

8 97

dtype: int64

[ ]

9.import pandas as pd  
import numpy as np  
arr=np.array([10,15,18,22,55,77])  
s=pd.Series(arr)  
print(s)  
print(s.loc[:2])  
print(s.loc[3:4])  
s.loc[2:3]

0 10

1 15

2 18

3 22

4 55

5 77

dtype: int64

0 10

1 15

2 18

dtype: int64

3 22

4 55

dtype: int64

2 18

3 22

dtype: int64

[ ]

10.import pandas as pd  
import numpy as np  
arr=np.array([10,15,18,22,55,77])  
s=pd.Series(arr)  
print(s)  
print(s.iloc[:2])  
print(s.iloc[3:4])  
s.iloc[2:3]

0 10

1 15

2 18

3 22

4 55

5 77

dtype: int64

0 10

1 15

dtype: int64

3 22

dtype: int64

2 18

dtype: int64

[ ]

11.import pandas as pd  
import numpy as np  
arr=np.array([10,15,18,22,55,77])  
s=pd.Series(arr)  
print(s)  
print(s[1])  
print('\n')  
print(s[3:4])  
s[:3]

0 10

1 15

2 18

3 22

4 55

5 77

dtype: int64

15

3 22

dtype: int64

0 10

1 15

2 18

dtype: int64

[ ]

12. import pandas as pd  
import numpy as np  
arr=np.array(['a','b','c','d'],)  
s=pd.Series(arr,index=['first','second','third','fourth'])  
print(s)  
print('\n indexes in series are')  
print(s.index)

first a

second b

third c

fourth d

dtype: object

indexes in series are

Index(['first', 'second', 'third', 'fourth'], dtype='object')

[ ]

13. import pandas as pd  
import numpy as  np  
arr=np.array([10,15,18,22,55,77])  
s=pd.Series(arr,index=['A','B','C','D','E','F'])  
print(s)  
print(s[1:5:2])  
print(s[0:6:2])

A 10

B 15

C 18

D 22

E 55

F 77

dtype: int64

B 15

D 22

dtype: int64

A 10

C 18

E 55

dtype: int64

[ ]

14.import pandas as pd  
df=pd.DataFrame()  
print(df)

Empty DataFrame

Columns: []

Index: []

[ ]

15.import pandas as pd  
  
s = pd.Series(['a','b','c','d'])  
df=pd.DataFrame(s)  
print(df)

0

0 a

1 b

2 c

3 d

[ ]

16.import pandas as pd  
name=pd.Series(['Riya','Jacey'])  
team=pd.Series(['Rugby','VolleyBall'])  
dic={'Name':name,'Team':team}  
df=pd.DataFrame(dic)  
print(df)

Name Team

0 Riya Rugby

1 Jacey VolleyBall

[ ]

17.import pandas as pd  
l=[{'Name':'Sachin','Sirname':'Bhardwaj'},  
   {'Name':'Sneha','Sirname':'kunal'},  
   {'Name':'Neha','Sirname':'Sanal'}]  
df1=pd.DataFrame(l)  
print(df1)

Name Sirname

0 Sachin Bhardwaj

1 Sneha kunal

2 Neha Sanal

[ ]

import pandas as pd  
l= [{'Name':'Sachin','SirName':'Banu'},  
    {'Name':'Neha' ,'SirName':'Menon'}]  
df1=pd.dataFrame(l)  
print(df1)  
for(row\_index,row\_value) in df1.iterrows():  
  print('')

[ ]

import pandas as pd  
s = pd.Series([10,15,18,22])  
df=pd.DataFrame(s)  
df.columns=['List1']  
df['List2']=20   
df['List3']=df['List1']+df['List2']  
print(df)  
del df['List3']   
df

[ ]

import pandas as pd  
s = pd.Series([10,15,18,22])  
df=pd.DataFrame(s)  
df.columns=['List1']  
df['List2']=20   
df['List3']=df['List1']+df['List2']  
df.pop('List2')  
df

