Multiple Linear regression

จากที่เราได้เรียนรู้ทฤษฎีการประมาณการแบบ multiple linear regression ไปแล้วว่ามีที่มาและการหาค่า coefficient หรือค่า Theta อย่างไร ต่อมาเราจะสอนเพื่อนๆเขียน code ในการประมาณการ โดยเราจะใช้ code จาก 2 library คือ

- Scikit-learn
- Statsmodels

โดยวิธีการประมาณการหลังบ้านจะเหมือนกัน แต่ว่าผลที่ได้นั้นก็ขึ้นอยู่กับว่าเราต้องการข้อมูลอะไร หรือต้องการให้แสดงผลอย่างไรบ้าง พร้อมทั้งเราจะสอนเพื่อนๆตีความผลที่ได้โดยใช้หลักทางสถิติเข้ามาอธิบาย ซึ่งมีขั้นตอนต่างๆ ดังนี้

1. การเตรียมข้อมูลก่อนทำการประมาณการ
2. Sklearn: Multiple linear regression
3. Statsmodels: Multiple Linear regression
4. การตีความโดยใช้หลักทางสถิติ

1. การเตรียมข้อมูลก่อนทำการประมาณการ

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

- **import pandas as pd** เป็นคำสั่ง เรียกใช้ library pandas โดยเวลาเรารันคำสั่งสามารถเขียนตัวย่อว่า pd ได้ ซึ่ง pandas นำมาใช้ในการเรียกอ่านไฟล์ข้อมูลของเรา
- **import seaborn as sns** เป็น library ที่ใช้ในการ plot graph
- **import matplotlib.pyplot as plt** เป็น library ที่ใช้ในการ plot graph เหมือน seaborn

In [2]:
```
print(f'pandas  version: {pd.__version__}')
print(f'seaborn version: {sns.__version__}')
```

```
pandas  version: 0.24.2
seaborn version: 0.9.0
```

- ข้างบนเป็นคำสั่งเพื่อให้คอมพิวเตอร์ตรวจสอบว่า library ที่เราเลือกมานั้นเป็น version อะไร ซึ่งที่เราใช้ก็จะเป็น pandas version: 0.24.2 seaborn version: 0.9.0

In [3]:
```
df=pd.read_csv('C:\\Users\\LENOVO\\Desktop\\files for example\\kc_house_test_data.csv')
df.head()
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	w
0	114101516	20140528T000000	310000.0	3	1.0	1430	19901	1.5	
1	9297300055	20150124T000000	650000.0	4	3.0	2950	5000	2.0	
2	1202000200	20141103T000000	233000.0	3	2.0	1710	4697	1.5	
3	8562750320	20141110T000000	580500.0	3	2.5	2320	3980	2.0	
4	7589200193	20141110T000000	535000.0	3	1.0	1090	3000	1.5	

5 rows × 21 columns

- **ต่อมาเราก็สร้าง object เพื่อบรรจุข้อมูลของเรา** ชื่อว่า df จากนั้น ก็เรียกกคำสั่ง pd.read_csv เพื่อให้มันอ่านข้อมูลที่เป็นไฟล์ csv จากนั้นเราก็ใส่ที่อยู่ข้อมูลของเราลงไป
- **df.head()** คือคำสั่งที่ให้มันแสดงข้อมูล object ของเราโดยอ่านแค่ตรงหัว (5แถวแรก)

In [4]:
```
df.shape
```

Out[4]: (4222, 21)

- **df.shape** เป็นคำสั่งให้มันแสดงข้อมูลทั้งหมดว่ามีกี่แถวและกี่คอลัมน์ ในที่นี้มีทั้งหมด 4222 แถว และ 21 คอลัมน์

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

- **df.columns** เป็นคำสั่งเรียกดูคอลัมน์ใน object หรือคอลัมน์ในข้อมูลของเรานั่นเอง

In [6]:
```
df.info()
```

