Question 1

In [90]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

In [91]:

```
dataset = pd.read_csv('heart.csv', header=0)
```

In [92]:

```
dataset.head()
```

Out[92]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [93]:

```
x = dataset.iloc[:,:-1].values
y = dataset.iloc[:,1].values #target column is dependent variable(output of model)
```

In [94]:

```
#dividing the dataset into testing and training set using sklearn module
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(x, y, train_size=0.8, shuffle=Tr
```

In [95]:

```
print(train_x.shape)
print(test_x.shape)
print(train_y.shape)
print(test_y.shape)
```

```
(242, 13)
(61, 13)
(242,)
(61,)
```

In [96]:

```
model = LinearRegression()
```

In [97]:

```
model.fit(train_x, train_y)
```

Out[97]:

```
LinearRegression()
```

In [98]:

```
prediction = model.predict(test_x)
```

In [99]:

```
np.round(prediction)
```

```
array([ 1.,  1.,  1.,  1., -0.,  1., -0.,  1.,  1.,  1.,  0.,  1., -0.,
```

Out[99]:

```
       -0.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1., -0.,
        0.,  1., -0., -0.,  1., -0., -0.,  1.,  1.,  1., -0.,  1.,  1.,
        1., -0.,  1.,  1.,  1., -0.,  1.,  1.,  1.,  1.,  0.,  0.,  1.,
        1.,  1.,  1.,  1.,  1.,  1., -0.,  1.,  1.])
```

In [100…

```
# test_y
np.round(prediction) == test_y
```

Out[100…

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

In [101…

```
error = np.sum(prediction-test_y)
```

In [102…

```
error
```

Out[102…

```
1.6460733146902606e-14
```

Question 2

In [103…

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

In [104…

```
dataset = pd.read_csv('heart.csv', header=0)

x = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
```

In [107…

```
#using sklearn module to split testing data and training dataset
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(x, y, train_size=0.8, shuffle=Tr
```

In [108…

```
#sigmoid = 1/(1+expt(-x.theta))
def sigmoid(x, theta):
    return 1/(1+ np.exp(-np.dot(x, theta)))
```

In [109…

```
print(train_x.shape)
print(test_x.shape)
print(train_y.shape)
print(test_y.shape)
```

```
(242, 13)
(61, 13)
(242,)
(61,)
```

In [110…

```
theta = np.zeros(x.shape[1])
```

In [117…

```
def logistic_regression(x, y, theta, alpha, iterations):
    m = x.shape[0]
    for _ in range(iterations):
        prec_y = sigmoid(x, theta)
        theta = theta - (alpha/m)*(np.dot(x.T, prec_y - y))
#         cost = (1/m)*(np.dot(y, np.log(prec_y)) + np.dot(1-y, np.log(1-prec_y)))
#         cost = (1/m)*(y*log(prec_y) + (1-y)*(log(1-prec_y))
#         if _%(iterations/10) == 0:
#             print(f"Cost: {cost}")
    return theta
```

In [119…

```
theta = logistic_regression(train_x, train_y, theta, 0.005, 30000)
```

In [120…

```
#precdiction
prec_y = np.round(sigmoid(test_x, theta))
prec_y
```

Out[120…

```
array([1., 1., 0., 1., 0., 1., 1., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [121…

```
error = np.sum(prec_y != test_y)
error #number of falsified outputs
```

Out[121…

```
20
```

In [122…

```
accuracy = 100*np.sum(test_y == prec_y)/test_y.shape[0]
accuracy
```

Out[122…

```
67.21311475409836
```

In [ ]:

```
```