[ ]

import pandas as pd  
s= pd.Series([10,20,30,40])  
df=pd.DataFrame(s)  
df.columns=['List1']  
df['List2']=40  
df1=df.drop('List2',axis=1)  
df2=df.drop(index=[2,3],axis=0)   
print(df)  
print("After deletion::")  
print(df1)  
print (" After row deletion::")  
print(df2)

List1 List2

0 10 40

1 20 40

2 30 40

3 40 40

After deletion::

List1

0 10

1 20

2 30

3 40

After row deletion::

List1 List2

0 10 40

1 20 40

[ ]

import pandas as pd  
Runs={'TCS':{'Qtr1':2500,'Qtr2':2000,'Qtr3':3000,'Qtr4':2000},  
   'WIPRO':{'Qtr1':2800,'Qtr2':2400,'Qtr3':3600,'Qtr4':2400},  
   'L&T':{'Qtr1':2100,'Qtr2':5700,'Qtr3':3500,'Qtr4':2100}}  
df=pd.DataFrame(Runs)  
print(df)  
print(df.loc['Qtr3',: ])  
print(df.loc['Qtr1' :'Qtr3',:])

TCS WIPRO L&T

Qtr1 2500 2800 2100

Qtr2 2000 2400 5700

Qtr3 3000 3600 3500

Qtr4 2000 2400 2100

TCS 3000

WIPRO 3600

L&T 3500

Name: Qtr3, dtype: int64

TCS WIPRO L&T

Qtr1 2500 2800 2100

Qtr2 2000 2400 5700

Qtr3 3000 3600 3500

[ ]

import pandas as pd  
dic1={'id':['1','2','3','4','5'],'Value1': ['A','C','E','G','I'],  
      'Value2':['B','D','F','H','J']}  
dic2={'id':['2','3','6','7','8'],'Value1': ['K','M','O','Q','S'],  
      'Value2':['L','N','P','R','T']}  
df1=pd.DataFrame(dic1)  
df2=pd.DataFrame(dic2)  
df3=pd.concat([df1,df2])  
print(df3)

id Value1 Value2

0 1 A B

1 2 C D

2 3 E F

3 4 G H

4 5 I J

0 2 K L

1 3 M N

2 6 O P

3 7 Q R

4 8 S T

[ ]

import pandas as pd  
dic1={'id':['1','2','3','4','5'],'Value1': ['A','C','E','G','I'],  
      'Value2':['B','D','F','H','J']}  
dic2={'id':['2','3','6','7','8'],'Value1': ['K','M','O','Q','S'],  
      'Value2':['L','N','P','R','T']}  
dic3={'id':['1','2','3','4','5','7','8','9','10','11'],  
      'Value3':[12,13,14,15,16,17,15,12,13,23]}  
df1=pd.DataFrame(dic1)  
df2=pd.DataFrame(dic2)  
df3=pd.concat([df1,df2])  
df4=pd.DataFrame(dic3)  
df5=pd.merge(df3,df4,on='id')  
print(df5)

id Value1 Value2 Value3

0 1 A B 12

1 2 C D 13

2 2 K L 13

3 3 E F 14

4 3 M N 14

5 4 G H 15

6 5 I J 16

7 7 Q R 17

8 8 S T 15

[ ]

import pandas as pd  
dic1={'id':['1','2','3','4','5'],'Value1': ['A','C','E','G','I'],  
      'Value2':['B','D','F','H','J']}  
dic2={'id':['2','3','6','7','8'],'Value1': ['K','M','O','Q','S'],  
      'Value2':['L','N','P','R','T']}  
df1=pd.DataFrame(dic1)  
df2=pd.DataFrame(dic2)  
df3=pd.merge(df1,df2,on='id',how='outer')  
print(df3)

id Value1\_x Value2\_x Value1\_y Value2\_y

0 1 A B NaN NaN

1 2 C D K L

2 3 E F M N

3 4 G H NaN NaN

4 5 I J NaN NaN

5 6 NaN NaN O P

6 7 NaN NaN Q R

7 8 NaN NaN S T

[ ]

import pandas as pd

dic1={'student\_id':['S1','S2','S3','S4','S5'],'Name': ['Daniella','Rider','Bryce','Ed','Kwame'],