```
bedrooms         4222 non-null int64
bathrooms        4222 non-null float64
sqft_living      4222 non-null int64
sqft_lot         4222 non-null int64
floors           4222 non-null float64
waterfront       4222 non-null int64
view             4222 non-null int64
condition        4222 non-null int64
grade            4222 non-null int64
sqft_above       4222 non-null int64
sqft_basement    4222 non-null int64
yr_built         4222 non-null int64
yr_renovated     4222 non-null int64
zipcode          4222 non-null int64
lat              4222 non-null float64
long             4222 non-null float64
sqft_living15    4222 non-null int64
sqft_lot15       4222 non-null int64
dtypes: float64(5), int64(15), object(1)
memory usage: 692.8+ KB
```

- **df.info()** เป็นคำสั่งที่ใช้เรียกดูว่า ข้อมูลในแต่ละคอลัมน์ของเรานั้นเป็นลักษณะอย่างไร อันนี้ถือว่าสำคัญมากในการประมาณการข้อมูล อันแรกที่เราจำเป็นต้องดูคือลักษณะของข้อมูล (dtypes)ในบรรทัดรองสุดท้าย ซึ่งมีทั้งหมด 3 แบบ (รูปแบบของข้อมูลในคอมพิวเตอร์มีหลายแบบ เช่น boolean , integer,floating point string เป็นต้น แต่ข้อมูลของเรามี 3 แบบเท่านั้น) คือ
  - **Float64(5)** นั่นคือมีคอลัมน์ที่เป็นจำนวนทศนิยม อยู่ 5 คอลัมน์
  - **int64(15)** มีคอลัมน์ที่เป็นจำนวนเต็มอยู่ 15 คอลัมน์
  - **object (1)** คือมีข้อมูลหลายๆ character ในที่นี้ช่อง date ของเราเป็น object เพราะมันรวมกันของ float+string

ซึ่งการประมาณการข้อมูลนั้นหากมีข้อมูลที่ไม่ใช่ float64 และ int64 จะไม่สามรถประมาณการได้ ดังนั้นเราจำเป็นที่จะต้องตัดข้อมูลที่เป็น object ออก

In [7]:
```
df2=df.drop(['id', 'date',  'condition', 'yr_built', 'yr_renovated', 'zipcode', 'long', 'sqft_living15', 'sqft
df2.head()
```

Out[7]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	grade	sqft_ab
0	310000.0	3	1.0	1430	19901	1.5	0	0	7	1
1	650000.0	4	3.0	2950	5000	2.0	0	3	9	1
2	233000.0	3	2.0	1710	4697	1.5	0	0	6	1
3	580500.0	3	2.5	2320	3980	2.0	0	0	8	2
4	535000.0	3	1.0	1090	3000	1.5	0	0	8	1

- **ต่อมาเราตั้ง object ใหม่** ชื่อว่า df2 โดย df2 เป็น dataframe ใหม่ที่เราจะตัดเอาคอลัมน์ที่เราไม่ต้องการออก คำสั่งที่เราใช้คือ df.drop([ใส่ชื่อคอลัมน์],แกนที่จะตัดคือคอมลัมน์ แทนด้วยเลข 1 ถ้าเป็น 0 คือแถว) ในที่นี้เราตัดอออกเยอะหน่อยเพื่อให้สอดคล้องกับทฤษฎีที่เราสอนไปก่อนหน้านั้น

In [8]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4222 entries, 0 to 4221
Data columns (total 11 columns):
price          4222 non-null float64
bedrooms       4222 non-null int64
bathrooms      4222 non-null float64
sqft_living    4222 non-null int64
sqft_lot       4222 non-null int64
floors         4222 non-null float64
waterfront     4222 non-null int64
view           4222 non-null int64
grade          4222 non-null int64
sqft_above     4222 non-null int64
sqft_basement  4222 non-null int64
dtypes: float64(3), int64(8)
memory usage: 362.9 KB
```

- **สุดท้ายเราก็ตรวจสอบข้อมูล** ว่ามีความสมบูรณ์พอที่จะนำไปทำการประมาณการแล้วหรือยังจากข้อมูลจะเห็นได้ว่าไม่มี object character และค่า missing value (non-null คือไม่มีช่องว่าในแต่ละแถว ทุกช่องมีค่าของมันเอง) ข้อมูลของเราก็พร้อมที่จะนำไปทำการประมาณการแล้ว เย่!

2. Sklearn: Multiple linear regression

ต่อมาจะเป็นการประมาณการโดยใช้ Sckit-learn

In [9]:
```
from sklearn.linear_model import LinearRegression
```

- คำสั่งแรกเป็นการ **เรียก library Scikit-learn** และการเรียกใช้คำสั่งการประมาณการแบบ Linear regression

In [10]:
```
model = LinearRegression()
model
```

Out[10]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

- ต่อมาเรา **สร้าง object ชื่อว่า model** และใส่คำสั่งให้มันประมาณการแบบ linear regression แต่ว่าเรายังไม่ได้กำหนดค่า X และ Y ดังนั้นค่าที่ออกมาจึงเป็น defult

