Student information

Name: Hoang Tuan Tu

ID: 21000709

In [ ]:
```
# Importing library
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
# Read data
data = []
with open("vidu3_lin_reg.txt") as f:
    data = f.readlines()

# Remove header row
data = data[1:]

# slit and cast data to float
data = [list(map(float, x.strip().split(','))) for x in data]

# Cast data to numpy array
data = np.array(data)

# Remove index row
data = data[:, 1:]

print(data)
```

```
[[ 56.    21.   160.    14.     6.     1.95]
 [ 76.    18.   150.    12.     4.97   1.33]
 [ 63.    16.   160.     4.4    6.39   0.83]
 [ 78.    20.   100.     4.     7.     2.  ]
 [ 87.    20.   110.     4.6    4.1    1.3 ]
 [ 76.    19.   150.     4.6    2.74   1.16]
 [ 55.    31.   160.     5.5    4.6    1.  ]
 [ 74.    22.   100.     6.8    5.04   1.  ]
 [ 81.    21.   120.     5.8    4.75   0.8 ]
 [ 77.    24.   160.     5.4    6.94   1.6 ]
 [ 29.    20.   120.     3.8    4.84   0.65]
 [ 71.    22.   160.     3.3    6.63   1.  ]
 [ 77.    21.   160.     5.1    4.93   0.97]
 [ 59.    18.   150.     6.     4.55   0.73]
 [ 58.    27.   130.     6.9    6.7    1.1 ]
 [ 34.    19.   130.     4.5    3.2    1.1 ]
 [ 74.    22.   100.    10.6    4.3    1.1 ]
 [ 61.    19.   170.    18.     6.8    0.8 ]
 [ 53.    20.   130.    25.     5.5    0.99]
 [ 65.    28.   140.     6.5    6.8    1.  ]
 [ 80.    19.   160.     4.8    5.74   1.13]
 [ 71.    25.   160.     6.2    6.9    1.  ]
 [ 90.    24.   160.     4.7    7.     1.7 ]
 [ 44.    24.   120.     6.     3.4    0.9 ]
 [ 91.    27.   150.     6.1    4.92   0.89]
 [ 75.    22.   160.     6.2    6.08   0.8 ]
 [ 60.    24.   140.     4.7    6.25   0.81]
 [ 51.    22.   150.     4.8    5.4    1.2 ]
 [ 91.    29.   120.     4.2    6.54   0.82]
 [ 45.    24.   170.     4.9    3.91   0.89]
 [ 62.    24.   140.     5.4    5.3    1.19]
 [ 65.    19.   150.    12.     2.6    0.97]
 [ 70.    22.   160.     3.6    6.85   0.97]
 [ 56.    27.   150.     5.7    3.75   0.97]
 [ 51.    19.   120.     4.7    5.84   0.88]
 [ 75.    18.   140.    10.1    6.91   0.97]
 [ 58.    32.   160.     4.7    5.01   0.9 ]
 [ 61.    19.   160.     5.2    4.     0.89]
 [ 72.    18.   120.    22.2    4.88   0.8 ]
 [ 82.    25.    90.     8.2    4.2    1.13]
 [ 95.    24.   120.    11.     4.47   1.2 ]
 [ 56.    21.   160.     4.9    6.9    0.9 ]
 [ 36.    20.   130.     4.5    4.71   0.81]
 [ 67.    18.   100.     5.5    5.7    0.8 ]
 [ 64.    22.   130.     6.2    3.     0.74]
 [ 81.    22.   140.     5.     5.06   2.66]
 [ 46.    22.   160.     3.3    4.61   0.89]
 [ 56.    22.   150.     4.1    4.15   0.79]
 [ 60.    24.   140.     7.1    5.3    0.8 ]
 [ 35.    19.   120.     7.4    4.1    0.56]
 [ 55.    21.   160.     5.4    3.     0.8 ]
 [ 70.    20.   150.     6.2    2.57   1.2 ]
 [ 64.    23.   130.     5.7    6.78   0.82]
 [ 64.    19.   160.     5.9    5.62   0.9 ]
 [ 58.    27.   160.    26.     8.07   1.  ]
 [ 73.    23.   140.     5.6    3.     1.15]
 [ 41.    24.   110.    10.     3.31   1.16]
 [ 74.    23.   100.     5.3    4.73   0.97]
 [ 21.    23.   160.     5.     4.     0.8 ]
 [ 67.    25.   150.     3.5    3.6    1.67]
 [ 57.    23.   140.     6.4    5.3    1.06]
 [ 69.    21.   120.     7.6    6.     1.1 ]
 [ 53.    34.   140.     8.1    6.49   0.8 ]
 [ 58.    23.   160.     9.     7.     1.7 ]
 [ 54.    29.   130.     6.4    7.48   0.99]
 [ 49.    17.   130.     6.3    5.19   1.16]
 [ 59.    22.   140.     7.     3.     0.62]
 [ 65.    23.   150.     5.9    6.7    1.  ]
 [ 42.    22.   150.     3.9    7.     0.82]
 [ 75.    24.   100.     6.4    6.6    1.  ]
 [ 72.    21.   140.    11.     5.75   1.7 ]
 [ 82.    24.   190.    18.     4.7    2.3 ]
 [ 70.    18.   160.     3.3    4.61   0.89]
 [ 42.    24.   160.     6.     6.3    0.97]
 [ 32.    19.   140.     4.     2.     0.7 ]
 [ 61.    21.   140.     5.2    2.5    1.1 ]
 [ 60.    26.   130.    11.3    4.79   1.01]
 [ 76.    19.   160.     5.1    5.31   1.15]
 [ 78.    27.   120.     4.9    3.8    0.92]
 [ 71.    26.   150.     6.6    7.13   1.1 ]
 [ 49.    24.   140.     4.3    5.5    0.8 ]
 [ 36.    23.   140.     4.3    4.2    0.7 ]
 [ 74.    21.   140.    17.     3.3    1.  ]
 [ 53.    21.   140.     5.6    5.9    0.8 ]
 [ 56.    19.   140.     4.1    4.73   0.89]
 [ 60.    20.   160.     4.9    3.     0.6 ]
 [ 83.    21.   120.     7.9    5.88   1.5 ]
 [ 68.    23.   130.     4.     5.39   0.7 ]
 [ 69.    19.   100.     4.4    6.15   1.1 ]
 [ 31.    21.   120.     4.1    3.94   0.81]
 [ 34.    21.   140.     6.7    3.83   0.7 ]
 [ 41.    20.   120.     2.7    4.93   0.71]
 [ 72.    22.   160.     6.4    7.     2.7 ]
 [ 54.    22.   170.     6.2    8.18   1.13]
 [ 54.    28.   150.     4.2    8.16   1.7 ]
 [ 55.    24.   160.     5.     7.2    0.9 ]
 [ 76.    15.   140.     3.1    5.24   1.16]
 [ 70.    25.   180.     4.     4.4    1.  ]
 [ 85.    21.   160.     5.2    5.2    0.97]
 [ 87.    22.   130.     9.     5.2    2.3 ]]
```

