Introduction to Machine Learning
HW3: Simple regression method & Tuning model complexity

Implementation

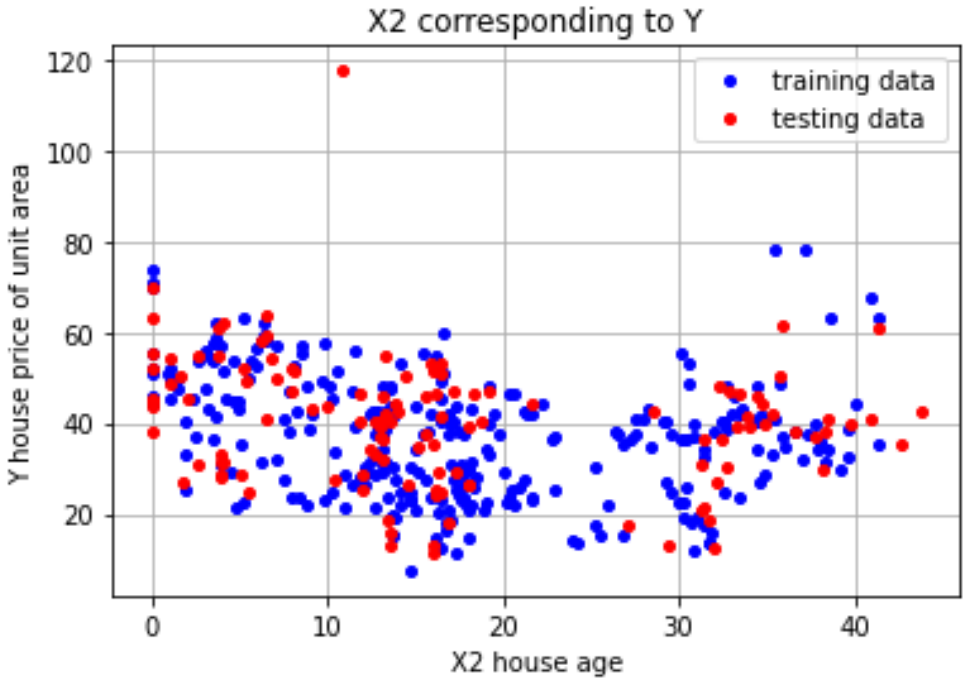
1. Dataset Loading & splitting:

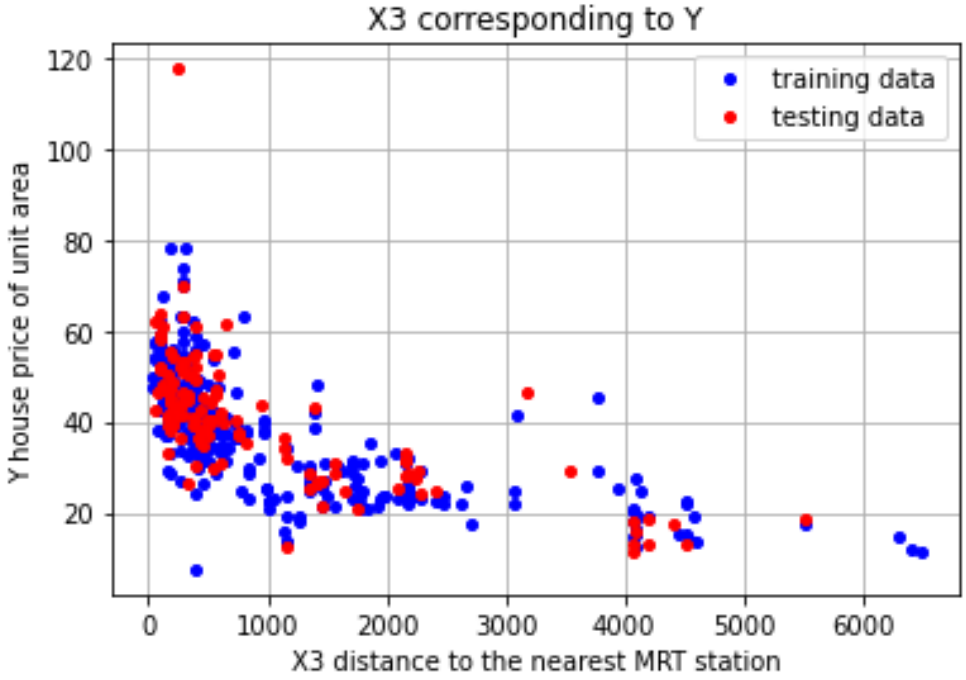
   下圖是隨機分割資料的副程式，將原本的 data 分成用來訓練模型的 data_train 和用來測試模型的 data_test，而 train_frac 設成 0.7、test_frac 設成 0.3。