Traning set

In [11]:
```
X=df2.drop(columns=['price'])[:2200]
Y=df2['price'][:2200]
```

เมื่อเรามีการเรียกใช้คำสั่งการประมาณการแล้ว ต่อมา เราจะมากำหนดตัวแปร ว่าให้คอลัมน์ไหนเป็นตัวแปรอิสระ(x) และคอลัมน์ไหนเป็นตัวแปรตาม(y)

โดยเราได้มีการกำหนดให้

- **X เป็น object** ที่ข้างในจะมีคอลัมน์ที่เรากำหนดให้เป็นตัวแปรอิสระทั้งหมด 10 ตัว ยกเว้นคอลัมน์ price (X ประกอบไปด้วย คอลัมน์ 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors','waterfront', 'view', 'grade', 'sqft_above', 'sqft_basement')
- **Y เป็น object** เป็นตัวแปรที่เราต้องการทราบผล(ตัวแปรตาม) นั่นก็คือ ราคาบ้าน (ในที่นี้เราต้องการประมาณการว่า ตัวแปรอิสระ 10 ตัวนั้น ส่งผลต่อราคาบ้านหรือไม่ อย่างไร)

เพิ่มเติม :

- **[:2200]** เป็นคำสั่งที่เราบอกให้คอมพิวเตอร์ประมาณการข้อมูลตั้งแต่ตัวที่แถวที่ 0 ไปจนถึงแถวที่ 2200

In [12]:
```
X.head()
```

Out[12]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	grade	sqft_above	sqft_l
0	3	1.0	1430	19901	1.5	0	0	7	1430	
1	4	3.0	2950	5000	2.0	0	3	9	1980	
2	3	2.0	1710	4697	1.5	0	0	6	1710	
3	3	2.5	2320	3980	2.0	0	0	8	2320	
4	3	1.0	1090	3000	1.5	0	0	8	1090	

In [13]:
```
Y.head()
```

Out[13]:
```
0    310000.0
1    650000.0
2    233000.0
3    580500.0
4    535000.0
Name: price, dtype: float64
```

In [14]:
```
model.fit(X, Y)
```

Out[14]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

- **model.fit(X, Y)** เป็นการสั่งให้คอมพิวเตอร์ประมาณการข้อมูลออกมา

In [15]:
```
model.score(X,Y)
```

Out[15]:
```
0.5949154504578794
```

- **model.score(X, y)** เป็นคำสั่งดูค่า R-squared เป็นค่าที่บอกว่า Model multiple linear regression ที่เราใช้นี้มีความเหมาะสมกับข้อมูลไหม เดี๋ยวเราจะไปขยายผลในตอนท้ายอีกทีนะคะ

In [16]:
```
model.intercept_
```

Out[16]:
```
-444727.062095869
```

- **model.intercept_** คือคำสั่งดูค่า theta_0

In [17]:
```
model.coef_
```

Out[17]:
```
array([-1.55148181e+04, -2.98758130e+04,  1.04157719e+02, -2.37056771e-01,
        1.46689627e+04,  4.44280514e+05,  6.79949266e+04,  9.94681879e+04,
        3.35079583e+01,  7.06497604e+01])
```

- **model.coef_** เป็นการดูค่า coefficient หรือค่า Theta ของตัวแปร x ทั้งหมด 10 ตัว

Testing set

ต่อมาเมื่อเราลองประมาณการข้อมูล training set แล้ว เราจะได้สมการประมาณการออกมา 1 สมการนั่นคือ

Y = -444727.062 -1.55148181e+04 X(1) + (-2.98758130e+04) X(2) + 1.04157719e+02 X(3) -2.37056771e-01 X(4) + 1.46689627e+04 X(5) + 4.44280514e+05 X(6) + 6.79949266e+04 X(7) + 9.94681879e+04 X(8) + 3.35079583e+01 X(9) + 7.06497604e+01 (10)

ซึ่งเราจะนำสมการด้านบนที่ได้ ไปประมาณการข้อมูล Testing set ที่เหลือ

In [18]:
```
X_test=df2.drop(columns=['price'])[2201:]
X_test.head()
```

Out[18]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	grade	sqft_above	sc
2201	2	2.50	1390	1132	2.0	0	0	8	1130	
2202	2	1.50	1360	1934	2.0	0	0	7	1360	
2203	3	2.25	2260	54014	1.0	0	0	7	1450	
2204	4	2.50	4070	129808	2.0	0	0	10	4070	
2205	5	4.50	4400	15580	2.0	0	0	11	3390	