      'Marks':[200,210,190,150,194]}

dic2={'student\_id':['S4','S5','S6','S7','S8'],'Name': ['Scarlett','Maya','Sana','Quilton','Sovyal'],

      'Value2':[190,100,200,290,160]}

df1=pd.DataFrame(dic1)

df2=pd.DataFrame(dic2)

df1.to\_csv('E:\DataFrame1.csv')

df2.to\_csv('E:\DataFrame2.csv')

df3=pd.merge(df1,df2)

df3.to\_csv('E:\DataFrame3.csv')





Name : Sandra P M

Roll no :34

Batch : MCA B

Date :15/09/22

15/09/22

Experiment-4

1. from sklearn.datasets import load\_iris

from sklearn.preprocessing import MinMaxScaler

import numpy as np

X, y = load\_iris(return\_X\_y=True)

print(X.shape)

scaler = MinMaxScaler()

scaler.fit(X)

X\_scaled = scaler.transform(X)

output

(150, 4)

2.

import matplotlib.pyplot as plt

fig, axes = plt.subplots(1,2)

axes[0].scatter(X[:,0], X[:,1], c=y)

axes[0].set\_title("Original data")

axes[1].scatter(X\_scaled[:,0], X\_scaled[:,1], c=y)

axes[1].set\_title("MinMax scaled data")

plt.show()

3. 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=42, stratify = y)

4. from sklearn.neighbors import KNeighborsClassifier

model=KNeighborsClassifier(n\_neighbors=5,metric='minkowski')

model.fit(X\_train, y\_train)

y\_pred=model.predict(X\_test)

model.score(X\_test, y\_test)

5. # Example of making predictions

from math import sqrt

# calculate the Euclidean distance between two vectors

def euclidean\_distance(row1, row2):

   distance = 0.0

   for i in range(len(row1)-1):

     distance += (row1[i] - row2[i])\*\*2

   return sqrt(distance)

# Locate the most similar neighbors

def get\_neighbors(train, test\_row, num\_neighbors):

   distances = list()

   for train\_row in train:

   dist = euclidean\_distance(test\_row, train\_row)

   distances.append((train\_row, dist))

   distances.sort(key=lambda tup: tup[1])

  neighbors = list()

  for i in range(num\_neighbors):

    neighbors.append(distances[i][0])

  return neighbors

# Make a classification prediction with neighbors

def predict\_classification(train, test\_row, num\_neighbors):

  neighbors = get\_neighbors(train, test\_row, num\_neighbors)

  output\_values = [row[-1] for row in neighbors]

  prediction = max(set(output\_values), key=output\_values.count)

  return prediction

# Test distance function

dataset = [[2.7810836,2.550537003,0],

  [1.465489372,2.362125076,0],

  [3.396561688,4.400293529,0],

  [1.38807019,1.850220317,0],

  [3.06407232,3.005305973,0],

  [7.627531214,2.759262235,1],

  [5.332441248,2.088626775,1],

  [6.922596716,1.77106367,1],

  [8.675418651,-0.242068655,1],

  [7.673756466,3.508563011,1]]

prediction = predict\_classification(dataset, dataset[0], 3)

print('Expected %d, Got %d.' % (dataset[0][-1], prediction))

Aim : Implement K-NN Algorithm using iris data set

from math import sqrt

def euclidean\_distance(row1, row2):

  distance = 0.0

  for i in range(len(row1)-1):

    distance += (row1[i] - row2[i])\*\*2

  return sqrt(distance)

def predict\_classification(train, test\_row, num\_neighbors):

  neighbors = get\_neighbors(train, test\_row, num\_neighbors)

  output\_values = [row[-1] for row in neighbors]

  prediction = max(set(output\_values), key=output\_values.count)

  return prediction

samples = [[0., 0., 0.], [0., .5, 0.], [1., 1., .5]]

from sklearn.neighbors import NearestNeighbors

neigh = NearestNeighbors(n\_neighbors=1)

neigh.fit(samples)

NearestNeighbors(n\_neighbors=1)

print(neigh.kneighbors([[1., 1., 1.]]))

