Feng Chia University

Analyzing the Impact of COVID-19 on Housing Price Prediction: A Comparative Study of Pre and Post-Pandemic Models in Taichung City

International Business

D0832094

Shu-Xi Chen

Machine Learning and Python Marketing Data Analysis MKT444
May 19, 2023

Table of Contents

Introduction and Background

Housing Market & Covid-19

(Source: https://www.businesstoday.com.tw/article/category/192008/post/202201250035/)

Over the past 20 years, the Taiwan housing market has experienced multiple fluctuations, mostly influenced by the overall economic environment. For example, in 2003, the SARS outbreak had a negative impact on the housing market, with both housing prices and transaction volumes declining. The market didn't start to rebound until the second half of that year when the outbreak was under control. In 2008, the financial crisis and stock market crash caused housing prices to plummet.

Afterward, as the economy recovered, housing prices continued to rise until 2016. However, due to the implementation of various housing policies, including the introduction of the "Land Value Increment Tax," the housing market experienced another decline. Nevertheless, because the tax policy lacked a long-term effect, the market rebounded soon after.

Since the COVID-19 outbreak in 2020, Taiwan's housing market has remained optimistic due to the country's effective epidemic prevention measures, low-interest rates, and capital inflows. Despite the lack of negative sentiment in the housing market, the question remains: did the pandemic have any impact on the housing market, and have the factors that influence housing prices remained relatively constant?

Macroeconomics

The real estate market is not only influenced by the supply and demand structure and external variables of real estate, but also significantly affected by the financial environment and macroeconomic factors. For example:

I. **Economic growth rate**: Real GDP annual growth rate is the foundation of the development of the real estate industry, and is often used internationally to measure a country's economic strength.
II. **Consumer Price Index (CPI)**: It measures the price level of household consumption in food, clothing, housing, transportation, and entertainment. If the annual growth rate of CPI rises, it means that prices are rising, which will reduce the purchasing power of money, resulting in a decrease in real income, and may lead to a reduction in the demand for real estate. On the other hand, real estate is a relatively stable investment option, which may lead to an increase in demand.
III. **Money supply**: It can be used to measure the abundance of social idle funds, and also affects the cost of funds and the demand for the real estate industry.
IV. **Mortgage interest rates**: Home loans are stable and long-term cash expenditures for families. Therefore, consumers need to consider the current level of interest rates and their potential changes when buying a house. Rising interest rates will increase the burden of loans, reduce the affordability of housing prices, and lower the willingness to buy a house.

Economic booms and busts will occur before the real estate market. When the economy is in a prosperous period, the real estate market will also prosper, and vice versa. This means that **economic indicators are leading indicators of the real estate market**, and to some extent, can predict housing prices. Including these variables in analysis can make predicting housing prices more accurate.

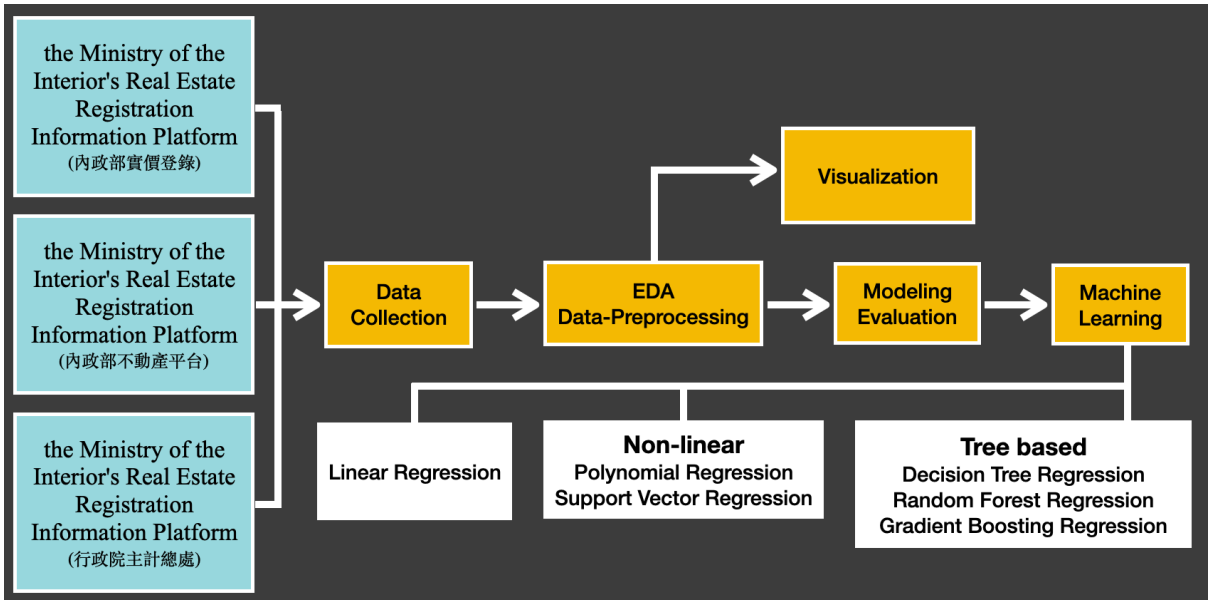
Under the global spread of Covid-19, both the domestic and international macro economies have been significantly impacted. For example, the unemployment rate and the proportion of unpaid leave have risen, and overall consumption has gradually decreased, along with the uncertainty surrounding the pandemic.

Objective

I'm curious about whether the factors that have been identified as impacting housing prices have remained relatively constant before and after the pandemic. It would be interesting to include macroeconomic indicators in the prediction of house prices while exploring the impact of COVID-19 on the housing market.

The purpose of this study is to combine the four macroeconomic indicators mentioned above with the characteristics of the properties themselves, and use machine learning methods to predict housing prices. The housing transaction data can be divided into **unit price** and **total price**, and this study will use **both** as prediction targets to explore the relationship between housing prices and individual variables.

Flow Chart of Research Process



Approach

The aim of this study was to use machine learning methods to predict housing prices by combining the four macroeconomic indicators mentioned above with the characteristics of the houses.

Housing data from eight administrative districts of Taichung City from August 2012 to December 2022, totaling 234,213 raw data records, was collected from the Ministry of the Interior's Real Estate Registration Information Platform. Additionally, macroeconomic indicators were obtained from the Ministry of the Interior's Real Estate Platform and the Directorate-General of Budget, Accounting and Statistics (DGBAS).

The prediction targets are the total and unit prices of the houses, and therefore, **regression analysis** using supervised learning is the primary analysis method.

The following table shows the features that were initially expected to be included in this study:

Basic Variables of the Building		
Data Type	Name of Variables	Feature Description
Target	Total Price	Actual transaction total price (NTD)
	Unit Price	Price per square meter (NTD)
Features	District	Top 8 administrative districts of Taichung City.
	Addresses	Building address, including the street name.
	Longitude	Longitude of the building
	Latitude	Latitude of the building
	Age	Years since construction
	Building Type	Only select data of apartment(公寓), residential building(住宅大樓), townhouse(華廈), suite(套房), and detached house(透天厝) types.
	Building Area	The total floor area of the main building in square meters.
	Main Purpose	Only select data with the main use of the building as residential(住家用).
	Layout	Floor plan (number of bedrooms, living rooms, bathrooms)
	Floor / Building Height	Floor sold for the transaction / Total floors of the building for the transaction
	Elevator	Presence of elevator
	Manager	Presence of property management staff
	Parking numbers	Number of parking spaces in the transaction
	Biuld_share1	Building area to total transfer area ratio. (excluding parking space area)
	Note	Additional notes or details about the transaction, including special transactions.
	Transaction Date	Transaction date of the building in the format of ROC year/month/day.

Macroeconomic indicators			
Data Type	Name of Variables	Time Granularity	Feature Description
Features	GDP%	Quarter	Real GDP growth rate
	CPI%	Month	year-on-year growth rate of CPI
	Mortgage Rate	Month	Average mortgage interest rate of the five major banks.
	M1b	Month	M1b money supply (100 million NTD)

Experimental Design

To test the impact of the COVID-19 pandemic on the housing market and evaluate the performance of appropriate methods, this study established two models and **divided the data into training, validation, and testing sets** using random sampling.

Model_1 uses all data from 2012 to 2022, including pandemic years, to predict housing prices. The data is randomly distributed into the training, validation, and testing sets to understand the accuracy of predictions when pandemic years are included in the training data. Depending on the prediction target, this model is further divided into **Model_1_1**, with total price as the target, and **Model_1_2**, with unit price as the target.

For Model_2, only data from 2012 to the end of 2019, before the pandemic occurred, is used for model training and validation. The testing set uses all data from 2020 to 2022. The purpose of this model is to understand the accuracy of predictions when pandemic years are not included in the training data. Depending on the prediction target, this model is further divided into **Model_2_1,** with total price as the target, and **Model_2_2**, with unit price as the target.

Finally, the results of both models are compared to observe whether there are significant differences in the model prediction performance. If the model trained on pre-pandemic data (model 2) performs better than the model trained on all available data (model_1), this could suggest that the patterns and relationships in the data have changed significantly due to the pandemic. Model_2, which was trained on data that does not include the pandemic period, may have learned patterns that are more robust and representative of the underlying dynamics

of the housing market, whereas model_1 may have been affected by the significant changes in the data caused by the pandemic.

If, on the other hand, model_1 performs similarly or better than model_2, it may indicate that the pandemic did not have a significant impact on the data, or that the model is able to adapt to the changes in the data caused by the pandemic. This would suggest that the patterns and relationships in the data remain stable over time, and that we can use the same model to make predictions on data from different time periods.

To balance the data volume in the three sets of data for both models, the data splitting ratios are slightly different. The data splitting settings and model explanations are summarized in the following table:

<table>
<tr><th colspan="6">Data Splitting</th></tr>
<tr><td colspan="2">Model</td><td>Target</td><td>Sample Period</td><td>Training and Validation</td><td>Test</td></tr>
<tr><td rowspan="2">Model_1</td><td>Model_1_1</td><td>Total Price</td><td rowspan="2">2012~2022</td><td rowspan="2">Training 60% ; Validation 20%</td><td rowspan="2">20%</td></tr>
<tr><td>Model_1_2</td><td>Unit Price</td></tr>
<tr><td rowspan="2">Model_2</td><td>Model_2_1</td><td>Total Price</td><td rowspan="2">2012~2019</td><td rowspan="2">Training 80% ; Validation 20%</td><td rowspan="2">2020~2022 100%</td></tr>
<tr><td>Model_2_2</td><td>Unit Price</td></tr>
<tr><td colspan="6">Description</td></tr>
<tr><td>Mode1_1</td><td colspan="5">All data from 2012 to 2022 were used for training, validation, and testing to understand the accuracy of predictions with the inclusion of years affected by the pandemic in the training data.</td></tr>
<tr><td>Model_2</td><td colspan="5">Using data from 2012 to 2019 as training and validation, the model predicts housing prices from 2019 to 2022 after the pandemic to understand the accuracy of predictions when pandemic years are not included in the training data.</td></tr>
</table>

Method of Model Evaluation

During the model testing phase, the performance of each method in predicting house prices will be explored based on the testing data. Considering that R square increases with the

increase in regression variables, this study uses **adjusted R square** with penalties and **MSE** indicators to measure the accuracy of the predictions.

EDA

- Change the data type of some features
- build_share1, build_share2 is in '%' string format, it should be transform to float
- area, build_share1, build_share2, unit_price, price should be numerical
- deal_date is not in date format

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234213 entries, 0 to 234212
Data columns (total 20 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   district       234213 non-null  object
 1   address        234213 non-null  object
 2   community      114945 non-null  object
 3   lon            234213 non-null  float64
 4   lat            234213 non-null  float64
 5   age            179460 non-null  float64
 6   area           234213 non-null  object
 7   build_type     234211 non-null  object
 8   main_purpose   234199 non-null  object
 9   floor          234213 non-null  object
 10  layout         222843 non-null  object
 11  elevator       234213 non-null  object
 12  manager        234213 non-null  object
 13  parking_num    234213 non-null  int64
 14  build_share1   198887 non-null  object
 15  build_share2   90607 non-null   object
 16  note           63424 non-null   object
 17  deal_date      234213 non-null  object
 18  unit_price     204264 non-null  object
 19  price          234213 non-null  object
dtypes: float64(3), int64(1), object(16)
```

→

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234213 entries, 0 to 234212
Data columns (total 20 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   district       234213 non-null  object
 1   address        234213 non-null  object
 2   community      114945 non-null  object
 3   lon            234213 non-null  float64
 4   lat            234213 non-null  float64
 5   age            179460 non-null  float64
 6   area           234213 non-null  float64
 7   build_type     234211 non-null  object
 8   main_purpose   234199 non-null  object
 9   floor          234213 non-null  object
 10  layout         222843 non-null  object
 11  elevator       234213 non-null  object
 12  manager        234213 non-null  object
 13  parking_num    234213 non-null  int64
 14  build_share1   198887 non-null  float64
 15  build_share2   90607 non-null   float64
 16  note           63424 non-null   object
 17  deal_date      234213 non-null  object
 18  unit_price     204264 non-null  float64
 19  price          234213 non-null  float64
dtypes: float64(8), int64(1), object(11)
```

```
df["build_share1"] = df["build_share1"].str.replace('%', '').astype(float)/100
df["build_share2"] = df["build_share2"].str.replace('%', '').astype(float)/100
```

- take a glimpes of missing value
  - build_share2, community is undesired, will not discuss them from now on

```
district             0
address              0
community       119268
lon                  0
lat                  0
age              54753
area                 0
build_type           2
main_purpose        14
floor                0
layout           11370
elevator             0
manager              0
parking_num          0
build_share1     35326
build_share2    143606
note            170789
deal_date            0
unit_price       29949
price                0
```

- check duplicates
- 192 duplicates

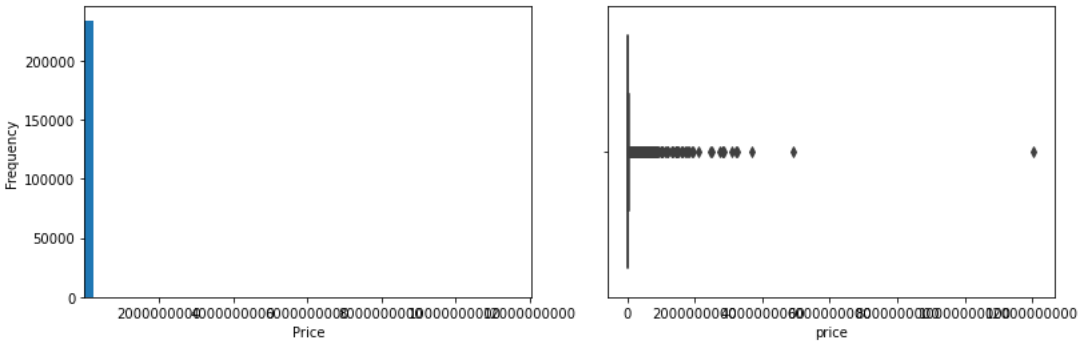
```
In [9]: len(df[df.duplicated()])
Out[9]: 192
```

Numerical data

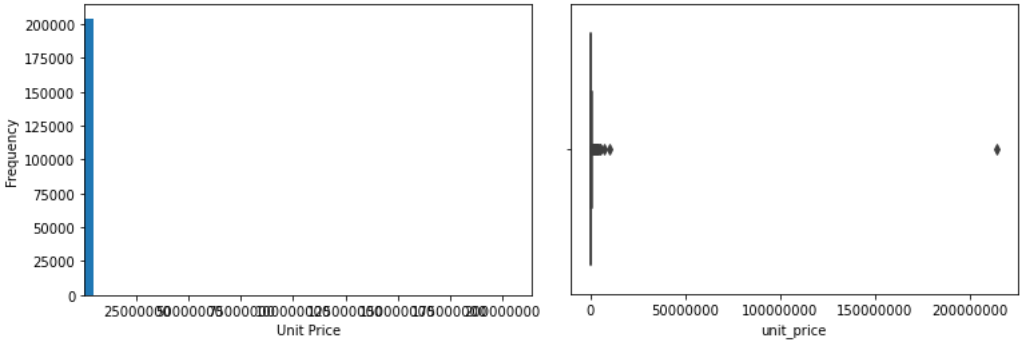
- Distribution overview
- age, area, parking_num, price, unit_price seems to have outliers

```
In [13]: df.describe()
Out[13]:
                lon            lat            age           area  \
count  234213.000000  234213.000000  179460.000000  234213.000000
mean      120.661829      24.157523      18.906898      51.229366
std         0.864283       0.174177      12.379440     143.446232
min         0.000000       0.000000       1.000000       0.000000
25%       120.645976      24.144112       7.000000      29.620000
50%       120.667005      24.161403      20.000000      42.370000
75%       120.689367      24.174401      27.000000      58.940000
max       120.788263      24.214521     109.000000   38035.630000

         parking_num   build_share1   build_share2     unit_price         price
count  234213.000000  198887.000000  90607.000000  2.042640e+05  2.342130e+05
mean        0.760748       0.579026      0.598287  2.157503e+05  1.251895e+07
std         1.569062       0.145221      0.044050  4.851880e+05  4.161615e+07
min         0.000000       0.000700     -1.547300  1.700000e+01  0.000000e+00
25%         0.000000       0.476500      0.576400  1.525850e+05  5.180000e+06
50%         1.000000       0.536900      0.593700  1.993320e+05  8.400000e+06
75%         1.000000       0.646800      0.611900  2.584315e+05  1.360000e+07
max       355.000000       1.000000      1.000000  2.145531e+08  1.205000e+10
```

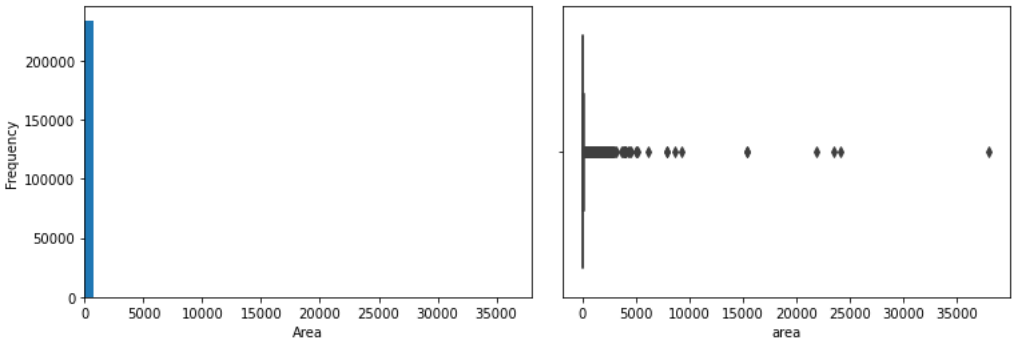
- price
- highly right skewed and has outliers



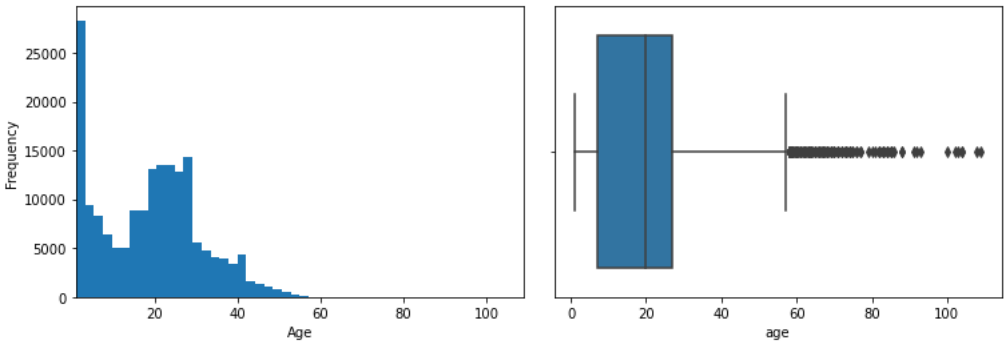
- unit price
- highly right skewed and has outliers



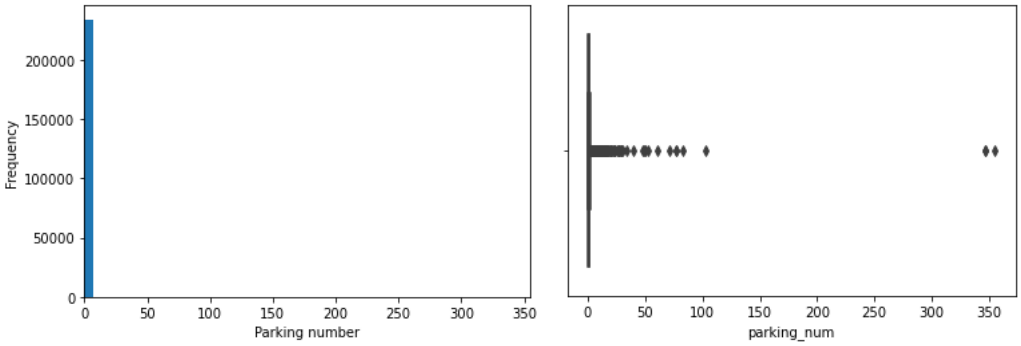
- area
- highly right skewed and has outliers



- age
- right skewed and has outliers

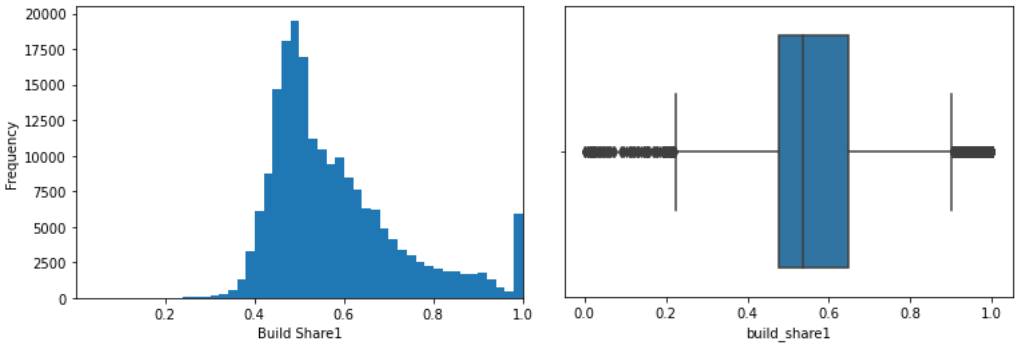


- parking numers
- highly right skewed and has outliers

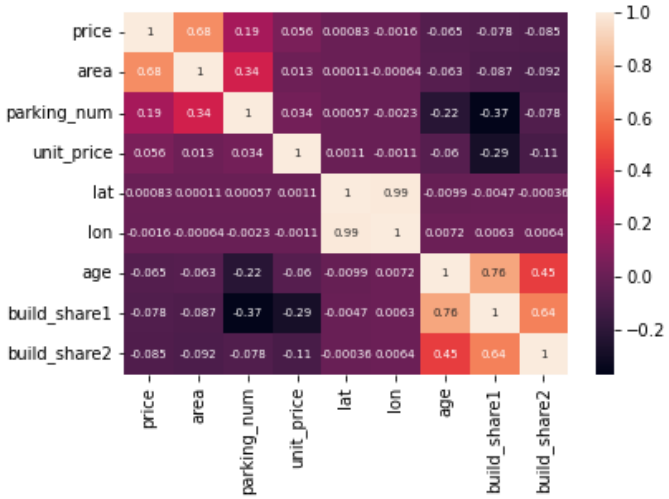


- build share1

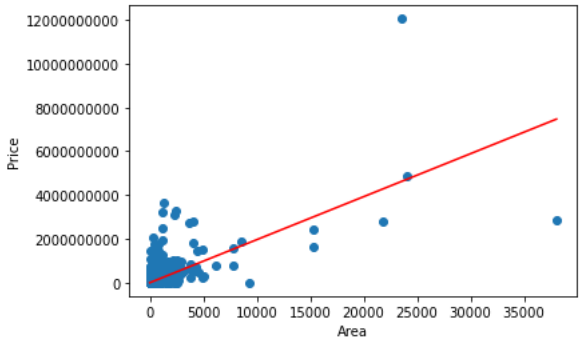
- has outliers



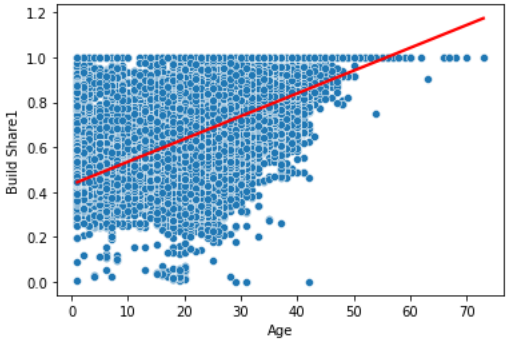
- correlation



- correlation between area & price is almost 0.7

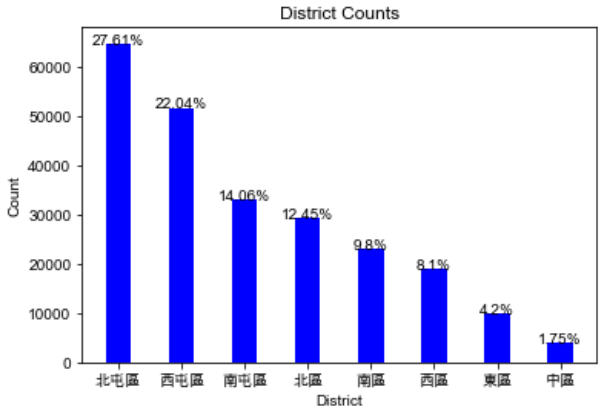


- correlation between age & build share1 > 0.7

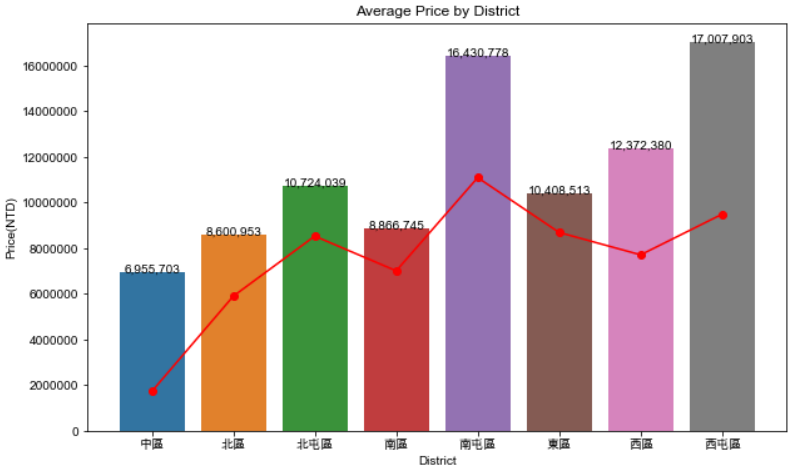


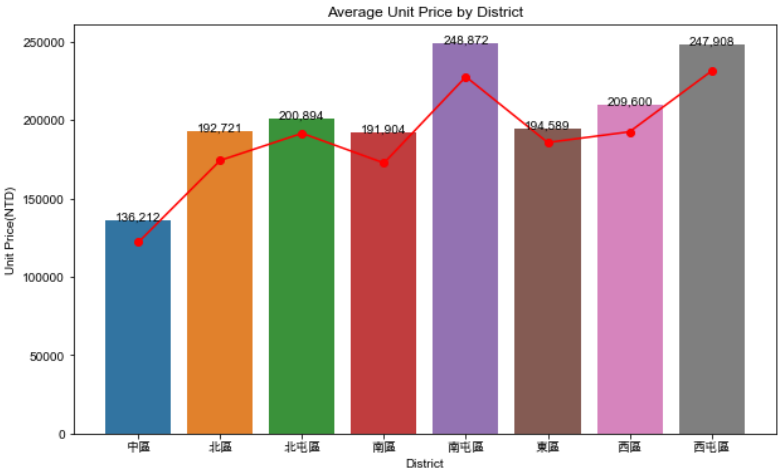
Categorical data

- district
- The majority of housing transactions took place in Beitun district(北屯區), followed by Xitun district(西屯區) and Nantun district(南屯區).