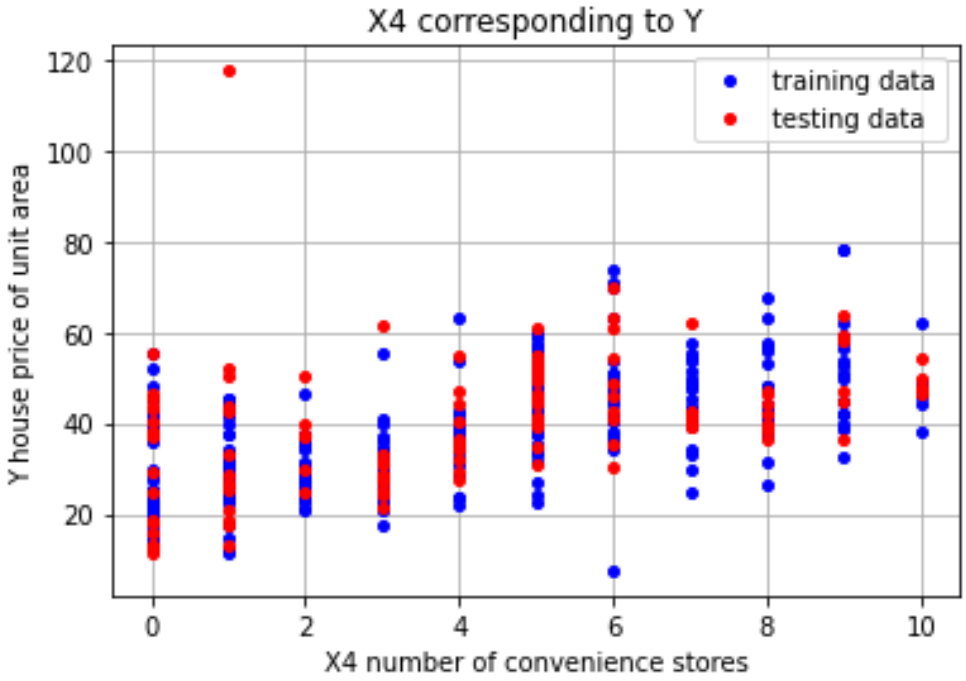
```
#%% 1
# Dataset Loading & splitting
def random_split_train_test(data, train_frac, test_frac):
    index = np.arange(0, len(data), 1)
    random.shuffle(index)

    train_number = round(len(data) * train_frac)
    test_number = round(len(data) * test_frac)
    index_train = index[:train_number]
    index_test = index[-test_number:]
    data_train = pd.DataFrame(np.zeros((train_number, data.shape[1])))
    data_test = pd.DataFrame(np.zeros((test_number, data.shape[1])))

    for i in range(data_train.shape[0]):
        for column in range(data_train.shape[1]):
            data_train[column][i] = data[column][index_train[i]]
    for i in range(data_test.shape[0]):
        for column in range(data_test.shape[1]):
            data_test[column][i] = data[column][index_test[i]]

    data_train = data_train.sort_values(by=0, ignore_index=True)
    data_test = data_test.sort_values(by=0, ignore_index=True)
    return data_train, data_test

csv = pd.read_csv("./Real_estate.csv")
data = pd.read_csv("./Real_estate.csv", header=None, skiprows=1)
data_train, data_test = random_split_train_test(data, 0.7, 0.3)
```

data	DataFrame	(414, 8)	Column names: 0, 1, 2, 3, 4, 5, 6, 7
data_test	DataFrame	(124, 8)	Column names: 0, 1, 2, 3, 4, 5, 6, 7
data_train	DataFrame	(290, 8)	Column names: 0, 1, 2, 3, 4, 5, 6, 7

2. Plot scatter figure including training data and testing data:


X2 corresponding to Y
training data
testing data
Y house price of unit area
X2 house age
0
10
20
30
40
20
40
60
80
100
120