X = [[0], [1], [2], [3]]

y = [0, 0, 1, 1]

from sklearn.neighbors import KNeighborsClassifier

neigh = KNeighborsClassifier(n\_neighbors=3)

neigh.fit(X, y)

KNeighborsClassifier(...)

print(neigh.predict([[1.1]]))

[0]

print(neigh.predict\_proba([[0.9]]))

o/p

[0]

[[0.66666667 0.33333333]]

def get\_neighbors(train, test\_row, num\_neighbors):

  distances = list()

  for train\_row in train:

    dist = euclidean\_distance(test\_row, train\_row)

    distances.append((train\_row, dist))

  distances.sort(key=lambda tup: tup[1])

  neighbors = list()

  for i in range(num\_neighbors):

    neighbors.append(distances[i][0])

  return neighbors

def predict\_classification(train, test\_row, num\_neighbors):

  neighbors = get\_neighbors(train, test\_row, num\_neighbors)

  output\_values = [row[-1] for row in neighbors]

  prediction = max(set(output\_values), key=output\_values.count)

  return prediction

dataset = [[2.7810836,2.550537003,0],

  [1.465489372,2.362125076,0],

  [3.396561688,4.400293529,0],

  [1.38807019,1.850220317,0],

  [3.06407232,3.005305973,0],

  [7.627531214,2.759262235,1],

  [5.332441248,2.088626775,1],

  [6.922596716,1.77106367,1],

  [8.675418651,-0.242068655,1],

  [7.673756466,3.508563011,1]]

prediction = predict\_classification(dataset, dataset[0], 3)

print('Expected %d, Got %d.' % (dataset[0][-1], prediction))

Experiment-5

Name : Sandra P M

Roll no :34

Batch : MCA B

Date :

Program using Iris dataset

import pandas as pd

data = pd.read\_csv ("/content/iris\_csv.csv")

data.head()

data.sample(10)

data.columns

data.shape

print(data)

print(data[10:21])

sliced\_data=data[10:21]

print(sliced\_data)

output

sepallength sepalwidth petallength petalwidth class

0 5.1 3.5 1.4 0.2 Iris-setosa

1 4.9 3.0 1.4 0.2 Iris-setosa

2 4.7 3.2 1.3 0.2 Iris-setosa

3 4.6 3.1 1.5 0.2 Iris-setosa

4 5.0 3.6 1.4 0.2 Iris-setosa

.. ... ... ... ... ...

145 6.7 3.0 5.2 2.3 Iris-virginica

146 6.3 2.5 5.0 1.9 Iris-virginica

147 6.5 3.0 5.2 2.0 Iris-virginica

148 6.2 3.4 5.4 2.3 Iris-virginica

149 5.9 3.0 5.1 1.8 Iris-virginica

[150 rows x 5 columns]

sepallength sepalwidth petallength petalwidth class

10 5.4 3.7 1.5 0.2 Iris-setosa

11 4.8 3.4 1.6 0.2 Iris-setosa

12 4.8 3.0 1.4 0.1 Iris-setosa

13 4.3 3.0 1.1 0.1 Iris-setosa

14 5.8 4.0 1.2 0.2 Iris-setosa

15 5.7 4.4 1.5 0.4 Iris-setosa

16 5.4 3.9 1.3 0.4 Iris-setosa

17 5.1 3.5 1.4 0.3 Iris-setosa

18 5.7 3.8 1.7 0.3 Iris-setosa

19 5.1 3.8 1.5 0.3 Iris-setosa

20 5.4 3.4 1.7 0.2 Iris-setosa

sepallength sepalwidth petallength petalwidth class

10 5.4 3.7 1.5 0.2 Iris-setosa

11 4.8 3.4 1.6 0.2 Iris-setosa

12 4.8 3.0 1.4 0.1 Iris-setosa

13 4.3 3.0 1.1 0.1 Iris-setosa

14 5.8 4.0 1.2 0.2 Iris-setosa

15 5.7 4.4 1.5 0.4 Iris-setosa

16 5.4 3.9 1.3 0.4 Iris-setosa

17 5.1 3.5 1.4 0.3 Iris-setosa

18 5.7 3.8 1.7 0.3 Iris-setosa

19 5.1 3.8 1.5 0.3 Iris-setosa

20 5.4 3.4 1.7 0.2 Iris-setosa

Experiment-6

Name : Sandra P M

Roll no :34

Batch : MCA B

Date :