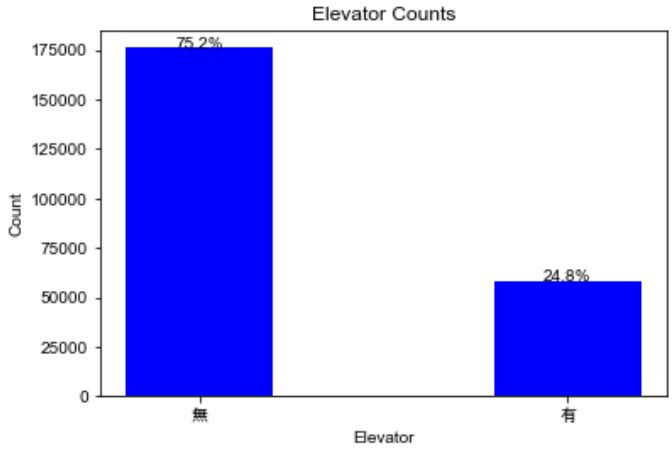


- The highest average and median prices, as well as unit prices, are found in Xitun district (西屯區) and Nantun district (南屯區).

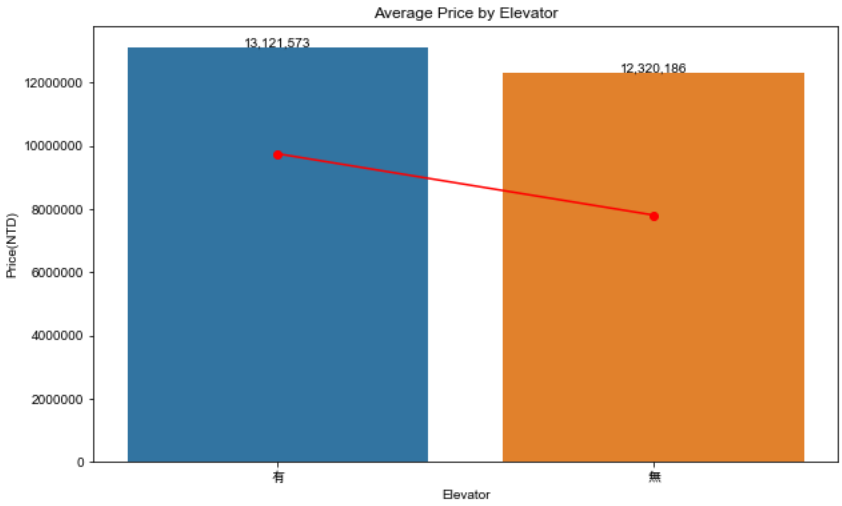


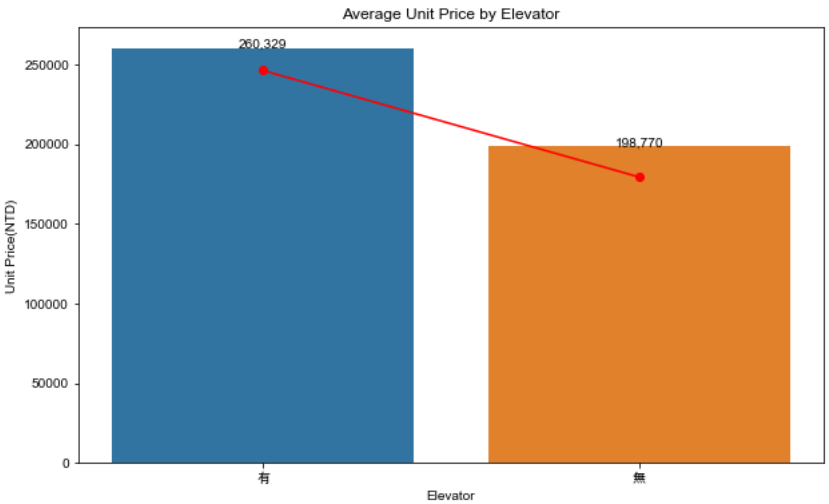


- elevator
- 75% of the deals involved buildings equipped with elevators.



- Buildings equipped with elevators have higher average and median prices, as well as unit prices.

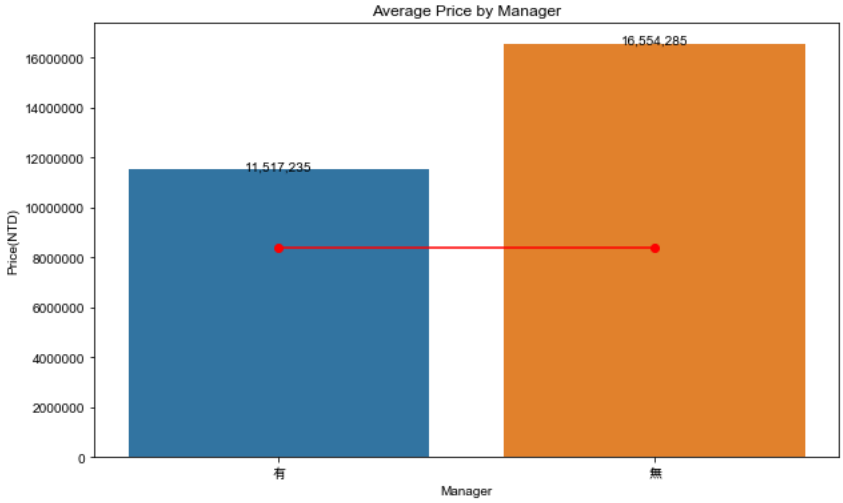


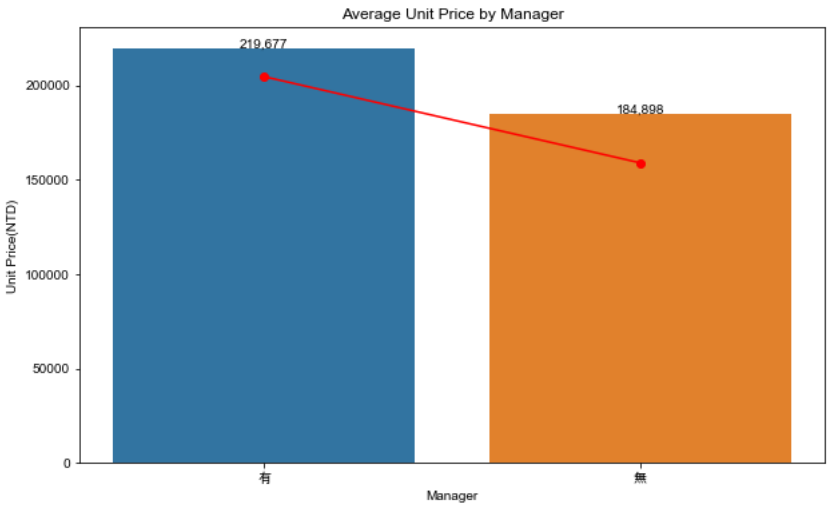


- manager

- 80% of the deals involved buildings with a presence of managers.

- The average price of buildings without a presence of managers are higher, while the unit prices have a lower average and median.





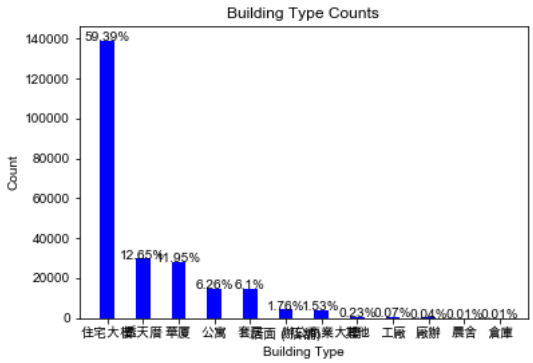
- build_type
- Only consider top 5, hence shorten their name

```
In [31]: df["build_type"].value_counts()
Out[31]:
住宅大樓(11層含以上有電梯)     139103
透天厝                    29636
華廈(10層含以下有電梯)        27997
公寓(5樓含以下無電梯)         14662
套房(1房(1廳)1衛)           14296
店面（店舖）                 4119
辦公商業大樓                 3574
其他                     535
工廠                     165
廠辦                      82
農舍                      26
倉庫                      16
Name: build_type, dtype: int64
```

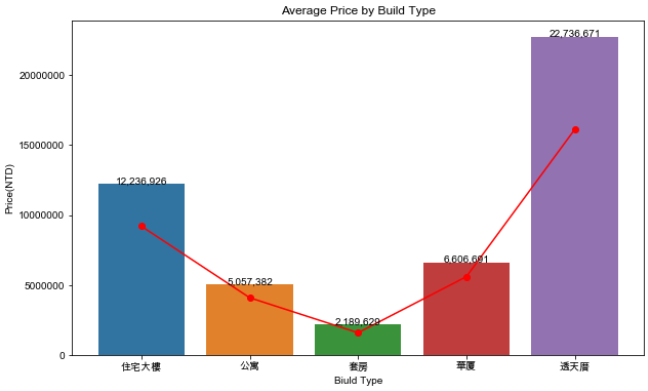
→

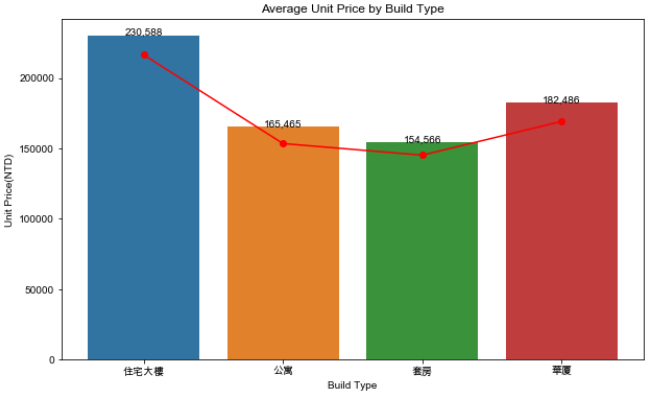
```
In [33]: df["build_type"].value_counts()
Out[33]:
住宅大樓       139103
透天厝         29636
華廈          27997
公寓          14662
套房          14296
店面（店舖）       4119
辦公商業大樓       3574
其他            535
工廠            165
廠辦             82
農舍             26
倉庫             16
Name: build_type, dtype: int64
```

- (住宅大樓) accounts for almost 60% of the deals.



- The highest average and median prices, as well as unit prices, are found in residential building (住宅大樓).





- main_purpose
  - only keep residential(住家用) in the following analysis

```
In [39]: df["main_purpose"].value_counts()
Out[39]:
住家用         206559
住商用          10796
商業用           9484
辦公用           4033
其他            1924
商辦用            548
工業用            292
住商辦用           251
住工用            208
農業用             71
工商用             32
見其他登記事項          1
Name: main_purpose, dtype: int64
```

Data Preprocessing

- remove duplicates
  - 192 rows be removed

```
In [40]: len(df[df.duplicated()]) # 192
    ...: df = df.drop_duplicates().reset_index(drop = True)
```

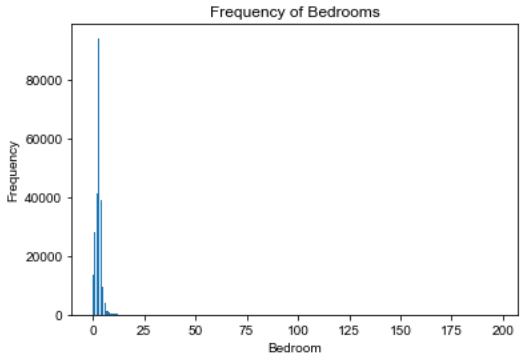
- address
  - extract "road name" from "address" and create a new column

```
In [41]: def extract_road(address):
    ...:     pattern = re.compile(r'^[\u4e00-\u9fa5]+')
    ...:     match = pattern.search(address)
    ...:     if match:
    ...:         return match.group(0)
    ...:     else:
    ...:         return ""
    ...: df["road_name"] = df["address"].apply(extract_road)
```

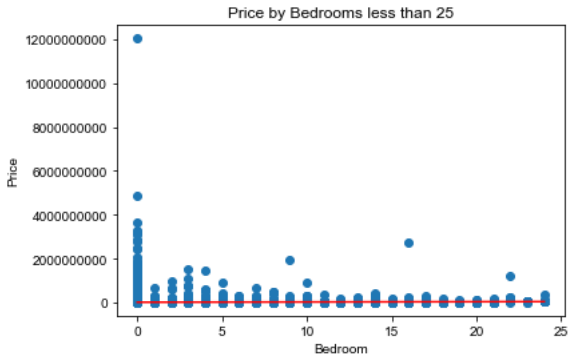
- layout
  - extract bedroom, hall, bathroom from layout and store them in new columns
  - convert their data types from string to int

```
In [42]: df[["bedroom_num", "hall_num", "bathroom_num"]] = df["layout"].str.extract(r'(\d+)房(\d+)廳(\d+)衛')
    ...: df[["bedroom_num", "hall_num", "bathroom_num"]] = df[["bedroom_num", "hall_num",
"bathroom_num"]].fillna(0).astype(int)
```

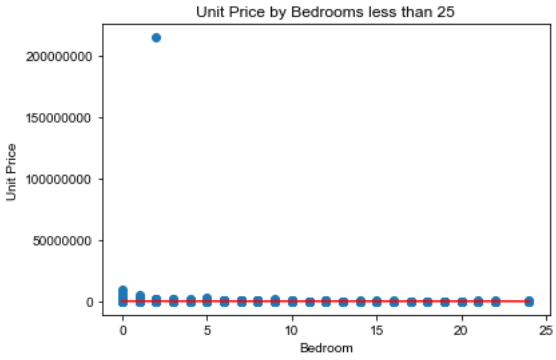
  - bedroom_num
    - most bedrooms in the deals are less than 25
    - has outliers



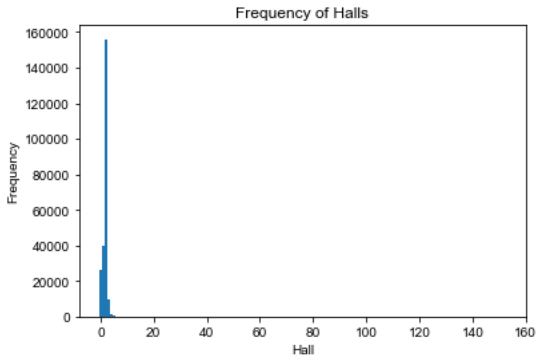
- bedroom_num < 25 vs. price



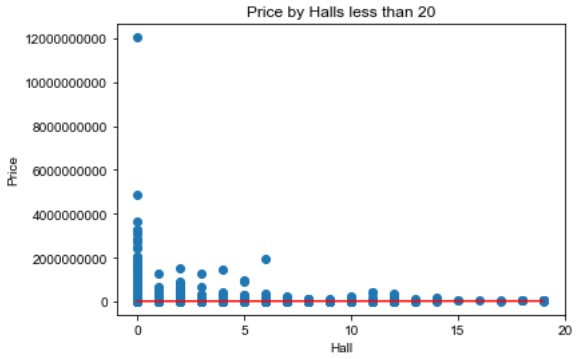
- bedrooms < 25 vs. unit price



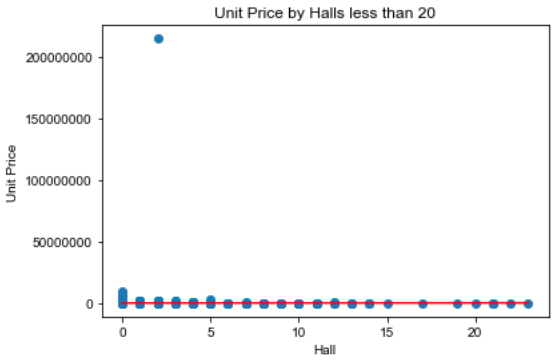
- hall_num
  - most hulls in the deals are less than 20
  - has outliers



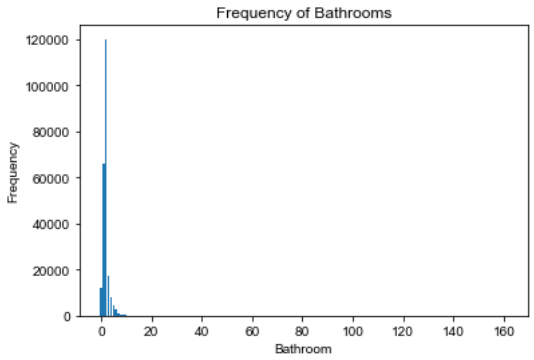
- hull_num < 20 vs. price



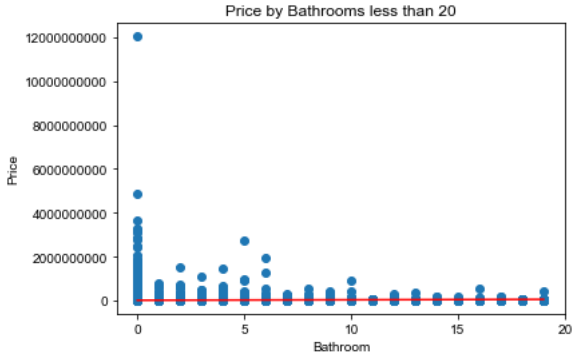
- hull_num < 20 vs. unit price



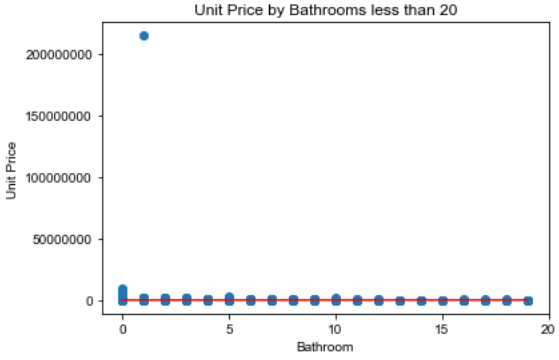
  - bathroom_num
    - most bathrooms in the deals are less than 20
    - has outliers



- bathroom_num < 20 vs. price



- bathroom_num < 20 vs. unit price



- floor
  - Extract "floor_extracted " and "story".

```
In [52]: df[["floor_extracted", "story"]]= df["floor"].str.split('/', expand=True)
```

  - Find the highest floor in story and convert it to numbers.

```
In [53]: df["story"].unique()
Out[53]:
array(['七層', '十二層', '五層', '八層', '四層', '二層', '三層', '一層', '十層', '十三層',
       '二十四層', '十五層', '十四層', '十六層', '十一層', '九層', '六層', '二十二層', '(空白)',
       '二十七層', '十七層', '二十三層', '十九層', '二十層', '二十五層', '三十四層', '三十二層',
       '二十一層', '三十五層', '二十九層', '二十六層', '三十七層', '十八層', '三十層', '二十八層',
       '三十九層', '四十二層', '三十八層', '三十六層', '三十三層', '四十一層', '三十一層', '四十三層'],
      dtype=object)
```

```
In [55]: floor_dict = {
    ...:   1: '一層', 2: '二層', 3: '三層', 4: '四層', 5: '五層', 6: '六層', 7: '七層', 8: '八層',
    ...:   9: '九層', 10: '十層', 11: '十一層', 12: '十二層', 13: '十三層', 14: '十四層',
    ...:   15: '十五層', 16: '十六層', 17: '十七層', 18: '十八層', 19: '十九層', 20: '二十層',
    ...:   21: '二十一層', 22: '二十二層', 23: '二十三層', 24: '二十四層', 25: '二十五層',
    ...:   26: '二十六層', 27: '二十七層', 28: '二十八層', 29: '二十九層', 30: '三十層', 31: '三十一層',
    ...:   32: '三十二層', 33: '三十三層', 34: '三十四層', 35: '三十五層', 36: '三十六層', 37: '三十七層',
    ...:   38: '三十八層', 39: '三十九層',40: '四十層', 41: '四十一層', 42: '四十二層', 43: '四十三層',
    ...:   None: '(空白)'}

In [56]: df["story"] = df["story"].map({v: k for k, v in floor_dict.items()})
```

  - There are multiple types in floor_extracted.

```
In [58]: df["floor_extracted"].value_counts()
Out[58]:
全                          29548
五層                         19237
四層                         19205
三層                         17658
六層                         15140
                           ...
四層,五層,六層                      1
四層,五層,夾層                      1
二層,三層,四層,屋頂突出物                 1
一層,屋頂突出物,地下層                  1
一層,二層,三層,四層,騎樓,電梯樓梯間         1
Name: floor_extracted, Length: 413, dtype: int64
```

  - Screen out those are not in floor_dict
    - Purchase all floor(全) and purchase basement floor(地下) are most important
    - Maybe purchase several floor at one time could have influence on total price and unit price, we create another features ‘total_deal_floor’ to store it.

```
In [73]: item_counts = Counter(floor_extracted_items)
    ...: filtered_counts = {k: v for k, v in item_counts.items() if k not in floor_dict.values()}
    ...: print(filtered_counts)
{'騎樓': 5768, '全': 29548, '地下一層': 725, '見其他登記事項': 1267, '夾層': 816, '地下層': 1601, '屋頂突出物': 221, '地下二層': 63, '': 637
'瞭望室': 2, '地下三層': 22, '地下四層': 1, '防空避難室': 1, '電梯樓梯間': 6, '通道': 2, '見其它登記事項': 77, '走廊': 21, '平台': 1}
```

  - Delete other type of floor except for all(全), basement(地下) and above-ground floors, and check if element in new_floor_extracted_list are removed successfully.

```
In [74]: drop_floor_items = ['見其他登記事項', '夾層', '屋頂突出物', '見其它登記事項', '走廊',
    ...:                     '電梯樓梯間', '瞭望室', '通道', '防空避難室', '平台', '騎樓', '']
    ...: new_floor_extracted_list = [[item for item in inner_list if item not in
drop_floor_items] for inner_list in floor_extracted_list]

In [75]:
    ...: for inner_list in new_floor_extracted_list:
    ...:     for item in inner_list:
    ...:         if item in drop_floor_items:
    ...:             print("Failed: {} was not removed".format(item))
    ...:             break
    ...:     else:
    ...:         continue
    ...:     break
    ...: else:
    ...:     print("Successful")
Successful
```

- Assign the value of total deal floor by the len of the new_floor_extracted_list, basement floor counts as one floor. For those purchase all floors(全), assign the value of total story of the building to it.

```
In [76]: df["total_deal_floor"] = [len(item) for item in new_floor_extracted_list]
    ...: df.loc[df["floor_extracted"] == "全", "total_deal_floor"] = df["story"]
```

- It is meaningless to count the total deal floor of a detached house (透天厝). If the transaction involves the purchase of all floors (全), then assign a value of 0 to the total deal floor for detached houses.

```
In [77]: df.loc[(df["floor_extracted"] == "全") & (df["build_type"] == "透天厝"), "total_deal_floor"] = 0
```

- Create floor_sold to store the value of floor sold. If a deal contains more than one floors, assign its mean value to the floor_sold.
  - First remove those whose floor_extracted are not in floor_dict in case a calculation error or other situation.

```
In [78]: target_floor = dict(Counter([item for item in floor_extracted_items if item not in floor_dict.values()]))
    ...: target_floor = dict(sorted(target_floor.items(), key=lambda item: item[1], reverse = True))
    ...: drop_floor_items_for_cal = list(target_floor.keys())
    ...: new_floor_extracted_list_for_cal = [[item for item in inner_list
    ...:                                      if item not in drop_floor_items_for_cal]
    ...:                                     for inner_list in floor_extracted_list]
```

  - check if element in new_floor_extracted_list_for_col are removed

```
In [79]: for inner_list in new_floor_extracted_list_for_cal:
    ...:     for item in inner_list:
    ...:         if item in drop_floor_items_for_cal:
    ...:             print("Failed: {} was not removed".format(item))
    ...:             break
    ...:     else:
    ...:         continue
    ...:     break
    ...: else:
    ...:     print("Successful")
Successful
```

  - assign value to new_floor_extracted_list_for_cal and store it in cal_floor

```
In [80]: cal_floor = [[list(floor_dict.keys())[list(floor_dict.values()).index(item)]
    ...:               for item in inner_list]
    ...:              for inner_list in new_floor_extracted_list_for_cal]
```

  - find the mean of floor sold for each list in cal_floor, and store it to a new feature named 'floor_sold'

```
In [82]: floor_sold = []
    ...: for inner_list in cal_floor:
    ...:     if len(inner_list) > 0:
    ...:         floor_sold.append(sum(inner_list) / len(inner_list))
    ...:     else:
    ...:         floor_sold.append("")

In [83]: df["floor_sold"] = floor_sold
    ...: df['floor_sold'] = df['floor_sold'].replace('', 0).astype(float)
```

  - assign the half value of total story of the building to it

```
In [84]: df.loc[df["floor_extracted"] == "全", "floor_sold"] = df["story"] / 2
```

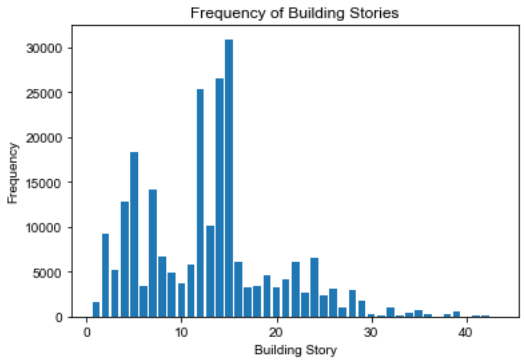
- It is meaningless to count the floor_sold of a detached house (透天厝). If the transaction involves the purchase of all floors (全), then assign a value of 0 to the floor_sold for detached houses.

```
In [85]: df.loc[(df["floor_extracted"] == "全") & (df["build_type"] == "透天厝"), "floor_sold"] = 0
```

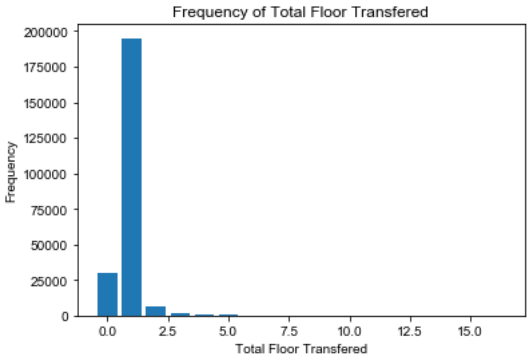
- Overview floor related features

```
In [86]: df[["story", "total_deal_floor", "floor_sold"]].describe()
Out[86]:
                story  total_deal_floor     floor_sold
count  233832.000000     234021.000000  234021.000000
mean       12.890297          0.925767       6.890238
std         7.081927          0.488752       5.635782
min         1.000000          0.000000       0.000000
25%         7.000000          1.000000       3.000000
50%        13.000000          1.000000       6.000000
75%        15.000000          1.000000      10.000000
max        43.000000         16.000000      42.000000
```

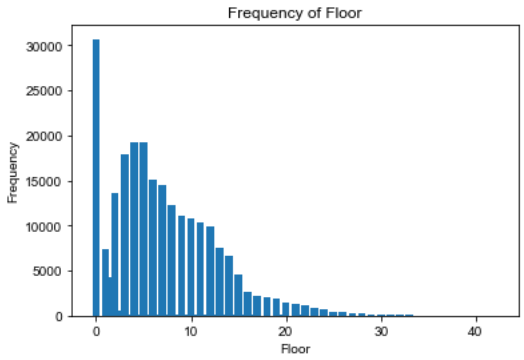
- story



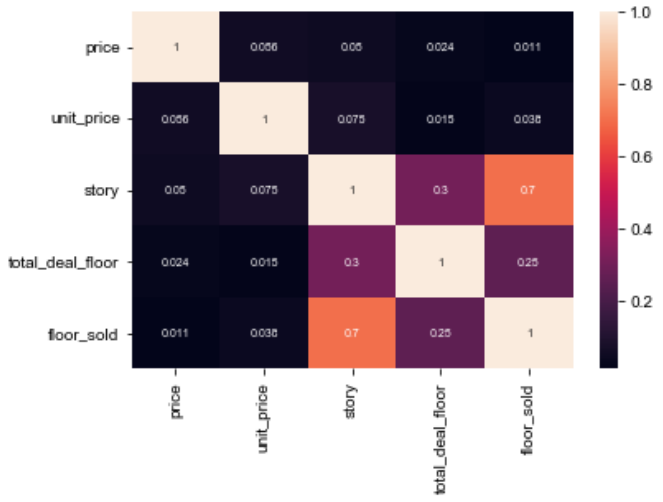
- total_deal_floor



- floor_sold



- correlation plot(between floor derivative features and total_price, unit_price)



- note

*** In the real estate transaction data registered in Taiwan, there is a "note field(備註欄位)" that allows people to declare special transaction types, such as "with illegal structures(含增建或未登記建物)" and "transaction between relatives or friends(親友交易)". However, the note field contains unstructured data that needs to be analyzed using text mining techniques in order to effectively analyze this field. According to a paper named "Application of Big Data in Real Estate Registration Data - Processing of Note Field(大數據於實價登錄資料的應用-備註欄的處理)" written by three land management professors, they used the JIEBA Chinese word segmentation program as the Chinese word segmentation tool, extracted keywords based on semantics, conducted frequency analysis on the keywords, and classified and compared them according to the nature of the word.