- เราก็ทำเหมือนขั้นตอนแรก คือการกำหนดตัวแปร ในที่นี้เราสร้าง object ขึ้นมาใหม่ ให้ชื่อว่า X_test ซึ่งเราก็ให้คอลัมน์ทั้งหมดเป็นตัวแปร X และให้ตัดคอลัมน์ price ออกเหมือนเดิม ต่างกันตรงที่เราให้มันประมาณการณ์ตั้งแต่แถวที่ 2201 ไปจนถึงแถวสุดท้าย

In [19]:
```
Y_hat=model.predict(X_test)
Y_hat
```

Out[19]:
```
array([475381.0091982 , 391811.67592343, 480859.08765269, ...,
       688565.46269421, 560237.26451124, 367550.6466532 ])
```

- และเราก็สร้าง object ที่เราต้องการทราบค่าและตั้งชื่อ object นี้ว่า Y_hat จากนั้นก็ใช้คำสั่ง model.predict(X_test) เป็นคำสั่งที่เราจะให้ใช้สมการที่ได้จากการประมาณการ training set ให้นำสมการที่ได้มาประมาณการข้อมูล X_test เพื่อที่จะดูว่า ค่า Y_hat ที่ได้ มีค่าใกล้เคียงกับค่า Y จริงมากน้อยแค่ไหน หรืออีกความหมายคือ สมการที่ได้นั้นมีความแม่นยำกับข้อมูลหรือไม่
- ซึ่งค่า Y_hat ที่ได้นั้นจะอยู่ในรูปของ array

In [21]:
```
dc=pd.concat([df2[2200:].reset_index(), pd.Series(Y_hat, name='predicted')], axis='columns')
dc.head()
```

Out[21]:

	index	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	grade
0	2200	246900.0	3	1.50	1370	9800	1.0	0	0	7
1	2201	635000.0	2	2.50	1390	1132	2.0	0	0	8
2	2202	150000.0	2	1.50	1360	1934	2.0	0	0	7
3	2203	514000.0	3	2.25	2260	54014	1.0	0	0	7
4	2204	710000.0	4	2.50	4070	129808	2.0	0	0	10

- **หากเราต้องการที่จะรวมตาราง** เพื่อที่จะได้ดูเปรียบเทียบกันระหว่างค่าที่ได้จากการประมาณการ(คอลัมน์ predicted) และค่า Yจริง (คอลัมน์ price) เราสามารถใช้คำสั่ง pd.concat ด้านบนได้ **"หากค่าทั้งสองมีความใกล้เคียงกัน แสดงว่าสมการที่เราใช้นั้นมีความแม่นยำกับข้อมูล"**
- โดยเราตั้ง object ว่า dc และใส่คำสั่งรวมตารางคือ pd.concat

In [22]:
```
model.predict([[200, 40, 70,70,66,50,77,40,90,30]])
```

Out[22]:
```
array([27666200.5590749])
```

- หรือหากเราต้องการที่จะลองกำหนดค่าตัวแปรเองก็สามารถทำได้ โดยการใส่ตัวเลขของแต่ละ feature(แต่ละตัวแปรอิสระ) ลงไปให้ครบ คำตอบที่ได้คือ Y =27666200.5590749

3. Statsmodels: Multiple linear regression

ต่อมาเราจะใช้ library Statsmodels ในการประมาณการ ที่เราเลือกใช้ตัวนี้เนื่องจากว่ามันจะแสดงผลทางสถิติครบเลย และขั้นตอนไม่ยุ่งยากมาก

In [23]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

- เรียกใช้ library Statsmodels และสามารถเขียนตัวย่อได้ว่า sm
- เรียกใช้คำสั่งในการเขียนสูตร และสามารถเขียนตัวย่อได้ว่า smf

In [26]:
```
model_a = smf.ols(formula='price ~ + bedrooms + bathrooms + sqft_living + sqft_lot + floors +
```

- **เราสามารถทำการประมาณการได้เพียงแค่ code 1 บรรทัดเท่านั้น** ง่ายๆ โดย ตั้ง object ว่า model_a จากนั้นก็เรียกคำสั่งเขียน code ว่า smf.ols จากนั้นก็เขียนสูตรลงไปได้เลย
- **data = df2[:2201]** เป็นการประมาณการโดยใช้ข้อมูล Testing set
- **.fit()** คำสั่งให้ทำการประมาณการ
- **print(model_a.summary())** คำสั่งแสดงผล จะได้ผลออกมาดังตารางด้านล่างนี้

