EE2211 Tutorial 5

Dr Feng LIN

Q1

Given the following data pairs for training:

Input Output

$$\{x = -10\} \rightarrow \{y = 5\}$$
$$\{x = -8\} \rightarrow \{y = 5\}$$
$$\{x = -3\} \rightarrow \{y = 4\}$$
$$\{x = -1\} \rightarrow \{y = 3\}$$
$$\{x = -2\} \rightarrow \{y = 2\}$$
$$\{x = -8\} \rightarrow \{y = 2\}$$

a) Perform a linear regression with addition of a bias/offset term to the input feature vector and sketch the result of line fitting.

b) Perform a linear regression **without** inclusion of any bias/offset term and sketch the result of line fitting.

c) What is the effect of adding a bias/offset term to the input feature vector?

Q1

(a) This is an over-determined system.

The input feature including bias/offset can be written as $X^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -10 & -8 & -3 & -1 & 2 & 8 \end{bmatrix}$

bias

$$\overbrace{\begin{bmatrix} 1 & -10 \\ 1 & -8 \\ 1 & -3 \\ 1 & -1 \\ 1 & 2 \\ 1 & 8 \end{bmatrix}}^{X} \overbrace{\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}}^{w} = \overbrace{\begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix}}^{y}$$

$$\hat{w} = (X^T X)^{-1} X^T y = \begin{bmatrix} 6 & -12 \\ -12 & 242 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -10 & -8 & -3 & -1 & 2 & 8 \end{bmatrix} \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.1055 \\ -0.1972 \end{bmatrix}$$

Q1

(b) This is an over-determined system.

In this case, the input feature without inclusion of bias/offset is a vector given by $[-10 \quad -8 \quad -3 \quad -1 \quad 2 \quad 8]^T$.

$$\underbrace{\begin{bmatrix} -10 \\ -8 \\ -3 \\ -1 \\ 2 \\ 8 \end{bmatrix}}_{X} \underbrace{[w]}_{w} = \underbrace{\begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix}}_{y}$$

$$\hat{w} = (x^T x)^{-1} x^T y = [242]^{-1} [-10 \quad -8 \quad -3 \quad -1 \quad 2 \quad 8] \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix} = -0.3512$$

Q1

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

# define the matrix X, bias vector and y
X   = np.array([[-10], [-8], [-3], [-1], [2], [8]])
b   = np.ones( (len(X),1) )
X_b = np.hstack((b, X)) # X matrix with bias

y = np.array([[5], [5], [4], [3], [2], [2]])

#(a) Perform a linear regression with addition of a bias/offset term
w_b = inv(X_b.T@X_b)@X_b.T@y
print("w_b=")
print(w_b)
print("\n")
```

```
w_b=
[[ 3.10550459]
 [-0.19724771]]
```

Q1

```
#(b) Perform a linear regression without inclusion of any bias/offset term
w = inv(X.T@X)@X.T@y
print("w=", w)

from numpy.linalg import pinv
w_p = pinv(X)@y
print("w_p=", w_p)

from numpy.linalg import lstsq
w_ls, residuals, rank, s = lstsq(X, y, rcond=-1)
print("w_sl", w_ls)
```

Left-inverse

pseudoinverse

Cutoff condition number

Least square solution

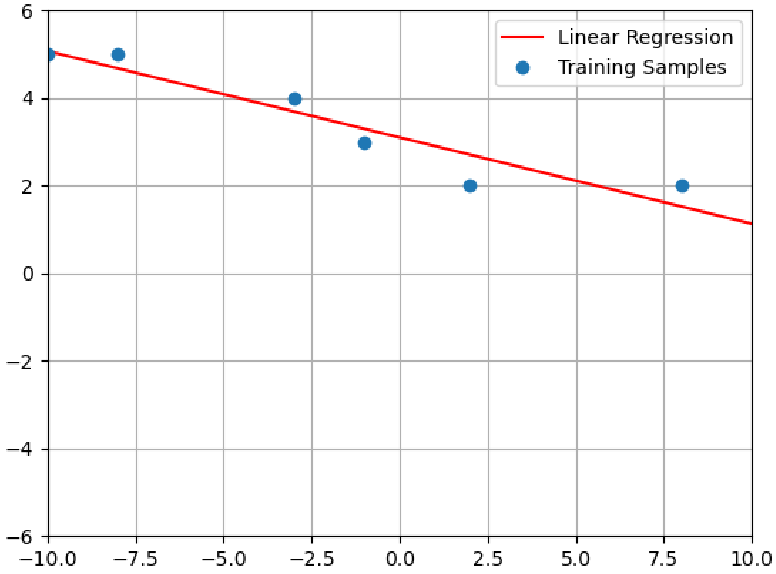
```
w= [[-0.35123967]]
w_p= [[-0.35123967]]
w_sl [[-0.35123967]]
```

Q1

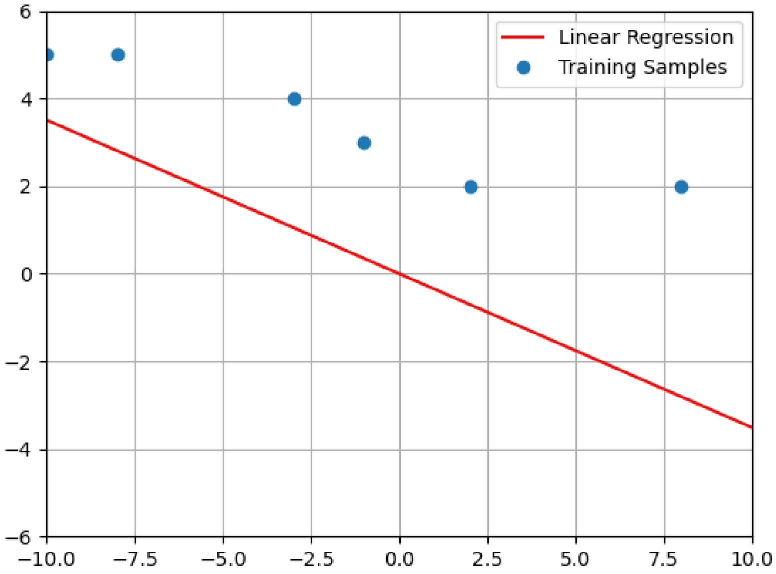
```python
# show the effect of adding a bias/offset term
X_test = np.linspace(-10,10, 400)
X_test = X_test.reshape(-1, 1)
b_test = np.ones((len(X_test),1)) # generate a bias column vector