X3 corresponding to Y
training data
testing data
Y house price of unit area
X3 distance to the nearest MRT station
0
1000
2000
3000
4000
5000
6000
20
40
60
80
100
120



3. Define loss function (Mean Square Error):

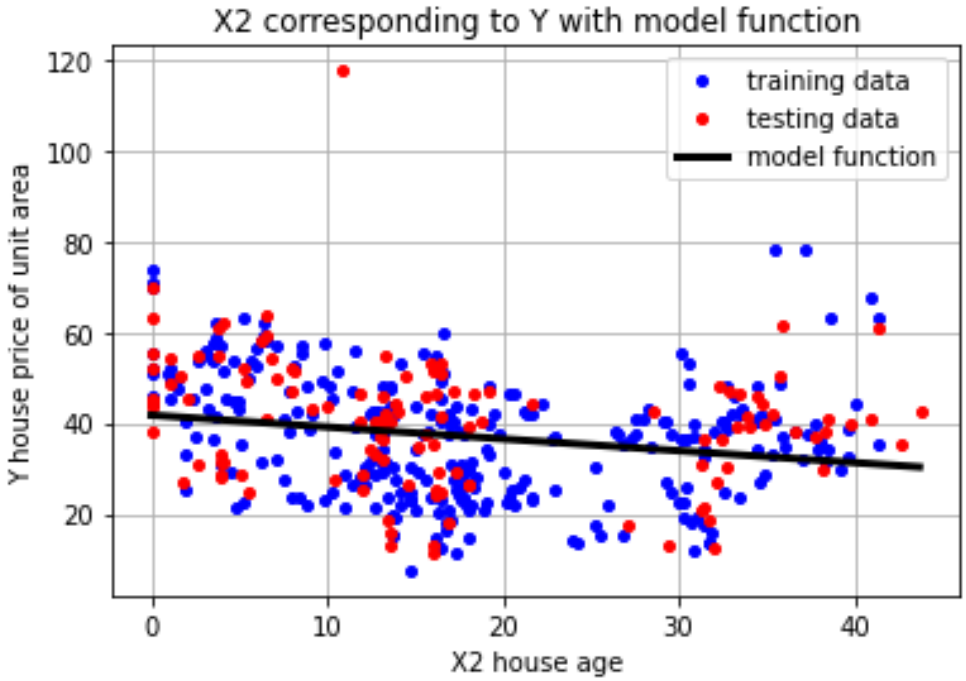
   此為找出 loss function 的副程式，train_pred、test_pred 是用模型的參數所算出來的 Y 值，分別跟真實的 Y_train、Y_test 做 MSE，得出 train_loss 和 test_loss。

```
#%% 3
# Define loss function (Mean Square Error)
def LossFunction(Y_train, Y_test, train_pred, test_pred):
    train_loss = 0.0
    test_loss = 0.0
    for i in range(len(Y_train)):
        train_loss = train_loss + (Y_train[i] - train_pred[i])**2
    for i in range(len(Y_test)):
        test_loss = test_loss + (Y_test[i] - test_pred[i])**2
    train_loss = train_loss / len(Y_train)
    test_loss = test_loss / len(Y_test)
    return train_loss, test_loss