Aim :implementation of NavieBaye’s theorm

NavieBaye’s

1.# load the iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

2.# store the feature matrix (X) and response vector (y)

X = iris.data

y = iris.target

3.# splitting 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=0.2, random\_state=42)

4.# training the model on training set

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X\_train, y\_train)

5.# making predictions on the testing set

y\_pred = gnb.predict(X\_test)

6.# comparing actual response values (y\_test) with predicted response values (y\_pred)

from sklearn import metrics

print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy\_score(y\_test, y\_pred)\*100)

output

Gaussian Naive Bayes model accuracy(in %): 100.0

GINI INDEX

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

import pandas as pd

import numpy as np

from sklearn import tree

from sklearn.datasets import load\_iris

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(df[data.feature\_names], df['target'], random\_state=0)

# Step 1: Import the model you want to use

# This was already imported earlier in the notebook so commenting out

# from sklearn.tree import DecisionTreeClassifier

# Step 2: Make an instance of the Model

clf = DecisionTreeClassifier(max\_depth=2,

                             random\_state=0)

# Step 3: Train the model on the data

clf.fit(X\_train, Y\_train)

# Step 4: Predict labels of unseen (test) data

# Not doing this step in the tutorial

# clf.predict(X\_test)

# tree.plot\_tree(clf);

fn = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

cn = ['setosa', 'versicolor', 'virginica']

# fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(4, 4), dpi=300)

tree.plot\_tree(clf,

               feature\_names=fn,

               class\_names=cn,

               filled=True

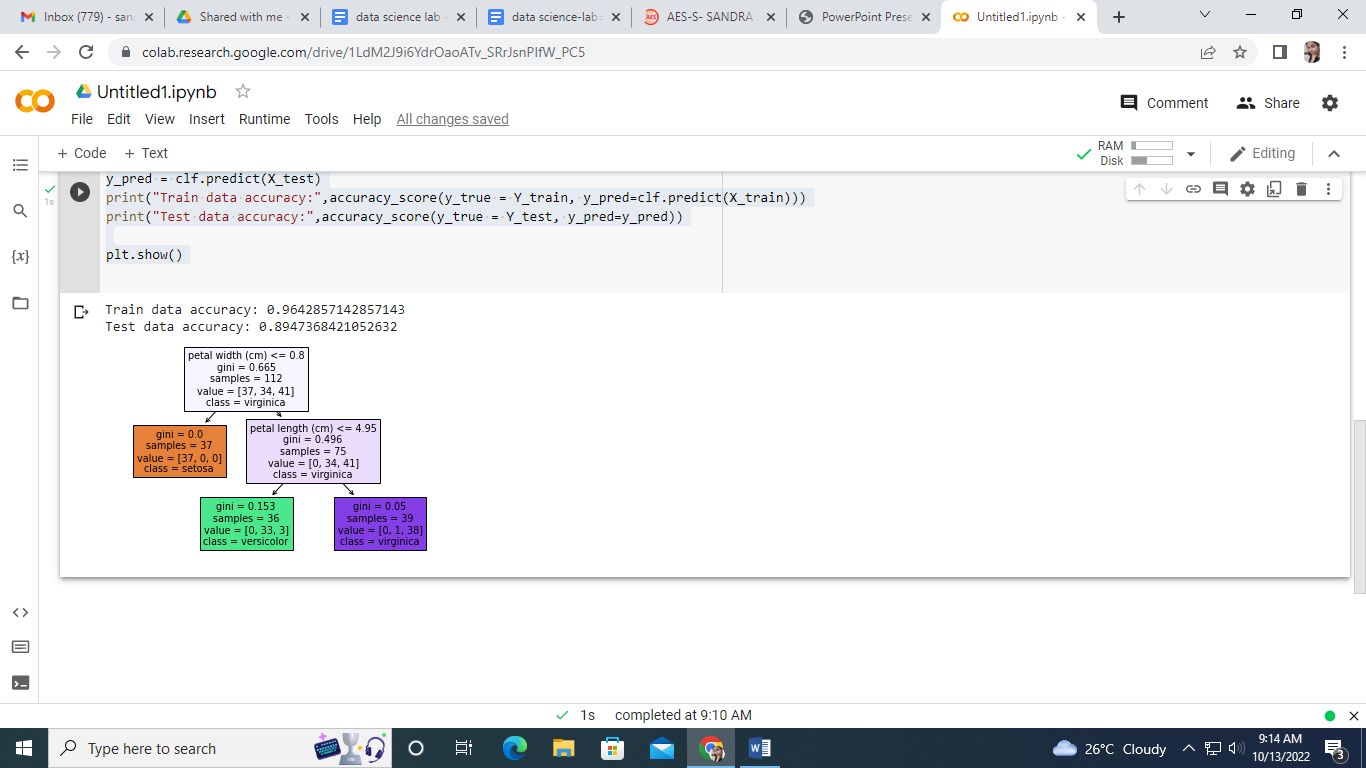
               )