X_b_test = np.hstack((b_test, X_test))
y_b_test = X_b_test@w_b

plt.figure()
plt.plot(X_test, y_b_test, color='red', label='Linear Regression')
plt.plot(X, y, 'o', label='Training Samples')
plt.xlim(-10,10)
plt.ylim(-6, 6)
plt.grid(True)
plt.legend()
# plt.show()

# show the effect without a bias/offset term
y_test = X_test@w

plt.figure()
plt.plot(X_test, y_test, color='red', label='Linear Regression')
plt.plot(X, y, 'o', label='Training Samples')
plt.xlim(-10,10)
plt.ylim(-6, 6)
plt.grid(True)
plt.legend()
plt.show()
```

(c) The bias/offset term allows the line to move away from the origin (moved vertically in this case).

With bias term



Without bias term



Q2

(Linear Regression, prediction, even/under-determined)

Given the following data pairs for training:

$$\{x_1 = 1, \quad x_2 = 0, \quad x_3 = 1\} \rightarrow \{y = 1\}$$
$$\{x_1 = 2, \quad x_2 = -1, \quad x_3 = 1\} \rightarrow \{y = 2\}$$
$$\{x_1 = 1, \quad x_2 = 1, \quad x_3 = 5\} \rightarrow \{y = 3\}$$

(a) Predict the following test data without inclusion of an input bias/offset term.

(b) Predict the following test data with inclusion of an input bias/offset term.

$$\{x_1 = -1, \quad x_2 = 2, \quad x_3 = 8\} \rightarrow \{y = ?\}$$
$$\{x_1 = 1, \quad x_2 = 5, \quad x_3 = -1\} \rightarrow \{y = ?\}$$

Q2

(a) Without bias, this is an even-determined system and X is invertible

$$\begin{array}{ccc} X & w & y \\ \begin{bmatrix} 1 & 0 & 1 \\ 2 & -1 & 1 \\ 1 & 1 & 5 \end{bmatrix} & \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \\ \boldsymbol{w_3} \end{bmatrix} = & \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \end{array}$$

$$\hat{w} = X^{-1}y = \begin{bmatrix} 1 & 0 & 1 \\ 2 & -1 & 1 \\ 1 & 1 & 5 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.3333 \\ -0.6667 \\ 0.6667 \end{bmatrix}$$

Make prediction $$\hat{y}_t = X_t^{-1}\hat{w} = \begin{bmatrix} -1 & 2 & 8 \\ 1 & 5 & -1 \end{bmatrix} \begin{bmatrix} 0.3333 \\ -0.6667 \\ 0.6667 \end{bmatrix} = \begin{bmatrix} 3.6667 \\ -3.6667 \end{bmatrix}$$

Q2

(b) After adding bias, it becomes an under-determined system.

bias

$$\begin{array}{ccc} X & w & y \\ \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 2 & -1 & 1 \\ 1 & 1 & 1 & 5 \end{bmatrix} & \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = & \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \end{array}$$

$$\hat{w} = X^{\mathrm{T}}(XX^{\mathrm{T}})^{-1}y = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & -1 & 1 \\ 1 & 1 & 5 \end{bmatrix} \begin{bmatrix} 3 & 4 & 7 \\ 4 & 7 & 7 \\ 7 & 7 & 28 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -0.1429 \\ 0.5238 \\ -0.4762 \\ 0.6190 \end{bmatrix}$$

Make prediction $$\hat{y}_{\mathrm{t}} = X_{\mathrm{t}}\ \hat{w} = \begin{bmatrix} 1 & -1 & 2 & 8 \\ 1 & 1 & 5 & -1 \end{bmatrix} \begin{bmatrix} -0.1429 \\ 0.5238 \\ -0.4762 \\ 0.6190 \end{bmatrix} = \begin{bmatrix} 3.3333 \\ -2.6190 \end{bmatrix}$$

Q2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy.linalg import inv

# Without bias, this is an even-determined system and X is invertible.
X   = np.array([[1, 0, 1], [2, -1, 1], [1, 1, 5]])
y = np.array([[1], [2], [3]])

b   = np.ones( (len(X),1) )
X_b = np.hstack((b, X)) # X matrix with bias

#(a) Perform a linear regression with addition of a bias/offset term
w = inv(X)@y
print(f"w={w}\n")

X_t = np.array([[-1, 2, 8], [1, 5,-1]])
y_t = X_t@w
print(f"y_t={y_t}\n")

#(b) After adding bias, it becomes an under-determined system.
w_b = X_b.T@inv(X_b@X_b.T)@y
print(f"w_b={w_b}\n")
```

```
w=[[ 0.33333333]
 [-0.66666667]
 [ 0.66666667]]

y_t=[[ 3.66666667]
 [-3.66666667]]

w_b=[[-0.14285714]
 [ 0.52380952]
 [-0.47619048]
 [ 0.61904762]]
```

Q3

(Linear Regression, prediction, extrapolation)

A college bookstore must order books two months before each semester starts. They believe that the number of books that will ultimately be sold for any particular course is related to the number of students registered for the course when the books are ordered.