```

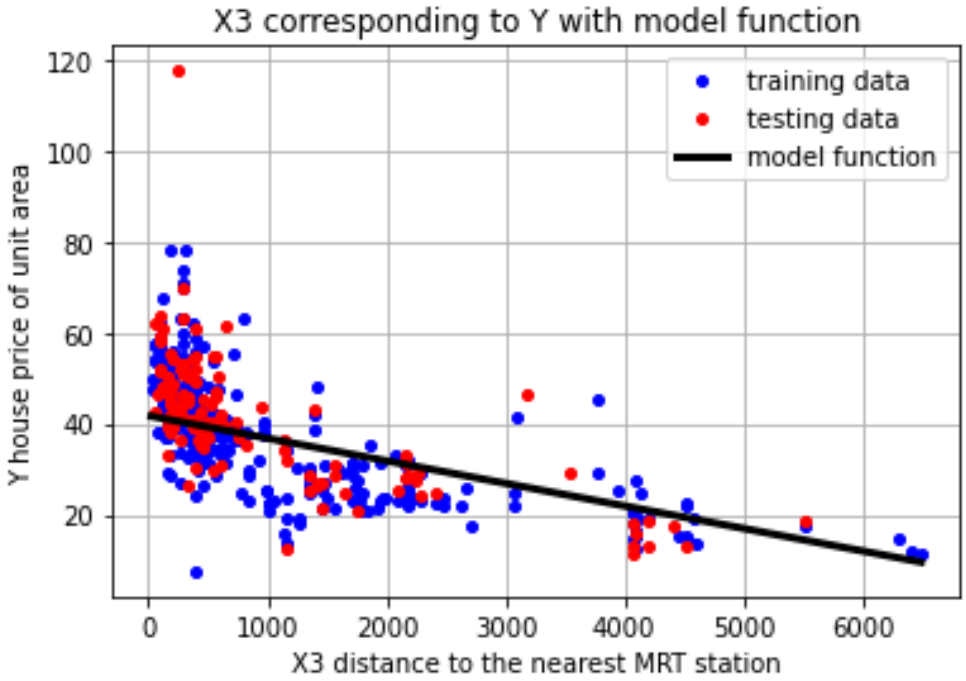
4. Using gradient method:

β_0	β_1	β_2	β_3
41.7414	-0.2609	-0.0049	1.4165

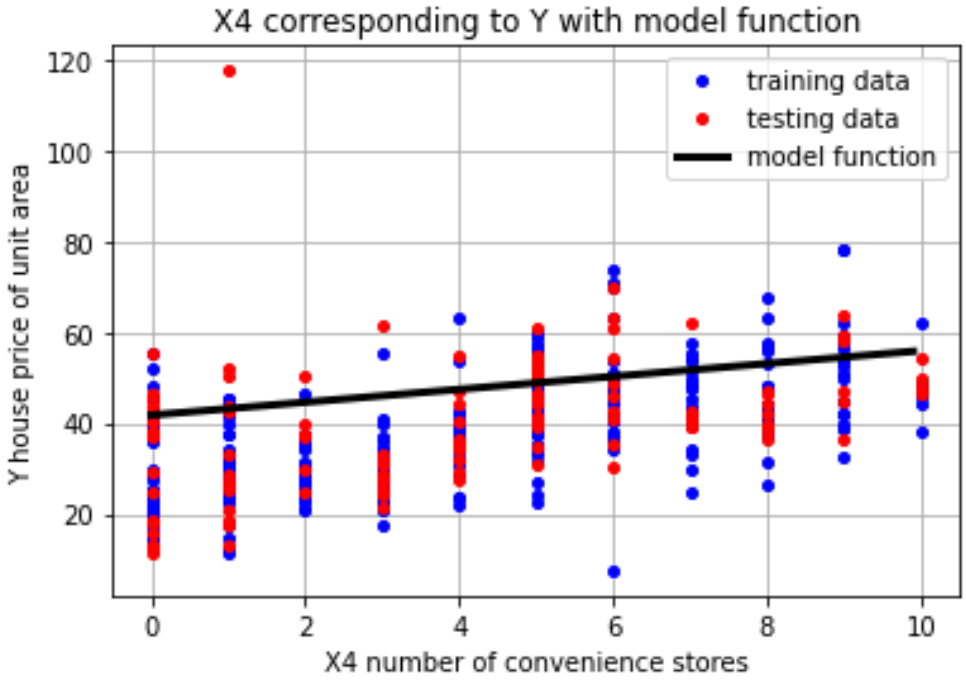
下圖為 X_2 和 Y 的關係圖，而 model function：$Y = \beta_0 + \beta_1 X_2 = 41.47 - 0.26X$，此時設 X_3, X_4 為 0。由圖可以看出 data 和 model function 的趨勢大致相同。