In [ ]:
```
# Slit data to input data set and label
label = data[:, len(data[0]) - 1:]
data = data[:, :-1]


print(data[:5])
```

```
[[ 56.    21.   160.    14.     6.  ]
 [ 76.    18.   150.    12.     4.97]
 [ 63.    16.   160.     4.4    6.39]
 [ 78.    20.   100.     4.     7.  ]
 [ 87.    20.   110.     4.6    4.1 ]]
```

In [ ]:
```
# Slit data to test and train
## Train dataset
X_train = data[:80]
Y_train = label[:80]

## Test dataset
X_test = data[80:]
Y_test = label[80:]
```

In [ ]:
```
# Train data
print(X_train[:5])
print(Y_train[:5])
```

```
[[ 56.    21.   160.    14.     6.  ]
 [ 76.    18.   150.    12.     4.97]
 [ 63.    16.   160.     4.4    6.39]
 [ 78.    20.   100.     4.     7.  ]
 [ 87.    20.   110.     4.6    4.1 ]]
[[1.95]
 [1.33]
 [0.83]
 [2.  ]
 [1.3 ]]
```

In [ ]:
```
# Test data
print(X_test[:5])
print(Y_test[:5])
```

```
[[ 49.    24.   140.     4.3    5.5 ]
 [ 36.    23.   140.     4.3    4.2 ]
 [ 74.    21.   140.    17.     3.3 ]
 [ 53.    21.   140.     5.6    5.9 ]
 [ 56.    19.   140.     4.1    4.73]]
[[0.8 ]
 [0.7 ]
 [1.  ]
 [0.8 ]
 [0.89]]
```