They would like to develop a linear regression equation to help plan how many books to order. From past records, the bookstore obtains the number of students registered, X, and the number of books actually sold for a course, Y, for 12 different semesters. These data are shown below.

	X	y
Semester	Students	Books
1	36	31
2	28	29
3	35	34
4	39	35
5	30	29
6	30	30
7	31	30
8	38	38
9	36	34
10	38	33
11	29	29
12	26	26

(a) Obtain a scatter plot of the number of books sold versus the number of registered students.

(b) Write down the regression equation and calculate the coefficients for this fitting.

(c) Predict the number of books that would be sold in a semester when 30 students have registered.

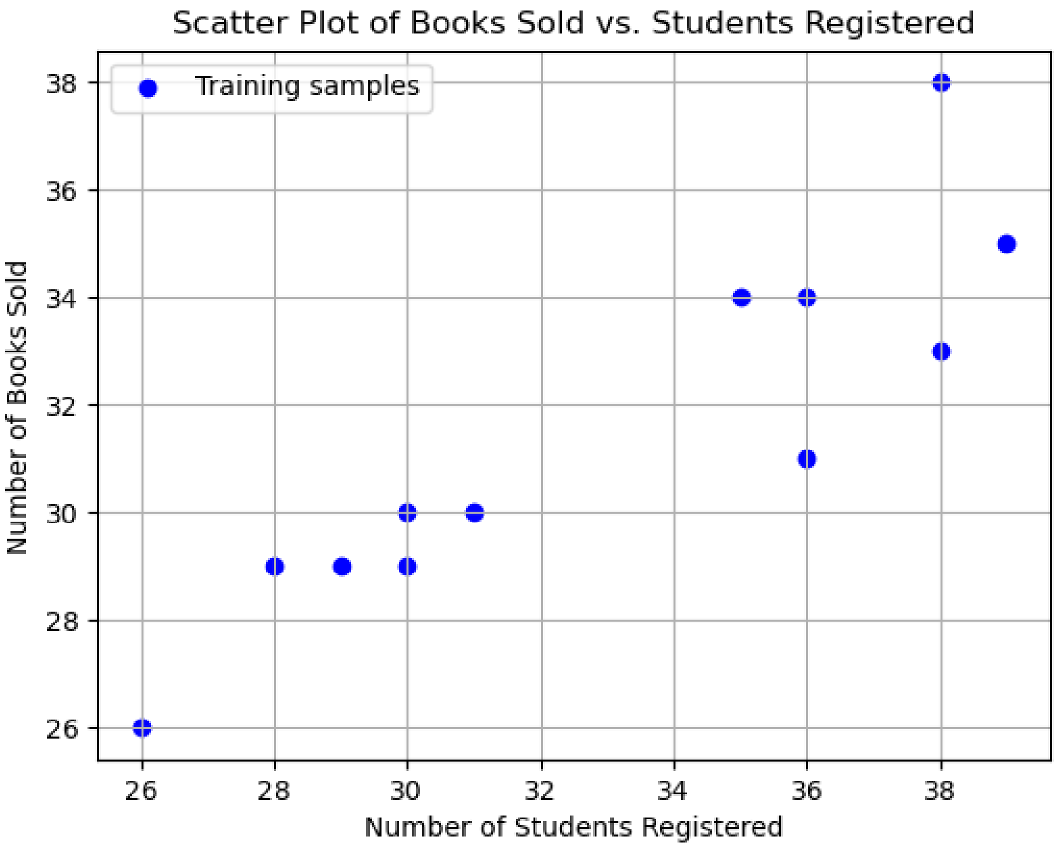
(d) Predict the number of books that would be sold in a semester when 5 students have registered.

Q3(a)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from numpy.linalg import inv

# Define Data: Number of students (X) and number of books sold (y)
X = np.array([36, 28, 35, 39, 30, 30, 31, 38, 36, 38, 29, 26])
X = X.reshape(-1, 1) # reshape as a vertical vector

y = np.array([31, 29, 34, 35, 29, 30, 30, 38, 34, 33, 29, 26])
y = y.reshape(-1, 1) # reshape as a vertical vector