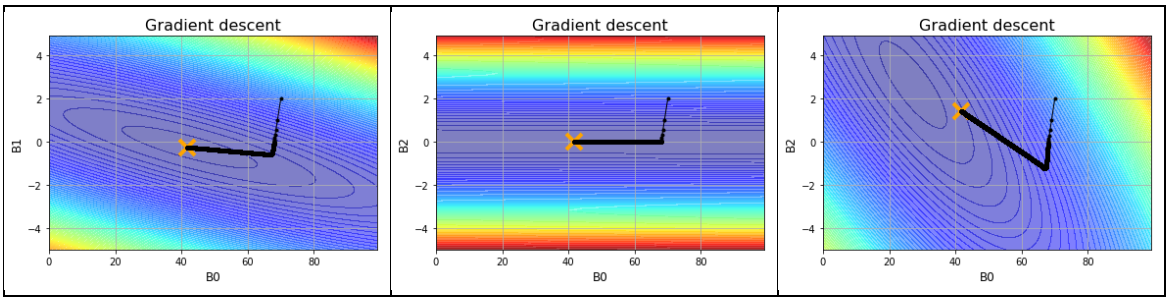
下圖為 X_3 和 Y 的關係圖，而 model function：$Y = \beta_0 + \beta_2 X_3 = 41.47 - 0.005X$，此時設 X_2, X_4 為 0。由圖可以看出 data 和 model function 的趨勢有些不同，在 X_3=0~1000 的區間有很多 data 落在 model function 之上。



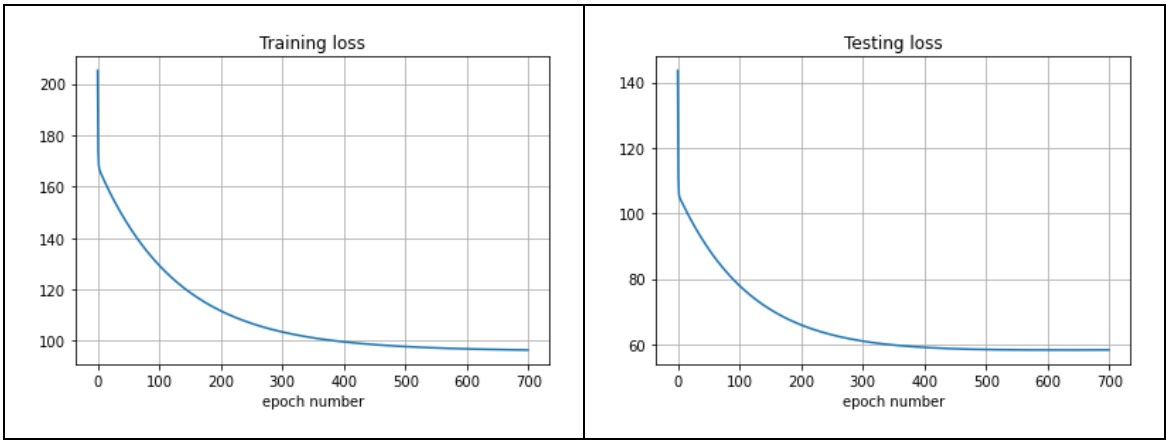
下圖為 X_4 和 Y 的關係圖，而 model function：$Y = \beta_0 + \beta_3 X_4 = 41.47 + 1.42X$，此時設 X_2, X_3 為 0。由圖可以看出 data 和 model function 的趨勢大致相同。



補充一：β_0 對 β_1、β_0 對 β_2、β_0 對 β_3 的 gradient descent



補充二：Training loss 和 Testing loss 的趨勢



5. Using least square method:

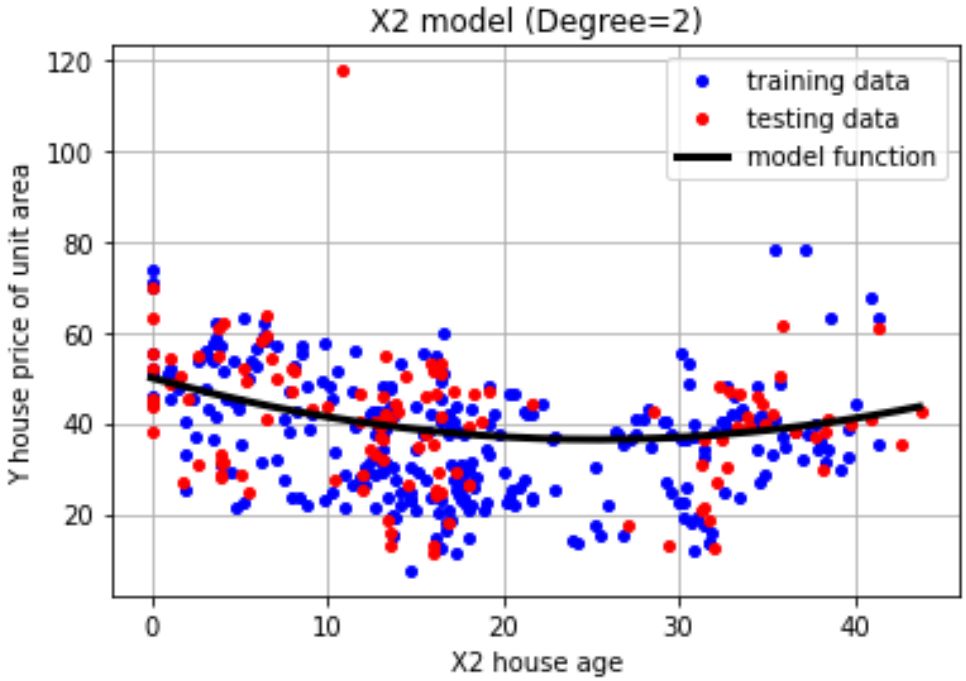
下表列出 1 種 gradient descent 模型和 8 種 least square 模型的所有參數。其中可