In addition, the study found, after word segmentation analysis, that there are more than 40 different ways to record special relationships between family members in the note field, such as "relatives(親戚), family(親友), kin(家人), blood relatives(血親), etc." These different words were automatically recognized by the computer and then classified according to the transaction attributes.

Based on this report, this study analyzed and combined the labeled 13 special transaction types and removed fields containing these keywords to avoid bias and noise in house prices and market trends caused by transactions based on specific reasons. ***

  - delete 47,852 rows whose note feature contain the following characters

```
In [91]: del_notes = ["親友、員工、共有人或其他特殊關係間之交易", "建商與地主合建案",
    ...:              "陽台外推", "頂樓加蓋", "夾層", "其他增建", "未登記建物",
    ...:              "農作物", "機電設備", "農業設施", "急買急賣", "民情風俗",
    ...:              "瑕疵物件", "含租約", "毛胚屋", "具重建或重劃、都更等效益",
    ...:              "夥零地", "借名登記", "受債權債務影響或債務抵償之交易",
    ...:              "雙方合意解除契約", "土地交易案件之價格含未來興建房屋成本",
    ...:              "地上權房屋", "市場攤位", "公共設施保留", "政府機關標讓售",
    ...:              "地清或未辦繼承標售", "水利地承購", "協議價購", "僅車位交易",
    ...:              "預售屋、或土地及建物分件登記案件", "親等","等親","親友","親屬",
    ...:              "血親","親人","親戚","姻親","親子","親朋","夫妻","父子","父女",
    ...:              "母子","子母","母女","兄弟","兄妹","姊弟","姊妹","姐弟","姐妹",
    ...:              "大伯","大嫂","女兒","小叔","小孩","父親","母親","弟媳","叔侄",
    ...:              "妹妹","妻舅","姊夫","姊姊","姐姐","姑姑","直系","表哥","近親",
    ...:              "姪女","祖孫","婚姻","甥舅","媳婦","朋友"]
    ...: df["note"].str.contains('|'.join(del_notes)).sum() # 47852 rows
    ...: df = df[~df['note'].fillna('').str.contains('|'.join(del_notes)) | df['note'].isna()].reset_index(drop = True)
```

- main_purpose
  - delete 21,048 rows that are other than residential (住家用)

```
In [92]:
    ...: len(df[df["main_purpose"] != "住家用"]) # 21048
    ...: df = df[df["main_purpose"] == "住家用"].reset_index(drop=True)
```

- build_type
  - delete 475 rows that are other than apartment(公寓), residential building(住宅大樓), townhouse(華廈), suite(套房), and detached house(透天厝)

```
In [93]: len(df[~df["build_type"].isin(df["build_type"].value_counts()[:5].index)]) # 475
    ...: df = df[df["build_type"].isin(df["build_type"].value_counts()[:5].index)].reset_index(drop=True)
```

- drop redundant features

```
In [95]: new_df_cols = ['district','road_name', 'lon', 'lat', 'age', 'area', 'build_type',
    ...:                'bedroom_num', 'hall_num', 'bathroom_num', 'story',
    ...:                'total_deal_floor', 'floor_sold','elevator', 'manager', 'parking_num',
    ...:                'build_share1', 'unit_price', 'price', 'deal_date']
    ...: df_new = df[new_df_cols]
```

Data Preprocessing (Outliers and Missing Values)

Data splitting

- extract year out for splitting data, and create year features for df_new

```
# extract year out for splitting data
year_month_day = []
for date in df["deal_date"]:
    year, month, day = date.split("/")
    new_year = int(year) + 1911
    year_month_day.append((str(new_year), month, day))
# create year features for df_new
time = []
for inner_list in year_month_day:
    time.append(inner_list[0])
df_new["year"] = time
df_new["year"].isna().sum() # 0
df_new["year"] = df_new["year"].astype(int)
```

- Model_1

```
In [97]: X_1 = df_new.drop("price", axis=1)
    ...: y_1 = df_new.price
    ...: X_train, X_test_1, y_train, y_test_1 = train_test_split(X_1, y_1, test_size=0.2, random_state=42)
    ...: X_train_1, X_val_1, y_train_1, y_val_1 = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
    ...: df_train_1 = pd.concat([X_train_1, y_train_1], axis=1) #105396
    ...: df_val_1 = pd.concat([X_val_1, y_val_1], axis=1) #26350
    ...: df_test_1 = pd.concat([X_test_1, y_test_1], axis=1) #32937
```

- Model_2

- Split by year

```
In [117]: pre_covid = df_new[df_new["year"] <= 2019]
     ...: post_covid = df_new[df_new["year"] > 2019]
     ...: X_2 = pre_covid.drop("price", axis = 1)
     ...: y_2 = pre_covid.price
     ...: X_test_2 = post_covid.drop("price", axis = 1)
     ...: y_test_2 = post_covid.price
     ...: X_train_2, X_val_2, y_train_2, y_val_2 = train_test_split(X_2, y_2, test_size = 0.2, random_state = 42)
     ...: df_train_2 = pd.concat([X_train_2, y_train_2], axis = 1) #103342
     ...: df_val_2 = pd.concat([X_val_2, y_val_2], axis = 1) #25836
     ...: df_test_2 = pd.concat([X_test_2, y_test_2], axis = 1) #35505
```

Data imputing

- Model_1 training
  - check overall NA and unreasonable values

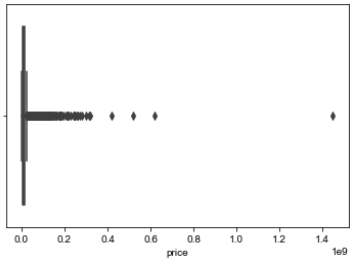
```
In [118]: len(df_train_1[df_train_1["lon"] == 0]) # 1
     ...: len(df_train_1[df_train_1["lat"] == 0]) # 1
     ...: # nan was replaced with "" in previous manipulation
     ...: len(df_train_1[df_train_1["road_name"] == ""]) # 1670
     ...: # it seems wierd that bedroom_num, hall_num and bathroom_num be 0 at the same time
     ...: len(df_train_1[(df_train_1["bedroom_num"] == 0) & (df_train_1["hall_num"] == 0) & (df_train_1["bathroom_num"] == 0)]) # 2362
     ...: zero_counts = {}
     ...: for build_type in df_train_1["build_type"].unique():
     ...:     zero_counts[build_type] = {}
     ...:     for col in ["bedroom_num", "hall_num", "bathroom_num"]:
     ...:         zero_count = len(df_train_1[(df_train_1[col] == 0) & (df_train_1["build_type"] == build_type)])
     ...:         zero_counts[build_type][col] = zero_count
     ...: # 0 is normal in total_deal_floor and floor_sold of "透天厝"
     ...: len(df_train_1[(df_train_1["total_deal_floor"] == 0) & (df_train_1["build_type"] != "透天厝")]) # 340
     ...: len(df_train_1[(df_train_1["floor_sold"] == 0) & (df_train_1["build_type"] != "透天厝")]) # 411
```

  - Since buildings features would be similar in same road and same build type, it would be more real if we impute outliers or NA with the value grouped by them. So, firstly, we need to extract road_name from address for the following imputation.

    *If road_name is blank, imputing it with the road whose median price has the smallest distance with the price of blank.

```
In [119]: road_name_median = df_train_1[df_train_1["road_name"] != ""].groupby(["district","road_name"])["price"].median()
     ...: for index in df_train_1[df_train_1["road_name"] == ""].index:
     ...:     district_medians = road_name_median[road_name_median.index.get_level_values("district") == df_train_1.loc[index, "district"]]
     ...:     distances = []
     ...:     for median in district_medians:
     ...:         distances.append(np.absolute(df_train_1.loc[index, "price"] - median))
     ...:     j = distances.index(min(distances))
     ...:     df_train_1.loc[index, "road_name"] = district_medians.index[j][1]
     ...: # check all blank be filled in road_name
     ...: len(df_train_1[df_train_1["road_name"] == ""])
Out[119]: 0
```

  - Price outliers




```
In [121]: df_train_1["price"].describe()
Out[121]:
count    1.053960e+05
mean     1.098405e+07
std      1.287106e+07
min      2.880000e+04
25%      4.813750e+06
50%      7.900000e+06
75%      1.280000e+07
max      1.447676e+09
Name: price, dtype: float64
```

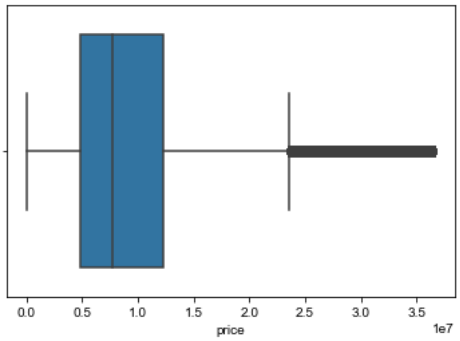
    * There are 3,444 data greater than 3IQR. Replace them with the median prices grouped by road name & build type.

```
In [122]: Q1 = df_train_1['price'].quantile(0.25)
     ...: Q3 = df_train_1['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_1[df_train_1["price"]> Q3 + 3*IQR]) # 3444
     ...: # transformed to the median groupby road_name
     ...: df_train_1 = df_train_1.reset_index(drop=True)
     ...: roadname_buildtype_median = df_train_1.groupby(["road_name", "build_type"])["price"].median()
     ...: for i, price in enumerate(df_train_1["price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_train_1.loc[i, "road_name"]
     ...:         build_type = df_train_1.loc[i, "build_type"]
     ...:         df_train_1.loc[i, "price"] = roadname_buildtype_median[(road_name, build_type)]
```

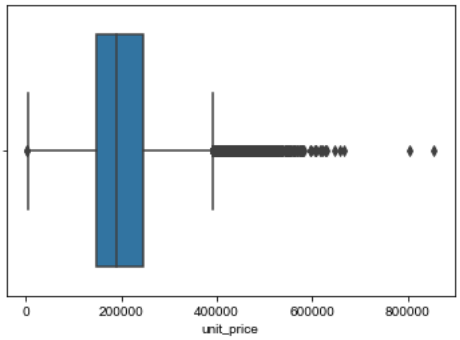
* There are 2,074 data greater than the new 3IQR. Remove them from dataset.

```
In [123]: Q1 = df_train_1['price'].quantile(0.25)
     ...: Q3 = df_train_1['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_1[df_train_1["price" ] > Q3 + 3*IQR]) # 2074
     ...: df_train_1 = df_train_1[~(df_train_1["price"] > Q3 + 3*IQR) | df_train_1["price"].isna()].reset_index(drop = True)
```

* box plot looked better



- price NA(no NA in price)
- Unit_price outliers




```
In [126]: df_train_1["unit_price"].describe()
Out[126]:
count     95025.000000
mean     202940.446693
std       73948.271392
min        1838.000000
25%      149230.000000
50%      190397.000000
75%      246407.000000
max      855425.000000
Name: unit_price, dtype: float64
```

* There are 57 data greater than 3IQR. Replace them with the median prices of grouped by road name and building type.

```
In [128]: Q1 = df_train_1['unit_price'].quantile(0.25)
     ...: Q3 = df_train_1['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_1[df_train_1["unit_price"]> Q3 + 3*IQR]) # 57
     ...: # transformed to the median groupby road_name
     ...: roadname_buildtype_median = df_train_1.groupby(["road_name", "build_type"])["unit_price"].median()
     ...: for i, price in enumerate(df_train_1["unit_price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_train_1.loc[i, "road_name"]
     ...:         build_type = df_train_1.loc[i, "build_type"]
     ...:         df_train_1.loc[i, "unit_price"] = roadname_buildtype_median[(road_name, build_type)]
```

* There are 1 data greater than the new 3IQR. Remove them from dataset.

```
In [129]: Q1 = df_train_1['unit_price'].quantile(0.25)
     ...: Q3 = df_train_1['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_1[df_train_1["unit_price"] > Q3 + 3*IQR]) # 1
     ...: df_train_1 = df_train_1[~(df_train_1["unit_price"] > Q3 + 3*IQR) |
df_train_1["unit_price"].isna()].reset_index(drop = True)
```

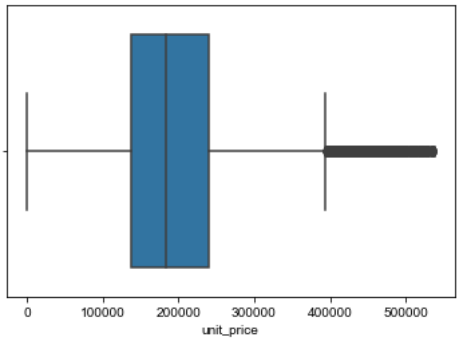
- Unit_price NA

  * Fill NA with the median value grouped by road name and build type. If there are no matched median value, then fill NA with the median value grouped by **district**

and build type.

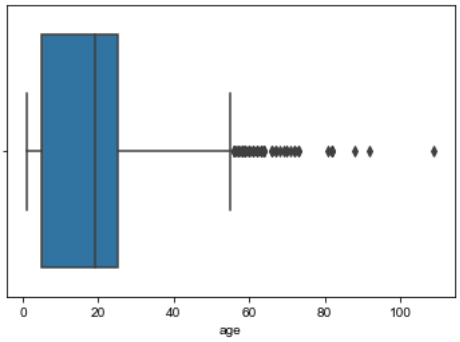
```
In [130]: district_buildtype_median = df_train_1.dropna(subset = ["unit_price"]).groupby(["district", "build_type"])
["unit_price"].median()
     ...: for i, unit_price in enumerate(df_train_1["unit_price"]):
     ...:     if math.isnan(unit_price):
     ...:         if df_train_1.loc[i, "build_type"] == "透天厝":
     ...:             df_train_1.loc[i, "unit_price"] = 0
     ...:         elif df_train_1.loc[i, "road_name"] in roadname_buildtype_median:
     ...:             df_train_1.loc[i, "unit_price"] =
roadname_buildtype_median[(roadname_buildtype_median.index.get_level_values("road_name") == df_train_1.loc[i, "road_name"]) &
(roadname_buildtype_median.index.get_level_values("build_type") == df_train_1.loc[i, "build_type"])].values[0]
     ...:         else:
     ...:             df_train_1.loc[i, "unit_price"] =
district_buildtype_median[(district_buildtype_median.index.get_level_values("district") == df_train_1.loc[i, "district"]) &
(district_buildtype_median.index.get_level_values("build_type") == df_train_1.loc[i, "build_type"])].values[0]
     ...: # check no NA
     ...: df_train_1["unit_price"].isna().sum() # 1
     ...: df_train_1.loc[df_train_1.unit_price.isna(), "unit_price"] = 191015
```

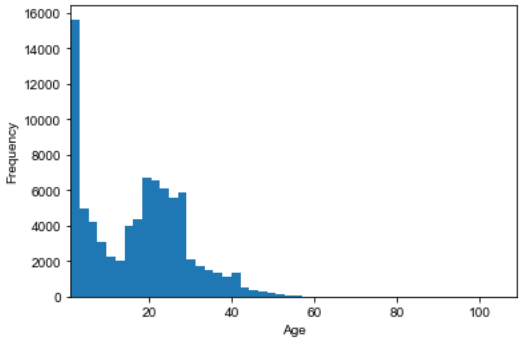
* box plot looked better



- Age outliers

  * Since it is possible for the age of houses to be older than applicable limit (60 years), I'm not going to to remove those outliers.



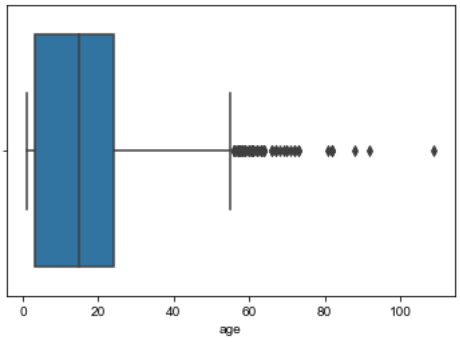


- age NA

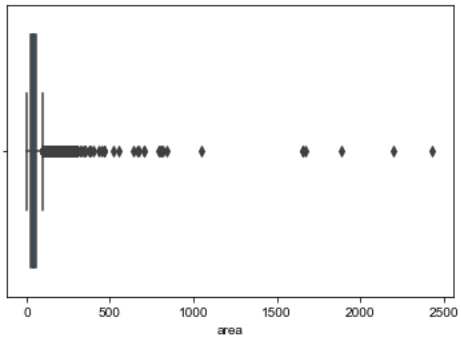
  * Fill NA with the median value grouped by road name and build type. If there are no matched median value, then fill NA with the median value grouped by **district** and build type.

```
In [134]: len(df_train_1[df_train_1["age"].isna()]) # 21305
     ...: age_df = df_train_1[~df_train_1["age"].isna()]
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["age"].median()
     ...: district_buildtype_median = df_train_1.groupby(["district", "build_type"])["age"].median()
     ...: for i, age in enumerate(df_train_1["age"]):
     ...:     if math.isnan(age):
     ...:         district = df_train_1.loc[i, "district"]
     ...:         road_name = df_train_1.loc[i, "road_name"]
     ...:         build_type = df_train_1.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_train_1.loc[i, "age"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_train_1.loc[i, "age"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_train_1["age"].isna().sum()
Out[134]: 0
```

* box plot after imputing



- area outliers



* There are 681 data greater than 3IQR. Replace them with the median prices of grouped by road name and building type.

```
In [137]: Q1 = df_train_1['area'].quantile(0.25)
     ...: Q3 = df_train_1['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_1[df_train_1["area"]> Q3 + 3*IQR]) # 681
     ...: # transformed to the median groupby road_name, buil_type
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: district_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: for i, area in enumerate(df_train_1["area"]):
     ...:     if area > Q3 + 3*IQR:
     ...:         road_name = df_train_1.loc[i, "road_name"]
     ...:         build_type = df_train_1.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median:
     ...:             df_train_1.loc[i, "area"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         elif (district, build_type) in district_buildtype_median:
     ...:             district = df_train_1.loc[i, "district"]
     ...:             df_train_1.loc[i, "area"] = district_buildtype_median[(district, build_type)]
```

* There are 89 data greater than the new 3IQR. Remove them from dataset.

```
Q1 = df_train_1['area'].quantile(0.25)
Q3 = df_train_1['area'].quantile(0.75)
IQR = Q3 - Q1
len(df_train_1[df_train_1["area"] > Q3 + 3*IQR]) # 89
df_train_1 = df_train_1[~(df_train_1["area"] > Q3 + 3*IQR) | df_train_1["area"].isna()].reset_index(drop = True)
```

- area NA(no NA in area)
- build_share1 NA

  * Since it is normal for the building share of detached house(透天厝) to be 0, there are still 592 (8,835-8,243) data have NA.

```
In [139]: len(df_train_1[df_train_1["build_share1"].isna()]) # 8835
     ...: len(df_train_1[df_train_1["build_share1"].isna() & (df_train_1["build_type"] == "透天厝")]) # 8243
     ...: # fill NaN with median build_share1 groupby road_name & build_type
     ...: share_buildtype_df = df_train_1[~(df_train_1["build_share1"].isna()) & (df_train_1["build_type"] != "透天厝")]
     ...: roadname_buildtype_median = share_buildtype_df.groupby(["road_name", "build_type"])["build_share1"].median()
     ...: district_buildtype_median = df_train_1.groupby(["district", "build_type"])["build_share1"].median()
     ...: for i, share in enumerate(df_train_1["build_share1"]):
     ...:     if math.isnan(share):
     ...:         road_name = df_train_1.loc[i, "road_name"]
     ...:         build_type = df_train_1.loc[i, "build_type"]
     ...:         if build_type == "透天厝":
     ...:             df_train_1.loc[i, "build_share1"] = 0
     ...:         elif (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_train_1.loc[i, "build_share1"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_train_1.loc[i, "build_share1"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA

In [140]: df_train_1["build_share1"].isna().sum()
Out[140]: 0
```

- bedroom_num, hall_num, bathroom_num NA

  * There are 1,887 data whose bedroom_num, hall_num, bathroom_num be 0 at the same time. It is unreasonable.

```
In [142]: len(df_train_1[(df_train_1["bedroom_num"] == 0) & (df_train_1["hall_num"] == 0) &
(df_train_1["bathroom_num"] == 0)]) # 1887
Out[142]: 1887
```

  * Fill mode value grouped by road name & building type. If there are multiple modes, then fill in with mean value.

```
In [143]: df_train_1.loc[(df_train_1["bedroom_num"] == 0) & (df_train_1["hall_num"] == 0) & (df_train_1["bathroom_num"] == 0),
["bedroom_num", "hall_num", "bathroom_num"]] = np.nan
     ...: df_train_1 = df_train_1[(df_train_1["bedroom_num"].notna()) & (df_train_1["hall_num"].notna()) &
(df_train_1["bathroom_num"].notna())].reset_index(drop = True)
```

- total_deal_floor NA

  * Since it is normal for the building share of detached house(透天厝) to be 0, there are still 378 data have NA.

```
In [144]: len(df_train_1[(df_train_1["total_deal_floor"] == 0) & (df_train_1["build_type"] != "透天厝")]) # 301
     ...: df_train_1.loc[(df_train_1["total_deal_floor"] == 0) & (df_train_1["build_type"] != "透天厝"), "total_deal_floor"] = np.nan
     ...: df_train_1["total_deal_floor"].isna().sum() #378
```

  * Fill NA with the median value grouped by build type.

```
In [145]: type_totalfloor = df_train_1[df_train_1["build_type"] != "透天厝"]
     ...: type_totalfloor = type_totalfloor[type_totalfloor["total_deal_floor"].notna()]
     ...: buildtype_median = type_totalfloor.groupby("build_type")["total_deal_floor"].median()
     ...: for i, total in enumerate(df_train_1["total_deal_floor"]):
     ...:     if math.isnan(total) :
     ...:         df_train_1.loc[i, "total_deal_floor"] = buildtype_median[df_train_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_train_1["total_deal_floor"].isna().sum()
Out[145]: 0
```

- floor_sold NA

  * Since it is normal for the building share of detached house(透天厝) to be 0, there are still 362 data have NA.

```
In [146]: len(df_train_1[(df_train_1["floor_sold"] == 0) & (df_train_1["build_type"] != "透天厝")]) # 362
     ...: df_train_1.loc[(df_train_1["floor_sold"] == 0) & (df_train_1["build_type"] != "透天厝"), "floor_sold"] = np.nan
     ...: df_train_1["floor_sold"].isna().sum()
Out[146]: 362
```

  * Fill NA with the median value grouped by build type.

```
In [148]: floor_sold = df_train_1[df_train_1["build_type"] != "透天厝"]
     ...: floor_sold = floor_sold[floor_sold["floor_sold"].notna()]
     ...: floor_median = floor_sold.groupby("build_type")["floor_sold"].median()
     ...: for i, floor in enumerate(df_train_1["floor_sold"]):
     ...:     if math.isnan(floor) :
     ...:         df_train_1.loc[i, "floor_sold"] = floor_median[df_train_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_train_1["floor_sold"].isna().sum()
Out[148]: 0
```

- story NA

  * There are still 32 data have NA. Fill in with the median value grouped by build type.

```
In [149]: df_train_1["story"].isna().sum() # 32
     ...: story_median = df_train_1.dropna().groupby("build_type")["story"].median()
     ...: for i, story in enumerate(df_train_1["story"]):
     ...:     if math.isnan(story) :
     ...:         df_train_1.loc[i, "story"] = story_median[df_train_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_train_1["story"].isna().sum()
Out[149]: 0
```

- lon, lat NA

  * One data has 0 values in longitude and latitude.

```
In [150]: df_train_1[(df_train_1["lon"] == 0) | (df_train_1["lat"] == 0)]
Out[150]:
      district road_name  lon  lat  age   area build_type  bedroom_num  \
85834      西屯區       廣興巷  0.0  0.0  2.0  72.21        透天厝          4.0

       hall_num  bathroom_num  story  total_deal_floor  floor_sold elevator  \
85834       2.0           6.0    4.0               0.0         0.0        有

      manager  parking_num  build_share1  unit_price  deal_date       price
85834       有            0           0.0         0.0  111/11/03  27000000.0
```

  * Manual imputation by the mean of its road name

```
In [151]: lon_index = df_train_1[(df_train_1["lon"] == 0) | (df_train_1["lat"] == 0)].index
     ...: df_train_1.loc[lon_index, "lon"] = round(df_train_1[(df_train_1["road_name"] == "廣興巷") & (df_train_1["lon"] != 0)]["lon"].mean(), 4)
     ...: df_train_1.loc[lon_index, "lat"] = round(df_train_1[(df_train_1["road_name"] == "廣興巷") & (df_train_1["lat"] != 0)]["lat"].mean(), 4)
```

- deal_date

  * there exist some typos (month = 00)

```
In [153]: df_date_split = df_train_1["deal_date"].str.split("/", expand = True)

In [154]: df_date_split.columns = ["year", "month", "day"]

In [155]: df_date_split["year"].value_counts()
Out[155]:
102    15523
103    12270
106    10343
108     9685
107     9393
104     8910
109     8271
105     7917
110     7622
111     5902
101     5509
Name: year, dtype: int64
```

```
In [156]: df_date_split["month"].value_counts()
Out[156]:
11    9484
12    9381
10    9306
03    9154
04    8809
09    8776
08    8592
05    8521
07    8103
06    7887
01    7687
02    5644
00       1
Name: month, dtype: int64
```

```
In [157]: df_date_split["day"].value_counts()
Out[157]:
07    3664
25    3649
01    3548
26    3510
15    3502
10    3467
16    3448
05    3407
28    3407
06    3396
20    3395
30    3382
12    3365
22    3327
17    3319
02    3302
11    3293
21    3288
27    3269
18    3258
23    3241
24    3219
13    3171
08    3132
14    3115
09    3087
03    3079
04    3046
29    3033
19    2978
31    2048
Name: day, dtype: int64
```

  * replace 00 month with 3rd mode = 10

```
In [158]: df_train_1["deal_date"] = df_train_1["deal_date"].str.replace(r'(?<=\/)00(?=\/)', '10', regex=True)
```

* transform to datetime and create month feature

```
In [159]:
     ...: year_month_day = []
     ...: for date in df_train_1["deal_date"]:
     ...:     year, month, day = date.split("/")
     ...:     new_year = int(year) + 1911
     ...:     year_month_day.append(str(new_year) + '-' + month + '-' + day)
     ...: df_train_1["deal_date"] = year_month_day
     ...: # extract "month" from "deal year" and create a new column
     ...: df_train_1["month"] = pd.to_datetime(df_train_1['deal_date']).dt.month
```