b   = np.ones( (len(X), 1) )
X_b = np.hstack((b, X)) # add bias to the X matrix

# (a) Scatter plot
plt.scatter(X, y, color='blue', label='Training samples')
plt.title('Scatter Plot of Books Sold vs. Students Registered')
plt.xlabel('Number of Students Registered')
plt.ylabel('Number of Books Sold')
plt.grid(True)
plt.legend()
```

Q3(a)

(a) Scatter plot



Q3(b)

Regression equation: $y = Xw$

where $w = [w_0, \quad w_1]^T$,

$X^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 36 & 28 & 35 & 39 & 30 & 30 & 31 & 38 & 36 & 38 & 29 & 26 \end{bmatrix}$,

$y^T = [31 \quad 29 \quad 34 \quad 35 \quad 29 \quad 30 \quad 30 \quad 38 \quad 34 \quad 33 \quad 29 \quad 26]$

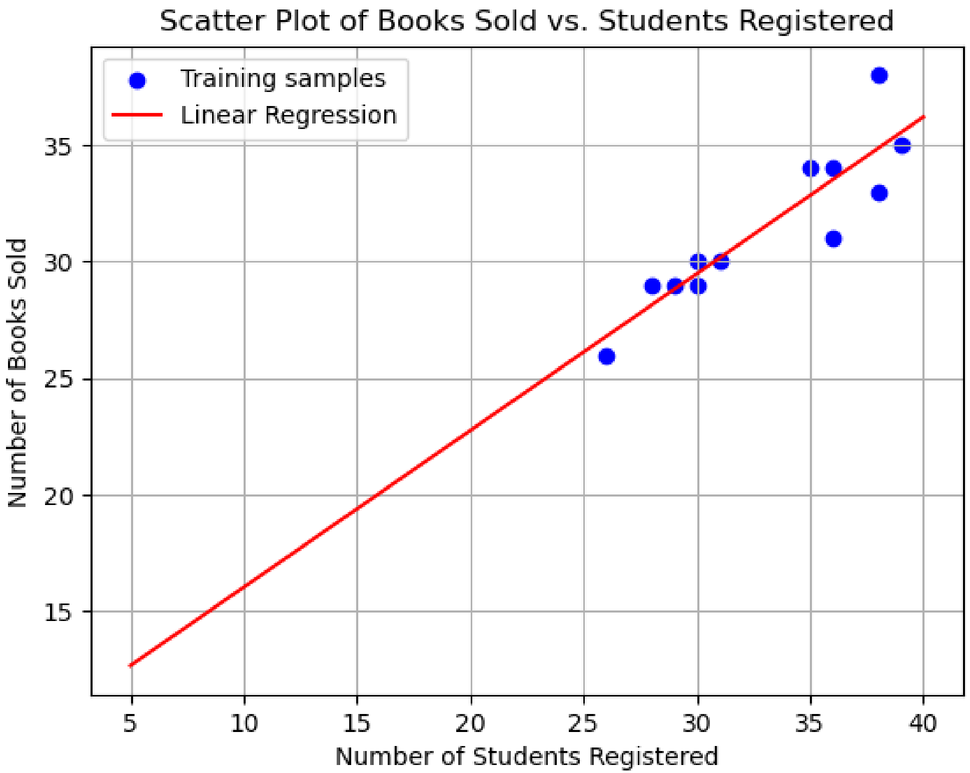
$$\hat{w} = (X^T X)^{-1} X^T y = \begin{bmatrix} 12 & 396 \\ 396 & 13288 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 36 & 28 & 35 & 39 & 30 & 30 & 31 & 38 & 36 & 38 & 29 & 26 \end{bmatrix} \begin{bmatrix} 31 \\ 29 \\ 34 \\ 35 \\ 29 \\ 30 \\ 30 \\ 38 \\ 34 \\ 33 \\ 29 \\ 26 \end{bmatrix} = \begin{bmatrix} 9.30 \\ 0.6727 \end{bmatrix}$$

Q3(b)

```python
# (b) Linear Regression: Calculate w
w = inv(X_b.T@X_b)@X_b.T@y
print(f"w = \n{w} \n");

# draw the estimated line
X_t = np.linspace(5,40, 50)
X_t = X_t.reshape(-1, 1)
b_t = np.ones((len(X_t),1)) # generate a bias column vector

X_b_t = np.hstack((b_t, X_t))
y_b_t = X_b_t@w

plt.plot(X_t, y_b_t, color='red', label='Linear Regression')
plt.legend()
plt.show()
```

```
w =
[[9.3       ]
 [0.67272727]]
```

Q3(b)


Scatter Plot of Books Sold vs. Students Registered
Training samples
Linear Regression
35
30
25
20
15
Number of Books Sold
5
10
15
20
25
30
35
40
Number of Students Registered

Q3

(c)

$$\hat{y}_t = X_t\hat{w} = [1 \quad 30]\begin{bmatrix} 9.30 \\ 0.6727 \end{bmatrix} = 29.4818$$

(d) $(\hat{y}_t = 12.6636)$ This prediction appears to be somewhat over optimistic. Since 5 students is not within the range of the sampled number of students, it might not be appropriate to use the regression equation to make this prediction. We do not know if the straight-line model would fit data at this point, and we might not want to extrapolate far beyond the observed range.

Q3

```python
# (c) Predict books sold when 30 students are registered
X_new_30 = 30
y_pred_30 = np.array([ [1, X_new_30]]) @ w
print(f"y_pred_30 = {y_pred_30} \n")

# (d) Predict books sold when 5 students are registered
X_new_5 = 5
y_pred_5 = np.array([ [1, X_new_5]]) @ w
print(f"y_pred_5 = {y_pred_5} \n")
```

```
y_pred_30 = [[29.48181818]]