In [27]:
```
print(model_a.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.595
Model:                            OLS   Adj. R-squared:                  0.593
Method:                 Least Squares   F-statistic:                     357.7
Date:                Tue, 21 Jan 2020   Prob (F-statistic):               0.00
Time:                        15:45:17   Log-Likelihood:                -29931.
No. Observations:                2201   AIC:                         5.988e+04
Df Residuals:                    2191   BIC:                         5.994e+04
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -4.449e+05   3.79e+04    -11.734      0.000   -5.19e+05   -3.71e+05
bedrooms    -1.553e+04   5738.222     -2.706      0.007   -2.68e+04   -4272.265
bathrooms   -2.987e+04   9359.673     -3.192      0.001   -4.82e+04   -1.15e+04
sqft_living   104.2042      6.567     15.867      0.000      91.326     117.083
sqft_lot       -0.2368      0.110     -2.161      0.031      -0.452      -0.022
floors       1.479e+04   1.11e+04      1.330      0.184   -7010.939    3.66e+04
waterfront    4.443e+05   5.32e+04      8.347      0.000     3.4e+05    5.49e+05
view          6.799e+04   6120.825     11.109      0.000     5.6e+04       8e+04
grade         9.946e+04   6064.534     16.400      0.000    8.76e+04    1.11e+05
sqft_above      33.4704      6.490      5.158      0.000      20.744      46.197
sqft_basement   70.7338      7.831      9.032      0.000      55.376      86.091
==============================================================================
Omnibus:                      662.290   Durbin-Watson:                   1.961
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3755.254
Skew:                           1.299   Prob(JB):                         0.00
Kurtosis:                       8.848   Cond. No.                     3.22e+17
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 3.91e-23. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

4.การตีความโดยใช้หลักทางสถิติ

ในขั้นตอนสุดท้ายเราจะมาดูว่า สมการที่เราใช้ประมารการนั้นมีความแม่นยำกับข้อมูลของเราหรือไม่ ซึ่งเราสามารถดูจากตารางสถิตินี้ได้ ซึ่งตัวหลักๆที่เราต้องดูมีดังนี้

- **1. R-squared** เป็นค่าที่ใช้บ่งบอกว่า สมการและตัวแปรอิสระทั้งหมด 10 ตัวที่เราใช้ สามารถใช้อธิบายการเปลี่ยนแปลงในตัวแปร Y ได้มากน้อยแค่ไหน ทั้งนี้การที่ข้อมูลของเราได้ค่า R-squared = 0.595 ถือว่ากลางๆ (ค่ายิ่งสูงยิ่งดี โดยค่า R- squared จะอยู่ระหว่าง 0<R<1) สาเหตุที่ทำให้ได้ค่านี้อาจจะเป็นเพราะว่า อาจจะมีตัวแปรอิสระบางตัวที่ไม่ได้มีความสัมพันธ์กับ Y หรืออาจจะมีความสัมพันธ์กับ Y แบบที่ไม่ใช่เส้นตรง