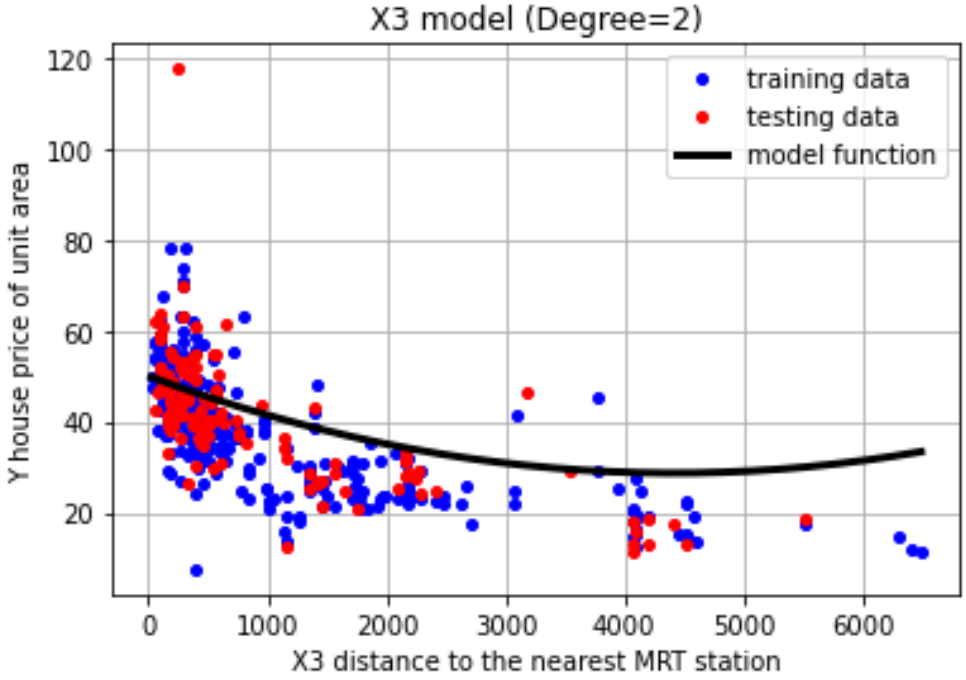
看出 gradient descent 的($\beta_0, \beta_1, \beta_2, \beta_3$)和 least square 的($\beta_0', \beta_1', \beta_2', \beta_3'$)數值相近但不相等，我認為原因在於我在做 gradient descent 時 iteration 的次數不夠，或者是一開始設定($\beta_0, \beta_1, \beta_2, \beta_3$)的初始值離實際值太遠，導致其跑完 iteration 後還是沒有到達最佳值，由上圖的補充一、補充二也可看出端倪。