y_pred_5 = [[12.66363636]]
```

Q4

Repeat the above problem using the following training data:

a) Calculate the regression coefficients for this fitting.

b) Predict the number of books that would be sold in a semester when 30 students have registered.

c) Purge those duplicating data and re-fit the line and observe the impact on predicting the number of books that would be sold in a semester when 30 students have registered.

d) Sketch and compare the two fitting lines.

Semester	Students	Books
1	36	31
2	26	20
3	35	34
4	39	35
5	26	20
6	30	30
7	31	30
8	38	38
9	36	34
10	38	33
11	26	20
12	26	20

Q4

(a) and (b) using the full data:

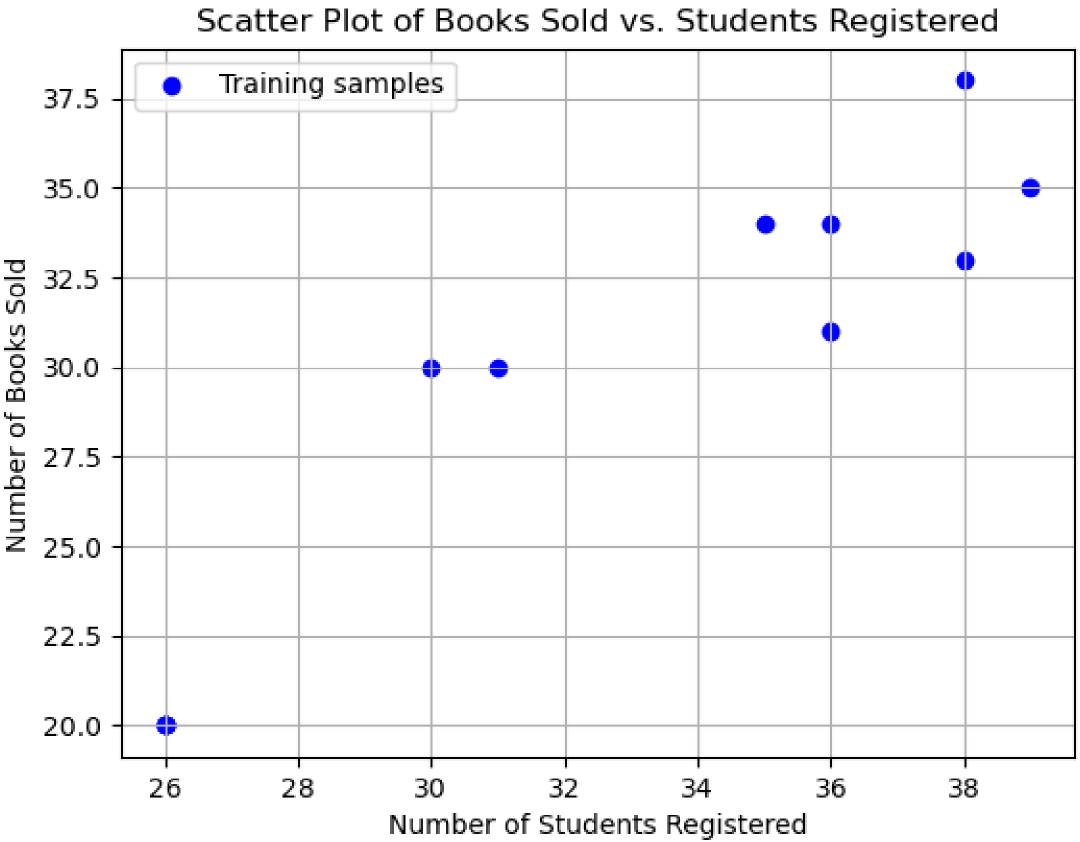
$$\hat{w} = (X^T X)^{-1} X^T y = \begin{bmatrix} 12 & 387 \\ 387 & 13288 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 36 & 26 & 35 & 39 & 26 & 30 & 31 & 38 & 36 & 38 & 26 & 26 \end{bmatrix} \begin{bmatrix} 31 \\ 20 \\ 34 \\ 35 \\ 20 \\ 30 \\ 30 \\ 38 \\ 34 \\ 33 \\ 20 \\ 20 \end{bmatrix} = \begin{bmatrix} -10.4126 \\ 1.2143 \end{bmatrix}$$

$$\hat{y}_t = X_t \hat{w} = \begin{bmatrix} 1 & 30 \end{bmatrix} \begin{bmatrix} -10.4126 \\ 1.2143 \end{bmatrix} = 26.0177$$

Q4

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from numpy.linalg import inv

# Define Data: Number of students (X) and number of books sold (y)
X = np.array([36, 26, 35, 39, 26, 30, 31, 38, 36, 38, 26, 26])
X = X.reshape(-1, 1) # reshape as a vertical vector

y = np.array([31, 20, 34, 35, 20, 30, 30, 38, 34, 33, 20, 20])
y = y.reshape(-1, 1) # reshape as a vertical vector

b   = np.ones( (len(X), 1) )
X_b = np.hstack((b, X)) # add bias to the X matrix

# Scatter plot
plt.scatter(X, y, color='blue', label='Training samples')
plt.title('Scatter Plot of Books Sold vs. Students Registered')
plt.xlabel('Number of Students Registered')
plt.ylabel('Number of Books Sold')
plt.grid(True)
plt.legend()
#plt.show()
```