- **2. Adjust. R - squared** เป็นการดูว่า model ที่เราใช้เหมาะสมกับข้อมูลจริง หรือเป็นเพราะตัวแปร x ที่เพิ่มขึ้น จึงทำให้ค่า R-squared นั้นสูง(โดยปกติแล้ว ค่า R-squared จะมีค่าสูงขึ้นหรือเท่าเดิม เมื่อมีตัวแปร x หรือตัวแปรอิสระเพิ่มขึ้น แต่ค่า Adj.R-squaresd จะเพิ่มขึ้นก็ต่อเมื่อตัวแปรอิสระที่เพิ่มเข้ามานั้นอธิบายตัวแปร Y ได้ดีขึ้น)
  ดังนั้น วิธีการของ Adj. R-squared คือ การนำตัวแปรอิสระ x ออก 1 ตัว แล้วทำการประมาณค่าอีกรอบ หากค่า Adj.R-squared มีค่าต่างจาก R-squared มากๆ อาจจะเป็นไปได้ว่า เราใส่ตัวแปรที่สามารถอธิบายการเปลี่ยนแปลง Y น้อย หรืออาจจะไม่สามารถอิธบายการเปลี่ยนแปลงของ Y ได้เลย คือ ประมาณว่าเราใส่ตัวแปรเกินจำเป็นนั่นเอง

- **F-statistic** เป็นค่าที่จะนำไปทดสอบสมมติฐาน ซึ่งหากผลออกมาว่า significant (significat คือการที่ค่า F-statistic เมื่อนำไปคำนวณแล้วค่า F ที่ได้ *มากกว่า* ค่า F-statistic ณ ระดับนัยสำคัญที่เรากำหนด ในที่นี้เรากำหนดไว้ที่ 0.05 ซึ่งเป็นที่นิยมโดยทั่วไป) **แสดงว่า มี x อย่างน้อย 1 ตัวที่มีความสัมพันธ์กับ Y** ยิ่ง F-statistic มีค่าสูง>>มีโอกาส significant มาก

- **Prob(F-statistic)** เป็นค่า probability ของ F-statistic หากเราไม่ต้องการคำนวณค่า F-statistic แล้วนำไปเทียบกับค่า F-statistic ณ ระดับนัยสำคัญให้ยุ่งยาก เราสามารถที่จะดูค่า probability ได้ ซึ่งหากค่าที่ได้คือ 0.000 แสดงว่า significant ทุกระดับนัยสำคัญ(ค่า Prob จะดูว่า significant หรือไม่ ดูได้จาก ถ้า Prob < ระดับนัยสำคัญ = significant ซึ่งในที่นี้ 0.000 < 0.05)

- **t, P>l t l ,confident interval [0.025 0.975]** เป็นค่าที่ใช้บอกว่า ตัวแปรอิสระนั้นๆ มีความสัมพันธ์กับ Y หรือไม่(ดูเจาะลึกลงไปทีละตัวแปรเลย)ซึ่งในที่นี้เรามีการกำหนดระดับนัยสำคัญไว้ที่ 0.05 นั่นคือ หากค่าของตัวแปรใดในช่อง P-value มีค่ามากกว่า 0.05 แสดงว่าตัวแปรนั้นไม่มีความสัมพันธ์กับตัวแปร Y อย่างมีนัยสำคัญทางสถิติ ซึ่งจากเราดูจะเห็นได้ว่า ตัวแปร floors นั้นมีค่า coefficient หรือค่า Theta เท่ากับ 0.187 และค่า confident interval มีค่าคร่อม 0 จึงสรุปได้ว่า ตัวแปร Floors ไม่มีความสัมพันธ์กันกับตัวแปรตาม Y อย่างมีนัยสำคัญทางสถิติ

ทำอย่างไรถึงจะได้สมการที่เหมาะสมกับข้อมูลมากที่สุด?

หากเราต้องการประมาณการข้อมูลใดๆ ให้แม่นยำที่สุดนั้น เราจำเป็นที่จะต้องระวังในหลายๆจุด ตั้งแต่การได้มาของข้อมูล ตลอดจนการตีความผลที่ได้ เพื่อไม่ให้เพื่อนๆสับสน เราขอแบ่งขั้นตอนการประมาณการออกเป็น 4 steps หลักๆแล้วกันนะคะ นั่นคือ

- การรวบรวมข้อมูล
- การ clean ข้อมูล

- การเลือก model
- การแก้ปัญหาเมื่อ model ไม่ fit

1. การรวบรวมข้อมูล

เราจำเป็นที่จะต้องดูให้ออกก่อนว่าข้อมูลที่เราได้มา หรือข้อมูลที่เราจะไปเก็บเพื่อนำมาทำการประมาณการ เป็นข้อมูลประเภทใด ซึ่งมันมีหลักการแบ่งมากมาย แต่เราจะขอยกตัวอย่างการแบ่งข้อมูลเพื่อใช้ในการวิเคราะห์ทางสถิติ ซึ่งแบ่งออกเป็น 3 แบบหลักๆ นั่นคือ

