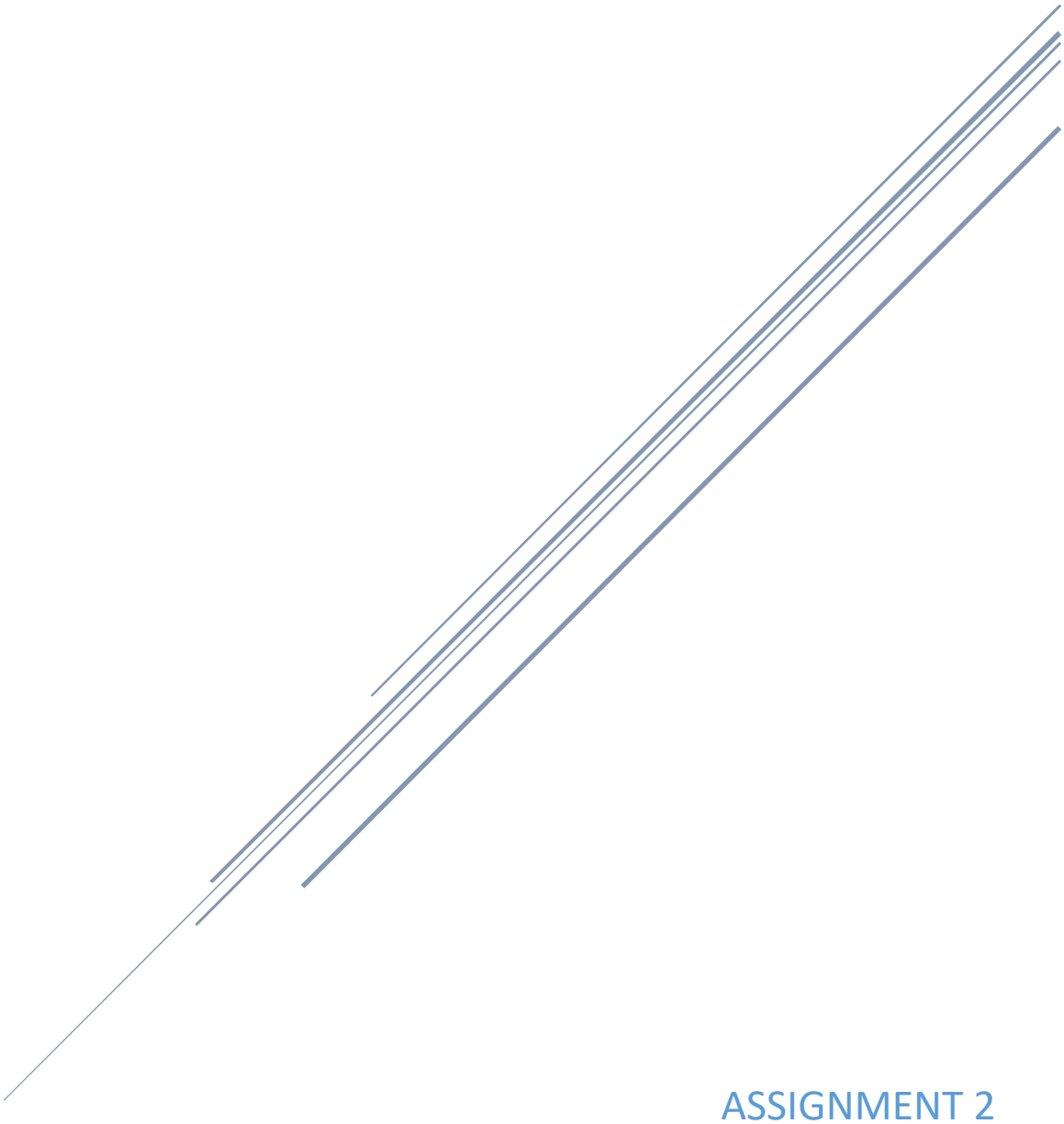
RUMAISA MARAYM

MACHINE LEARNING



ASSIGNMENT 2

29-03-2020

PERFORMING LINEAR AND POLYNOMIAL REGRESSION ON PROVIDED DATASET

LINEAR REGRESSION:

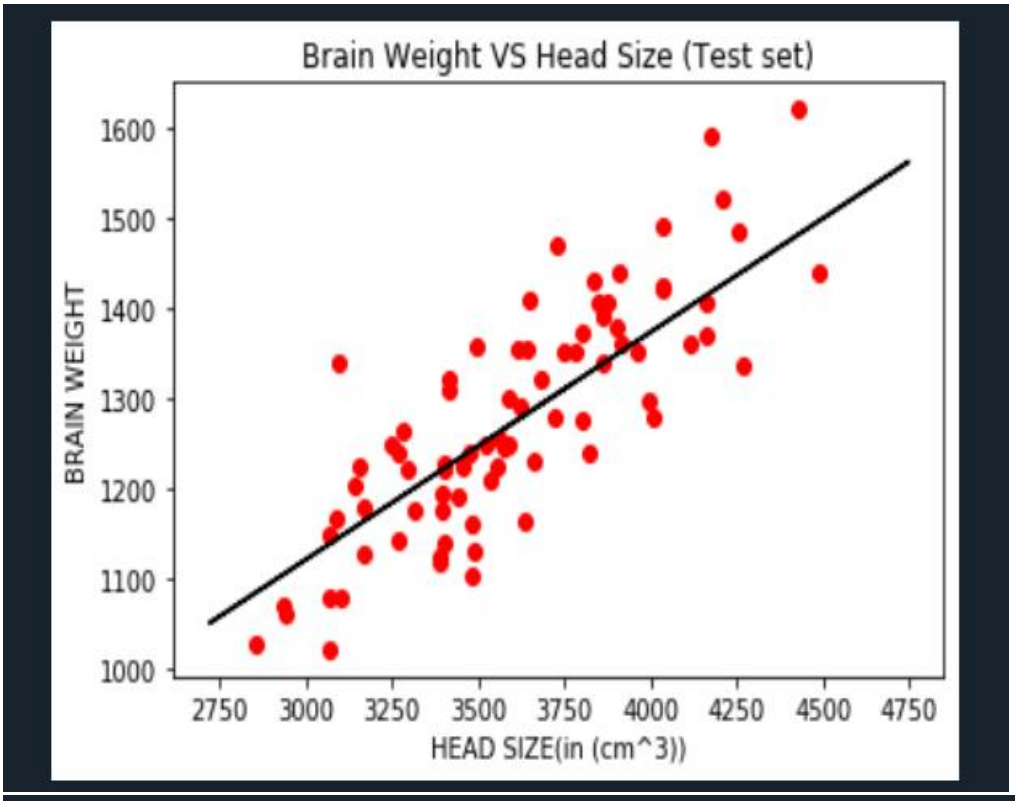
temp.py | polynomial_regression_headweightset.py | headweightset.csv | simple_linear_regression_headweightset.py