Q4

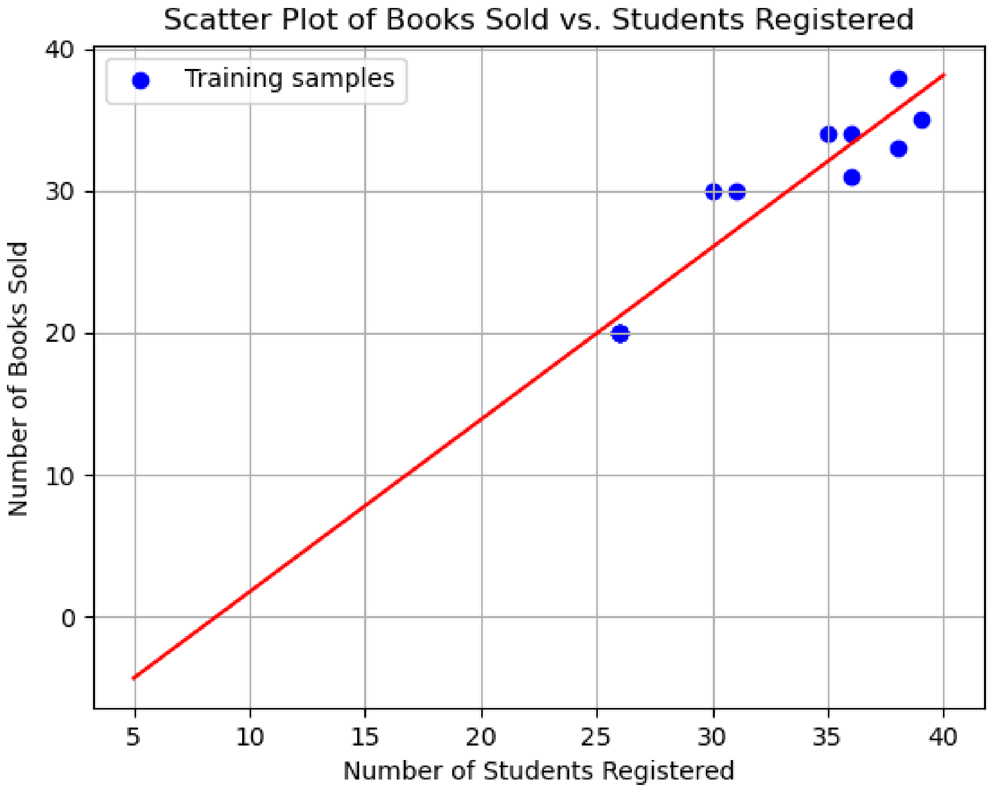

Scatter Plot of Books Sold vs. Students Registered
Training samples
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
Number of Books Sold
26
28
30
32
34
36
38
Number of Students Registered

Q4

```
# (a) Linear Regression: Calculate w
w = inv(X_b.T@X_b)@X_b.T@y
print(f"w=\n{w}\n")

# draw the estimated line
X_t = np.linspace(5,40, 50)
X_t = X_t.reshape(-1, 1)
b_t = np.ones((len(X_t),1)) # generate a bias column vector

X_b_t = np.hstack((b_t, X_t))
y_t = X_b_t@w

plt.plot(X_t, y_t, color='red', label='Linear regression
using all 12 samples')
# plt.show()

# (b) Predict books sold when 30 students are registered
X_new_30 = 30
y_pred_30 = np.array([ [1, X_new_30]]) @ w
print(f"y_pred_30 = {y_pred_30} \n")
```

```
w=
[[-10.41257051]
 [  1.21434327]]

y_pred_30 = [[26.01772764]]
```

Q4



Scatter Plot of Books Sold vs. Students Registered
Training samples
Number of Books Sold
Number of Students Registered
40
30
20
10
0
5
10
15
20
25
30
35
40

Q4

(c) after having the duplicating data purged:

$$\hat{w} = (X^T X)^{-1} X^T y = \begin{bmatrix} 9 & 309 \\ 309 & 10763 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 36 & 35 & 39 & 30 & 31 & 38 & 36 & 38 & 26 \end{bmatrix} \begin{bmatrix} 31 \\ 34 \\ 35 \\ 30 \\ 30 \\ 38 \\ 34 \\ 33 \\ 20 \end{bmatrix} = \begin{bmatrix} -3.5584 \\ 1.0260 \end{bmatrix}$$

$$\hat{y}_t = X_t \hat{w} = \begin{bmatrix} 1 & 30 \end{bmatrix} \begin{bmatrix} -3.5584 \\ 1.0260 \end{bmatrix} = 27.2208$$

Note: these results show that duplicating samples can influence the learning and decision too. In this case, purging seems to give a more optimistic prediction for a relatively small number of students (<37) and more conservative prediction for a relatively large number of students (>37).

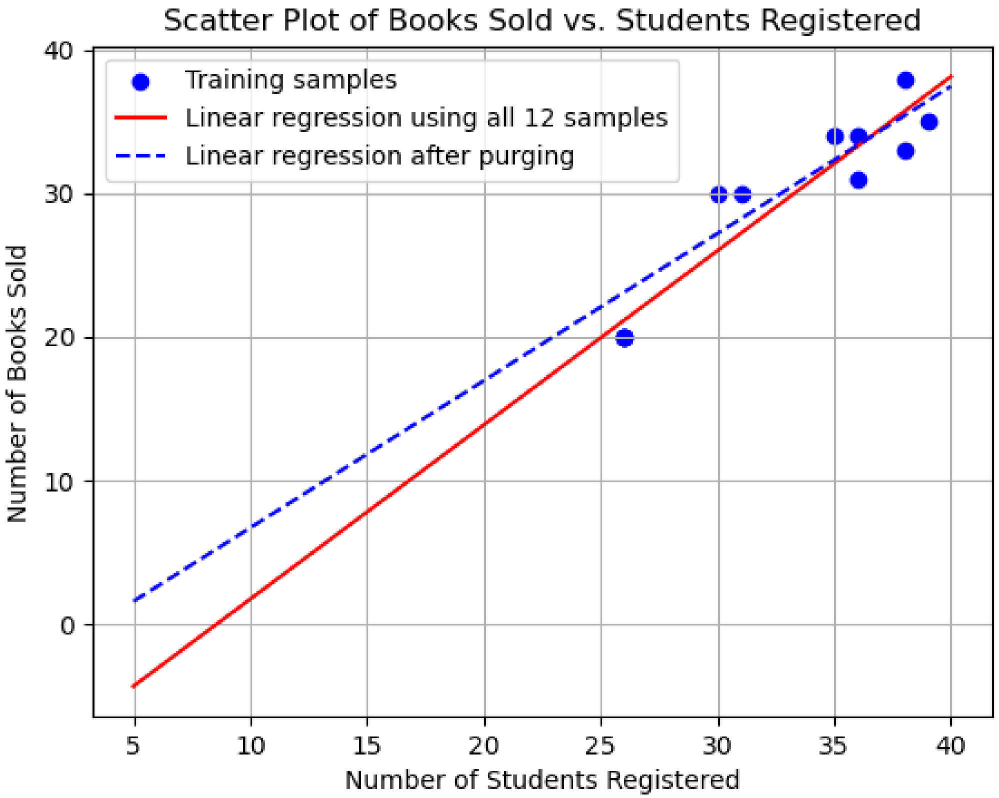
```python
# (c) Purge duplicates (keep only one instance where X = 26 and Y = 20)
# X_cleaned = np.array([36, 35, 39, 30, 31, 38, 36, 38, 26])
# X_cleaned = X_cleaned.reshape(-1,1)
# y_cleaned = np.array([31, 34, 35, 30, 30, 38, 34, 33, 20])
# y_cleaned = y_cleaned.reshape(-1,1)

# find the unique data
duplicated = np.hstack((X,y))
cleaned = np.unique(duplicated , axis=0)
print(f"cleaned data = \n{cleaned}\n")

X_cleaned = cleaned[:,0]
X_cleaned = X_cleaned.reshape(-1,1)
y_cleaned = cleaned[:,1]
y_cleaned = y_cleaned.reshape(-1,1)

b_cleaned   = np.ones( (len(X_cleaned), 1) )
X_b_cleaned = np.hstack((b_cleaned, X_cleaned)) # add bias to the X
matrix
print(f"X_b_cleaned=\n{X_b_cleaned}\n")

w_cleaned = inv(X_b_cleaned.T@X_b_cleaned)@X_b_cleaned.T@y_cleaned
print(f"w_cleaned=\n{w_cleaned}\n")

# Predict books sold when 5 students are registered
X_new_30 = 30
y_pred_30_cleaned = np.array([ [1, X_new_30]]) @ w_cleaned
print("y_pred_30_cleaned={y_pred_30_cleaned}\n")
```