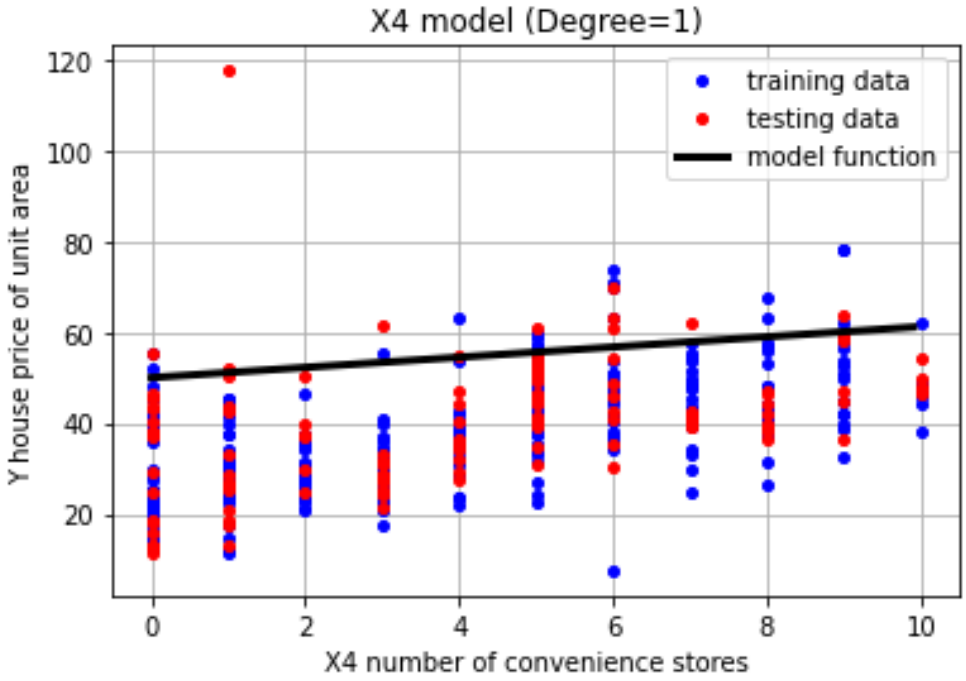
β_0	β_1	β_2	β_3	-	-	-
41.7414	-0.2609	-0.0049	1.4165	-	-	-
β_0'	β_1'	β_2'	β_3'	-	-	-
39.3145	-0.2282	-0.0044	1.6694	-	-	-
β_4'	β_5'	β_6'	β_7'	β_8'	-	-
45.8939	-1.1951	0.0244	-0.0039	1.5109	-	-
β_9'	β_{10}'	β_{11}'	β_{12}'	β_{13}'	-	-
45.1738	-0.2333	-0.0112	1.2953e-6	1.1839	-	-
β_{14}'	β_{15}'	β_{16}'	β_{17}'	β_{18}'	-	-
40.7432	-0.2350	-0.0046	0.8027	0.0957	-	-
β_{19}'	β_{20}'	β_{21}'	β_{22}'	β_{23}'	β_{24}'	-
49.9832	-1.0744	0.0212	-0.0096	1.0921e-6	1.1221	-
β_{25}'	β_{26}'	β_{27}'	β_{28}'	β_{29}'	β_{30}'	-
46.7385	-1.1838	0.0240	-0.0040	0.9357	0.0638	-
β_{31}'	β_{32}'	β_{33}'	β_{34}'	β_{35}'	β_{36}'	-
45.9694	-0.2376	-0.0111	1.2693e-6	0.6396	0.0612	-
β_{37}'	β_{38}'	β_{39}'	β_{40}'	β_{41}'	β_{42}'	β_{43}'
50.4377	-1.0692	0.0210	-0.0096	1.0774e-6	0.7808	0.0384

下表列出 8 種 least square 模型的 training loss 和 testing loss。可看出 Model 5 的 training loss 或 testing loss 都比較小，也就是有較好的準確率。

Model	1	2	3	4	5	6	7	8
Training loss	73.79	66.12	67.26	73.24	61.61	65.87	67.04	61.52
Testing loss	115.87	113.80	100.67	115.92	101.12	113.72	101.18	101.34


X2 model (Degree=2)
training data
testing data
model function
Y house price of unit area
X2 house age
0
10
20
30
40
20
40
60
80
100
120


X3 model (Degree=2)
training data
testing data
model function
Y house price of unit area
X3 distance to the nearest MRT station
0
1000
2000
3000
4000
5000
6000
20
40
60
80
100
120


X4 model (Degree=1)
training data
testing data
model function
Y house price of unit area
X4 number of convenience stores
120
100
80
60
40
20
0
2
4
6
8
10