y\_pred = clf.predict(X\_test)

print("Train data accuracy:",accuracy\_score(y\_true = Y\_train, y\_pred=clf.predict(X\_train)))

print("Test data accuracy:",accuracy\_score(y\_true = Y\_test, y\_pred=y\_pred))

plt.show()



Name: Sandra PM

RMCA

Roll no:34

Multiple linear regression

1. import numpy as np

from sklearn.linear\_model import LinearRegression

x = [[0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55, 34], [60, 35]]

y = [4, 5, 20, 14, 32, 22, 38, 43]

x, y = np.array(x), np.array(y)

2. print(x)

print(y)

output

[[ 0 1]

[ 5 1]

[15 2]

[25 5]

[35 11]

[45 15]

[55 34]

[60 35]]

[ 4 5 20 14 32 22 38 43]

3. model = LinearRegression().fit(x, y)

4. r\_sq = model.score(x, y)

print(f"coefficient of determination: {r\_sq}")

print(f"intercept: {model.intercept\_}")

print(f"coefficients: {model.coef\_}")

output

coefficient of determination: 0.8615939258756775

intercept: 5.52257927519819

coefficients: [0.44706965 0.25502548]

5. y\_pred = model.predict(x)

print(f"predicted response:\n{y\_pred}")

output

predicted response:

[ 5.77760476 8.012953 12.73867497 17.9744479 23.97529728 29.4660957

38.78227633 41.27265006]

6. x\_new = np.arange(10).reshape((-1, 2))

print(x\_new)

y\_new = model.predict(x\_new)

y\_new

output

[[0 1]

[2 3]

[4 5]

[6 7]

[8 9]]

array([ 5.77760476, 7.18179502, 8.58598528, 9.99017554, 11.3943658 ])

Name: Sandra PM

RMCA

Roll no:34

KMeans

from sklearn .cluster import KMeans

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

%matplotlib inline

df = pd.read\_csv("/income.csv")

df.head

Output

<bound method NDFrame.head of Name Age Income($) 0 Rob 27 70000 1 Michael 29 90000 2 Mohan 29 61000 3 Ismail 28 60000 4 Kory 42 150000 5 Gautam 39 155000 6 David 41 160000 7 Andrea 38 162000 8 Brad 36 156000 9 Angelina 35 130000 10 Donald 37 137000 11 Tom 26 45000 12 Arnold 27 48000 13 Jared 28 51000 14 Stark 29 49500 15 Ranbir 32 53000 16 Dipika 40 65000 17 Priyanka 41 63000 18 Nick 43 64000 19 Alia 39 80000 20 Sid 41 82000 21 Abdul 39 58000>

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])

df['Income($)'] = scaler.transform(df[['Income($)']])