In [ ]:
```
# Define function
def qr_householder(A):
    M = A.shape[0]
    N = A.shape[1]

    # set Q to the identity matrix
    Q = np.identity(M)

    # set R to zero matrix
    R = np.copy(A)

    for n in range(N):
    # vector to transform
        x = A[n:, n]
        k = x.shape[0]

        # compute ro=-sign(x0)||x||
        ro = -np.sign(x[0]) * np.linalg.norm(x)

        # compute the householder vector v
        e = np.zeros(k)
        e[0] = 1
        v = (1 / (x[0] - ro)) * (x - (ro * e))

        # apply v to each column of A to find R
        for i in range(N):
            R[n:, i] = R[n:, i] - (2 / (v@v)) * ((np.outer(v, v)) @ R[n:, i])

        # apply v to each column of Q
        for i in range(M):
            Q[n:, i] = Q[n:, i] - (2 / (v@v)) * ((np.outer(v, v)) @ Q[n:, i])

    return Q.transpose(), R

def linear_regression(x_data, y_data):
    # add column 1
    x_bars = np.concatenate((np.ones((x_data.shape[0], 1)), x_data), axis=1)

    Q, R = qr_householder(x_bars) # QR decomposition
    R_pinv = np.linalg.pinv(R) # calculate inverse matrix of R
    A = np.dot(R_pinv, Q.T) # apply formula

    return np.dot(A, y_data)
```

In [ ]:
```
# Solve
w = linear_regression(X_train, Y_train)
w = w.T.tolist()
intercept = w[0][0]
coef = w[0][1:]
print('Intercept:', intercept)
print("Coefficient: ", coef)
```

```
Intercept: 0.04306436410329317
Coefficient:  [0.08898919688929797, -0.0004774242218527069, 0.002602179867555783, 0.008086342231978141, 0.007085352341923808]
```

In [ ]:
```
# Predict with test data
x = np.array(X_test)

y_pred = np.array([intercept] * len(x))

for i in range(len(x)):
    for j in range(len(x[0])):
        y_pred[i] += coef[j] * x[i, j]

print(y_pred)
```

```
[0.91012272 0.78452963 1.22339369 0.96085817 0.96836123 1.05009561
 1.19694736 1.0521678  0.9936207  0.68503543 0.78429172 0.77109845
 1.19748204 1.06844174 0.99721921 1.03380704 1.14558205 1.19228584
 1.29236178 1.26252546]
```

In [ ]:
```
# Skit learn solution
model = LinearRegression()

model.fit(X_train, Y_train)
print(model.coef_[0])
print(model.intercept_[0])
```

```
[ 0.0089892  -0.00047742  0.00260218  0.00808634  0.00708535]
0.04306436410329706
```

In [ ]:
```
# Predict with sklearn solution
pred = model.predict(X_test)
print(pred.T[0])
```

```
[0.91012272 0.78452963 1.22339369 0.96085817 0.96836123 1.05009561
 1.19694736 1.0521678  0.9936207  0.68503543 0.78429172 0.77109845
 1.19748204 1.06844174 0.99721921 1.03380704 1.14558205 1.19228584
 1.29236178 1.26252546]
```

In [ ]:
```
df = pd.DataFrame({"My Solution" :  y_pred,"Sklearn Solution" : pred[:, 0], "Label": Y_test[:, 0]})
print(df)
```

```
    My Solution  Sklearn Solution  Label
0      0.910123          0.910123   0.80
1      0.784530          0.784530   0.70
2      1.223394          1.223394   1.00
3      0.960858          0.960858   0.80
4      0.968361          0.968361   0.89
5      1.050096          1.050096   0.60
6      1.196947          1.196947   1.50
7      1.052168          1.052168   0.70
8      0.993621          0.993621   1.10
9      0.685035          0.685035   0.81
10     0.784292          0.784292   0.70
11     0.771098          0.771098   0.71
12     1.197482          1.197482   2.70
13     1.068442          1.068442   1.13
14     0.997219          0.997219   1.70
15     1.033807          1.033807   0.90
16     1.145582          1.145582   1.16
17     1.192286          1.192286   1.00
18     1.292362          1.292362   0.97
19     1.262525          1.262525   2.30
```

In [ ]:
```
# Caculate Mean Squared Error
mse = np.sum((Y_test - y_pred) ** 2) / len(y_pred)
print(mse)
```

```
0.5084895842612145
```

In [ ]:
```
# caculate Variance of the Error Term
var = np.sum((Y_test - y_pred - mse) ** 2) / len(y_pred)
print(var)
```

```
832.8930197387374
```