- Returns the sorted unique elements of an array

```
cleaned data=
[[26 20]
 [30 30]
 [31 30]
 [35 34]
 [36 31]
 [36 34]
 [38 33]
 [38 38]
 [39 35]]

X_b_cleaned=
[[ 1. 26.]
 [ 1. 30.]
 [ 1. 31.]
 [ 1. 35.]
 [ 1. 36.]
 [ 1. 36.]
 [ 1. 38.]
 [ 1. 38.]
 [ 1. 39.]]

w_cleaned=
[[-3.55844156]
 [ 1.02597403]]

y_pred_30_cleaned=
[[27.22077922]]
```

Q4

```
# (d) Sketch and compare the two
fitting lines
y_t_cleaned = X_b_t@w_cleaned
plt.plot(X_t, y_t_cleaned,
color='blue', linestyle='--',
label='Linear regression after
purging')
plt.legend()
plt.show()
```



Q5

Download the data file “government-expenditure-on-education.csv” from Canvas Tutorial Folder. It depicts the government’s educational expenditure over the years (downloaded in July 2021 from https://data.gov.sg/dataset/government-expenditure-on-education)

Predict the educational expenditure of year 2021 based on linear regression. Solve the problem using Python with a plot. Note: please use the file from the canvas link.

Hint: use Python packages like numpy, pandas, matplotlib.pyplot, numpy.linalg.

Q5

```
df.head()
```

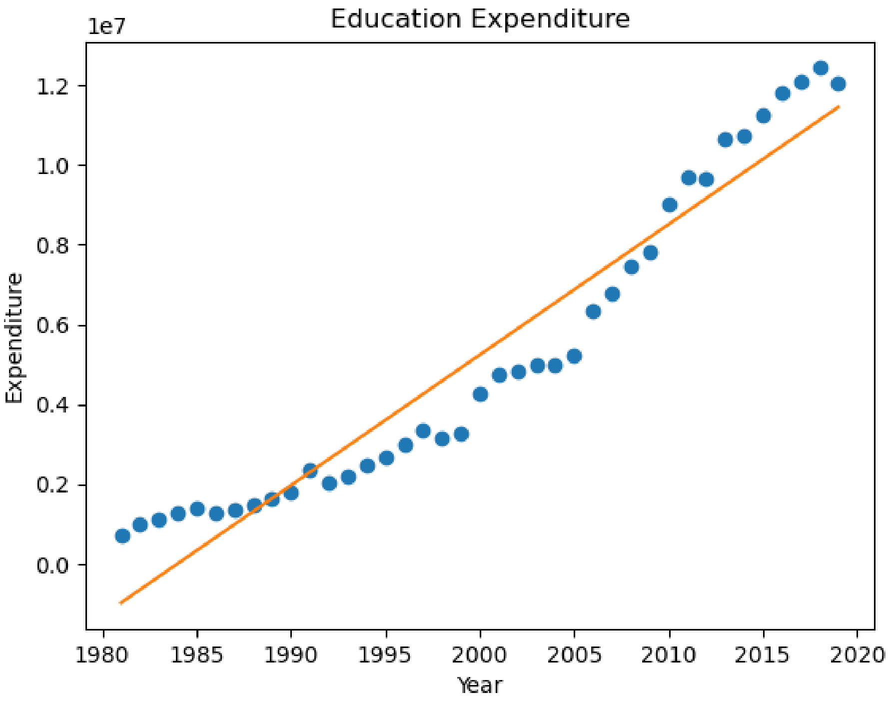
	year	recurrent_expenditure_total
0	1981	712732
1	1982	983751
2	1983	1107113
3	1984	1272559
4	1985	1388186

Q5

```
# read data
df = pd.read_csv("government-expenditure-on-education.csv")

# convert the data to matrices: X and y
expenditureList = df ['recurrent_expenditure_total'].tolist()
yearList = df ['year'].tolist()
m_list = [[1]*len(yearList), yearList]
X = np.array(m_list).T
y = np.array(expenditureList)

# Linear regression
w = inv(X.T @ X) @ X.T @ y
print(f"w=\n{w}\n")

# plot the results
y_line = X.dot(w)
plt.plot(yearList, expenditureList, 'o', label = 'Expenditure
over the years')
plt.plot(yearList, y_line)
plt.xlabel('Year')
plt.ylabel('Expenditure')
plt.title('Education Expenditure')
plt.show()

# prediction
y_predict = np.array([1, 2021]).dot(w)
print(f"y_predict=\n{y_predict}\n")
```

```
w=
[-6.4843247e+08
3.2683591e+05]

y_predict=
12102904.270643115
```

Q5



w=
[-6.4843247e+08
3.2683591e+05]

y_predict=
12102904.270643115

Answer:

The predicted educational expenditure in year 2021 is **12102904.270637512**

Q6

Download the CSV file for red-wine using " wine = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learningdatabases/wine-quality/winequality-red.csv",sep=';') " . Use Python to perform the following tasks.