- **Cross section data** ภาษาไทยเรียกว่า ข้อมูลภาคตัดขวาง *เป็นข้อมูลที่เก็บ ณ ช่วงเวลาใดเวลาหนึ่ง* เช่น เป็นจำนวนบ้านในเมือง A ที่เก็บในวันที่ 20 เดือน 1 ปี 2020 หรือ ข้อมูล GDP ของประเทศต่างๆในปี 2020 เป็นต้น ซึ่งข้อมูลที่เราใช้ประมาณการที่ผ่านมาเป็นข้อมูลภาคตัดขวางนี้เอง

- **Time series data** หรือข้อมูลอนุกรมเวลา *เป็นข้อมูลที่มีเวลาเข้ามาเกี่ยวข้อง* เป็นข้อมูลที่เก็บเรียงตามเวลา เช่น ข้อมูลจำนวนบ้านในเมือง A เก็บตั้งแต่ปี 1995 ไปจนถึงปี 2020 หรือ ข้อมูล GDP ของประเทศไทยตั้งแต่ปี 1995-2020 เป็นต้น

- **Panel data** หรือข้อมูลผสม *เป็นข้อมูลที่ผสมระหว่าง Time series และ Cross section* เช่น ข้อมูลจำนวนบ้านในเมือง A และข้อมูลบ้านในเมือง B ในปี 1995-2020 หรือ ข้อมูล GDP ของทั้งโลกในปี 1995-2020 เป็นต้น

หากเราไม่สามารถแยกได้ว่าข้อมูลของเราเป็นแบบไหน จะส่งผลให้เราเลือกสมการไม่เหมาะสม รวมไปถึงการตีความผลที่ได้ก็จะไม่ถูกต้อง เพราะแต่ละแบบจะมีการตีความที่แตกต่างกันออกไป

2. การ clean ข้อมูล

ขั้นตอนนี้เป็นการดูว่าเมื่อเราได้ข้อมูลมาแล้วนั้น เราต้องมาดูว่าข้อมูลเรามีความสมบูรณ์มากแค่ไหน เราจะยกตัวอย่างคร่าวๆนะคะ

- **Missing value** แปลง่ายๆคือ ค่าที่หายไป เวลาเราได้ตารางข้อมูลมา เราจะมีคำสั่งเพื่อให้มันแสดงออกมาว่า แถวไหน ช่องไหน มีค่าที่หายไป หรือขาดหายไป เราจำเป็นที่จะต้องกำจัดข้อมูลเหล่านี้ออกเพราะมันเป็นข้อมูลที่ไม่มีคุณภาพ จะทำให้การประมาณการคลาดเคลื่อน แต่ก็มีข้อควรระวังนิดหน่อย เพราะว่าบางทีข้อมูล มี missing value ในตัวแปรอิสระ(x)ที่เราไม่ได้สนใจ แบบนี้สามารถเอามาใช้ได้

- **Outlier** หรือจุด High leverage เป็นจุดที่ค่า x มีค่าต่างจากข้อมูลมากๆ เช่น ข้อมูลของอยู่ใน range 1-10 แต่อยู่ดีๆอีกค่าโผล่มา 1000 แบบนี้ เมื่อเราทำการประมาณการ เส้น linear regression ของเรามันจะพยายามให้เส้นใกล้เคียงค่าทุกค่า ซึ่งสมการที่ได้จะมีความคลาดเคลื่อนสูงมาก

- **จำนวนข้อมูล** หรือข้อมูลใน Training set บางครั้ง modelของเราอาจจะเหมาะสมกับข้อมูลแล้วแหละ แต่พอสมการประมาณการออกมากลับไม่ fit กับข้อมูล อาจจะเป็นเพราะเรามีจำนวนตัวอย่างข้อมูลน้อยไป โดยส่วนใหญ่แล้ว การเพิ่มจำนวนข้อมูลนี้ถือเป็นการแก้ปัญหาในเรื่องที่ model ไม่ fit ในเบื้องต้นเลย

3. การเลือก model

คือการเลือกรูปแบบการประมาณการข้อมูล ทั้งขึ้นอยู่กับข้อมูลของเราด้วย หากเป็นข้อมูลที่ซับซ้อน เราอาจจะต้องมีการออกแบบสมการเอง แต่ทั้งนี้ต้องอ้างอิงหลักการทางคณิตศาสตร์และสถิติว่า X และ Y มีความสัมพันธ์กันแบบไหน อย่างไร และอีกอย่างคือ ความรู้เกี่ยวกับเรื่องนั้นๆ ที่จะมารองรับได้ว่าสมการของเราถูกต้อง หรือเราอาจจะใช้หลักการประมาณการทั่วไป ยกตัวอย่างจากที่เราได้เรียนมา เช่น

- **Simple linear regression** เป็นการดูความสัมพันธ์ระหว่างตัวแปรอิสระ(x) 1 ตัว กับตัวแปรตาม (Y) ว่ามีความสัมพันธ์แบบเส้นตรงมากน้อยเพียงใด

- **Multiple linear regression** เป็นการดูความสัมพันธ์ระหว่างตัวแปรอิสระ(x) มากกว่า 1 ตัว กับตัวแปรตาม(Y) ว่ามีความสัมพันธ์แบบเส้นตรงมากน้อยเพียงใด