- Model_1 validation
  - check overall NA and unreasonable values

```
In [160]: len(df_val_1[df_val_1["lon"] == 0]) # 0
     ...: len(df_val_1[df_val_1["lat"] == 0]) # 0
     ...: # nan was replaced with "" in previous manipulation
     ...: len(df_val_1[df_val_1["road_name"] == ""]) # 439
     ...: # it seems wierd that bedroom_num, hall_num and bathroom_num be 0 at the same time
     ...: len(df_val_1[(df_val_1["bedroom_num"] == 0) & (df_val_1["hall_num"] == 0) & (df_val_1["bathroom_num"] == 0)]) # 552
     ...: zero_counts = {}
     ...: for build_type in df_val_1["build_type"].unique():
     ...:     zero_counts[build_type] = {}
     ...:     for col in ["bedroom_num", "hall_num", "bathroom_num"]:
     ...:         zero_count = len(df_val_1[(df_val_1[col] == 0) & (df_val_1["build_type"] == build_type)])
     ...:         zero_counts[build_type][col] = zero_count
     ...: # 0 is normal in total_deal_floor and floor_sold of "透天厝"
     ...: len(df_val_1[(df_val_1["total_deal_floor"] == 0) & (df_val_1["build_type"] != "透天厝")]) # 64
     ...: len(df_val_1[(df_val_1["floor_sold"] == 0) & (df_val_1["build_type"] != "透天厝")]) # 75
```

  - Extract road_name from address for the following imputation.

```
In [161]: road_name_median = df_val_1[df_val_1["road_name"] != ""].groupby(["district","road_name"])["price"].median()
     ...: for index in df_val_1[df_val_1["road_name"] == ""].index:
     ...:     district_medians = road_name_median[road_name_median.index.get_level_values("district") == df_val_1.loc[index, "district"]]
     ...:     distances = []
     ...:     for median in district_medians:
     ...:         distances.append(np.absolute(df_val_1.loc[index, "price"] - median))
     ...:     j = distances.index(min(distances))
     ...:     df_val_1.loc[index, "road_name"] = district_medians.index[j][1]
     ...: # check all blank be filled in road_name
     ...: len(df_val_1[df_val_1["road_name"] == ""])
Out[161]: 0
```

  - Price outliers

    * Same process with training dataset

```
In [164]: Q1 = df_val_1['price'].quantile(0.25)
     ...: Q3 = df_val_1['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_1[df_val_1["price"]> Q3 + 3*IQR]) # 853
     ...: # transformed to the median groupby road_name
     ...: df_val_1 = df_val_1.reset_index(drop=True)
     ...: roadname_buildtype_median = df_val_1.groupby(["road_name", "build_type"])["price"].median()
     ...: for i, price in enumerate(df_val_1["price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_val_1.loc[i, "road_name"]
     ...:         build_type = df_val_1.loc[i, "build_type"]
     ...:         df_val_1.loc[i, "price"] = roadname_buildtype_median[(road_name, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median
     ...: Q1 = df_val_1['price'].quantile(0.25)
     ...: Q3 = df_val_1['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_1[df_val_1["price" ] > Q3 + 3*IQR]) # 592
     ...: df_val_1 = df_val_1[~(df_val_1["price"] > Q3 + 3*IQR) | df_val_1["price"].isna()].reset_index(drop = True)
```

  - Unit_price outliers & NA

    * Same process with training dataset

```
In [165]: df_val_1["unit_price"].describe()
     ...: sns.boxplot(x = "unit_price", data = df_val_1)
     ...: Q1 = df_val_1['unit_price'].quantile(0.25)
     ...: Q3 = df_val_1['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_1[df_val_1["unit_price"]> Q3 + 3*IQR]) # 22
     ...: # transformed to the median groupby road_name
     ...: roadname_buildtype_median = df_val_1.groupby(["road_name", "build_type"])["unit_price"].median()
     ...: for i, price in enumerate(df_val_1["unit_price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_val_1.loc[i, "road_name"]
     ...:         build_type = df_val_1.loc[i, "build_type"]
     ...:         df_val_1.loc[i, "unit_price"] = roadname_buildtype_median[(road_name, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median groupby road_name
     ...: Q1 = df_val_1['unit_price'].quantile(0.25)
     ...: Q3 = df_val_1['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_1[df_val_1["unit_price"] > Q3 + 3*IQR]) # 0
     ...: df_val_1 = df_val_1[~(df_val_1["unit_price"] > Q3 + 3*IQR) | df_val_1["unit_price"].isna()].reset_index(drop = True)
     ...: # fill NaN with roadname_buildtype_median in unit_price
     ...: district_buildtype_median = df_val_1.dropna(subset = ["unit_price"]).groupby(["district", "build_type"])["unit_price"].median()
     ...: for i, unit_price in enumerate(df_val_1["unit_price"]):
     ...:     if math.isnan(unit_price):
     ...:         if df_val_1.loc[i, "build_type"] == "透天厝":
     ...:             df_val_1.loc[i, "unit_price"] = 0
     ...:         elif df_val_1.loc[i, "road_name"] in roadname_buildtype_median:
     ...:             df_val_1.loc[i, "unit_price"] = roadname_buildtype_median[(roadname_buildtype_median.index.get_level_values("road_name"
== df_val_1.loc[i, "road_name"]) & (roadname_buildtype_median.index.get_level_values("build_type") == df_val_1.loc[i,
"build_type"])].values[0]
     ...:         else:
     ...:             df_val_1.loc[i, "unit_price"] = district_buildtype_median[(district_buildtype_median.index.get_level_values("district")
== df_val_1.loc[i, "district"]) & (district_buildtype_median.index.get_level_values("build_type") == df_val_1.loc[i, "build_type"])].values[0
     ...: # check no NA
     ...: df_val_1["unit_price"].isna().sum()
```

- Age NA

  * Same process with training dataset

```
In [166]: len(df_val_1[df_val_1["age"].isna()]) # 5231
     ...: age_df = df_val_1[~df_val_1["age"].isna()]
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["age"].median()
     ...: district_buildtype_median = df_val_1.groupby(["district", "build_type"])["age"].median()
     ...: for i, age in enumerate(df_val_1["age"]):
     ...:     if math.isnan(age):
     ...:         district = df_val_1.loc[i, "district"]
     ...:         road_name = df_val_1.loc[i, "road_name"]
     ...:         build_type = df_val_1.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_val_1.loc[i, "age"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_val_1.loc[i, "age"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_val_1["age"].isna().sum()
Out[166]: 0
```

- area outliers

  * Same process with training dataset

```
In [167]: Q1 = df_val_1['area'].quantile(0.25)
     ...: Q3 = df_val_1['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_1[df_val_1["area"]> Q3 + 3*IQR]) # 143
     ...: # transformed to the median groupby road_name, buil_type
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: district_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: for i, area in enumerate(df_val_1["area"]):
     ...:     if area > Q3 + 3*IQR:
     ...:         road_name = df_val_1.loc[i, "road_name"]
     ...:         build_type = df_val_1.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median:
     ...:             df_val_1.loc[i, "area"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         elif (district, build_type) in district_buildtype_median:
     ...:             district = df_val_1.loc[i, "district"]
     ...:             df_val_1.loc[i, "area"] = district_buildtype_median[(district, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median groupby road_name, buil_type
     ...: Q1 = df_val_1['area'].quantile(0.25)
     ...: Q3 = df_val_1['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_1[df_val_1["area"] > Q3 + 3*IQR]) # 28
     ...: df_val_1 = df_val_1[~(df_val_1["area"] > Q3 + 3*IQR) | df_val_1["area"].isna()].reset_index(drop = True)
```

- build_share1 NA

  * Same process with training dataset

```
In [168]: len(df_val_1[df_val_1["build_share1"].isna()]) # 2209
     ...: len(df_val_1[df_val_1["build_share1"].isna() & (df_val_1["build_type"] == "透天厝")]) # 2061
     ...: # fill NaN with median build_share1 groupby road_name & build_type
     ...: share_buildtype_df = df_val_1[~(df_val_1["build_share1"].isna()) & (df_val_1["build_type"] != "透天厝")]
     ...: roadname_buildtype_median = share_buildtype_df.groupby(["road_name", "build_type"])["build_share1"].median()
     ...: district_buildtype_median = df_val_1.groupby(["district", "build_type"])["build_share1"].median()
     ...: for i, share in enumerate(df_val_1["build_share1"]):
     ...:     if math.isnan(share):
     ...:         road_name = df_val_1.loc[i, "road_name"]
     ...:         build_type = df_val_1.loc[i, "build_type"]
     ...:         if build_type == "透天厝":
     ...:             df_val_1.loc[i, "build_share1"] = 0
     ...:         elif (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_val_1.loc[i, "build_share1"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_val_1.loc[i, "build_share1"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_val_1["build_share1"].isna().sum()
Out[168]: 0
```

- bedroom_num, hall_num, bathroom_num NA

  * Same process with training dataset

```
In [169]:
    ...: len(df_val_1[(df_val_1["bedroom_num"] == 0) & (df_val_1["hall_num"] == 0) & (df_val_1["bathroom_num"] == 0)]) # 391
    ...: # fill mode after groupby road_name&build_type, if there is multiple modes, then fill in with mean
    ...: df_val_1.loc[(df_val_1["bedroom_num"] == 0) & (df_val_1["hall_num"] == 0) & (df_val_1["bathroom_num"] == 0), ["bedroom_num", "hall_num", "bathroom_num"]] = np.nan
    ...: df_val_1 = df_val_1[(df_val_1["bedroom_num"].notna()) & (df_val_1["hall_num"].notna()) & (df_val_1["bathroom_num"].notna())].reset_index(drop = True)
```

- total_deal_floor NA

  * Same process with training dataset

```
In [170]: len(df_val_1[(df_val_1["total_deal_floor"] == 0) & (df_val_1["build_type"] != "透天厝")]) # 60
     ...: df_val_1.loc[(df_val_1["total_deal_floor"] == 0) & (df_val_1["build_type"] != "透天厝"), "total_deal_floor"] = np.nan
     ...: df_val_1["total_deal_floor"].isna().sum() #378
     ...: type_totalfloor = df_val_1[df_val_1["build_type"] != "透天厝"]
     ...: type_totalfloor = type_totalfloor[type_totalfloor["total_deal_floor"].notna()]
     ...: buildtype_median = type_totalfloor.groupby("build_type")["total_deal_floor"].median()
     ...: for i, total in enumerate(df_val_1["total_deal_floor"]):
     ...:     if math.isnan(total) :
     ...:         df_val_1.loc[i, "total_deal_floor"] = buildtype_median[df_val_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_val_1["total_deal_floor"].isna().sum()
Out[170]: 0
```

- floor_sold NA

  * Same process with training dataset

```
In [171]: len(df_val_1[(df_val_1["floor_sold"] == 0) & (df_val_1["build_type"] != "透天厝")]) # 70
     ...: df_val_1.loc[(df_val_1["floor_sold"] == 0) & (df_val_1["build_type"] != "透天厝"), "floor_sold"] = np.nan
     ...: floor_sold = df_val_1[df_val_1["build_type"] != "透天厝"]
     ...: floor_sold = floor_sold[floor_sold["floor_sold"].notna()]
     ...: floor_median = floor_sold.groupby("build_type")["floor_sold"].median()
     ...: for i, floor in enumerate(df_val_1["floor_sold"]):
     ...:     if math.isnan(floor) :
     ...:         df_val_1.loc[i, "floor_sold"] = floor_median[df_val_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_val_1["floor_sold"].isna().sum()
Out[171]: 0
```

- story NA

  * Same process with training dataset

```
In [172]: df_val_1["story"].isna().sum() # 11
     ...: story_median = df_val_1.dropna().groupby("build_type")["story"].median()
     ...: for i, story in enumerate(df_val_1["story"]):
     ...:     if math.isnan(story) :
     ...:         df_val_1.loc[i, "story"] = story_median[df_val_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_val_1["story"].isna().sum()
Out[172]: 0
```

- deal_date

  * there exist some typos (month = 00, day = .5)

```
In [173]: df_date_split = df_val_1["deal_date"].str.split("/", expand = True)
     ...: df_date_split.columns = ["year", "month", "day"]
     ...: df_date_split["year"].value_counts()
Out[173]:
102    3880
103    3028
106    2656
107    2358
108    2311
104    2169
109    2108
105    2041
110    1871
101    1461
111    1456
Name: year, dtype: int64

In [174]: df_date_split["month"].value_counts() # 00 month, replace it with 3ed mode = 10
Out[174]:
10    2437
11    2380
03    2352
12    2252
04    2188
05    2155
09    2143
08    2126
06    2030
07    1968
01    1910
02    1396
00       2
Name: month, dtype: int64
```

```
In [175]: df_date_split["day"].value_counts()
Out[175]:
16    928
26    919
07    903
25    896
22    895
21    891
15    865
10    863
17    857
18    855
01    855
02    846
28    844
12    829
06    825
24    824
23    822
20    812
09    811
05    809
13    807
11    803
30    800
27    779
04    773
03    762
08    758
29    752
19    741
14    708
31    506
.5      1
```

* replace 00 month with 3rd mode = 10, replace .5 day with 25

* transform to datetime and create month feature

```
In [176]:
     ...: df_val_1["deal_date"] = df_val_1["deal_date"].str.replace(r'(?<=\/)00(?=\/)', '10', regex=True)
     ...: # replace .5 with the mode 25
     ...: df_val_1["deal_date"] = df_val_1["deal_date"].str.replace(r'\.5$', "25", regex=True)
     ...: # transform date format and replace deal_date with it
     ...: year_month_day = []
     ...: for date in df_val_1["deal_date"]:
     ...:     year, month, day = date.split("/")
     ...:     new_year = int(year) + 1911
     ...:     year_month_day.append(str(new_year) + '-' + month + '-' + day)
     ...: df_val_1["deal_date"] = year_month_day
     ...: # extract "month" from "deal_year" and create a new column
     ...: df_val_1["month"] = pd.to_datetime(df_val_1['deal_date']).dt.month
```

- Model_1 test
  - check overall NA and unreasonable values

```
In [178]: len(df_test_1[df_test_1["lon"] == 0]) # 0
     ...: len(df_test_1[df_test_1["lat"] == 0]) # 0
     ...: # nan was replaced with "" in previous manipulation
     ...: len(df_test_1[df_test_1["road_name"] == ""]) # 534
     ...: # it seems wierd that bedroom_num, hall_num and bathroom_num be 0 at the same time
     ...: len(df_test_1[(df_test_1["bedroom_num"] == 0) & (df_test_1["hall_num"] == 0) & (df_test_1["bathroom_num"] == 0)]) # 757
     ...: zero_counts = {}
     ...: for build_type in df_test_1["build_type"].unique():
     ...:     zero_counts[build_type] = {}
     ...:     for col in ["bedroom_num", "hall_num", "bathroom_num"]:
     ...:         zero_count = len(df_test_1[(df_test_1[col] == 0) & (df_test_1["build_type"] == build_type)])
     ...:         zero_counts[build_type][col] = zero_count
     ...: # 0 is normal in total_deal_floor and floor_sold of "透天厝"
     ...: len(df_test_1[(df_test_1["total_deal_floor"] == 0) & (df_test_1["build_type"] != "透天厝")]) # 103
     ...: len(df_test_1[(df_test_1["floor_sold"] == 0) & (df_test_1["build_type"] != "透天厝")]) # 119
Out[178]: 119
```

  - Extract road_name from address for the following imputation.

```
In [179]: road_name_median = df_test_1[df_test_1["road_name"] != ""].groupby(["district","road_name"])["price"].median()
     ...: for index in df_test_1[df_test_1["road_name"] == ""].index:
     ...:     district_medians = road_name_median[road_name_median.index.get_level_values("district") == df_test_1.loc[index, "district"]]
     ...:     distances = []
     ...:     for median in district_medians:
     ...:         distances.append(np.absolute(df_test_1.loc[index, "price"] - median))
     ...:     j = distances.index(min(distances))
     ...:     df_test_1.loc[index, "road_name"] = district_medians.index[j][1]
     ...: # check all blank be filled in road_name
     ...: len(df_test_1[df_test_1["road_name"] == ""])
Out[179]: 0
```

  - Price outliers

    * Same process with training dataset

```
In [180]: df_test_1["price"].describe()
     ...: Q1 = df_test_1['price'].quantile(0.25)
     ...: Q3 = df_test_1['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_1[df_test_1["price"]> Q3 + 3*IQR]) # 1099
     ...: # transformed to the median groupby road_name
     ...: df_test_1 = df_test_1.reset_index(drop=True)
     ...: roadname_buildtype_median = df_test_1.groupby(["road_name", "build_type"])["price"].median()
     ...: for i, price in enumerate(df_test_1["price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_test_1.loc[i, "road_name"]
     ...:         build_type = df_test_1.loc[i, "build_type"]
     ...:         df_test_1.loc[i, "price"] = roadname_buildtype_median[(road_name, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median
     ...: Q1 = df_test_1['price'].quantile(0.25)
     ...: Q3 = df_test_1['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_1[df_test_1["price" ] > Q3 + 3*IQR]) # 638
     ...: df_test_1 = df_test_1[~(df_test_1["price"] > Q3 + 3*IQR) |
df_test_1["price"].isna()].reset_index(drop = True)
```

  - Unit_price outliers & NA

    * Same process with training dataset

```
In [181]: df_test_1["unit_price"].describe()
     ...: sns.boxplot(x = "unit_price", data = df_test_1)
     ...: Q1 = df_test_1['unit_price'].quantile(0.25)
     ...: Q3 = df_test_1['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_1[df_test_1["unit_price"]> Q3 + 3*IQR]) # 23
     ...: # transformed to the median groupby road_name
     ...: roadname_buildtype_median = df_test_1.groupby(["road_name", "build_type"])["unit_price"].median()
     ...: for i, price in enumerate(df_test_1["unit_price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_test_1.loc[i, "road_name"]
     ...:         build_type = df_test_1.loc[i, "build_type"]
     ...:         df_test_1.loc[i, "unit_price"] = roadname_buildtype_median[(road_name, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median groupby road_name
     ...: Q1 = df_test_1['unit_price'].quantile(0.25)
     ...: Q3 = df_test_1['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_1[df_test_1["unit_price"] > Q3 + 3*IQR]) # 1
     ...: df_test_1 = df_test_1[~(df_test_1["unit_price"] > Q3 + 3*IQR) | df_test_1["unit_price"].isna()].reset_index(drop = True)
     ...: # fill NaN with roadname_buildtype_median in unit_price
     ...: district_buildtype_median = df_test_1.dropna(subset = ["unit_price"]).groupby(["district", "build_type"])["unit_price"].median()
     ...: for i, unit_price in enumerate(df_test_1["unit_price"]):
     ...:     if math.isnan(unit_price):
     ...:         if df_test_1.loc[i, "build_type"] == "透天厝":
     ...:             df_test_1.loc[i, "unit_price"] = 0
     ...:         elif df_test_1.loc[i, "road_name"] in roadname_buildtype_median:
     ...:             df_test_1.loc[i, "unit_price"] =
roadname_buildtype_median[(roadname_buildtype_median.index.get_level_values("road_name") == df_test_1.loc[i, "road_name"]) &
(roadname_buildtype_median.index.get_level_values("build_type") == df_test_1.loc[i, "build_type"])].values[0]
     ...:         else:
     ...:             df_test_1.loc[i, "unit_price"] =
district_buildtype_median[(district_buildtype_median.index.get_level_values("district") == df_test_1.loc[i, "district"]) &
(district_buildtype_median.index.get_level_values("build_type") == df_test_1.loc[i, "build_type"])].values[0]
```

* There are still NA. Manual imputation by the median of its road name

```
In [183]: df_test_1[df_test_1["unit_price"].isna()] # index: 21170, 22472, 26909
Out[183]:
      district  road_name          lon        lat  age  area build_type  \
21170       北區  賴興里梅川西路四段  120.678779  24.172394  NaN  7.93       住宅大樓
22472       北區  賴興里梅川西路四段  120.678887  24.172494  NaN  7.93       住宅大樓
26909      南屯區         惠中路  120.643351  24.146474  NaN  9.01       住宅大樓

       bedroom_num  hall_num  bathroom_num  story  total_deal_floor  \
21170            0         0             0    NaN               0.0
22472            0         0             0    NaN               0.0
26909            0         0             0    NaN               0.0

       floor_sold elevator manager  parking_num  build_share1  unit_price  \
21170         0.0        無       有            1           NaN         NaN
22472         0.0        無       有            1           NaN         NaN
26909         0.0        無       有            1           NaN         NaN

       deal_date     price
21170  102/12/03  700000.0
22472  102/12/31  750000.0
26909  102/03/01  450000.0

In [184]: df_test_1.loc[df_test_1.index == 21170, "unit_price"] = 215180 # 北區住宅大樓

In [185]: df_test_1.loc[df_test_1.index == 22472, "unit_price"] = 215180 # 北區住宅大樓

In [186]: df_test_1.loc[df_test_1.index == 26909, "unit_price"] = 225002 # 南屯區住宅大樓
```

- Age NA

  * Same process with training dataset

```
In [187]: len(df_test_1[df_test_1["age"].isna()]) # 6584
     ...: age_df = df_test_1[~df_test_1["age"].isna()]
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["age"].median()
     ...: district_buildtype_median = df_test_1.groupby(["district", "build_type"])["age"].median()
     ...: for i, age in enumerate(df_test_1["age"]):
     ...:     if math.isnan(age):
     ...:         district = df_test_1.loc[i, "district"]
     ...:         road_name = df_test_1.loc[i, "road_name"]
     ...:         build_type = df_test_1.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_test_1.loc[i, "age"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_test_1.loc[i, "age"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_test_1["age"].isna().sum()
Out[187]: 0
```

- area outliers

  * Same process with training dataset

```
In [188]: Q1 = df_test_1['area'].quantile(0.25)
     ...: Q3 = df_test_1['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_1[df_test_1["area"]> Q3 + 3*IQR]) # 228
     ...: # transformed to the median groupby road_name, buil_type
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: district_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: for i, area in enumerate(df_test_1["area"]):
     ...:     if area > Q3 + 3*IQR:
     ...:         road_name = df_test_1.loc[i, "road_name"]
     ...:         build_type = df_test_1.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median:
     ...:             df_test_1.loc[i, "area"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         elif (district, build_type) in district_buildtype_median:
     ...:             district = df_test_1.loc[i, "district"]
     ...:             df_test_1.loc[i, "area"] = district_buildtype_median[(district, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median groupby road_name, buil_type
     ...: Q1 = df_test_1['area'].quantile(0.25)
     ...: Q3 = df_test_1['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_1[df_test_1["area"] > Q3 + 3*IQR]) # 22
     ...: df_test_1 = df_test_1[~(df_test_1["area"] > Q3 + 3*IQR) | df_test_1["area"].isna()].reset_index(drop = True)
```

- build_share1 NA

  * Same process with training dataset

```
In [189]: len(df_test_1[df_test_1["build_share1"].isna()]) # 2724
     ...: len(df_test_1[df_test_1["build_share1"].isna() & (df_test_1["build_type"] == "透天厝")]) # 2554
     ...: # fill NaN with median build_share1 groupby road_name & build_type
     ...: share_buildtype_df = df_test_1[~(df_test_1["build_share1"].isna()) & (df_test_1["build_type"] != "透天厝")]
     ...: roadname_buildtype_median = share_buildtype_df.groupby(["road_name", "build_type"])["build_share1"].median()
     ...: district_buildtype_median = df_test_1.groupby(["district", "build_type"])["build_share1"].median()
     ...: for i, share in enumerate(df_test_1["build_share1"]):
     ...:     if math.isnan(share):
     ...:         road_name = df_test_1.loc[i, "road_name"]
     ...:         build_type = df_test_1.loc[i, "build_type"]
     ...:         if build_type == "透天厝":
     ...:             df_test_1.loc[i, "build_share1"] = 0
     ...:         elif (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_test_1.loc[i, "build_share1"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_test_1.loc[i, "build_share1"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_test_1["build_share1"].isna().sum()
Out[189]: 0
```

- bedroom_num, hall_num, bathroom_num NA

  * Same process with training dataset

```
In [190]: len(df_test_1[(df_test_1["bedroom_num"] == 0) & (df_test_1["hall_num"] == 0) & (df_test_1["bathroom_num"] == 0)]) # 613
     ...: # fill mode after groupby road_name&build_type, if there is multiple modes, then fill in with mean
     ...: df_test_1.loc[(df_test_1["bedroom_num"] == 0) & (df_test_1["hall_num"] == 0) & (df_test_1["bathroom_num"] == 0), ["bedroom_num", "hall_num", "bathroom_num"]] =
np.nan
     ...: df_test_1 = df_test_1[(df_test_1["bedroom_num"].notna()) & (df_test_1["hall_num"].notna()) & (df_test_1["bathroom_num"].notna())].reset_index(drop = True)
```

- total_deal_floor NA

  * Same process with training dataset

```
In [191]: len(df_test_1[(df_test_1["total_deal_floor"] == 0) & (df_test_1["build_type"] != "透天厝")]) # 94
     ...: df_test_1.loc[(df_test_1["total_deal_floor"] == 0) & (df_test_1["build_type"] != "透天厝"), "total_deal_floor"] = np.nan
     ...: df_test_1["total_deal_floor"].isna().sum() #378
     ...: type_totalfloor = df_test_1[df_test_1["build_type"] != "透天厝"]
     ...: type_totalfloor = type_totalfloor[type_totalfloor["total_deal_floor"].notna()]
     ...: buildtype_median = type_totalfloor.groupby("build_type")["total_deal_floor"].median()
     ...: for i, total in enumerate(df_test_1["total_deal_floor"]):
     ...:     if math.isnan(total) :
     ...:         df_test_1.loc[i, "total_deal_floor"] = buildtype_median[df_test_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_test_1["total_deal_floor"].isna().sum()
Out[191]: 0
```

- floor_sold NA

  * Same process with training dataset

```
In [192]: len(df_test_1[(df_test_1["floor_sold"] == 0) & (df_test_1["build_type"] != "透天厝")]) # 107
     ...: df_test_1.loc[(df_test_1["floor_sold"] == 0) & (df_test_1["build_type"] != "透天厝"), "floor_sold"] = np.nan
     ...: floor_sold = df_test_1[df_test_1["build_type"] != "透天厝"]
     ...: floor_sold = floor_sold[floor_sold["floor_sold"].notna()]
     ...: floor_median = floor_sold.groupby("build_type")["floor_sold"].median()
     ...: for i, floor in enumerate(df_test_1["floor_sold"]):
     ...:     if math.isnan(floor) :
     ...:         df_test_1.loc[i, "floor_sold"] = floor_median[df_test_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_test_1["floor_sold"].isna().sum()
Out[192]: 0
```

- story NA

  * Same process with training dataset

```
In [193]: df_test_1["story"].isna().sum() # 15
     ...: story_median = df_test_1.dropna().groupby("build_type")["story"].median()
     ...: for i, story in enumerate(df_test_1["story"]):
     ...:     if math.isnan(story) :
     ...:         df_test_1.loc[i, "story"] = story_median[df_test_1.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_test_1["story"].isna().sum()
Out[193]: 0
```

- deal_date

  * No typos.

```
In [194]: df_date_split = df_test_1["deal_date"].str.split("/", expand = True)
     ...: df_date_split.columns = ["year", "month", "day"]
     ...: df_date_split["year"].value_counts()
Out[194]:
102    4765
103    3824
106    3183
108    3079
107    3048
104    2737
109    2620
105    2471
110    2353
111    1905
101    1681
Name: year, dtype: int64

In [195]: df_date_split["month"].value_counts()
Out[195]:
11    2945
12    2939
10    2885
03    2806
09    2737
08    2730
04    2715
05    2644
06    2548
07    2520
01    2465
02    1732
Name: month, dtype: int64
```

```
In [196]: df_date_split["day"].value_counts()
Out[196]:
25    1156
26    1144
07    1137
01    1095
17    1082
15    1080
10    1078
02    1072
06    1070
21    1070
05    1067
28    1064
27    1061
20    1055
16    1048
23    1028
24    1023
14    1023
03    1015
22    1011
19    1007
18     989
30     989
13     987
11     985
08     984
04     966
12     960
09     929
29     915
31     576
Name: day, dtype: int64
```

  * transform to datetime and create month feature

```
In [197]: year_month_day = []
     ...: for date in df_test_1["deal_date"]:
     ...:     year, month, day = date.split("/")
     ...:     new_year = int(year) + 1911
     ...:     year_month_day.append(str(new_year) + '-' + month + '-' + day)
     ...: df_test_1["deal_date"] = year_month_day
     ...: # extract "month" from "deal year" and create a new column
     ...: df_test_1["month"] = pd.to_datetime(df_test_1['deal_date']).dt.month
```

- Model_2 training
  - check overall NA and unreasonable values

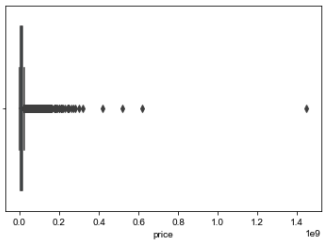
```
In [198]: len(df_train_2[df_train_2["lon"] == 0]) # 1
     ...: len(df_train_2[df_train_2["lat"] == 0]) # 1
     ...: # nan was replaced with "" in previous manipulation
     ...: len(df_train_2[df_train_2["road_name"] == ""]) # 1518
     ...: # it seems wierd that bedroom_num, hall_num and bathroom_num be 0 at the same time
     ...: len(df_train_2[(df_train_2["bedroom_num"] == 0) & (df_train_2["hall_num"] == 0) & (df_train_2["bathroom_num"] == 0)]) # 2407
     ...: zero_counts = {}
     ...: for build_type in df_train_2["build_type"].unique():
     ...:     zero_counts[build_type] = {}
     ...:     for col in ["bedroom_num", "hall_num", "bathroom_num"]:
     ...:         zero_count = len(df_train_2[(df_train_2[col] == 0) & (df_train_2["build_type"] == build_type)])
     ...:         zero_counts[build_type][col] = zero_count
     ...: # 0 is normal in total_deal_floor and floor_sold of "透天厝"
     ...: len(df_train_2[(df_train_2["total_deal_floor"] == 0) & (df_train_2["build_type"] != "透天厝")]) # 405
     ...: len(df_train_2[(df_train_2["floor_sold"] == 0) & (df_train_2["build_type"] != "透天厝")]) # 475
```

  - Since buildings features would be similar in same road and same build type, it would be more real if we impute outliers or NA with the value grouped by them. So, firstly, we need to extract road_name from address for the following imputation.