Hint: use Python packages like numpy, pandas, matplotlib.pyplot, numpy.linalg, and sklearn.metrics.

a) Take y = wine.quality as the target output and x = wine.drop('quality',axis = 1) as the input features. Assume the given list of data is already randomly indexed (i.e., not in particular order), split the database into two sets: [0:1500] samples for regression training, and [1500:1599] samples for testing.
b) Perform linear regression on the training set and print out the learned parameters.
c) Perform prediction using the test set and provide the prediction accuracy in terms of the mean of squared errors (MSE).

Q6

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
```

[4]:
```
wine
```

[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

Q6

```
import pandas as pd
#import matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv
from sklearn.metrics import mean_squared_error

## get data from web
# wine = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-
databases/winequality/winequality-red.csv",sep=';')
wine = pd.read_csv("winequality-red.csv",sep=';'); # get data from the downloaded local
file
wine.info()
y = wine.quality
x = wine.drop('quality',axis = 1)

## Include the offset/bias term
x0 = np.ones((len(y),1))
X = np.hstack((x0,x))
```

- x = wine.drop('quality', axis=1) : creates a variable x that stores the features by dropping the quality column from ‘wine’;
- The axis=1 argument indicates that the column (quality) is being removed (axis 1 is for columns, axis 0 is for rows).

Q6

```
## split data into training and test sets
## (Note: this exercise introduces the basic protocol of using the training-test
partitioning of samples for evaluation assuming the list of data is already randomly
indexed)
## In case you really want a general random split to have a better training/test
distributions:
## from sklearn.model_selection import train_test_split
## train_X,test_X,train_y,test_y = train_test_split(X,y,test_size=99/1599, random_state=0)

# randomly split the data into training and testing sets
# from sklearn.model_selection import train_test_split
# train_X,test_X,train_y,test_y = train_test_split(X,y,test_size=99/1599, random_state=0)
```

- test_size=99/1599: This specifies the proportion of the dataset to include in the test set
- random_state=4: This ensures reproducibility.

```
# deterministically split data into training and testing sets
train_X = X[0:1500]
train_y = y[0:1500]
test_X = X[1500:1599]
test_y = y[1500:1599]
```

- Split the database into two sets: [0:1500] samples for regression training, and [1500:1599] samples for testing.

Q6

```
## linear regression
w = inv(train_X.T @ train_X) @ train_X.T @ train_y
print(f"w=\n{w}\n")
yt_est = test_X.dot(w)
MSE = np.square(np.subtract(test_y,yt_est)).mean()
print(f"MSE={MSE}\n")
MSE = mean_squared_error(test_y,yt_est)
print(f"MSE={MSE}\n")
```

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

```
w=
[ 2.22330327e+01  2.68702621e-02 -1.12838019e+00 -2.06141685e-01
  1.22000584e-02 -1.77718503e+00  4.29357454e-03 -3.18953315e-03
 -1.81795124e+01 -3.98142390e-01  8.92474793e-01  2.77147239e-01]

MSE=0.3435263813857623

MSE=0.3435263813857623
```

Q7

This question is related to understanding of modelling assumptions. The function given by $f(\mathbf{x}) = 1 + x_1 + x_2 - x_3 - x_4$ is affine.

a) True

b) False

Q7

Answer: a)

Q8

MCQ: There could be more than one answer.

Suppose $f(\mathbf{x})$ is a scalar function of d variables where $\mathbf{x}$ is a $d \times 1$ vector. Then, without taking data points into consideration, differentiation of $f(\mathbf{x})$ $w.r.t.\,\mathbf{x}$ is

a) a scalar

b) a $d \times 1$ vector

c) a $d \times d$ matrix

d) a $d \times d \times d$ tensor

e) None of the above

Q8

Answer: b)

Q9

(Linear regression with multiple outputs)

The values of feature vector **x** and their corresponding values of target vector **y** are shown in the table below:

$\boldsymbol{x}$	[3, -1, 0]	[5, 1, 2]	[9, -1, 3]	[-6, 7, 2]	[3, -2, 0]
y	[1, -1]	[-1, 0]	[1, 2]	[0, 3]	[1, -2]

Find the least square solution of $\boldsymbol{w}$ using linear regression of multiple outputs and then estimate the value of $\boldsymbol{y}$ when $\boldsymbol{x}$ = [8, 0, 2].

Q9

```
import numpy as np
from numpy.linalg import inv

X = np.array([[1, 3, -1, 0], [1, 5, 1, 2], [1, 9, -1, 3], [1, -6, 7, 2], [1, 3, -2, 0]])
Y = np.array([[1, -1], [-1, 0], [1, 2], [0, 3], [1, -2]])

W = inv(X.T @ X) @ X.T @ Y
print(f"W=\n{W}\n")

newX=np.array([1, 8, 0, 2])
newY=newX@W
print(f"newY=\n{newY}\n")
```

```
W=
[[ 1.14668974 -0.95997404]
 [-0.630463   -0.33427088]
 [-1.10601471 -0.24426655]
 [ 1.3595846   1.77953267]]

newY=
[-1.17784509 -0.07507572]
```

THANK YOU