1 a. Implement Multilinear Regression on Data1.csv. Display the coefficients. (that is B1, B2, B3)

b. Predict Y values for

Y X1 X2 X3

? 50 70 80

? 30 40 50

```
# Import necessary libraries
import numpy as np
import pandas as pd

# Read data from CSV file into a DataFrame
df = pd.read_csv('C:/Users/raosu/Documents/Assignment 6
aiml/Data1.csv')

# Display the first few rows of the DataFrame
df.head()
```

	Y	X1	X2	X3
0	43	30	63	33
1	63	45	47	52
2	71	68	67	62
3	61	46	83	42
4	81	66	84	42

```
# Create a new DataFrame 'x' by dropping the 'Y' column from the
original DataFrame 'df'
x = df.drop('Y', axis=1)

# Display the modified DataFrame 'x'
print(x)

# Insert a new column 'x0' at the beginning of the DataFrame with
```

```
constant values [1, 1, 1, 1, 1, 1]
x.insert(0, 'x0', [1, 1, 1, 1, 1, 1])

# Convert the DataFrame 'x' to a NumPy array
x = x.values

# Display the resulting NumPy array
x
```

```
   X1  X2  X3
0  30  63  33
1  45  47  52
2  68  67  62
3  46  83  42
4  66  84  42
5  36  50  66
```

```
array([[ 1, 30, 63, 33],
       [ 1, 45, 47, 52],
       [ 1, 68, 67, 62],
       [ 1, 46, 83, 42],
       [ 1, 66, 84, 42],
       [ 1, 36, 50, 66]], dtype=int64)
```

```
# Transpose the NumPy array 'x' to obtain the transpose 'x_transpose'
x_transpose = x.T

# Display the transposed array
x_transpose
```

```
array([[ 1,  1,  1,  1,  1,  1],
       [30, 45, 68, 46, 66, 36],
       [63, 47, 67, 83, 84, 50],
       [33, 52, 62, 42, 42, 66]], dtype=int64)
```

```
# Calculate the dot product of the transposed matrix 'x_transpose' and
the original matrix 'x' to get 'x_transposeX'
x_transposeX = x_transpose.dot(x)

# Display the resulting matrix 'x_transposeX'
x_transposeX
```

```
array([[    6,   291,   394,   297],
       [  291, 15317, 19723, 14626],
       [  394, 19723, 27112, 18991],
       [  297, 14626, 18991, 15521]], dtype=int64)
```

```
# Calculate the inverse of the matrix 'x_transposeX' using NumPy's
linear algebra module
x_transposeX_inv = np.linalg.inv(x_transposeX)
```

```
# Display the inverse matrix 'x_transposeX_inv'
x_transposeX_inv
```

```
array([[ 1.69580230e+01,  8.31914459e-02, -1.73130574e-01,
        -1.91055228e-01],
       [ 8.31914459e-02,  1.89410821e-03, -1.54994977e-03,
        -1.48031635e-03],
       [-1.73130574e-01, -1.54994977e-03,  2.35593088e-03,
         1.89084867e-03],
       [-1.91055228e-01, -1.48031635e-03,  1.89084867e-03,
         2.80171396e-03]])
```

```
# Extract the dependent variable 'Y' from the DataFrame 'df' and
convert it to a NumPy array
y = df['Y'].values

# Display the resulting NumPy array 'y'
y
```

```
array([43, 63, 71, 61, 81, 43], dtype=int64)
```

```
# Calculate the dot product of the transposed matrix 'x_transpose' and
the dependent variable 'y' to get 'x_transposeY'
x_transposeY = x_transpose.dot(y)

# Display the resulting vector 'x_transposeY'
x_transposeY
```

```
array([  362, 18653, 24444, 17899], dtype=int64)
```

```
# Calculate the coefficients of the linear regression model using the
normal equation
B_hat = x_transposeX_inv.dot(x_transposeY)

# Extract individual coefficients B0, B1, B2, B3 from the vector B_hat
B0, B1, B2, B3 = B_hat[0], B_hat[1], B_hat[2], B_hat[3]

# Display the calculated coefficients
print("B0:", B0, "\nB1:", B1, "\nB2:", B2, "\nB3:", B3)
```

```
B0: 38.873096698266636
B1: 1.0629493797738405
B2: -0.1518057423441661
B3: -0.4065501253204644
```

```
# Predict the dependent variable 'Y' for new data (50, 70, 80) using
the calculated coefficients
y_predicted = B0 + (B1 * 50) + (B2 * 70) + (B3 * 80)

# Display the predicted value of 'Y'
y_predicted
```

```
48.87015369722931
```

```
# Predict the dependent variable 'Y' for new data (30, 40, 50) using the calculated coefficients
y_predicted = B0 + (B1 * 30) + (B2 * 40) + (B3 * 50)

# Display the predicted value of 'Y'
y_predicted
```

```
44.361842131691986
```

2. Implement Multiple Linear Regression to predict the price given the data set below. Do data preprocessing to fill the null value. Display the coefficients. (that is B1, B2, B3)

Area---Bedrooms---Age---Price

2600---3----------20----550000

3000---4----------15----565000

3200----------18----610000

3600---3----------30----595000

4000---5----------8-----760000

Predict the price for the below

3000---3----------40----?

