Student information

Name: Hoang Tuan Tu

ID: 21000709

In [ ]:

```python
# Importing library
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
```

In [ ]:

```python
# Reading data
data = pd.read_csv('real_estate.csv')
```

In [ ]:

```python
# Pre-processing

## Removing first column (index column)
data = data.drop(columns=['No'])

## Change data type of X1
data['X1 transaction date'] = data['X1 transaction date'].astype(int)
```

In [ ]:

```python
# View first 5 row of data
print(data.head(5))
x_data = data.iloc[:,:-1]
y_data = data['Y house price of unit area']
```

```
   X1 transaction date  X2 house age  X3 distance to the nearest MRT station  \
0                 2012          32.0                                84.87882   
1                 2012          19.5                               306.59470   
2                 2013          13.3                               561.98450   
3                 2013          13.3                               561.98450   
4                 2012           5.0                               390.56840   

   X4 number of convenience stores  X5 latitude  X6 longitude  \
0                               10     24.98298     121.54024   
1                                9     24.98034     121.53951   
2                                5     24.98746     121.54391   
3                                5     24.98746     121.54391   
4                                5     24.97937     121.54245   

   Y house price of unit area  
0                        37.9  
1                        42.2  
2                        47.3  
3                        54.8  
4                        43.1  
```

Slit data to train and test

In [ ]:

```python
## Train data
x_train = x_data[:350]
y_train = y_data[:350]

## Test data
x_test = x_data[350:]
y_test = y_data[350:]
```

In [ ]:

```python
# 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 [ ]:

```python
w = linear_regression(x_data, y_data) # get result
w = w.T.tolist()
coef = w[1:]
intercept = w[0]
print('Intercept:', intercept)
print("Coefficient: ", coef)
```

```
Intercept: -9859.500752139651
Coefficient:  [2.937206600866556, -0.27472311191143695, -0.004370149788463273, 1.1618225695590005, 234.46769598208985, -15.33576369370165]
```

In [ ]:

```python
# 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)
```

```
[42.78059274 32.60526828 26.12637597 35.59343372 31.37180724 49.41742525
 39.44793757 52.41284316 48.64901623 27.97318862 46.15159586 41.33257179
 44.04764989 48.86819503 41.78884723 29.42844377 25.77338657 30.24324187
 39.88498541 28.1293385  43.85247872 43.86422703 41.07439298 45.07987782
 48.62394007 30.71086237 33.69054696 49.16013422 40.4073473  51.21047775
 47.28418693 54.57145766 15.41823335 37.34790486 12.93726214 53.11844413
 40.48342432 31.83090876 33.75508788 40.76207676 43.94046628 30.28869431
 39.33940067 44.10259452 15.23183325 38.45392515 28.16482452 44.94990368
 33.69054696 38.86361493 41.2188481  34.52971949 38.82809755 45.3458884
 46.65232377 37.88962235 50.51327045 28.08307786 31.59242114 16.12745646
 50.39892957 47.27143635 46.1969649  53.08888537]
```

Solution Using Scikit learn

In [ ]:

```python
# Build model
model = LinearRegression()
model.fit(x_data, y_data)

print(model.coef_)
print(model.intercept_)
```

```
[ 2.93720660e+00 -2.74723112e-01 -4.37014979e-03  1.16182257e+00
  2.34467696e+02 -1.53357637e+01]
-9859.500751985148
```

In [ ]:

```python
# predict with test data
sklearn_pred = model.predict(x_test)
print(sklearn_pred)
```

```
[42.78059274 32.60526828 26.12637597 35.59343372 31.37180724 49.41742525
 39.44793757 52.41284316 48.64901623 27.97318862 46.15159586 41.33257179
 44.04764989 48.86819503 41.78884723 29.42844377 25.77338657 30.24324187
 39.88498541 28.1293385  43.85247872 43.86422703 41.07439298 45.07987782
 48.62394007 30.71086237 33.69054696 49.16013422 40.4073473  51.21047775
 47.28418693 54.57145766 15.41823335 37.34790486 12.93726214 53.11844413
 40.48342432 31.83090876 33.75508788 40.76207676 43.94046628 30.28869431
 39.33940067 44.10259452 15.23183325 38.45392515 28.16482452 44.94990368
 33.69054696 38.86361493 41.2188481  34.52971949 38.82809755 45.3458884
 46.65232377 37.88962235 50.51327045 28.08307786 31.59242114 16.12745646
 50.39892957 47.27143635 46.1969649  53.08888537]
```

Comprraring solution and real house price

In [ ]:

```python
df = pd.DataFrame({'My Solution': y_pred,
                   'Sklearn solution': sklearn_pred, 'Real house price': y_test})
print(df)
```

```
     My Solution  Sklearn solution  Real house price
350    42.780593         42.780593              42.3
351    32.605268         32.605268              28.6
352    26.126376         26.126376              25.7
353    35.593434         35.593434              31.3
354    31.371807         31.371807              30.1
..           ...               ...               ...
409    16.127456         16.127456              15.4
410    50.398930         50.398930              50.0
411    47.271436         47.271436              40.6
412    46.196965         46.196965              52.5
413    53.088885         53.088885              63.9

[64 rows x 3 columns]
```

In [ ]:

```
# Caculate Sum of Squared Errors
sse = np.sum((y_test - y_pred) ** 2)
print(sse)
```

```
3978.2585329312396
```