    *If road_name is blank, imputing it with the road whose median price has the smallest distance with the price of blank.

```
In [199]: road_name_median = df_train_2[df_train_2["road_name"] != ""].groupby(["district","road_name"])["price"].median()
     ...: for index in df_train_2[df_train_2["road_name"] == ""].index:
     ...:     district_medians = road_name_median[road_name_median.index.get_level_values("district") == df_train_2.loc[index, "district"]]
     ...:     distances = []
     ...:     for median in district_medians:
     ...:         distances.append(np.absolute(df_train_2.loc[index, "price"] - median))
     ...:     j = distances.index(min(distances))
     ...:     df_train_2.loc[index, "road_name"] = district_medians.index[j][1]
     ...: # check all blank be filled in road_name
     ...: len(df_train_2[df_train_2["road_name"] == ""])
Out[199]: 0
```

- Price outliers




```
In [201]: df_train_2["price"].describe()
Out[201]:
count    1.033420e+05
mean     1.063401e+07
std      1.253459e+07
min      9.320000e+02
25%      4.700000e+06
50%      7.600000e+06
75%      1.240000e+07
max      1.447676e+09
Name: price, dtype: float64
```

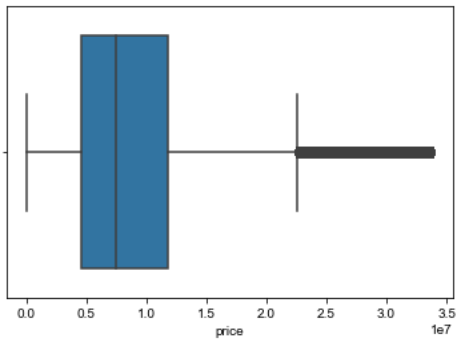
* There are 3,338 data greater than 3IQR. Replace them with the median prices grouped by road name & build type.

```
In [202]: Q1 = df_train_2['price'].quantile(0.25)
     ...: Q3 = df_train_2['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_2[df_train_2["price"]> Q3 + 3*IQR]) # 3338
     ...: # transformed to the median groupby road_name
     ...: df_train_2 = df_train_2.reset_index(drop=True)
     ...: roadname_buildtype_median = df_train_2.groupby(["road_name", "build_type"])["price"].median()
     ...: for i, price in enumerate(df_train_2["price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_train_2.loc[i, "road_name"]
     ...:         build_type = df_train_2.loc[i, "build_type"]
     ...:         df_train_2.loc[i, "price"] = roadname_buildtype_median[(road_name, build_type)]
```

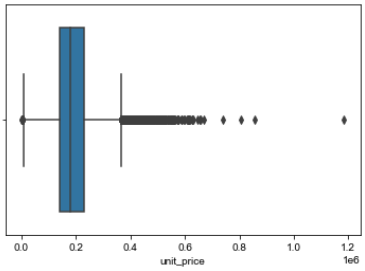
* There are 2,016 data greater than the new 3IQR. Remove them from dataset.

```
In [204]: Q1 = df_train_2['price'].quantile(0.25)
     ...: Q3 = df_train_2['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_2[df_train_2["price" ] > Q3 + 3*IQR]) # 2016
     ...: df_train_2 = df_train_2[~(df_train_2["price"] > Q3 + 3*IQR) | df_train_2["price"].isna()].reset_index(drop = True)
```

* box plot looked better



- price NA(no NA in price)
- Unit_price outliers




```
In [207]: df_train_2["unit_price"].describe()
Out[207]:
count    9.235100e+04
mean     1.906896e+05
std      6.899002e+04
min      1.700000e+01
25%      1.409120e+05
50%      1.781190e+05
75%      2.311110e+05
max      1.184020e+06
Name: unit_price, dtype: float64
```

* There are 124 data greater than 3IQR. Replace them with the median prices of grouped by road name and building type.

```
In [209]: Q1 = df_train_2['unit_price'].quantile(0.25)
     ...: Q3 = df_train_2['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_2[df_train_2["unit_price"]> Q3 + 3*IQR]) # 124
     ...: # transformed to the median groupby road_name
     ...: roadname_buildtype_median = df_train_2.groupby(["road_name", "build_type"])["unit_price"].median()
     ...: for i, price in enumerate(df_train_2["unit_price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_train_2.loc[i, "road_name"]
     ...:         build_type = df_train_2.loc[i, "build_type"]
     ...:         df_train_2.loc[i, "unit_price"] = roadname_buildtype_median[(road_name, build_type)]
```

* No data greater than the new 3IQR.

```
In [210]: Q1 = df_train_2['unit_price'].quantile(0.25)
     ...: Q3 = df_train_2['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_2[df_train_2["unit_price"] > Q3 + 3*IQR]) # 0
```

- Unit_price NA

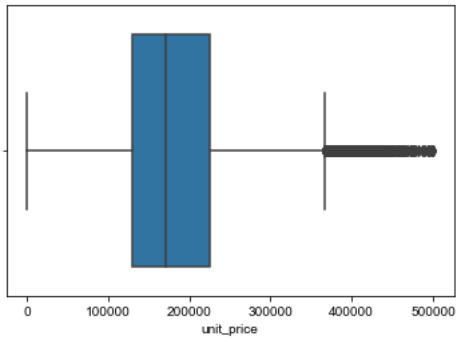
  * Fill NA with the median value grouped by road name and build type. If there are no matched median value, then fill NA with the median value grouped by **district** and build type.

```
In [211]: district_buildtype_median = df_train_2.dropna(subset = ["unit_price"]).groupby(["district", "build_type"])
["unit_price"].median()
     ...: for i, unit_price in enumerate(df_train_2["unit_price"]):
     ...:     if math.isnan(unit_price):
     ...:         if df_train_2.loc[i, "build_type"] == "透天厝":
     ...:             df_train_2.loc[i, "unit_price"] = 0
     ...:         elif df_train_2.loc[i, "road_name"] in roadname_buildtype_median:
     ...:             df_train_2.loc[i, "unit_price"] =
roadname_buildtype_median[(roadname_buildtype_median.index.get_level_values("road_name") == df_train_2.loc[i,
"road_name"]) & (roadname_buildtype_median.index.get_level_values("build_type") == df_train_2.loc[i,
"build_type"])].values[0]
     ...:         else:
     ...:             df_train_2.loc[i, "unit_price"] =
district_buildtype_median[(district_buildtype_median.index.get_level_values("district") == df_train_2.loc[i,
"district"]) & (district_buildtype_median.index.get_level_values("build_type") == df_train_2.loc[i,
"build_type"])].values[0]
```

* There is still one NA. Manual imputation by the median of its road name

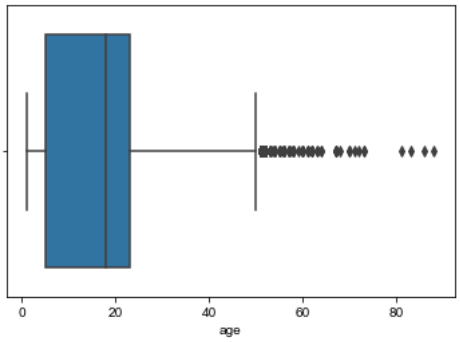
```
In [212]: df_train_2["unit_price"].isna().sum()
Out[212]: 1

In [213]: df_train_2[df_train_2["unit_price"].isna()] # index = 14225
Out[213]:
      district road_name          lon        lat  age   area build_type  \
14122      北屯區         子巷  120.723288  24.165908  1.0  78.03       住宅大樓

       bedroom_num  hall_num  bathroom_num  story  total_deal_floor  \
14122            4         2             2   15.0               1.0

       floor_sold elevator manager  parking_num  build_share1  unit_price  \
14122         7.0        無       有            2        0.4483         NaN

       deal_date  year       price
14122  103/08/09  2014  15210000.0
In [214]: df_train_2.loc[df_train_2.unit_price.isna(), "unit_price"] = 180792
```

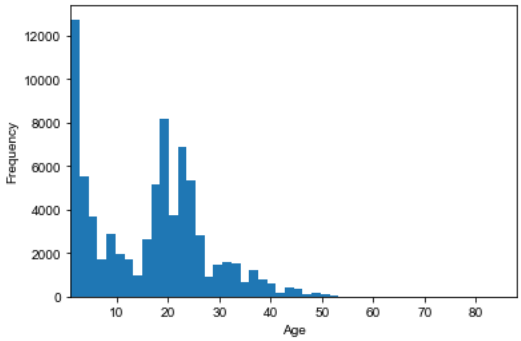
* box plot looked better



- Age outliers

  * Since it is possible for the age of houses to be older than applicable limit (60 years), I'm not going to to remove those outliers.



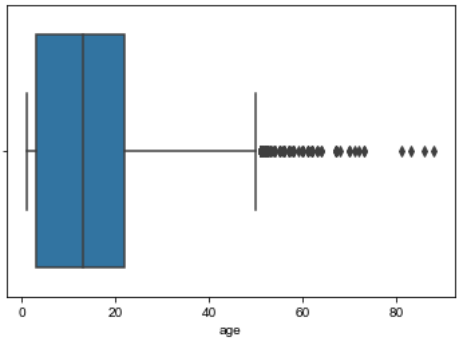


- age NA

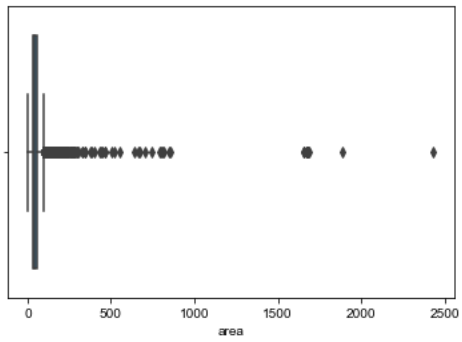
  * Fill NA with the median value grouped by road name and build type. If there are no matched median value, then fill NA with the median value grouped by **district** and build type.

```
In [219]: len(df_train_2[df_train_2["age"].isna()]) # 24485
     ...: age_df = df_train_2[~df_train_2["age"].isna()]
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["age"].median()
     ...: district_buildtype_median = df_train_2.groupby(["district", "build_type"])["age"].median()
     ...: for i, age in enumerate(df_train_2["age"]):
     ...:     if math.isnan(age):
     ...:         district = df_train_2.loc[i, "district"]
     ...:         road_name = df_train_2.loc[i, "road_name"]
     ...:         build_type = df_train_2.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_train_2.loc[i, "age"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_train_2.loc[i, "age"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_train_2["age"].isna().sum()
Out[219]: 0
```

  * box plot after imputing



- area outliers



* There are 688 data greater than 3IQR. Replace them with the median prices of grouped by road name and building type.

```
In [222]: Q1 = df_train_2['area'].quantile(0.25)
     ...: Q3 = df_train_2['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_2[df_train_2["area"]> Q3 + 3*IQR]) # 688
     ...: # transformed to the median groupby road_name, buil_type
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: district_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: for i, area in enumerate(df_train_2["area"]):
     ...:     if area > Q3 + 3*IQR:
     ...:         road_name = df_train_2.loc[i, "road_name"]
     ...:         build_type = df_train_2.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median:
     ...:             df_train_2.loc[i, "area"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         elif (district, build_type) in district_buildtype_median:
     ...:             district = df_train_2.loc[i, "district"]
     ...:             df_train_2.loc[i, "area"] = district_buildtype_median[(district, build_type)]
```

* There are 75 data greater than the new 3IQR. Remove them from dataset.

```
In [223]: Q1 = df_train_2['area'].quantile(0.25)
     ...: Q3 = df_train_2['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_train_2[df_train_2["area"] > Q3 + 3*IQR]) # 75
     ...: df_train_2 = df_train_2[~(df_train_2["area"] > Q3 + 3*IQR) | df_train_2["area"].isna()].reset_index(drop = True)
```

- area NA(no NA in area)
- build_share1 NA

* Since it is normal for the building share of detached house(透天厝) to be 0, there are still 656 (9,085-8,429) data have NA.

```
In [224]: len(df_train_2[df_train_2["build_share1"].isna()]) # 9085
     ...: len(df_train_2[df_train_2["build_share1"].isna() & (df_train_2["build_type"] == "透天厝")]) # 8429
     ...: # fill NaN with median build_share1 groupby road_name & build_type
     ...: share_buildtype_df = df_train_2[~(df_train_2["build_share1"].isna()) & (df_train_2["build_type"] != "透天厝")]
     ...: roadname_buildtype_median = share_buildtype_df.groupby(["road_name", "build_type"])["build_share1"].median()
     ...: district_buildtype_median = df_train_2.groupby(["district", "build_type"])["build_share1"].median()
     ...: for i, share in enumerate(df_train_2["build_share1"]):
     ...:     if math.isnan(share):
     ...:         road_name = df_train_2.loc[i, "road_name"]
     ...:         build_type = df_train_2.loc[i, "build_type"]
     ...:         if build_type == "透天厝":
     ...:             df_train_2.loc[i, "build_share1"] = 0
     ...:         elif (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_train_2.loc[i, "build_share1"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_train_2.loc[i, "build_share1"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_train_2["build_share1"].isna().sum()
Out[224]: 0
```

- bedroom_num, hall_num, bathroom_num NA

* There are 1,776 data whose bedroom_num, hall_num, bathroom_num be 0 at the same time. It is unreasonable.

```
In [226]: len(df_train_2[(df_train_2["bedroom_num"] == 0) & (df_train_2["hall_num"] == 0) & (df_train_2["bathroom_num"] == 0)]) # 1776
Out[226]: 1776
```

* Fill mode value grouped by road name & building type. If there are multiple modes, then fill in with mean value.

```
In [227]: df_train_2.loc[(df_train_2["bedroom_num"] == 0) & (df_train_2["hall_num"] == 0) & (df_train_2["bathroom_num"] == 0),
["bedroom_num", "hall_num", "bathroom_num"]] = np.nan
     ...: df_train_2 = df_train_2[(df_train_2["bedroom_num"].notna()) & (df_train_2["hall_num"].notna()) &
(df_train_2["bathroom_num"].notna())].reset_index(drop = True)
```

- total_deal_floor NA

  * Since it is normal for the building share of detached house(透天厝) to be 0, there are still 371 data have NA.

  * Fill NA with the median value grouped by build type.

```
In [228]: len(df_train_2[(df_train_2["total_deal_floor"] == 0) & (df_train_2["build_type"] != "透天厝")]) # 371
     ...: df_train_2.loc[(df_train_2["total_deal_floor"] == 0) & (df_train_2["build_type"] != "透天厝"), "total_deal_floor"] = np.nan
     ...: df_train_2["total_deal_floor"].isna().sum()
     ...: type_totalfloor = df_train_2[df_train_2["build_type"] != "透天厝"]
     ...: type_totalfloor = type_totalfloor[type_totalfloor["total_deal_floor"].notna()]
     ...: buildtype_median = type_totalfloor.groupby("build_type")["total_deal_floor"].median()
     ...: for i, total in enumerate(df_train_2["total_deal_floor"]):
     ...:     if math.isnan(total) :
     ...:         df_train_2.loc[i, "total_deal_floor"] = buildtype_median[df_train_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_train_1["total_deal_floor"].isna().sum()
Out[228]: 0
```

- floor_sold NA

  * Since it is normal for the building share of detached house(透天厝) to be 0, there are still 431 data have NA.

```
In [229]: len(df_train_2[(df_train_2["floor_sold"] == 0) & (df_train_2["build_type"] != "透天厝")]) # 431
Out[229]: 431
```

  * Fill NA with the median value grouped by build type.

```
In [230]: df_train_2.loc[(df_train_2["floor_sold"] == 0) & (df_train_2["build_type"] != "透天厝"), "floor_sold"] = np.nan
     ...: floor_sold = df_train_2[df_train_2["build_type"] != "透天厝"]
     ...: floor_sold = floor_sold[floor_sold["floor_sold"].notna()]
     ...: floor_median = floor_sold.groupby("build_type")["floor_sold"].median()
     ...: for i, floor in enumerate(df_train_2["floor_sold"]):
     ...:     if math.isnan(floor) :
     ...:         df_train_2.loc[i, "floor_sold"] = floor_median[df_train_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_train_2["floor_sold"].isna().sum()
Out[230]: 0
```

- story NA

  * There are still 43 data have NA. Fill in with the median value grouped by build type.

```
In [231]: df_train_2["story"].isna().sum() # 43
     ...: story_median = df_train_2.dropna().groupby("build_type")["story"].median()
     ...: for i, story in enumerate(df_train_2["story"]):
     ...:     if math.isnan(story) :
     ...:         df_train_2.loc[i, "story"] = story_median[df_train_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_train_2["story"].isna().sum()
Out[231]: 0
```

- lon, lat NA

  * One data has 0 values in longitude and latitude.

```
In [150]: df_train_1[(df_train_1["lon"] == 0) | (df_train_1["lat"] == 0)]
Out[150]:
      district road_name  lon  lat  age   area build_type  bedroom_num  \
85834       西屯區       廣興巷  0.0  0.0  2.0  72.21        透天厝          4.0

       hall_num  bathroom_num  story  total_deal_floor  floor_sold elevator  \
85834       2.0           6.0    4.0               0.0         0.0        有

      manager  parking_num  build_share1  unit_price  deal_date       price
85834       有            0           0.0         0.0  111/11/03  27000000.0
```

* Manual imputation by the mean of its road name

```
In [151]: lon_index = df_train_1[(df_train_1["lon"] == 0) | (df_train_1["lat"] == 0)].index
     ...: df_train_1.loc[lon_index, "lon"] = round(df_train_1[(df_train_1["road_name"] == "廣興巷") & (df_train_1["lon"] != 0)]["lon"].mean(), 4)
     ...: df_train_1.loc[lon_index, "lat"] = round(df_train_1[(df_train_1["road_name"] == "廣興巷") & (df_train_1["lat"] != 0)]["lat"].mean(), 4)
```

- deal_date

  * there exist some typos (month = 00)

```
In [239]: df_date_split = df_train_2["deal_date"].str.split("/", expand = True)
     ...: df_date_split.columns = ["year", "month", "day"]
     ...: df_date_split["year"].value_counts()
Out[239]:
102    19203
103    15246
106    12875
108    12018
107    11710
104    10980
105     9889
101     6872
Name: year, dtype: int64

In [240]: df_date_split["month"].value_counts() # 00 month, replace it with 2nd mode = 10
Out[240]:
11    9439
10    9387
12    9337
03    8642
09    8526
08    8445
04    8404
05    8318
06    8070
07    7734
01    7395
02    5094
00       2
Name: month, dtype: int64
```

```
In [241]: df_date_split["day"].value_counts()
Out[241]:
25    3637
07    3605
26    3583
15    3462
01    3404
16    3360
10    3358
28    3349
06    3339
02    3311
21    3310
30    3309
22    3270
17    3267
05    3261
20    3248
27    3170
18    3168
12    3130
24    3126
11    3109
23    3094
13    3059
08    3054
03    3047
04    3030
09    3009
14    2971
29    2897
19    2852
31    2004
Name: day, dtype: int64
```

  * replace 00 month with 3rd mode = 10

```
In [242]: df_train_2["deal_date"] = df_train_2["deal_date"].str.replace(r'(?<=\/)00(?=\/)', '10', regex=True)
```

  * transform to datetime and create month feature

```
In [243]: year_month_day = []
     ...: for date in df_train_2["deal_date"]:
     ...:     year, month, day = date.split("/")
     ...:     new_year = int(year) + 1911
     ...:     year_month_day.append(str(new_year) + '-' + month + '-' + day)
     ...: df_train_2["deal_date"] = year_month_day
     ...: # extract "month" from "deal year" and create a new column
     ...: df_train_2["month"] = pd.to_datetime(df_train_2['deal_date']).dt.month
```

- Model_1 validation

  - check overall NA and unreasonable values

```
In [244]: len(df_val_2[df_val_2["lon"] == 0]) # 0
     ...: len(df_val_2[df_val_2["lat"] == 0]) # 0
     ...: # nan was replaced with "" in previous manipulation
     ...: len(df_val_2[df_val_2["road_name"] == ""]) # 380
     ...: # it seems wierd that bedroom_num, hall_num and bathroom_num be 0 at the same time
     ...: len(df_val_2[(df_val_2["bedroom_num"] == 0) & (df_val_2["hall_num"] == 0) & (df_val_2["bathroom_num"] == 0)]) # 622
     ...: zero_counts = {}
     ...: for build_type in df_val_2["build_type"].unique():
     ...:     zero_counts[build_type] = {}
     ...:     for col in ["bedroom_num", "hall_num", "bathroom_num"]:
     ...:         zero_count = len(df_val_2[(df_val_2[col] == 0) & (df_val_2["build_type"] == build_type)])
     ...:         zero_counts[build_type][col] = zero_count
     ...: # 0 is normal in total_deal_floor and floor_sold of "透天厝"
     ...: len(df_val_2[(df_val_2["total_deal_floor"] == 0) & (df_val_2["build_type"] != "透天厝")]) # 99
     ...: len(df_val_2[(df_val_2["floor_sold"] == 0) & (df_val_2["build_type"] != "透天厝")]) # 111
```

  - Extract road_name from address for the following imputation.

```
In [245]: road_name_median = df_val_2[df_val_2["road_name"] != ""].groupby(["district","road_name"])["price"].median()
     ...: for index in df_val_2[df_val_2["road_name"] == ""].index:
     ...:     district_medians = road_name_median[road_name_median.index.get_level_values("district") == df_val_2.loc[index, "district"]]
     ...:     distances = []
     ...:     for median in district_medians:
     ...:         distances.append(np.absolute(df_val_2.loc[index, "price"] - median))
     ...:     j = distances.index(min(distances))
     ...:     df_val_2.loc[index, "road_name"] = district_medians.index[j][1]
     ...: # check all blank be filled in road_name
     ...: len(df_val_2[df_val_2["road_name"] == ""])
Out[245]: 0
```

  - Price outliers

    * Same process with training dataset

```
In [246]: df_val_2["price"].describe()
     ...: Q1 = df_val_2['price'].quantile(0.25)
     ...: Q3 = df_val_2['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_2[df_val_2["price"]> Q3 + 3*IQR]) # 836
     ...: # transformed to the median groupby road_name
     ...: df_val_2 = df_val_2.reset_index(drop=True)
     ...: roadname_buildtype_median = df_val_2.groupby(["road_name", "build_type"])["price"].median()
     ...: for i, price in enumerate(df_val_2["price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_val_2.loc[i, "road_name"]
     ...:         build_type = df_val_2.loc[i, "build_type"]
     ...:         df_val_2.loc[i, "price"] = roadname_buildtype_median[(road_name, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median
     ...: Q1 = df_val_2['price'].quantile(0.25)
     ...: Q3 = df_val_2['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_2[df_val_2["price" ] > Q3 + 3*IQR]) # 517
     ...: df_val_2 = df_val_2[~(df_val_2["price"] > Q3 + 3*IQR) | df_val_2["price"].isna()].reset_index(drop = True)
```

- Unit_price outliers & NA

  * Same process with training dataset

```
In [247]: Q1 = df_val_2['unit_price'].quantile(0.25)
     ...: Q3 = df_val_2['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_2[df_val_2["unit_price"]> Q3 + 3*IQR]) # 37
     ...: # transformed to the median groupby road_name
     ...: roadname_buildtype_median = df_val_2.groupby(["road_name", "build_type"])["unit_price"].median()
     ...: for i, price in enumerate(df_val_2["unit_price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_val_2.loc[i, "road_name"]
     ...:         build_type = df_val_2.loc[i, "build_type"]
     ...:         df_val_2.loc[i, "unit_price"] = roadname_buildtype_median[(road_name, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median groupby road_name
     ...: Q1 = df_val_2['unit_price'].quantile(0.25)
     ...: Q3 = df_val_2['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_2[df_val_2["unit_price"] > Q3 + 3*IQR]) # 0
     ...: df_val_2 = df_val_2[~(df_val_2["unit_price"] > Q3 + 3*IQR) | df_val_2["unit_price"].isna()].reset_index(drop = True)
     ...: # fill NaN with roadname_buildtype_median in unit_price
     ...: district_buildtype_median = df_val_2.dropna(subset = ["unit_price"]).groupby(["district", "build_type"])["unit_price"].median()
     ...: for i, unit_price in enumerate(df_val_2["unit_price"]):
     ...:     if math.isnan(unit_price):
     ...:         if df_val_2.loc[i, "build_type"] == "透天厝":
     ...:             df_val_2.loc[i, "unit_price"] = 0
     ...:         elif df_val_2.loc[i, "road_name"] in roadname_buildtype_median:
     ...:             df_val_2.loc[i, "unit_price"] =
roadname_buildtype_median[(roadname_buildtype_median.index.get_level_values("road_name") == df_val_2.loc[i, "road_name"]) &
(roadname_buildtype_median.index.get_level_values("build_type") == df_val_2.loc[i, "build_type"])].values[0]
     ...:         else:
     ...:             df_val_2.loc[i, "unit_price"] =
district_buildtype_median[(district_buildtype_median.index.get_level_values("district") == df_val_2.loc[i, "district"]) &
(district_buildtype_median.index.get_level_values("build_type") == df_val_2.loc[i, "build_type"])].values[0]
     ...: # check no NA
     ...: df_val_2["unit_price"].isna().sum()
Out[247]: 0
```

- Age NA

  * Same process with training dataset

```
In [249]: len(df_val_2[df_val_2["age"].isna()]) # 5978
     ...: age_df = df_val_2[~df_val_2["age"].isna()]
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["age"].median()
     ...: district_buildtype_median = df_val_2.groupby(["district", "build_type"])["age"].median()
     ...: for i, age in enumerate(df_val_2["age"]):
     ...:     if math.isnan(age):
     ...:         district = df_val_2.loc[i, "district"]
     ...:         road_name = df_val_2.loc[i, "road_name"]
     ...:         build_type = df_val_2.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_val_2.loc[i, "age"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_val_2.loc[i, "age"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_val_2["age"].isna().sum()
Out[249]: 0
```