2500---4----------5-----?

```
# Import necessary libraries
import numpy as np
import pandas as pd

# Read data from the CSV file 'HPriceData.csv' into a DataFrame 'df'
df = pd.read_csv('C:/Users/raosu/Documents/Assignment 6
```

```
aiml/HPriceData.csv')

# Display the first few rows of the DataFrame 'df'
df.head()
```

	Area	Bedrooms	Age	Price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	NaN	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000

```
# Replace non-numeric values ('?') with NaN
df = df.replace('?', np.nan)

# Convert columns to numeric type
df = df.apply(pd.to_numeric)

# Fill NaN values with the median of each column
df = df.fillna(df.median())

# Create a new DataFrame 'x' by dropping the 'Price' column from the
original DataFrame 'df'
x = df.drop('Price', axis=1)

# Display the modified DataFrame 'x'
x
```

	Area	Bedrooms	Age
0	2600	3.0	20
1	3000	4.0	15
2	3200	3.5	18
3	3600	3.0	30
4	4000	5.0	8
5	3000	3.0	40
6	2500	4.0	5

```
# Insert a new column 'x0' with constant value 1 at the beginning of
the DataFrame 'x'
x.insert(0, 'x0', [1, 1, 1, 1, 1, 1, 1])

# Display the modified DataFrame 'x' with the new column 'x0'
x
```

	x0	Area	Bedrooms	Age
0	1	2600	3.0	20
1	1	3000	4.0	15
2	1	3200	3.5	18
3	1	3600	3.0	30
4	1	4000	5.0	8

```
5   1  3000       3.0   40
6   1  2500       4.0    5

# Convert the DataFrame 'x' to a NumPy array
x = x.values

# Display the resulting NumPy array 'x'
x

array([[1.0e+00, 2.6e+03, 3.0e+00, 2.0e+01],
       [1.0e+00, 3.0e+03, 4.0e+00, 1.5e+01],
       [1.0e+00, 3.2e+03, 3.5e+00, 1.8e+01],
       [1.0e+00, 3.6e+03, 3.0e+00, 3.0e+01],
       [1.0e+00, 4.0e+03, 5.0e+00, 8.0e+00],
       [1.0e+00, 3.0e+03, 3.0e+00, 4.0e+01],
       [1.0e+00, 2.5e+03, 4.0e+00, 5.0e+00]])

# Transpose the NumPy array 'x' to get 'x_transpose'
x_transpose = x.T

# Display the transposed array 'x_transpose'
x_transpose

array([[1.0e+00, 1.0e+00, 1.0e+00, 1.0e+00, 1.0e+00, 1.0e+00,
1.0e+00],
       [2.6e+03, 3.0e+03, 3.2e+03, 3.6e+03, 4.0e+03, 3.0e+03,
2.5e+03],
       [3.0e+00, 4.0e+00, 3.5e+00, 3.0e+00, 5.0e+00, 3.0e+00,
4.0e+00],
       [2.0e+01, 1.5e+01, 1.8e+01, 3.0e+01, 8.0e+00, 4.0e+01,
5.0e+00]])

# Calculate the dot product of the transposed array 'x_transpose' and
the original array 'x' to get 'x_transposeX'
x_transposeX = x_transpose.dot(x)

# Display the resulting matrix 'x_transposeX'
x_transposeX

array([[7.000e+00, 2.190e+04, 2.550e+01, 1.360e+02],
       [2.190e+04, 7.021e+07, 8.080e+04, 4.271e+05],
       [2.550e+01, 8.080e+04, 9.625e+01, 4.530e+02],
       [1.360e+02, 4.271e+05, 4.530e+02, 3.538e+03]])

# Calculate the inverse of the matrix 'x_transposeX' using NumPy's
linear algebra module
x_transposeX_inv = np.linalg.inv(x_transposeX)

# Display the resulting inverse matrix 'x_transposeX_inv'
x_transposeX_inv
```

```
array([[ 1.65082071e+01,  7.84051843e-04, -4.02559849e+00,
        -2.13789879e-01],
       [ 7.84051843e-04,  1.25507721e-06, -1.02269182e-03,
        -5.07052381e-05],
       [-4.02559849e+00, -1.02269182e-03,  1.57552154e+00,
         7.64730964e-02],
       [-2.13789879e-01, -5.07052381e-05,  7.64730964e-02,
         4.83021991e-03]])
```

```
# Extract the 'Price' column from the DataFrame 'df' and convert it to a NumPy array 'y'
y = df['Price'].values

# Display the resulting array 'y'
y
```

```
array([550000., 565000., 610000., 595000., 760000., 595000., 595000.])
```

```
# Calculate the dot product of the transposed array 'x_transpose' and the array 'y' to get 'x_transposeY'
x_transposeY = x_transpose.dot(y)

# Display the resulting vector 'x_transposeY'
x_transposeY
```

```
array([4.27000e+06, 1.35315e+10, 1.57950e+07, 8.11600e+07])
```

```
# Calculate the coefficients 'B_hat' using the formula: B_hat = (x_transposeX_inv) * x_transposeY
B_hat = x_transposeX_inv.dot(x_transposeY)

# Extract individual coefficients B0, B1, B2, B3 from 'B_hat'
B0, B1, B2, B3 = B_hat[0], B_hat[1], B_hat[2], B_hat[3]

# Display the calculated coefficients
print("B0:", B0, "\nB1:", B1, "\nB2:", B2, "\nB3:", B3)
```

```
B0: 163927.0857784897
B1: 62.32421045124647
B2: 64059.25664011389
B3: 912.4937231284566
```

```
# Predict the target variable 'y' using the calculated coefficients
y_predicted = B0 + (B1 * 3000) + (B2 * 3) + (B3 * 40)

# Display the predicted value 'y_predicted'
y_predicted
```

```
579577.2359777091
```

```
# Predict the target variable 'y' using the calculated coefficients for a new set of input values
```

```
y_predicted = B0 + (B1 * 2500) + (B2 * 4) + (B3 * 5)

# Display the predicted value 'y_predicted'
y_predicted
```

```
580537.1070827037
```