- **Polynomial regression** เป็นการประมาณการที่ยังอยู่ในรูปของ linear regression แต่ว่าตัวแปรอิสระบางตัวอยู่ในรูปของเลขยกกำลัง เช่น Y = b + ax + cx^2... เป็นต้น

นี่ถือเป็น model เบื้องต้นเท่านั้น หากเจาะลึกการประมารการแบบ regression ลงไปจริงๆแล้วมี model มากมาย ซึ่งแต่ละ model ก็ออกแบบมาเพื่อให้สอดคล้องกับข้อมูลที่ได้มา

4. การแก้ปัญหาเมื่อ model ไม่ fit

อธิบายเพิ่มเติมนะคะ การที่ค่า R-squared ผลออกมามีค่าต่ำ นั้นเป็นเพราะว่าลักษณะของข้อมูลบางอย่างไปละเมิด assumption ของ OLS ทำให้การประมารการของเรานั้นมันเอนเอียง(bias)

หากว่าเราทำตามขั้นตอนทั้งหมดมาแล้ว แล้วผลวิเคราะห์ทางสถิติออกมาว่า Model เรา fit แค่ 0.59% (R-squared = 0.59) หรือ ตัวแปรอิสระบางตัวไม่มีความสัมพันธ์กับตัวแปรตาม เราจะทำอย่างไรดี การแก้ปัญหานี้ต้องใช้ความรู้ทางสถิติมากพอสมควร คือเรายิ่งรู้ลึก เราก็จะยิ่งแก้ปัญหาได้ตรงจุดมากขึ้น ในที่นี้ เราจะยกตัวอย่างปัญหาและการแก้ไขบื้องต้นกันก่อนนะคะ

- **Multicolinearity** เป็นปัญหาที่ตัวแปรอิสระมีความสัมพันธ์กันเองเป็นธรรมชาติอยู่แล้ว เช่น X2 เพิ่มขึ้น ส่งผลให้ X3 เพิ่มขึ้น เป็นต้น ซึ่งปัญหานี้แบ่งออกเป็น 2 ประเภทคือ Perfect multicolinearity และ multicolinearity

  ปัญหานี้ถ้าพูดแบบชาวบ้านคือ จะทำให้ผลการประมาณการที่ออกมาบอกว่า X2 และ X3 ไม่มีความสัมพันธ์กับ Y แต่จริงๆแล้ว X2 หรือ X3 นั้นอาจจะมีความสัมพันธ์กับ Y

  ในการแก้ไขปัญหาดังกล่าวนั้น เบื้องต้นคือ เราอาจจะเพิ่มจำนวนข้อมูลลงไปก่อน หรืออีกทางคือ เราอาจจะตัดตัวแปรอิสระที่เป็นปัญหาออก เช่นอาจจะตัด X3 ออก แต่ว่าเราต้องแน่ใจว่าตัวที่ตัดออกไป *ต้องเป็นตัวแปรที่ไม่มีอิทธิพลต่อตัวแปร y* ไม่งั้นเราจะเจอกับปัญหาใหม่คือ การกำหนดแบบจำลองผิดพลาดๆ (model missspecificaton)

- **Heteroscedasticity** เป็นปัญหาที่ค่า error มีค่าต่างกันมากๆๆๆ ปัญหานี้อาจจะมาจากข้อมูลมี outlier ซึ่งเราไม่ได้ใส่ตัวแปรนี้ลงไป เป็นต้น

ในการแก้ไขปัญหาดังกล่าวนั้น เบื้องต้น เราจะทำการประมาณการใหม่ให้เป็นแบบ Generalized least squared (GLS)

- **Autocorrelation** เป็นปัญหาที่เกิดกับข้อมูลที่มีเรื่องเวลาเข้ามาเกี่ยวข้อง ซึ่งข้อมูลของเราเป็นแบบ cross section ดังนั้นก็จะไม่เกิดปัญหาดังกล่าว ปัญหา Autocorrelation คือ ข้อมูลไม่เป็นอิสระต่อกัน เช่น ให้ x คือ การใช้จ่าย สมมติ การใช้จ่ายในเดือนนี้สูง อาจจะส่งผลทำให้การใช้จ่ายเดือนถัดไปลดลง นั่นคือตัวข้อมูลไม่ได้เป็นอิสระต่อกันนั่นเอง

ทั้งหมดที่กล่าวมานี้ ถือว่าเป็นพื้นฐานในการทำการประมาณการแบบ linear regression ซึ่งเพื่อนๆสามารถนำไปต่อยอดได้เพิ่มเติมได้ เช่น ต่อยอดด้านการพยากรณ์ราคาหุ้นในอนาคต การประมาณการเศรษฐกิจ เป็นต้น ทั้งนี้ก็ขึ้นอยู่กับงานที่เพื่อนๆทำแล้วว่า มีข้อมูลแบบไหน ต้องการที่จะวางแผนอะไร จุดประสงค์คืออะไร จะได้เลือกใช้ model ให้ถูกต้อง

ในบทต่อไป เราจะพูดถึงการทำ machine learning แบบ Classification กันค่ะ