- area outliers

  * Same process with training dataset

```
In [250]: Q1 = df_val_2['area'].quantile(0.25)
     ...: Q3 = df_val_2['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_2[df_val_2["area"]> Q3 + 3*IQR]) # 152
     ...: # transformed to the median groupby road_name, buil_type
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: district_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: for i, area in enumerate(df_val_2["area"]):
     ...:     if area > Q3 + 3*IQR:
     ...:         road_name = df_val_2.loc[i, "road_name"]
     ...:         build_type = df_val_2.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median:
     ...:             df_val_2.loc[i, "area"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         elif (district, build_type) in district_buildtype_median:
     ...:             district = df_val_2.loc[i, "district"]
     ...:             df_val_2.loc[i, "area"] = district_buildtype_median[(district, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median groupby road_name, buil_type
     ...: Q1 = df_val_2['area'].quantile(0.25)
     ...: Q3 = df_val_2['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_val_2[df_val_2["area"] > Q3 + 3*IQR]) # 20
     ...: df_val_2 = df_val_2[~(df_val_2["area"] > Q3 + 3*IQR) | df_val_2["area"].isna()].reset_index(drop = True)
```

- build_share1 NA

  * Same process with training dataset

```
In [251]: len(df_val_2[df_val_2["build_share1"].isna()]) # 2236
     ...: len(df_val_2[df_val_2["build_share1"].isna() & (df_val_2["build_type"] == "透天厝")]) # 2035
     ...: # fill NaN with median build_share1 groupby road_name & build_type
     ...: share_buildtype_df = df_val_2[~(df_val_2["build_share1"].isna()) & (df_val_2["build_type"] != "透天厝")]
     ...: roadname_buildtype_median = share_buildtype_df.groupby(["road_name", "build_type"])["build_share1"].median()
     ...: district_buildtype_median = df_val_2.groupby(["district", "build_type"])["build_share1"].median()
     ...: for i, share in enumerate(df_val_2["build_share1"]):
     ...:     if math.isnan(share):
     ...:         road_name = df_val_2.loc[i, "road_name"]
     ...:         build_type = df_val_2.loc[i, "build_type"]
     ...:         if build_type == "透天厝":
     ...:             df_val_2.loc[i, "build_share1"] = 0
     ...:         elif (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_val_2.loc[i, "build_share1"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_val_2.loc[i, "build_share1"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_val_2["build_share1"].isna().sum()
Out[251]: 0
```

- bedroom_num, hall_num, bathroom_num NA

  * Same process with training dataset

```
In [252]: len(df_val_2[(df_val_2["bedroom_num"] == 0) & (df_val_2["hall_num"] == 0) & (df_val_2["bathroom_num"] == 0)]) # 481
     ...: # fill mode after groupby road_name&build_type, if there is multiple modes, then fill in with mean
     ...: df_val_2.loc[(df_val_2["bedroom_num"] == 0) & (df_val_2["hall_num"] == 0) & (df_val_2["bathroom_num"] == 0), ["bedroom_num", "hall_num", "bathroom_num"]] = np.nan
     ...: df_val_2 = df_val_2[(df_val_2["bedroom_num"].notna()) & (df_val_2["hall_num"].notna()) & (df_val_2["bathroom_num"].notna())].reset_index(drop = True)
```

- total_deal_floor NA

  * Same process with training dataset

```
In [253]: len(df_val_2[(df_val_2["total_deal_floor"] == 0) & (df_val_2["build_type"] != "透天厝")]) # 83
     ...: df_val_2.loc[(df_val_2["total_deal_floor"] == 0) & (df_val_2["build_type"] != "透天厝"), "total_deal_floor"] = np.nan
     ...: df_val_2["total_deal_floor"].isna().sum() #378
     ...: type_totalfloor = df_val_2[df_val_2["build_type"] != "透天厝"]
     ...: type_totalfloor = type_totalfloor[type_totalfloor["total_deal_floor"].notna()]
     ...: buildtype_median = type_totalfloor.groupby("build_type")["total_deal_floor"].median()
     ...: for i, total in enumerate(df_val_2["total_deal_floor"]):
     ...:     if math.isnan(total) :
     ...:         df_val_2.loc[i, "total_deal_floor"] = buildtype_median[df_val_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_val_2["total_deal_floor"].isna().sum()
Out[253]: 0
```

- floor_sold NA

  * Same process with training dataset

```
In [254]: len(df_val_2[(df_val_2["floor_sold"] == 0) & (df_val_2["build_type"] != "透天厝")]) # 93
     ...: df_val_2.loc[(df_val_2["floor_sold"] == 0) & (df_val_2["build_type"] != "透天厝"), "floor_sold"] = np.nan
     ...: floor_sold = df_val_2[df_val_2["build_type"] != "透天厝"]
     ...: floor_sold = floor_sold[floor_sold["floor_sold"].notna()]
     ...: floor_median = floor_sold.groupby("build_type")["floor_sold"].median()
     ...: for i, floor in enumerate(df_val_2["floor_sold"]):
     ...:     if math.isnan(floor) :
     ...:         df_val_2.loc[i, "floor_sold"] = floor_median[df_val_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_val_2["floor_sold"].isna().sum()
Out[254]: 0
```

- story NA

  * Same process with training dataset

```
In [255]: df_val_2["story"].isna().sum() # 11
     ...: story_median = df_val_2.dropna().groupby("build_type")["story"].median()
     ...: for i, story in enumerate(df_val_2["story"]):
     ...:     if math.isnan(story) :
     ...:         df_val_2.loc[i, "story"] = story_median[df_val_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_val_2["story"].isna().sum()
Out[255]: 0
```

- deal_date

  * there exist some typos (month = 00, day = .5)

```
In [259]: df_date_split = df_val_2["deal_date"].str.split("/", expand = True)
     ...: df_date_split.columns = ["year", "month", "day"]
     ...: df_date_split["year"].value_counts()
Out[259]:
102    4865
103    3810
106    3226
107    3008
108    2975
104    2714
105    2483
101    1747
Name: year, dtype: int64

In [260]: df_date_split["month"].value_counts()
Out[260]:
11    2437
10    2359
12    2346
03    2214
04    2138
09    2130
08    2127
05    2099
06    1973
07    1918
01    1854
02    1233
Name: month, dtype: int64
```

```
In [261]: df_date_split["day"].value_counts()
Out[261]:
07    923
01    907
25    903
16    869
20    860
10    857
28    843
26    839
23    838
22    836
15    836
05    829
02    825
27    824
12    817
21    812
24    811
06    810
30    806
18    791
03    780
11    771
17    768
08    753
13    747
14    746
04    746
09    742
19    728
29    719
31    492
Name: day, dtype: int64
```

  * transform to datetime and create month feature

```
In [262]: year_month_day = []
     ...: for date in df_val_2["deal_date"]:
     ...:     year, month, day = date.split("/")
     ...:     new_year = int(year) + 1911
     ...:     year_month_day.append(str(new_year) + '-' + month + '-' + day)
     ...: df_val_2["deal_date"] = year_month_day
     ...: # extract "month" from "deal year" and create a new column
     ...: df_val_2["month"] = pd.to_datetime(df_val_2['deal_date']).dt.month
```

- Model_1 test

  - check overall NA and unreasonable values

```
In [263]: len(df_test_2[df_test_2["lon"] == 0]) # 1
     ...: len(df_test_2[df_test_2["lat"] == 0]) # 1
     ...: # nan was replaced with "" in previous manipulation
     ...: len(df_test_2[df_test_2["road_name"] == ""]) # 745
     ...: # it seems wierd that bedroom_num, hall_num and bathroom_num be 0 at the same time
     ...: len(df_test_2[(df_test_2["bedroom_num"] == 0) & (df_test_2["hall_num"] == 0) & (df_test_2["bathroom_num"] == 0)]) # 642
     ...: zero_counts = {}
     ...: for build_type in df_test_2["build_type"].unique():
     ...:     zero_counts[build_type] = {}
     ...:     for col in ["bedroom_num", "hall_num", "bathroom_num"]:
     ...:         zero_count = len(df_test_2[(df_test_2[col] == 0) & (df_test_2["build_type"] == build_type)])
     ...:         zero_counts[build_type][col] = zero_count
     ...: # 0 is normal in total_deal_floor and floor_sold of "透天厝"
     ...: len(df_test_2[(df_test_2["total_deal_floor"] == 0) & (df_test_2["build_type"] != "透天厝")]) # 3
     ...: len(df_test_2[(df_test_2["floor_sold"] == 0) & (df_test_2["build_type"] != "透天厝")]) # 19
```

  - Extract road_name from address for the following imputation.

```
In [264]: road_name_median = df_test_2[df_test_2["road_name"] != ""].groupby(["district","road_name"])["price"].median()
     ...: for index in df_test_2[df_test_2["road_name"] == ""].index:
     ...:     district_medians = road_name_median[road_name_median.index.get_level_values("district") == df_test_2.loc[index, "district"]]
     ...:     distances = []
     ...:     for median in district_medians:
     ...:         distances.append(np.absolute(df_test_2.loc[index, "price"] - median))
     ...:     j = distances.index(min(distances))
     ...:     df_test_2.loc[index, "road_name"] = district_medians.index[j][1]
     ...: # check all blank be filled in road_name
     ...: len(df_test_2[df_test_2["road_name"] == ""])
Out[264]: 0
```

  - Price outliers

    * Same process with training dataset

```
In [268]: Q1 = df_test_2['price'].quantile(0.25)
     ...: Q3 = df_test_2['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_2[df_test_2["price"]> Q3 + 3*IQR]) # 1131
     ...: # transformed to the median groupby road_name
     ...: df_test_2 = df_test_2.reset_index(drop=True)
     ...: roadname_buildtype_median = df_test_2.groupby(["road_name", "build_type"])["price"].median()
     ...: for i, price in enumerate(df_test_2["price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_test_2.loc[i, "road_name"]
     ...:         build_type = df_test_2.loc[i, "build_type"]
     ...:         df_test_2.loc[i, "price"] = roadname_buildtype_median[(road_name, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median
     ...: Q1 = df_test_2['price'].quantile(0.25)
     ...: Q3 = df_test_2['price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_2[df_test_2["price" ] > Q3 + 3*IQR]) # 639
     ...: df_test_2 = df_test_2[~(df_test_2["price"] > Q3 + 3*IQR) | df_test_2["price"].isna()].reset_index(drop = True)
```

- Unit_price outliers & NA

  * Same process with training dataset

```
In [269]: Q1 = df_test_2['unit_price'].quantile(0.25)
     ...: Q3 = df_test_2['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_2[df_test_2["unit_price"]> Q3 + 3*IQR]) # 107
     ...: # transformed to the median groupby road_name
     ...: roadname_buildtype_median = df_test_2.groupby(["road_name", "build_type"])["unit_price"].median()
     ...: for i, price in enumerate(df_test_2["unit_price"]):
     ...:     if price > Q3 + 3*IQR:
     ...:         road_name = df_test_2.loc[i, "road_name"]
     ...:         build_type = df_test_2.loc[i, "build_type"]
     ...:         df_test_2.loc[i, "unit_price"] = roadname_buildtype_median[(road_name, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median groupby road_name
     ...: Q1 = df_test_2['unit_price'].quantile(0.25)
     ...: Q3 = df_test_2['unit_price'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_2[df_test_2["unit_price"] > Q3 + 3*IQR]) # 0
     ...: df_test_2 = df_test_2[~(df_test_2["unit_price"] > Q3 + 3*IQR) | df_test_2["unit_price"].isna()].reset_index(drop = True)
     ...: # fill NaN with roadname_buildtype_median in unit_price
     ...: district_buildtype_median = df_test_2.dropna(subset = ["unit_price"]).groupby(["district", "build_type"])["unit_price"].median()
     ...: for i, unit_price in enumerate(df_test_2["unit_price"]):
     ...:     if math.isnan(unit_price):
     ...:         if df_test_2.loc[i, "build_type"] == "透天厝":
     ...:             df_test_2.loc[i, "unit_price"] = 0
     ...:         elif df_test_2.loc[i, "road_name"] in roadname_buildtype_median:
     ...:             df_test_2.loc[i, "unit_price"] =
roadname_buildtype_median[(roadname_buildtype_median.index.get_level_values("road_name") == df_test_2.loc[i, "road_name"]) &
(roadname_buildtype_median.index.get_level_values("build_type") == df_test_2.loc[i, "build_type"])].values[0]
     ...:         else:
     ...:             df_test_2.loc[i, "unit_price"] =
district_buildtype_median[(district_buildtype_median.index.get_level_values("district") == df_test_2.loc[i, "district"]) &
(district_buildtype_median.index.get_level_values("build_type") == df_test_2.loc[i, "build_type"])].values[0]
     ...: # check no NA
     ...: df_test_2["unit_price"].isna().sum() # 0
Out[269]: 0
```

- Age NA

  * Same process with training dataset

```
In [270]: len(df_test_2[df_test_2["age"].isna()]) # 2700
     ...: age_df = df_test_2[~df_test_2["age"].isna()]
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["age"].median()
     ...: district_buildtype_median = df_test_2.groupby(["district", "build_type"])["age"].median()
     ...: for i, age in enumerate(df_test_2["age"]):
     ...:     if math.isnan(age):
     ...:         district = df_test_2.loc[i, "district"]
     ...:         road_name = df_test_2.loc[i, "road_name"]
     ...:         build_type = df_test_2.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_test_2.loc[i, "age"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_test_2.loc[i, "age"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_test_2["age"].isna().sum()
Out[270]: 0
```

- area outliers

  * Same process with training dataset

```
In [271]: Q1 = df_test_2['area'].quantile(0.25)
     ...: Q3 = df_test_2['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_2[df_test_2["area"]> Q3 + 3*IQR]) # 268
     ...: # transformed to the median groupby road_name, buil_type
     ...: roadname_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: district_buildtype_median = age_df.groupby(["road_name", "build_type"])["area"].median()
     ...: for i, area in enumerate(df_test_2["area"]):
     ...:     if area > Q3 + 3*IQR:
     ...:         road_name = df_test_2.loc[i, "road_name"]
     ...:         build_type = df_test_2.loc[i, "build_type"]
     ...:         if (road_name, build_type) in roadname_buildtype_median:
     ...:             df_test_2.loc[i, "area"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         elif (district, build_type) in district_buildtype_median:
     ...:             district = df_test_2.loc[i, "district"]
     ...:             df_test_2.loc[i, "area"] = district_buildtype_median[(district, build_type)]
     ...: # delete rows greater than 3IQR after being transformed to the median groupby road_name, buil_type
     ...: Q1 = df_test_2['area'].quantile(0.25)
     ...: Q3 = df_test_2['area'].quantile(0.75)
     ...: IQR = Q3 - Q1
     ...: len(df_test_2[df_test_2["area"] > Q3 + 3*IQR]) # 26
     ...: df_test_2 = df_test_2[~(df_test_2["area"] > Q3 + 3*IQR) | df_test_2["area"].isna()].reset_index(drop = True)
```

- build_share1 NA

  * Same process with training dataset

```
In [272]: len(df_test_2[df_test_2["build_share1"].isna()]) # 2498
     ...: len(df_test_2[df_test_2["build_share1"].isna() & (df_test_2["build_type"] == "透天厝")]) # 2444
     ...: # fill NaN with median build_share1 groupby road_name & build_type
     ...: share_buildtype_df = df_test_2[~(df_test_2["build_share1"].isna()) & (df_test_2["build_type"] != "透天厝")]
     ...: roadname_buildtype_median = share_buildtype_df.groupby(["road_name", "build_type"])["build_share1"].median()
     ...: district_buildtype_median = df_test_2.groupby(["district", "build_type"])["build_share1"].median()
     ...: for i, share in enumerate(df_test_2["build_share1"]):
     ...:     if math.isnan(share):
     ...:         road_name = df_test_2.loc[i, "road_name"]
     ...:         build_type = df_test_2.loc[i, "build_type"]
     ...:         if build_type == "透天厝":
     ...:             df_test_2.loc[i, "build_share1"] = 0
     ...:         elif (road_name, build_type) in roadname_buildtype_median.index:
     ...:             df_test_2.loc[i, "build_share1"] = roadname_buildtype_median[(road_name, build_type)]
     ...:         else:
     ...:             df_test_2.loc[i, "build_share1"] = district_buildtype_median[(district, build_type)]
     ...: # check no NA
     ...: df_test_2["build_share1"].isna().sum()
Out[272]: 0
```

- bedroom_num, hall_num, bathroom_num NA

  * Same process with training dataset

```
In [273]: len(df_test_2[(df_test_2["bedroom_num"] == 0) & (df_test_2["hall_num"] == 0) & (df_test_2["bathroom_num"] == 0)]) # 613
     ...: # fill mode after groupby road_name&build_type, if there is multiple modes, then fill in with mean
     ...: df_test_2.loc[(df_test_2["bedroom_num"] == 0) & (df_test_2["hall_num"] == 0) & (df_test_2["bathroom_num"] == 0), ["bedroom_num", "hall_num", "bathroom_num"]] =
np.nan
     ...: df_test_2 = df_test_2[(df_test_2["bedroom_num"].notna()) & (df_test_2["hall_num"].notna()) & (df_test_2["bathroom_num"].notna())].reset_index(drop = True)
```

- total_deal_floor NA

  * Same process with training dataset

```
In [274]: len(df_test_2[(df_test_2["total_deal_floor"] == 0) & (df_test_2["build_type"] != "透天厝")]) # 528
     ...: df_test_2.loc[(df_test_2["total_deal_floor"] == 0) & (df_test_2["build_type"] != "透天厝"), "total_deal_floor"] = np.nan
     ...: df_test_2["total_deal_floor"].isna().sum() #378
     ...: type_totalfloor = df_test_2[df_test_2["build_type"] != "透天厝"]
     ...: type_totalfloor = type_totalfloor[type_totalfloor["total_deal_floor"].notna()]
     ...: buildtype_median = type_totalfloor.groupby("build_type")["total_deal_floor"].median()
     ...: for i, total in enumerate(df_test_2["total_deal_floor"]):
     ...:     if math.isnan(total) :
     ...:         df_test_2.loc[i, "total_deal_floor"] = buildtype_median[df_test_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_test_2["total_deal_floor"].isna().sum()
Out[274]: 0
```

- floor_sold NA

  * Same process with training dataset

```
In [275]: len(df_test_2[(df_test_2["floor_sold"] == 0) & (df_test_2["build_type"] != "透天厝")]) # 19
     ...: df_test_2.loc[(df_test_2["floor_sold"] == 0) & (df_test_2["build_type"] != "透天厝"), "floor_sold"] = np.nan
     ...: floor_sold = df_test_2[df_test_2["build_type"] != "透天厝"]
     ...: floor_sold = floor_sold[floor_sold["floor_sold"].notna()]
     ...: floor_median = floor_sold.groupby("build_type")["floor_sold"].median()
     ...: for i, floor in enumerate(df_test_2["floor_sold"]):
     ...:     if math.isnan(floor) :
     ...:         df_test_2.loc[i, "floor_sold"] = floor_median[df_test_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_test_2["floor_sold"].isna().sum()
Out[275]: 0
```

- story NA

  * Same process with training dataset

```
In [276]: df_test_2["story"].isna().sum() # 6
     ...: story_median = df_test_2.dropna().groupby("build_type")["story"].median()
     ...: for i, story in enumerate(df_test_2["story"]):
     ...:     if math.isnan(story) :
     ...:         df_test_2.loc[i, "story"] = story_median[df_test_2.loc[i, "build_type"]]
     ...: # check no NA
     ...: df_test_2["story"].isna().sum()
Out[276]: 0
```

- lon, lat NA

  * One data has 0 values in longitude and latitude.

```
In [277]: df_test_2[(df_test_2["lon"] == 0) | (df_test_2["lat"] == 0)]
Out[277]:
      district road_name  lon  lat  age   area build_type  bedroom_num  \
13212      西屯區       廣興巷  0.0  0.0  2.0  72.21        透天厝          4.0

       hall_num  bathroom_num  story  total_deal_floor  floor_sold elevator  \
13212       2.0           6.0    4.0               0.0         0.0        有

      manager  parking_num  build_share1  unit_price  deal_date  year  \
13212       有            0           0.0         0.0  111/11/03  2022

           price
13212  27000000.0
```

  * Manual imputation by the mean of its road name

```
In [278]: lon_index = df_test_2[(df_test_2["lon"] == 0) | (df_test_2["lat"] == 0)].index
     ...: df_test_2.loc[lon_index, "lon"] = round(df_test_2[(df_test_2["road_name"] == "廣興巷") & (df_test_2["lon"] != 0)]["lon"].mean(), 4)
     ...: df_test_2.loc[lon_index, "lat"] = round(df_test_2[(df_test_2["road_name"] == "廣興巷") & (df_test_2["lat"] != 0)]["lat"].mean(), 4)
```

- deal_date

  * No typos.

```
In [194]: df_date_split = df_test_1["deal_date"].str.split("/", expand = True)
     ...: df_date_split.columns = ["year", "month", "day"]
     ...: df_date_split["year"].value_counts()
Out[194]:
102    4765
103    3824
106    3183
108    3079
107    3048
104    2737
109    2620
105    2471
110    2353
111    1905
101    1681
Name: year, dtype: int64

In [195]: df_date_split["month"].value_counts()
Out[195]:
11    2945
12    2939
10    2885
03    2806
09    2737
08    2730
04    2715
05    2644
06    2548
07    2520
01    2465
02    1732
Name: month, dtype: int64
```

```
In [196]: df_date_split["day"].value_counts()
Out[196]:
25    1156
26    1144
07    1137
01    1095
17    1082
15    1080
10    1078
02    1072
06    1070
21    1070
05    1067
28    1064
27    1061
20    1055
16    1048
23    1028
24    1023
14    1023
03    1015
22    1011
19    1007
18     989
30     989
13     987
11     985
08     984
04     966
12     960
09     929
29     915
31     576
Name: day, dtype: int64
```

  * replace 00 month with 10

```
In [284]: df_test_2["deal_date"] = df_test_2["deal_date"].str.replace(r'(?<=\/)00(?=\/)', '10', regex=True)
```

  * replace .5 day with 25

```
In [285]: df_test_2["deal_date"] = df_test_2["deal_date"].str.replace(r'\.5$', "25", regex=True)
```

  * transform to datetime and create month feature

```
In [286]: year_month_day = []
     ...: for date in df_test_2["deal_date"]:
     ...:     year, month, day = date.split("/")
     ...:     new_year = int(year) + 1911
     ...:     year_month_day.append(str(new_year) + '-' + month + '-' + day)
     ...: df_test_2["deal_date"] = year_month_day
     ...: # extract "month" from "deal year" and create a new column
     ...: df_test_2["month"] = pd.to_datetime(df_test_2['deal_date']).dt.month
```

Merge with macroeconomic indicators

```
In [288]: m1 = pd.read_csv('M1b貨幣供給額.csv')
     ...: mortgage = pd.read_csv('五大行庫平均房貸利率.csv', encoding='utf-8')
     ...: gdp = pd.read_csv('實質GDP成長率.csv')
     ...: inflation = pd.read_csv('消費者物價指數年增率.csv')
```

* They are in different formats and need to be standardized.

```
In [289]: m1
Out[289]:
      時間  M1b貨幣供給額(億元)
0    111/12        258054
1    111/11        256377
2    111/10        254564
3    111/09        254517
4    111/08        258303
..      ...           ...
120  101/12        124184
121  101/11        120685
122  101/10        120120
123  101/09        120608
124  101/08        120441

[125 rows x 2 columns]
```

```
In [290]: mortgage
Out[290]:
      時間  五大行庫平均房貸利率(%)
0    111/12            1.86
1    111/11            1.85
2    111/10            1.83
3    111/09            1.73
4    111/08            1.73
..      ...             ...
120  101/12            1.91
121  101/11            1.91
122  101/10            1.90
123  101/09            1.90
124  101/08            1.90

[125 rows x 2 columns]
```

```
In [292]: inflation
Out[292]:
      統計期  總指數
0     101年  1.93
1   101年8月  3.43
2   101年9月  2.95
3  101年10月  2.33
4  101年11月  1.59
..      ...   ...
135 111年12月  2.71
136     NaN   NaN
137     NaN   NaN
138      註解：   NaN
139  由於受查者延誤或更正報價，最近3個月資料均可能修正。   NaN

[140 rows x 2 columns]
```

```
In [291]: gdp
Out[291]:
      統計期  經濟成長率(%)
0     101年       2.22
1   101年第3季       2.18
2   101年第4季       4.48
3     102年       2.48
4   102年第1季       1.50
5   102年第2季       2.68
6   102年第3季       1.95
7   102年第4季       3.73
8     103年       4.72
9   103年第1季       4.69
10  103年第2季       4.92
11  103年第3季       4.90
12  103年第4季       4.38
13    104年       1.47
14  104年第1季       4.80
15  104年第2季       1.89
16  104年第3季      -0.28
17  104年第4季      -0.20
18    105年       2.17
19  105年第1季      -0.09
20  105年第2季       1.69
21  105年第3季       3.00
22  105年第4季       3.92
23    106年       3.31
```

- m1b

  * extract year and month features for merging purpose

```
In [293]: year_month = []
     ...: for date in m1["時間"]:
     ...:     year, month = date.split("/")
     ...:     new_year = int(year) + 1911
     ...:     year_month.append(str(new_year) + '-' + month)
     ...: m1["時間"] = year_month
     ...: # extract "month" from "deal year" and create a new column
     ...: m1["year"] = pd.to_datetime(m1["時間"]).dt.year
     ...: m1["month"] = pd.to_datetime(m1["時間"]).dt.month
     ...: # drop column
     ...: m1 = m1.drop(["時間"], axis = 1)
     ...: # rename column
     ...: m1.rename(columns={'M1b貨幣供給額(億元)' : 'M1b'}, inplace=True)
```

```
In [294]: m1
Out[294]:
      M1b  year  month
0    258054  2022     12
1    256377  2022     11
2    254564  2022     10
3    254517  2022      9
4    258303  2022      8
..      ...   ...    ...
120  124184  2012     12
121  120685  2012     11
122  120120  2012     10
123  120608  2012      9
124  120441  2012      8

[125 rows x 3 columns]
```

- mortgage

  * extract year and month features for merging purpose

```
In [295]: df_date_split = mortgage["時間"].str.split("/", expand = True)
     ...: df_date_split.columns = ["year", "month"]
     ...: df_date_split["year"].value_counts()
     ...: df_date_split["month"].value_counts()
     ...: # transform date format and replace deal_date with it
     ...: year_month = []
     ...: for date in mortgage["時間"]:
     ...:     year, month = date.split("/")
     ...:     new_year = int(year) + 1911
     ...:     year_month.append(str(new_year) + '-' + month)
     ...: mortgage["時間"] = year_month
     ...: # extract "month" from "deal year" and create a new column
     ...: mortgage["year"] = pd.to_datetime(mortgage["時間"]).dt.year
     ...: mortgage["month"] = pd.to_datetime(mortgage["時間"]).dt.month
     ...: # drop column
     ...: mortgage = mortgage.drop(["時間"], axis = 1)
     ...: # rename column
     ...: mortgage.rename(columns={'五大行庫平均房貸利率(%)' : 'mortgage_rate'}, inplace=True)
```

```
In [296]: mortgage
Out[296]:
     mortgage_rate  year  month
0             1.86  2022     12
1             1.85  2022     11
2             1.83  2022     10
3             1.73  2022      9
4             1.73  2022      8
..             ...   ...    ...
120           1.91  2012     12
121           1.91  2012     11
122           1.90  2012     10
123           1.90  2012      9
124           1.90  2012      8

[125 rows x 3 columns]
```

