Regression Model and the Concept of Least Squares

For a multiple linear regression equation expressed as $\mathbf{X}.W = y$, the matrix of regression coefficients, W, is given by:

$$\left(\mathbf{X}^T\mathbf{X}\right)^{-1} X^T y$$

Step 1

```
import numpy as np

# Taking matrix X from the user
print("Enter the elements of matrix X (20x4) row-wise, separated by spaces (20 rows, 4 columns):")
X = []
for _ in range(20):
    row = list(map(float, input().split()))
    if len(row) != 4:
        raise ValueError("Each row must contain exactly 4 elements.")
    X.append(row)

X = np.array(X)

# Calculating the transpose of X
X_transpose = X.T

# Displaying the two matrices
print("\nMatrix X (20x4):")
print(X)
print("\nTranspose of X (X') (4x20):")
print(X_transpose)
```

```
Enter the elements of matrix X (20x4) row-wise, separated by spaces (20 rows, 4 columns):
1 8.4 8 1
1 2 6.5 8.5
1 3.5 6.2 6.5
1 10.4 5 1.5
1 6.5 6.5 7.5
1 6.2 7.3 4.5
1 12.4 6.4 4
1 7 6 10
1 5.8 6.1 3
1 3 5.4 11
1 6 7.3 4.5
1 5.5 6.6 5.5
1 9 6.5 2.5
1 1.1 5.8 7
1 2.1 7.1 9
1 10 8.5 2
1 7 5.5 3
1 5 5 4.5
1 9.3 7.9 3
1 4.4 4.5 7.9

Matrix X (20x4):
[[ 1.   8.4  8.   1. ]
 [ 1.   2.   6.5  8.5]
 [ 1.   3.5  6.2  6.5]
 [ 1.  10.4  5.   1.5]
 [ 1.   6.5  6.5  7.5]
 [ 1.   6.2  7.3  4.5]
 [ 1.  12.4  6.4  4. ]
 [ 1.   7.   6.  10. ]
 [ 1.   5.8  6.1  3. ]
 [ 1.   3.   5.4 11. ]
 [ 1.   6.   7.3  4.5]
 [ 1.   5.5  6.6  5.5]
 [ 1.   9.   6.5  2.5]
 [ 1.   1.1  5.8  7. ]
 [ 1.   2.1  7.1  9. ]
 [ 1.  10.   8.5  2. ]
 [ 1.   7.   5.5  3. ]
 [ 1.   5.   5.   4.5]
 [ 1.   9.3  7.9  3. ]
 [ 1.   4.4  4.5  7.9]]

Transpose of X (X') (4x20):
[[ 1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.
   1.   1.   1.   1.   1.   1. ]
 [ 8.4  2.   3.5 10.4  6.5  6.2 12.4  7.   5.8  3.   6.   5.5  9.   1.1
   2.1 10.   7.   5.   9.3  4.4]
```

```
 [ 8.   6.5  6.2  5.   6.5  7.3  6.4  6.   6.1  5.4  7.3  6.6  6.5  5.8
   7.1  8.5  5.5  5.   7.9  4.5]
 [ 1.   8.5  6.5  1.5  7.5  4.5  4.  10.   3.  11.   4.5  5.5  2.5  7.
   9.   2.   3.   4.5  3.   7.9]]
```

Step 2

```
# Multiplying X' and X to get matrix A
A = np.dot(X_transpose, X)

# Displaying the product matrix A
print("\nProduct Matrix A (4x4):")
print(A)
```

```
Product Matrix A (4x4):
[[ 20.   124.6  128.1  106.4 ]
 [124.6  953.78 816.01 542.91]
 [128.1  816.01 841.87 661.1 ]
 [106.4  542.91 661.1  731.66]]
```

Step 3

```
# Calculating the inverse of matrix A
B = np.linalg.inv(A)

# Displaying the inverse matrix B
print("\nInverse Matrix B (4x4):")
print(B)
```

```
Inverse Matrix B (4x4):
[[ 3.73954454e+00 -9.87413780e-02 -3.57360084e-01 -1.47649325e-01]
 [-9.87413780e-02  1.10996966e-02 -1.86729590e-03  7.81020644e-03]
 [-3.57360084e-01 -1.86729590e-03  5.32841382e-02  5.20835202e-03]
 [-1.47649325e-01  7.81020644e-03  5.20835202e-03  1.23368880e-02]]
```

Step 4

```
# Multiplying B and X' to get matrix C
C = np.dot(B, X_transpose)

# Displaying the product matrix C
print("\nProduct Matrix C (4x20):")
print(C)
```

```
Product Matrix C (4x20):
[[-9.64130405e-02 -3.57980330e-02  2.18596576e-01  7.04359794e-01
  -3.32484909e-01 -1.45802588e-01 -3.62550393e-01 -5.72298868e-01
   5.44000053e-01 -1.10566631e-01 -1.26054312e-01  2.58191109e-02
   1.58908272e-01  5.24695254e-01 -3.33912884e-01 -5.80728613e-01
   6.39926450e-01  7.94615260e-01 -4.44842922e-01  5.30532424e-01]
 [-1.26320871e-02 -2.22926534e-02 -2.07033325e-02  1.90742971e-02
   1.98457750e-02 -8.40859001e-03  5.81849922e-02  4.58547874e-02
  -2.23230232e-02  1.03865848e-02 -1.06285293e-02 -7.06106408e-03
   8.54398439e-03 -4.26905828e-02 -1.83979580e-02  1.20039860e-02
  -7.88300974e-03 -1.74334454e-02  1.31647823e-02  3.39508647e-03]
 [ 5.84360876e-02  2.95232142e-02  3.20324864e-04 -1.02546743e-01
   1.59120306e-02  4.34744739e-02 -1.86626611e-02  1.35719366e-03
  -2.75321016e-02 -1.79357536e-02  4.38479331e-02  1.26910363e-02
  -1.47979692e-02 -1.39076442e-02  6.39111436e-02  8.72988353e-02
  -6.17433396e-02 -7.68382889e-02  6.18438115e-02 -8.46515835e-02]
 [-2.80398868e-02  6.68892412e-03 -7.83204787e-03 -2.18760860e-02
   2.94979651e-02 -5.68907931e-03  3.18782398e-02  6.16411123e-02
  -3.35685163e-02  3.96121634e-02 -7.25112060e-03 -2.46518221e-03
  -1.26609589e-02 -2.24914401e-02  1.67634000e-02 -6.02492475e-04
  -2.73212798e-02 -2.70405367e-02  3.14223983e-03  7.61458271e-03]]
```

Step 5

```
# Taking matrix Y from the user
print("Enter the elements of matrix Y (20x1), separated by spaces (20 elements):")
Y = list(map(float, input().split()))

if len(Y) != 20:
    raise ValueError("Matrix Y must contain exactly 20 elements.")
```

```
Y = np.array(Y).reshape(20, 1)

# Multiplying C and Y to get matrix W
W = np.dot(C, Y)

# Displaying the product matrix W
print("\nMatrix of regression coefficients, W:")
print(W)
```

```
Enter the elements of matrix Y (20x1), separated by spaces (20 elements):
35 10 9 30 20 23 28 8 29 4 18 14 32 6 8 37 25 15 30 10

Matrix of regression coefficients, W:
[[12.01139276]
 [ 1.30894082]
 [ 1.6672016 ]
 [-2.12303013]]
```