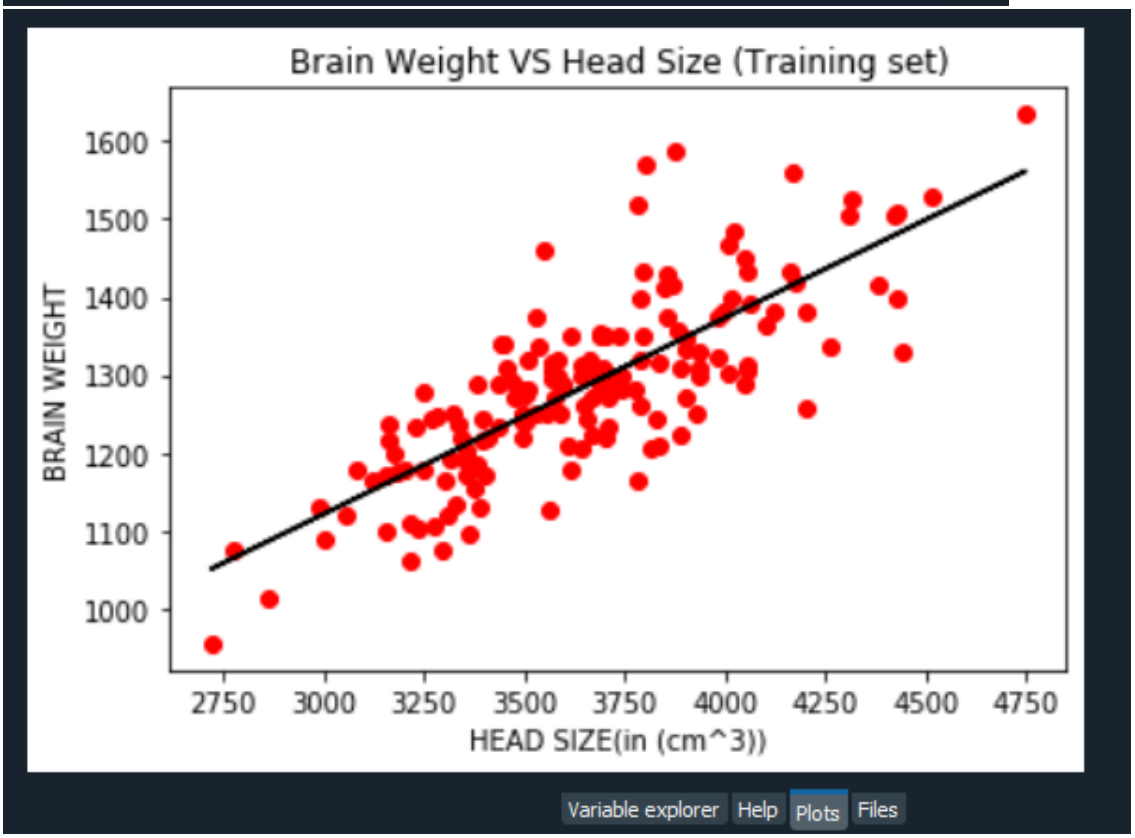
```python
#MACHINE LEARNING ASSIGNMENT2
#RUMAISA MARYAM
# Simple Linear Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the headweightset dataset
dataset = pd.read_csv('headweightset.csv')
A = dataset.iloc[:,2:3].values
B = dataset.iloc[:, 3:4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 1/3, random_state = 0)

# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressing = LinearRegression()
regressing.fit(A_train, B_train)

# Predicting the Test set results
B_pred = regressing.predict(A_test)

# Visualising the Training set results
plt.scatter(A_train, B_train, color = 'red')
plt.plot(A_train, regressing.predict(A_train), color = 'black')
plt.title('Brain Weight VS Head Size (Training set)')
plt.xlabel('HEAD SIZE(in (cm^3))')
plt.ylabel('BRAIN WEIGHT')
plt.show()

# Visualising the Test set results
plt.scatter(A_test, B_test, color = 'red')
plt.plot(A_train, regressing.predict(A_train), color = 'black')
```

conda: base (Python 3

temp.py | polynomial_regression_headweightset.py | headweightset.csv | simple_linear_regression_headweightset.py

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

# Importing the headweightset dataset
dataset = pd.read_csv('headweightset.csv')
A = dataset.iloc[:,2:3].values
B = dataset.iloc[:, 3:4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 1/3, random_state = 0)

# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressing = LinearRegression()
regressing.fit(A_train, B_train)

# Predicting the Test set results
B_pred = regressing.predict(A_test)

# Visualising the Training set results
plt.scatter(A_train, B_train, color = 'red')
plt.plot(A_train, regressing.predict(A_train), color = 'black')
plt.title('Brain Weight VS Head Size (Training set)')
plt.xlabel('HEAD SIZE(in (cm^3))')
plt.ylabel('BRAIN WEIGHT')
plt.show()