- Inflation

  * extract year and month features for merging purpose

```
In [300]: inflation = inflation[~inflation["總指數"].isna()]
     ...: inflation[['year', 'month']] = inflation['統計期'].str.split('年', expand=True)
     ...: inflation['year'] = inflation['year'].astype(int)
     ...: inflation['year'] = inflation['year'] + 1911
     ...: inflation = inflation[inflation["month"] != ""]
     ...: inflation['month'] = inflation['month'].str.replace('月', '').astype(int)
     ...: # drop column
     ...: inflation = inflation.drop(["統計期"], axis = 1)
     ...: # rename column
     ...: inflation.rename(columns={'總指數' : 'CPI%'}, inplace=True)
```

```
In [301]: inflation
Out[301]:
     CPI%  year  month
1    3.43  2012      8
2    2.95  2012      9
3    2.33  2012     10
4    1.59  2012     11
5    1.62  2012     12
..    ...   ...    ...
131  2.68  2022      8
132  2.76  2022      9
133  2.74  2022     10
134  2.35  2022     11
135  2.71  2022     12

[125 rows x 3 columns]
```

- Gdp

  * extract year and month features for merging purpose

```
In [302]: gdp[['year', 'quarter']] = gdp['統計期'].str.split('年', expand=True)
     ...: gdp = gdp[~gdp["經濟成長率(%)"].isna()]
     ...: gdp = gdp[gdp["quarter"] != ""]
     ...: gdp['year'] = gdp['year'].astype(int)
     ...: gdp['year'] = gdp['year'] + 1911
     ...: gdp['quarter'] = gdp['quarter'].apply(lambda x: int(re.findall(r'\d+', x)[0]))
     ...: d_map={1 : "Q1", 2 : "Q2", 3 : "Q3", 4 : "Q4"}
     ...: gdp['quarter'] = gdp['quarter'].map(d_map)
     ...: # quater month conversion
     ...: month_quarter = {"Q1": [1, 2, 3], "Q2": [4, 5, 6], "Q3": [7, 8, 9], "Q4": [10, 11, 12]}
     ...: months = [month for quarter in month_quarter.values() for month in quarter]
     ...: years = range(2012, 2023)
     ...: dates = pd.date_range(start=f"{years[0]}-{months[0]}", end=f"{years[-1]}-{months[-1]}", freq="MS")
     ...: gdp_month = pd.DataFrame({"year": dates.year, "month": dates.month})
     ...: gdp_month["quarter"] = ""
     ...: for q, months in month_quarter.items():
     ...:     gdp_month.loc[gdp_month["month"].isin(months), "quarter"] = q
     ...: gdp_month = gdp_month.drop(df.index[:7])
     ...: gdp_conv = gdp_month.merge(gdp, on=['year', 'quarter'], how='left')
     ...: gdp_conv = gdp_conv.drop(["quarter", "統計期"], axis = 1)
     ...: # rename column
     ...: gdp_conv.rename(columns={'經濟成長率(%)': 'GDP%'}, inplace=True)
```

```
In [303]: gdp
Out[303]:
      統計期  經濟成長率(%)  year quarter
1   101年第3季       2.18  2012      Q3
2   101年第4季       4.48  2012      Q4
4   102年第1季       1.50  2013      Q1
5   102年第2季       2.68  2013      Q2
6   102年第3季       1.95  2013      Q3
7   102年第4季       3.73  2013      Q4
9   103年第1季       4.69  2014      Q1
10  103年第2季       4.92  2014      Q2
11  103年第3季       4.90  2014      Q3
12  103年第4季       4.38  2014      Q4
14  104年第1季       4.80  2015      Q1
15  104年第2季       1.89  2015      Q2
16  104年第3季      -0.28  2015      Q3
17  104年第4季      -0.20  2015      Q4
19  105年第1季      -0.09  2016      Q1
20  105年第2季       1.69  2016      Q2
21  105年第3季       3.00  2016      Q3
22  105年第4季       3.92  2016      Q4
24  106年第1季       3.24  2017      Q1
25  106年第2季       2.64  2017      Q2
```

- merge to house data

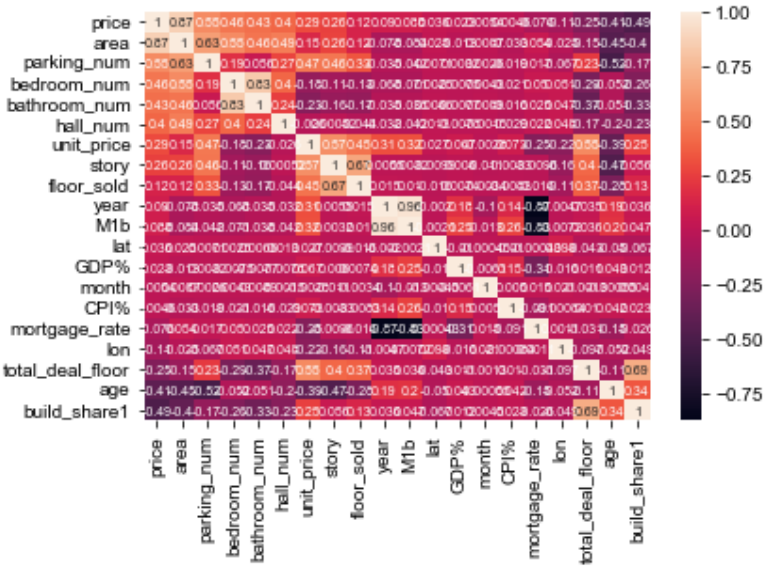
```
In [304]:
     ...: add = gdp_conv.merge(inflation, on = ["year", "month"], how = "left")
     ...: add = add.merge(m1, on = ["year", "month"], how = "left")
     ...: add = add.merge(mortgage, on = ["year", "month"], how = "left")
     ...:
     ...: # merge to house data
     ...: df_train_1 = df_train_1.merge(add, on = ["year", "month"], how = "left")
     ...: df_val_1 = df_val_1.merge(add, on = ["year", "month"], how = "left")
     ...: df_test_1 = df_test_1.merge(add, on = ["year", "month"], how = "left")
     ...: df_train_2 = df_train_2.merge(add, on = ["year", "month"], how = "left")
     ...: df_val_2 = df_val_2.merge(add, on = ["year", "month"], how = "left")
     ...: df_test_2 = df_test_2.merge(add, on = ["year", "month"], how = "left")
```

- Overall correlation

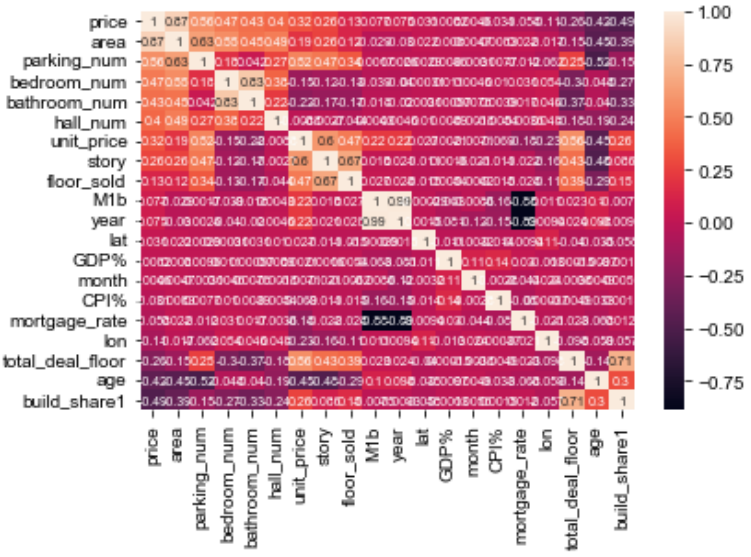
  * Highly correlated:

  - price vs. area
  - bathroom vs. bedroom
  - total_deal_floor vs. 透天厝
  - m1b vs. elevator

- build_share1 vs. 透天厝
- year vs. m1b
- year vs. mortgage_rate
- mortgage_rate vs. m1b
- Model_1



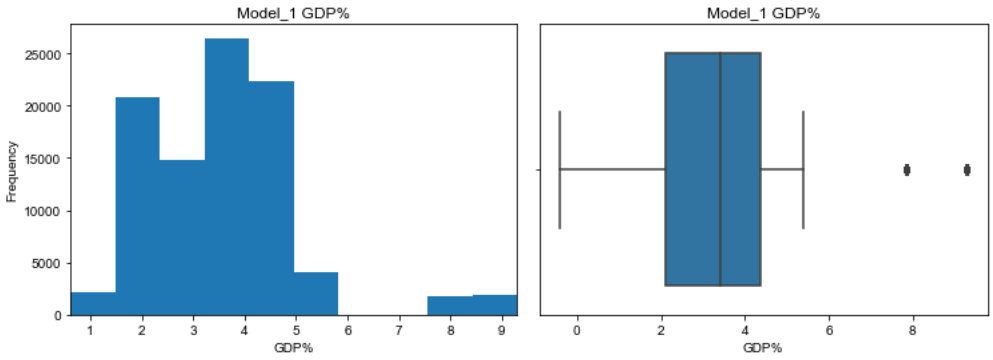
- Model_2



- skewness check

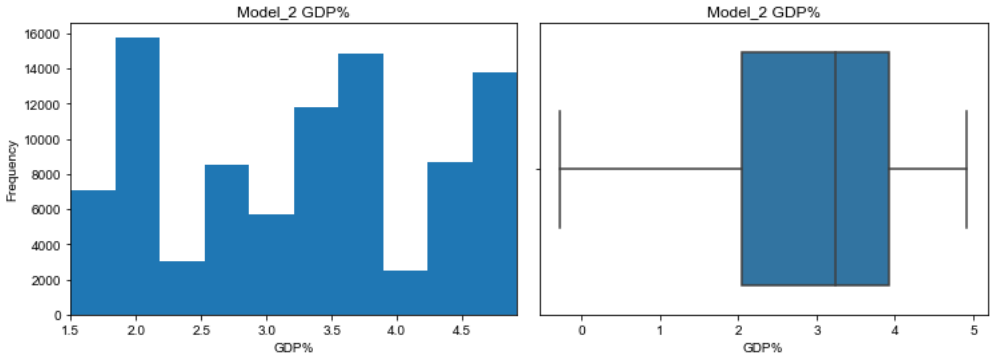
  * check those features with highly skewness in previous EDA

  - GDP%
    - Model_1



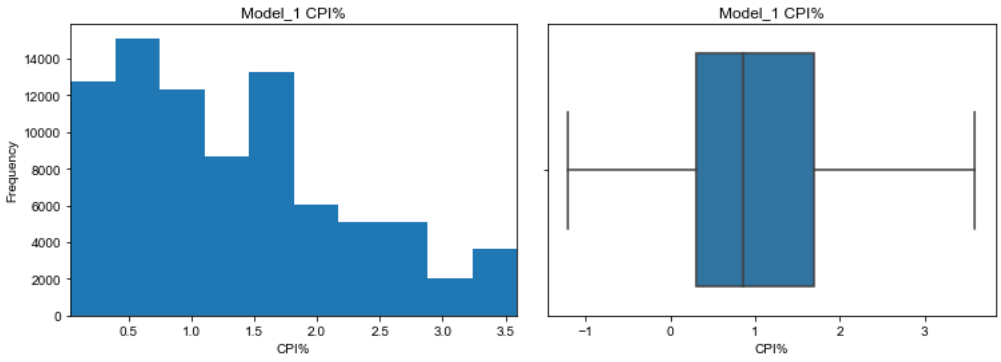

```
In [344]: print('Original Skewness:', skew(df_train_1["GDP%"])) # 0.4854
Original Skewness: 0.48544020494525164
```

- Model_2



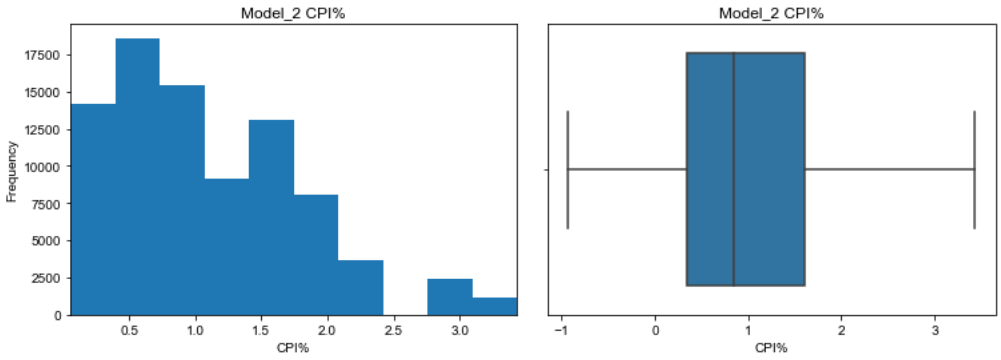

```
In [347]: print('Original Skewness:', skew(df_train_2["GDP%"])) # -0.6595
Original Skewness: -0.6584367897276777
```

- CPI%
  - Model_1



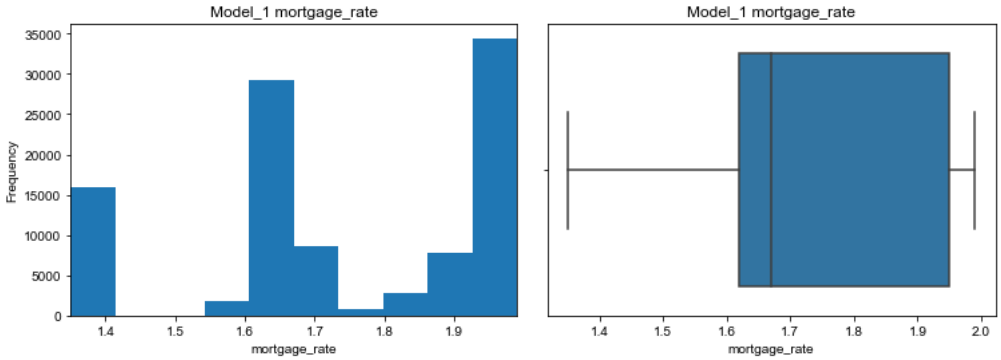

```
In [349]: print('Original Skewness:', skew(df_train_1["CPI%"])) # 0.2467
Original Skewness: 0.24666506222631848
```

- Model_2



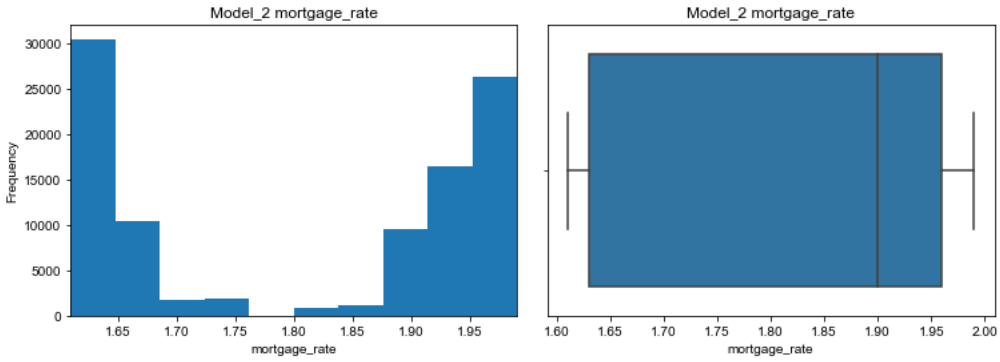

```
In [350]: print('Original Skewness:', skew(df_train_2["CPI%"])) # 0.2227
Original Skewness: 0.22252452531220435
```

- Mortgage rate
  - Model_1



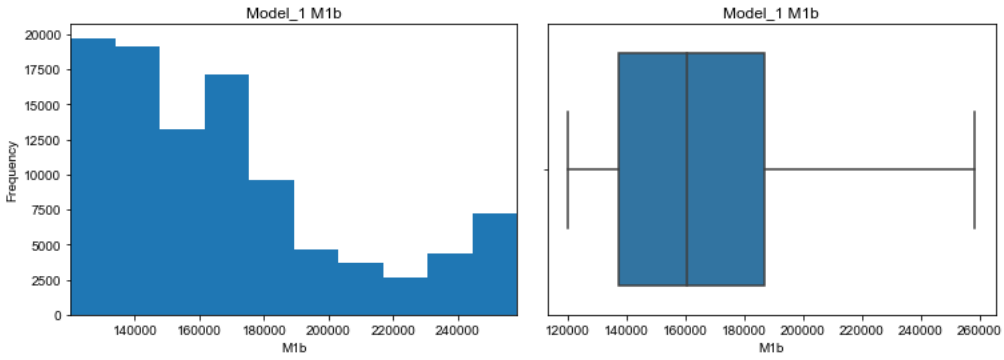

```
In [353]: print('Original Skewness:', skew(df_train_1["mortgage_rate"])) # -0.3991
Original Skewness: -0.3990785861326743
```

  - Model_2



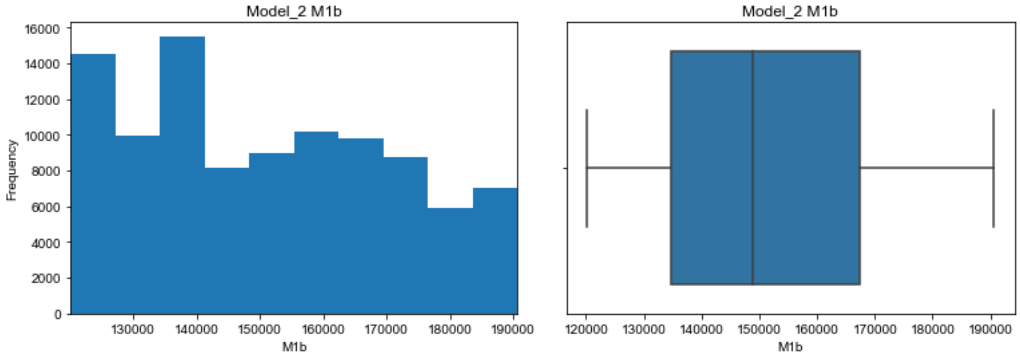

```
...: print('Original Skewness:', skew(df_train_2["mortgage_rate"])) # -0.1754
Original Skewness: -0.17631586050684453
```

- M1b
  - Model_1



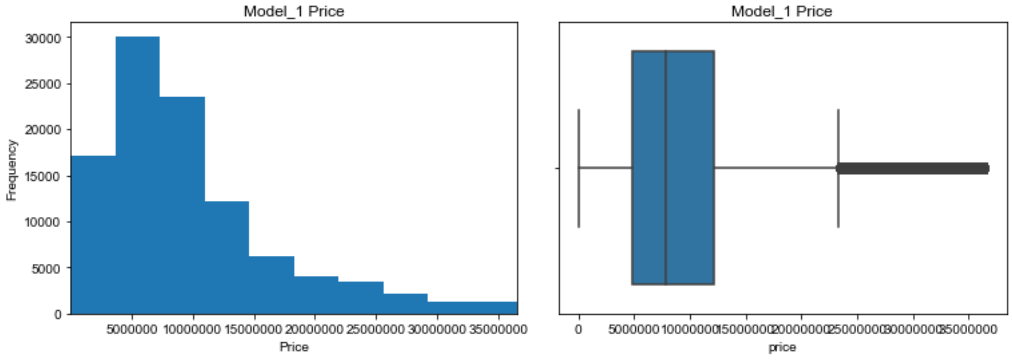

```
In [356]: print('Original Skewness:', skew(df_train_1["M1b"])) # 0.9029
Original Skewness: 0.9028526799975862
```

- Model_2




```
In [358]: print('Original Skewness:', skew(df_train_2["M1b"])) # 0.2316

Output from spyder call 'get_namespace_view':
Original Skewness: 0.23229881932756902
```

- price
  - Model_1

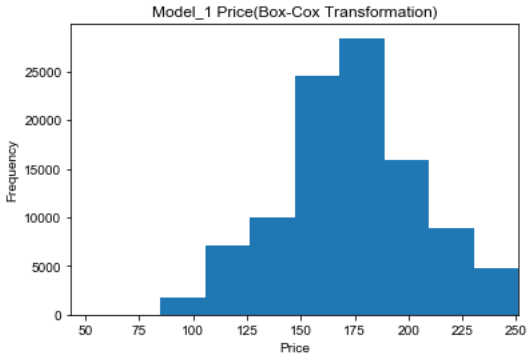



```
In [360]: print('Original Skewness:', skew(df_train_1["price"])) # 1.4220 > 1

Output from spyder call 'get_namespace_view':
Original Skewness: 1.4219955763771754
```

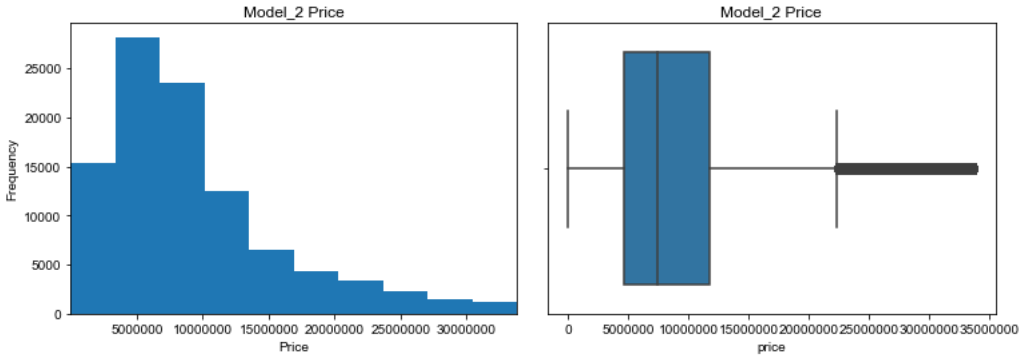
* skewness is considerably reduced

```
In [363]: df_train_1[df_train_1["price"] <= 0]  # 0
     ...: x_boxcox, lam = boxcox(df_train_1["price"])
     ...: print('Box-Cox Transformed Skewness:', skew(x_boxcox)) # -0.0099
Box-Cox Transformed Skewness: -0.009911471724470496
```

* data is more symmetric



- Model_2

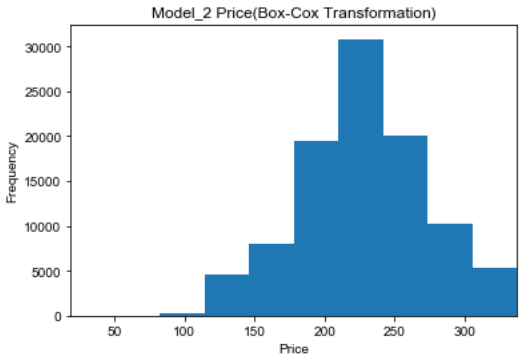



```
In [428]: print('Original Skewness:', skew(df_train_2["price"])) # 1.4221 > 1
Original Skewness: 1.4220558774008039
```

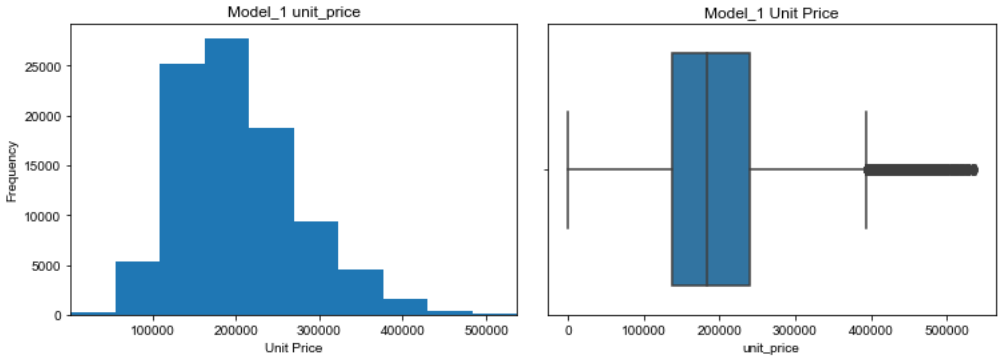
* skewness is considerably reduced

```
In [430]: print('Box-Cox Transformed Skewness:', skew(x_boxcox)) # -0.0124
Box-Cox Transformed Skewness: -0.012402341107796266
```

* data is more symmetric

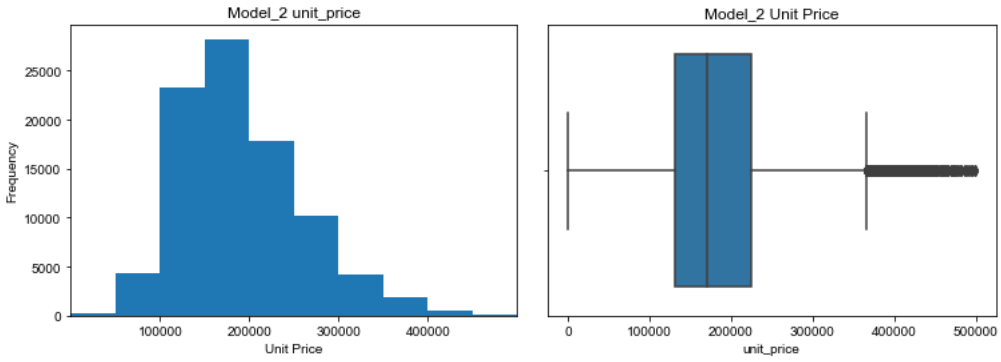


- Unit price
  - Model_1



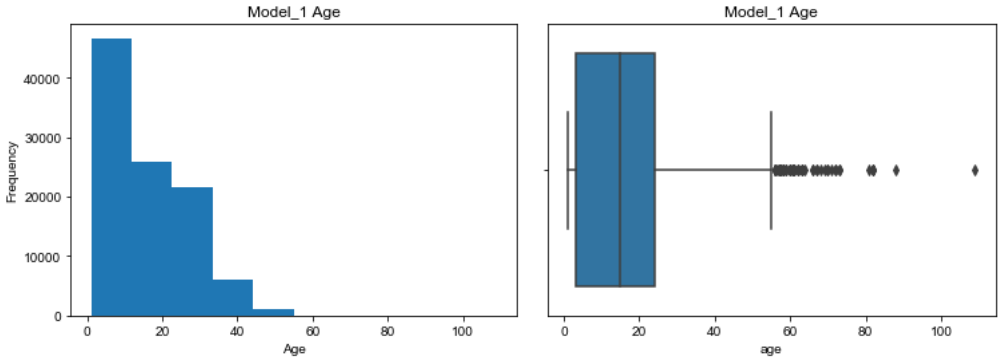

```
    ...: print('Original Skewness:', skew(df_train_1["unit_price"])) # 0.0075
Original Skewness: 0.007469196045697374
```

- Model_2



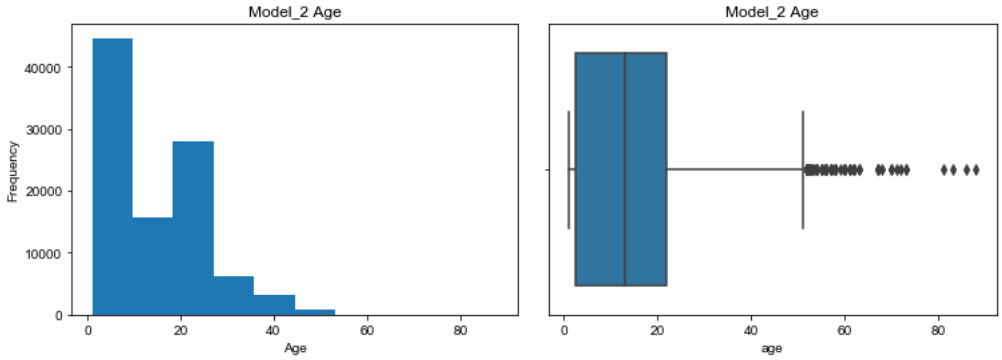

```
In [432]: print('Original Skewness:', skew(df_train_2["unit_price"])) # 0.0008
Original Skewness: 0.000771483815830837
```