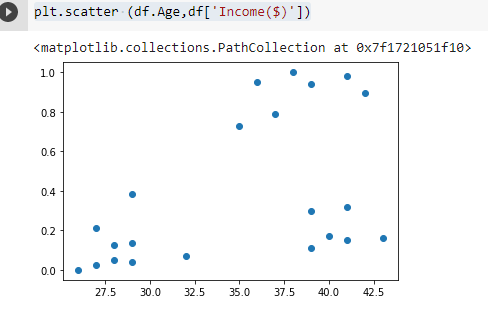
scaler.fit(df[['Age']])

df[' Age ']=scaler.transform(df[['Age']])

plt.scatter (df.Age,df['Income($)'])

<matplotlib.collections.PathCollection at 0x7f1721051f10>

Output



km=KMeans(n\_clusters=3)

y\_predicted = km.fit\_predict(df[['Age','Income($)']])

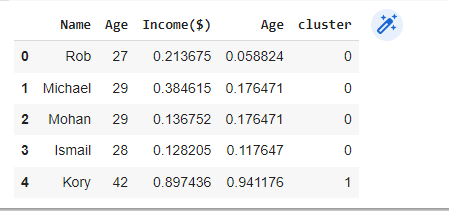
y\_predicted

output

array([0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1], dtype=int32)

df['cluster']=y\_predicted

df.head()



km.cluster\_centers\_

output

array([[28.33333333, 0.11633428], [40.55555556, 0.44824311], [36.5 , 0.86538462]])

df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

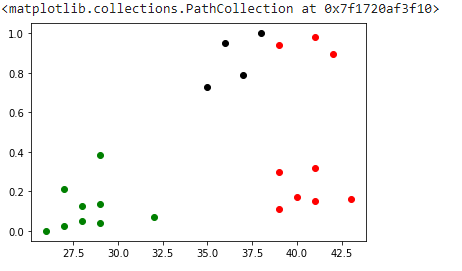
df3 = df[df.cluster==2]

plt.scatter(df1.Age,df1['Income($)'],color='green')

plt.scatter(df2.Age,df2['Income($)'],color='red')

plt.scatter(df3.Age,df3['Income($)'],color='black')

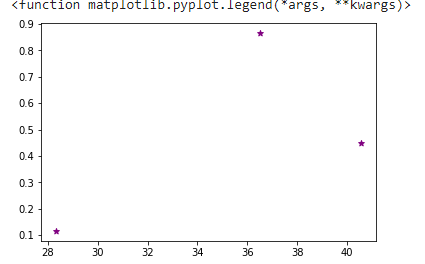
output



plt.scatter(km.cluster\_centers\_[:,0],km.cluster\_centers\_[:,1],color='purple',marker='\*',label='centroid')

plt.legend

output



Ngrams

Name: Sandra PM

RMCA

Roll no:34

import nltk

from nltk.util import ngrams

text = "this is a very good book to study";

Ngrams = ngrams(sequence=nltk.wordpunct\_tokenize(text), n=3)

for grams in Ngrams:

 print(grams)

output

('this', 'is', 'a')

('is', 'a', 'very')

('a', 'very', 'good')

('very', 'good', 'book')

('good', 'book', 'to')

('book', 'to', 'study')

import requests

from bs4 import BeautifulSoup

import csv

URL = "http://www.values.com/inspirational-quotes"

r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')

print(soup.prettify())

Output



import nltk

from nltk.tag import DefaultTagger

exptagger=DefaultTagger('NN')

exptagger.tag\_sents([['hi',','],['how','are','you','?']])

output

[[('hi', 'NN'), (',', 'NN')], [('how', 'NN'), ('are', 'NN'), ('you', 'NN'), ('?', 'NN')]]

import nltk

from nltk.tag import untag

untag([('Tutorials','NN'),('point','NN')])

output

['Tutorials', 'point']

import nltk

nltk.download("book")

output

[nltk\_data] Downloading collection 'book'

[nltk\_data] |

[nltk\_data] | Downloading package abc to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/abc.zip.

[nltk\_data] | Downloading package brown to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/brown.zip.

[nltk\_data] | Downloading package chat80 to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/chat80.zip.

[nltk\_data] | Downloading package cmudict to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/cmudict.zip.

[nltk\_data] | Downloading package conll2000 to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/conll2000.zip.

[nltk\_data] | Downloading package conll2002 to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/conll2002.zip.

[nltk\_data] | Downloading package dependency\_treebank to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping corpora/dependency\_treebank.zip.