# Visualising the Test set results
plt.scatter(A_test, B_test, color = 'red')
plt.plot(A_train, regressing.predict(A_train), color = 'black')
plt.title('Brain Weight VS Head Size (Test set)')
plt.xlabel('HEAD SIZE(in (cm^3))')
plt.ylabel('BRAIN WEIGHT')
plt.show()
```


Brain Weight VS Head Size (Test set)
BRAIN WEIGHT
HEAD SIZE(in (cm^3))
1000
1100
1200
1300
1400
1500
1600
2750
3000
3250
3500
3750
4000
4250
4500
4750


Brain Weight VS Head Size (Training set)
BRAIN WEIGHT
HEAD SIZE(in (cm^3))
1000
1100
1200
1300
1400
1500
1600
2750
3000
3250
3500
3750
4000
4250
4500
4750
Variable explorer
Help
Plots
Files

Name	Type	Size	Value
A	Array of int64	(237, 1)	[[4512] [3738]]
A_grid	Array of float64	(20270, 1)	[[2720.] [2720.1]
A_poly	Array of float64	(237, 5)	[[1.00000000e+00 4.51200000e+03 2.03581440 4.1445 ...
A_test	Array of int64	(48, 1)	[[3724] [3680]]
A_train	Array of int64	(189, 1)	[[3104] [3069]]
B	Array of int64	(237,)	[1530 1297 1335 ... 1104 1170 1120]
B_pred	Array of float64	(79, 1)	[[1303.83322923] [1292.73537163]]
B_test	Array of int64	(48,)	[1280 1321 1425 ... 1255 1140 1202]
B_train	Array of int64	(189,)	[1080 1022 1220 ... 1270 1215 1316]

POLYNOMIAL REGRESSION:

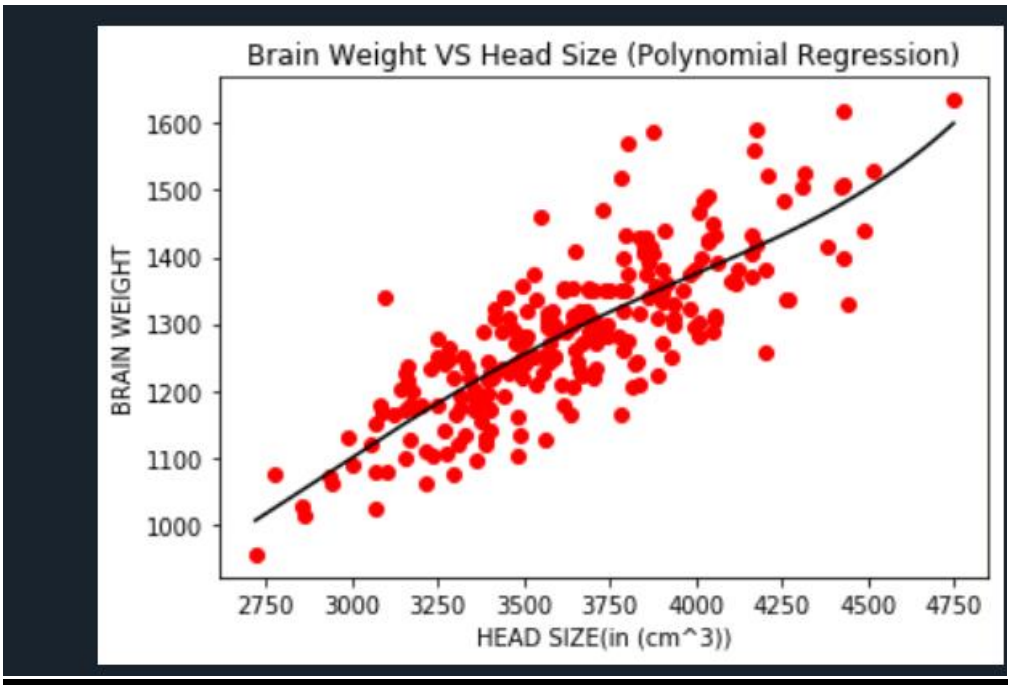
temp.py | polynomial_regression_headweightset.py | headweightset.csv | simple_linear_regression_headweightset.py

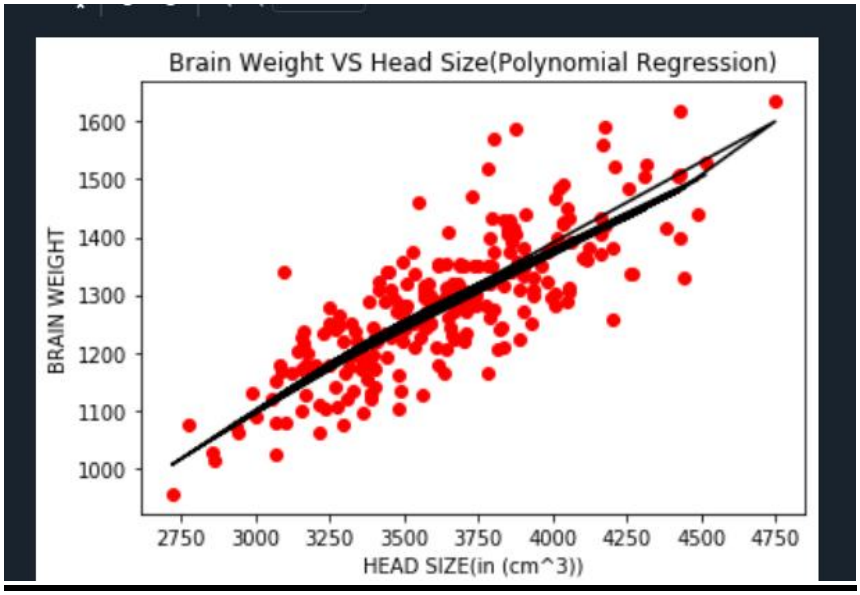
```python
#MACHINE LEARNING ASSIGNMENT2
#RUMAISA MARYAM
# Polynomial Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('headweightset.csv')
A = dataset.iloc[:, 2:-1].values
B = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 0.2, random_state = 0)


# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_regressor = LinearRegression()
lin_regressor.fit(A, B)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
A_poly = poly_reg.fit_transform(A)
poly_reg.fit(A_poly, B)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(A_poly, B)
```

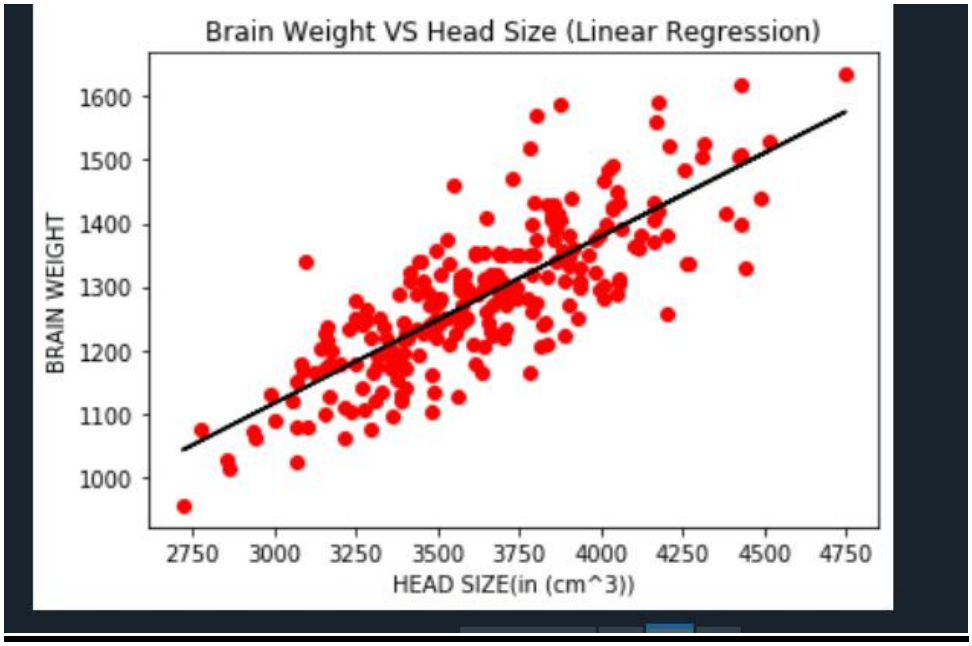
temp.py | polynomial_regression_headweightset.py | headweightset.csv | simple_linear_regression_headweightset.py