[nltk\_data] | Downloading package genesis to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/genesis.zip.

[nltk\_data] | Downloading package gutenberg to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/gutenberg.zip.

[nltk\_data] | Downloading package ieer to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/ieer.zip.

[nltk\_data] | Downloading package inaugural to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/inaugural.zip.

[nltk\_data] | Downloading package movie\_reviews to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping corpora/movie\_reviews.zip.

[nltk\_data] | Downloading package nps\_chat to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/nps\_chat.zip.

[nltk\_data] | Downloading package names to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/names.zip.

[nltk\_data] | Downloading package ppattach to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/ppattach.zip.

[nltk\_data] | Downloading package reuters to /root/nltk\_data...

[nltk\_data] | Downloading package senseval to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/senseval.zip.

[nltk\_data] | Downloading package state\_union to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/state\_union.zip.

[nltk\_data] | Downloading package stopwords to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/stopwords.zip.

[nltk\_data] | Downloading package swadesh to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/swadesh.zip.

[nltk\_data] | Downloading package timit to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/timit.zip.

[nltk\_data] | Downloading package treebank to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/treebank.zip.

[nltk\_data] | Downloading package toolbox to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/toolbox.zip.

[nltk\_data] | Downloading package udhr to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/udhr.zip.

[nltk\_data] | Downloading package udhr2 to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/udhr2.zip.

[nltk\_data] | Downloading package unicode\_samples to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping corpora/unicode\_samples.zip.

[nltk\_data] | Downloading package webtext to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/webtext.zip.

[nltk\_data] | Downloading package wordnet to /root/nltk\_data...

[nltk\_data] | Downloading package wordnet\_ic to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/wordnet\_ic.zip.

[nltk\_data] | Downloading package words to /root/nltk\_data...

[nltk\_data] | Unzipping corpora/words.zip.

[nltk\_data] | Downloading package maxent\_treebank\_pos\_tagger to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping taggers/maxent\_treebank\_pos\_tagger.zip.

[nltk\_data] | Downloading package maxent\_ne\_chunker to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping chunkers/maxent\_ne\_chunker.zip.

[nltk\_data] | Downloading package universal\_tagset to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping taggers/universal\_tagset.zip.

[nltk\_data] | Downloading package punkt to /root/nltk\_data...

[nltk\_data] | Unzipping tokenizers/punkt.zip.

[nltk\_data] | Downloading package book\_grammars to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping grammars/book\_grammars.zip.

[nltk\_data] | Downloading package city\_database to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping corpora/city\_database.zip.

[nltk\_data] | Downloading package tagsets to /root/nltk\_data...

[nltk\_data] | Unzipping help/tagsets.zip.

[nltk\_data] | Downloading package panlex\_swadesh to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Downloading package averaged\_perceptron\_tagger to

[nltk\_data] | /root/nltk\_data...

[nltk\_data] | Unzipping taggers/averaged\_perceptron\_tagger.zip.

[nltk\_data] |

[nltk\_data] Done downloading collection book

True

sentence="""At eight o'clock on thursday morning Arthur didn't feel very good"""

tokens=nltk.word\_tokenize(sentence)

print(tokens)

tagged=nltk.pos\_tag(tokens)

print(tagged)

output

['At', 'eight', "o'clock", 'on', 'thursday', 'morning', 'Arthur', 'did', "n't", 'feel', 'very', 'good']

[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'NN'), ('on', 'IN'), ('thursday', 'JJ'), ('morning', 'NN'), ('Arthur', 'NNP'), ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'), ('very', 'RB'), ('good', 'JJ')]

text="learn php from guru99 and make study easy".split()

print("after split:",text)

token\_tag=nltk.pos\_tag(text)

print("after token:",token\_tag)

output

after split: ['learn', 'php', 'from', 'guru99', 'and', 'make', 'study', 'easy']

after token: [('learn', 'JJ'), ('php', 'NN'), ('from', 'IN'), ('guru99', 'NN'), ('and', 'CC'), ('make', 'VB'), ('study', 'NN'), ('easy', 'JJ')] J')]