```python
# Visualising the Linear Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_regressor.predict(A), color = 'Black')
plt.title('Brain Weight VS Head Size (Linear Regression)')
plt.xlabel('HEAD SIZE(in (cm^3))')
plt.ylabel('BRAIN WEIGHT')
plt.show()

# Visualising the Polynomial Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_reg_2.predict(poly_reg.fit_transform(A)), color = 'Black')
plt.title('Brain Weight VS Head Size(Polynomial Regression)')
plt.xlabel('HEAD SIZE(in (cm^3))')
plt.ylabel('BRAIN WEIGHT')
plt.show()

# Visualising the Polynomial Regression results (for higher resolution and smoother curve)
A_grid = np.arange(min(A), max(A), 0.1)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
plt.plot(A_grid, lin_reg_2.predict(poly_reg.fit_transform(A_grid)), color = 'Black')
plt.title('Brain Weight VS Head Size (Polynomial Regression)')
plt.xlabel('HEAD SIZE(in (cm^3))')
plt.ylabel('BRAIN WEIGHT')
plt.show()

# Predicting a new result with Linear Regression
lin_regressor.predict([[6]])

# Predicting a new result with Polynomial Regression
lin_reg_2.predict(poly_reg.fit_transform([[6]]))
```

Name	Type	Size	Value
A	Array of int64	(237, 1)	[[4512] [3738]
A_grid	Array of float64	(20270, 1)	[[2720.] [2720.1]
A_poly	Array of float64	(237, 5)	[[1.00000000e+00 4.51200000e+03 2.03581440e+07… 4.1445 ...
A_test	Array of int64	(79, 1)	[[3724] [3680]
A_train	Array of int64	(158, 1)	[[3777] [3302]
B	Array of int64	(237, 1)	[[1530] [1297]
B_pred	Array of float64	(79, 1)	[[1303.83322923] [1292.73537163]
B_test	Array of int64	(79, 1)	[[1280] [1321]
B_train	Array of int64	(158, 1)	[[1282] [1165]




Brain Weight VS Head Size(Polynomial Regression)
BRAIN WEIGHT
1600
1500
1400
1300
1200
1100
1000
2750
3000
3250
3500
3750
4000
4250
4500
4750
HEAD SIZE(in (cm^3))


Brain Weight VS Head Size (Linear Regression)
BRAIN WEIGHT
1600
1500
1400
1300
1200
1100
1000
2750
3000
3250
3500
3750
4000
4250
4500
4750
HEAD SIZE(in (cm^3))

CLASS TASK ASSIGNMENT:

CLASS TASK:

Create two random arrays A and B, and multiply them. Get their result in C and add 1 to every element of C.

In [9]:
```
import numpy
A=numpy.random.randn(3,3)
print ("A=",A)
B= numpy.random.randn(3,3)
print("B=",B)
C= numpy.multiply(A,B)
print("C=",C)
D=numpy.ones((3,3))
print("D=",D)
R=numpy.add(C,D)
print ("RESULT=",R)
```

```
A= [[-0.91269733  0.00257767  0.27485019]
 [-1.61436977 -2.24815322 -0.76136557]
 [ 0.46606914  1.31785327 -0.8216844 ]]
B= [[-0.07989783  0.44257981  0.40138431]
 [-0.8850222   0.77147887 -0.04590463]
 [-0.94419722 -2.19816394 -0.64370823]]
C= [[ 7.29225359e-02  1.14082544e-03  1.10320555e-01]
 [ 1.42875308e+00 -1.73440271e+00  3.49502014e-02]
 [-4.40061187e-01 -2.89685753e+00  5.28925007e-01]]
D= [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
RESULT= [[ 1.07292254  1.00114083  1.11032056]
 [ 2.42875308 -0.73440271  1.0349502 ]
 [ 0.55993881 -1.89685753  1.52892501]]
```