- age
  - Model_1



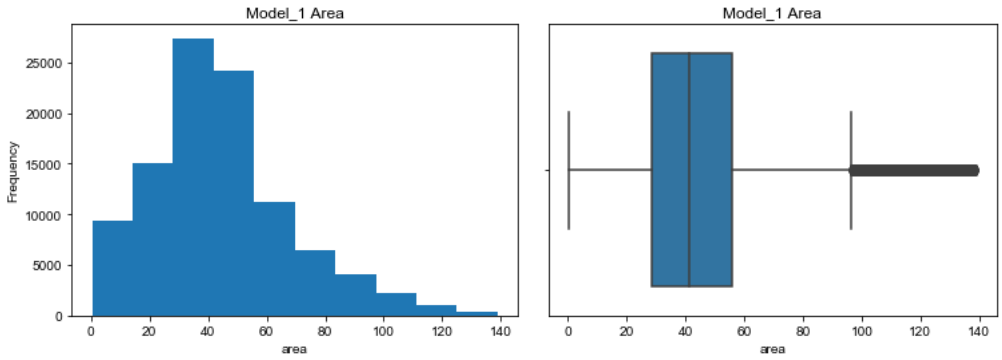

```
In [376]: print('Original Skewness:', skew(df_train_1["age"])) # 0.5529
Original Skewness: 0.5528558013914332
```

- Model_2



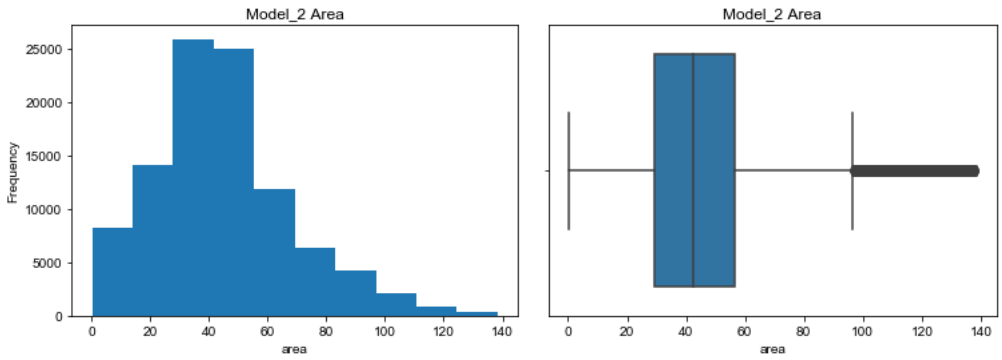

```
In [434]: print('Original Skewness:', skew(df_train_2["age"])) # 0.6402
Original Skewness: 0.6402109706218978
```

- area
  - Model_1



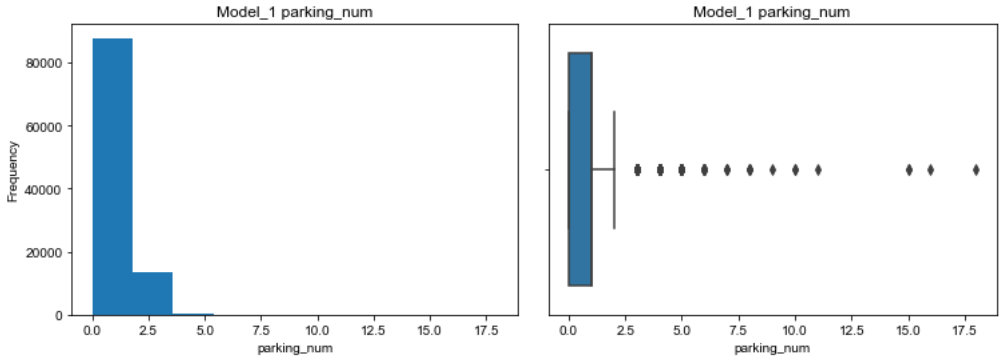

```
In [382]: print('Original Skewness:', skew(df_train_1["area"])) # 0.8837
Original Skewness: 0.8837006475286386
```

  - Model_2



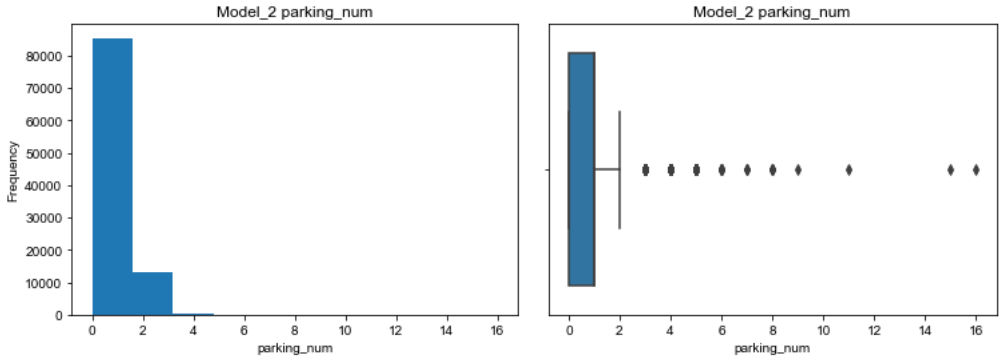

```
In [436]: print('Original Skewness:', skew(df_train_2["area"])) # 0.8629
Original Skewness: 0.8629064172865477
```

- Parking numbers
  - Model_1



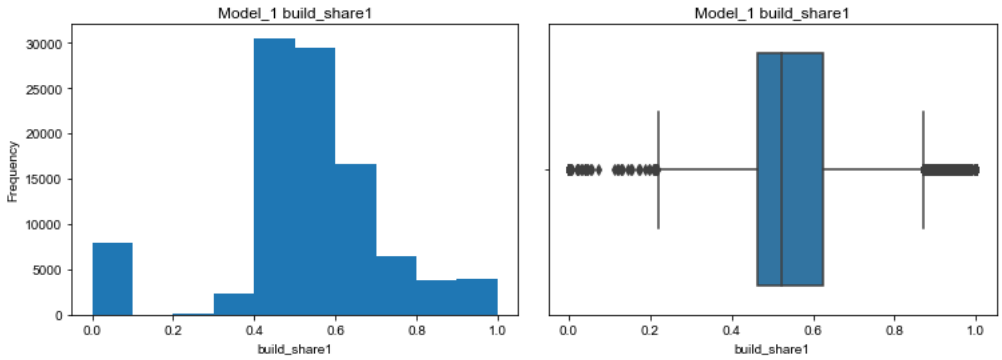

```
In [386]: print('Original Skewness:', skew(df_train_1["parking_num"])) # 1.2050 > 1
Original Skewness: 1.204952637293375
```

- Model_2



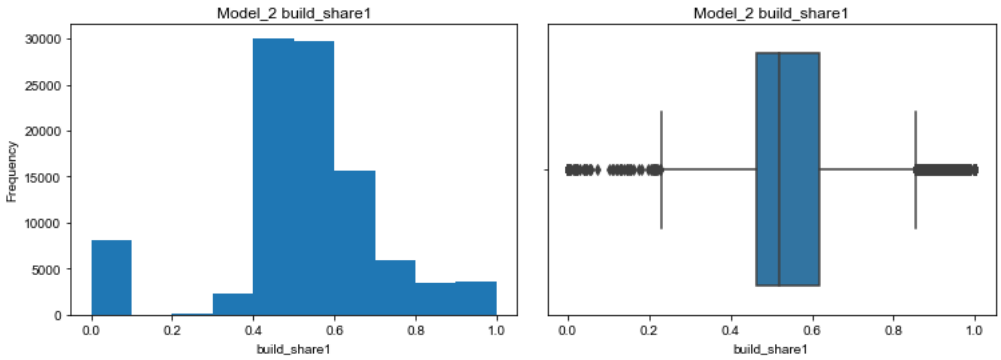

```
print('Original Skewness:', skew(df_train_2["parking_num"])) # 1.1326 > 1
```

- Building share 1
  - Model_1




```
In [391]: print('Original Skewness:', skew(df_train_1["build_share1"])) # -0.7469
Original Skewness: -0.7469358734397046
```

- Model_2




```
In [439]: print('Original Skewness:', skew(df_train_2["build_share1"])) # -0.7846
Original Skewness: -0.784635781602426
```

* There are several ways can deal with high skewness, including log transformation, box-cox transformation, power transformation. I've tried log transformation but the effects were not so good. So I'll use box-cox and power to address highly skewed data.

Box- cox transformation

* It can only be applied to data that is strictly positive. We should check there is **no zero or negative values in data before transforming.**

- Both the prices of model_1 and model_2 need to be transformed.

```
In [396]:
     ...: df_train_1[df_train_1["price"] <= 0]  # 0
     ...: x_boxcox, lam = boxcox(df_train_1["price"])
     ...: df_train_1["price_norm"] = x_boxcox
     ...: # Model_1(val): total_price
     ...: df_val_1[df_val_1["price"] <= 0]  # 0
     ...: x_boxcox, lam = boxcox(df_val_1["price"])
     ...: df_val_1["price_norm"] = x_boxcox
     ...: # Model_1(test): total_price
     ...: df_test_1[df_test_1["price"] <= 0]  # 0
     ...: x_boxcox, lam = boxcox(df_test_1["price"])
     ...: df_test_1["price_norm"] = x_boxcox
     ...: # Model_2(train): total_price
     ...: df_train_2[df_train_2["price"] <= 0]  # 0
     ...: x_boxcox, lam = boxcox(df_train_2["price"])
     ...: df_train_2["price_norm"] = x_boxcox
     ...: # Model_2(val): total_price
     ...: df_val_2[df_val_2["price"] <= 0]  # 0
     ...: x_boxcox, lam = boxcox(df_val_2["price"])
     ...: df_val_2["price_norm"] = x_boxcox
     ...: # Model_2(test): total_price
     ...: df_test_2[df_test_2["price"] <= 0]  # 0
     ...: x_boxcox, lam = boxcox(df_test_2["price"])
     ...: df_test_2["price_norm"] = x_boxcox
```

power transformation

* Since the numbers of parking number of model_1 and model_2 have 0, box-cox is not applicable. So, I use power transformation to correct their skewness.

```
In [399]: len(df_train_1[df_train_1["parking_num"] <= 0]) # 38718 (Box-cox is not usable for <= 0)
     ...: pt = PowerTransformer(method = 'yeo-johnson', standardize = False)
     ...: pt_trans = pt.fit_transform(df_train_1[["parking_num"]])
     ...: df_train_1["parking_num_norm"] = pt_trans
     ...: print(df_train_1["parking_num_norm"].skew())
     ...: # Model_1(val): parking_num
     ...: pt = PowerTransformer(method = 'yeo-johnson', standardize = False)
     ...: pt_trans = pt.fit_transform(df_val_1[["parking_num"]])
     ...: df_val_1["parking_num_norm"] = pt_trans
     ...: # Model_1(test): parking_num
     ...: pt = PowerTransformer(method = 'yeo-johnson', standardize = False)
     ...: pt_trans = pt.fit_transform(df_test_1[["parking_num"]])
     ...: df_test_1["parking_num_norm"] = pt_trans
     ...:
     ...: # Model_2: parking_num
     ...: len(df_train_2[df_train_2["parking_num"] <= 0])  # 36940
     ...: pt_trans = pt.fit_transform(df_train_2[["parking_num"]])
     ...: df_train_2["parking_num_norm"] = pt_trans
     ...: print(df_train_2["parking_num_norm"].skew())
     ...: # Model_2(val): parking_num
     ...: pt = PowerTransformer(method = 'yeo-johnson', standardize = False)
     ...: pt_trans = pt.fit_transform(df_val_2[["parking_num"]])
     ...: df_val_2["parking_num_norm"] = pt_trans
     ...: # Model_2(test): parking_num
     ...: pt = PowerTransformer(method = 'yeo-johnson', standardize = False)
     ...: pt_trans = pt.fit_transform(df_test_2[["parking_num"]])
     ...: df_test_2["parking_num_norm"] = pt_trans
-0.00028351399108496236
-0.009194248650095434
```

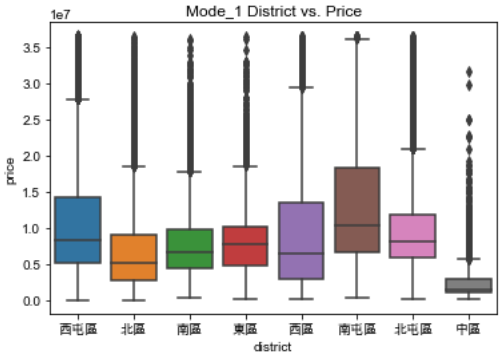
- categorical data

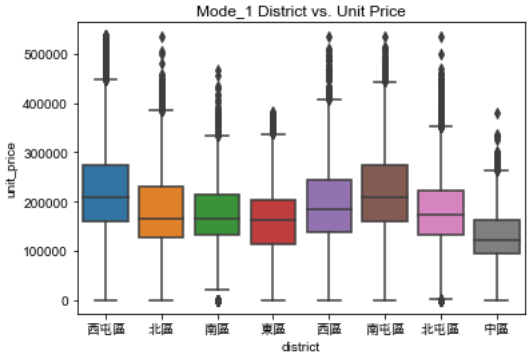
  * check if there are significant difference between levels

  - District

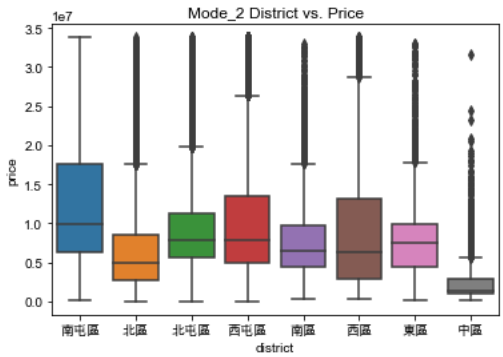
    * some of the difference between levels in both models are NOT significant by price and unit price, should perform t-test

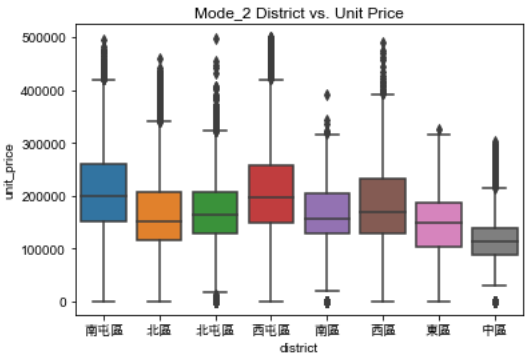
    - Model_1





    - Model_2

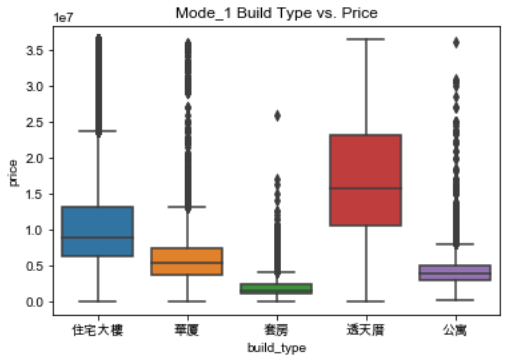


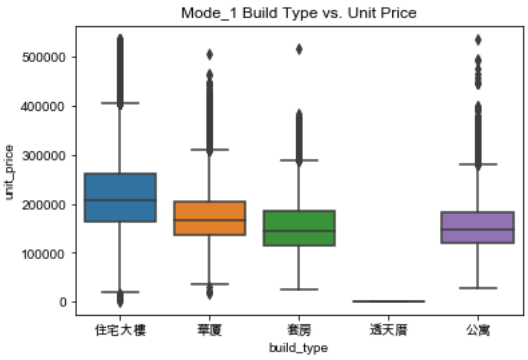


  - Building type

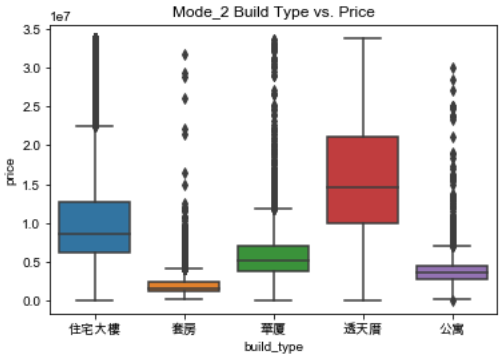
    * some of the difference between levels in both models are NOT significant by unit price, should perform t-test

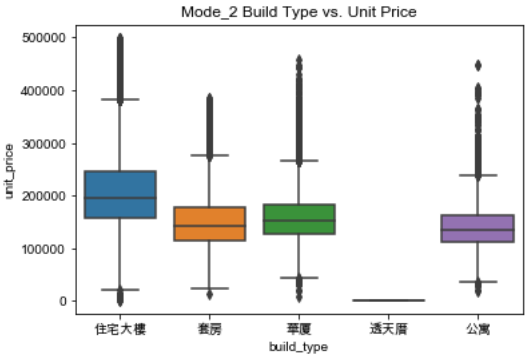
- Model_1





- Model_2

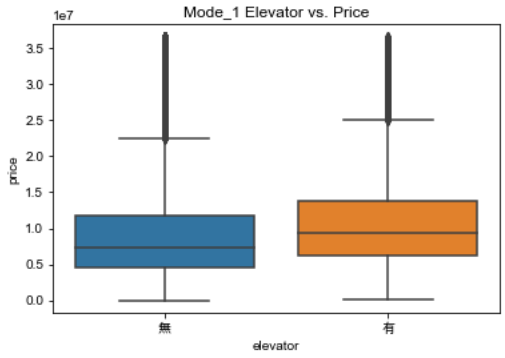


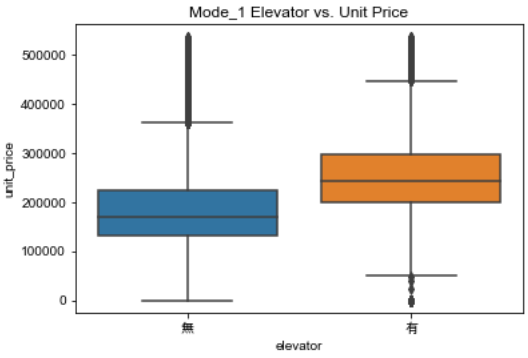


- elevator

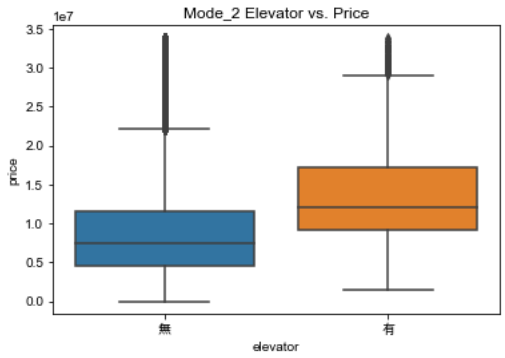
  * the difference between levels in model_1 are NOT significant by price, should perform t-test

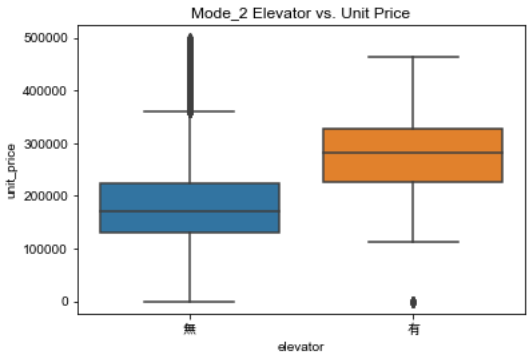
- Model_1





- Model_2

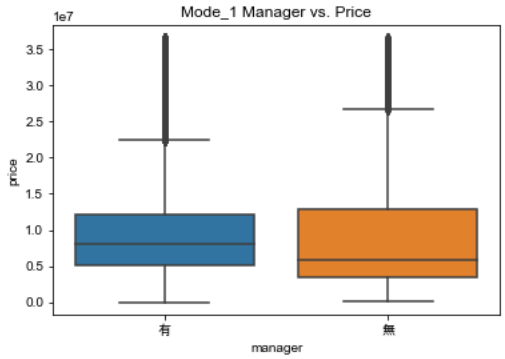


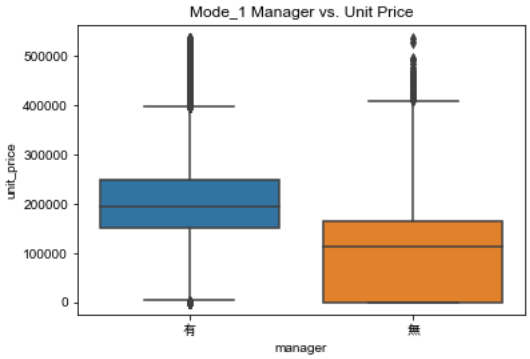


- Manager

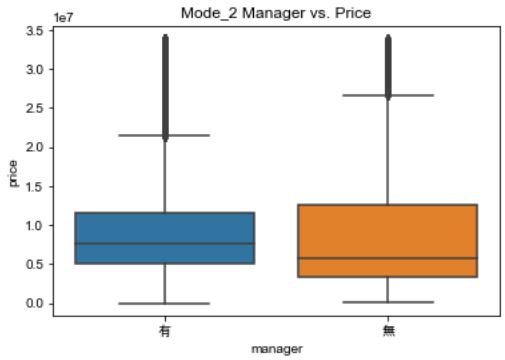
  * the difference between levels in both are NOT significant by price, should perform t-test

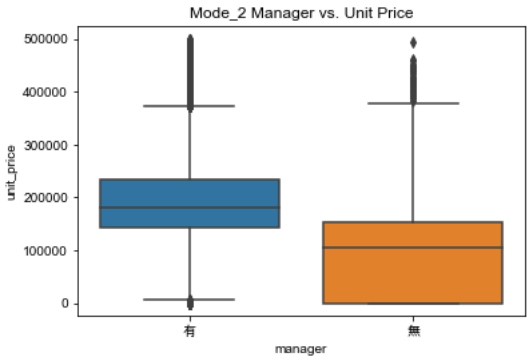
  - Model_1





  - Model_2





Scaling

* Minmax scaler

```
scaler = MinMaxScaler()
non_scaled_cols = ["district", "road_name", "build_type", "elevator", "manager", "deal_date", "year", "month"]
# df_train_1
scaled_cols = df_train_1.columns.difference(non_scaled_cols)
scaler.fit(df_train_1[scaled_cols])
df_train_1_scale = scaler.transform(df_train_1[scaled_cols])
df_train_1_scale = pd.DataFrame(df_train_1_scale, columns = scaled_cols)
df_train_1_scale = pd.concat([df_train_1_scale, df_train_1[non_scaled_cols]], axis=1)
# df_val_1
scaled_cols = df_val_1.columns.difference(non_scaled_cols)
scaler.fit(df_val_1[scaled_cols])
df_val_1_scale = scaler.transform(df_val_1[scaled_cols])
df_val_1_scale = pd.DataFrame(df_val_1_scale, columns = scaled_cols)
df_val_1_scale = pd.concat([df_val_1_scale, df_val_1[non_scaled_cols]], axis=1)
# df_test_1
sscaled_cols = df_test_1.columns.difference(non_scaled_cols)
scaler.fit(df_test_1[scaled_cols])
df_test_1_scale = scaler.transform(df_test_1[scaled_cols])
df_test_1_scale = pd.DataFrame(df_test_1_scale, columns = scaled_cols)
df_test_1_scale = pd.concat([df_test_1_scale, df_test_1[non_scaled_cols]], axis=1)
# df_train_2
scaled_cols = df_train_2.columns.difference(non_scaled_cols)
scaler.fit(df_train_2[scaled_cols])
df_train_2_scale = scaler.transform(df_train_2[scaled_cols])
df_train_2_scale = pd.DataFrame(df_train_2_scale, columns = scaled_cols)
df_train_2_scale = pd.concat([df_train_2_scale, df_train_2[non_scaled_cols]], axis=1)
# df_val_2
scaled_cols = df_val_2.columns.difference(non_scaled_cols)
scaler.fit(df_val_2[scaled_cols])
df_val_2_scale = scaler.transform(df_val_2[scaled_cols])
df_val_2_scale = pd.DataFrame(df_val_2_scale, columns = scaled_cols)
df_val_2_scale = pd.concat([df_val_2_scale, df_val_2[non_scaled_cols]], axis=1)
# df_test_2
sscaled_cols = df_test_2.columns.difference(non_scaled_cols)
scaler.fit(df_test_2[scaled_cols])
df_test_2_scale = scaler.transform(df_test_2[scaled_cols])
df_test_2_scale = pd.DataFrame(df_test_2_scale, columns = scaled_cols)
df_test_2_scale = pd.concat([df_test_2_scale, df_test_2[non_scaled_cols]], axis=1)
```

Pairwise t-test (Bonferroni)

* Based on the previous plot, test the features that exhibit a INSIGNIFICANT difference between the levels. (α = 0.05)

- Model_1
- district vs. price(t-test)

  * levels are not significant, should be in one group

    - 北屯區&西屯區
    - 南區&西區
    - 東區&西區

```
In [453]: comp1 = mc.MultiComparison(df_train_1_scale["price_norm"], df_train_1_scale["district"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('中區', '北區', -36.663 , 0.    , 0.    ,  True),
       ('中區', '北屯區', -74.2058, 0.    , 0.    ,  True),
       ('中區', '南區', -58.1165, 0.    , 0.    ,  True),
       ('中區', '南屯區', -74.2557, 0.    , 0.    ,  True),
       ('中區', '東區', -51.9742, 0.    , 0.    ,  True),
       ('中區', '西區', -38.7754, 0.    , 0.    ,  True),
       ('中區', '西屯區', -55.0706, 0.    , 0.    ,  True),
       ('北區', '北屯區', -59.8672, 0.    , 0.    ,  True),
       ('北區', '南區', -20.2497, 0.    , 0.    ,  True),
       ('北區', '南屯區', -75.7228, 0.    , 0.    ,  True),
       ('北區', '東區', -18.522 , 0.    , 0.    ,  True),
       ('北區', '西區', -17.3309, 0.    , 0.    ,  True),
       ('北區', '西屯區', -47.2881, 0.    , 0.    ,  True),
       ('北屯區', '南區',  36.569 , 0.    , 0.    ,  True),
       ('北屯區', '南屯區', -39.6478, 0.    , 0.    ,  True),
       ('北屯區', '東區',  16.9476, 0.    , 0.    ,  True),
       ('北屯區', '西區',  24.6764, 0.    , 0.    ,  True),
       ('北屯區', '西屯區',  -0.0973, 0.9225, 1.    , False),
       ('南區', '南屯區', -60.4973, 0.    , 0.    ,  True),
       ('南區', '東區',  -6.3594, 0.    , 0.    ,  True),
       ('南區', '西區',  -2.4955, 0.0126, 0.3524, False),
       ('南區', '西屯區', -28.1007, 0.    , 0.    ,  True),
       ('南屯區', '東區',  34.3914, 0.    , 0.    ,  True),
       ('南屯區', '西區',  42.7572, 0.    , 0.    ,  True),
       ('南屯區', '西屯區',  31.2814, 0.    , 0.    ,  True),
       ('東區', '西區',   2.5656, 0.0103, 0.2888, False),
       ('東區', '西屯區', -12.7252, 0.    , 0.    ,  True),
       ('西區', '西屯區', -19.4232, 0.    , 0.    ,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')])))
```

- district vs. unit_price(t-test)

  * levels are not significant, should be in one group

  - 南屯&西屯
  - 北屯&南區

```
In [454]:
     ...: comp1 = mc.MultiComparison(df_train_1_scale["unit_price"], df_train_1_scale["district"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('中區', '北區', -1.99018e+01, 0.    , 0.,  True),
       ('中區', '北屯區', -1.69408e+01, 0.    , 0.,  True),
       ('中區', '南區', -2.04504e+01, 0.    , 0.,  True),
       ('中區', '南屯區', -3.25745e+01, 0.    , 0.,  True),
       ('中區', '東區', -9.58180e+00, 0.    , 0.,  True),
       ('中區', '西區', -2.49762e+01, 0.    , 0.,  True),
       ('中區', '西屯區', -3.51809e+01, 0.    , 0.,  True),
       ('北區', '北屯區',  9.16670e+00, 0.    , 0.,  True),
       ('北區', '南區',  9.12130e+00, 0.    , 0.,  True),
       ('北區', '南屯區', -3.41594e+01, 0.    , 0.,  True),
       ('北區', '東區',  1.37662e+01, 0.    , 0.,  True),
       ('北區', '西區', -1.00397e+01, 0.    , 0.,  True),
       ('北區', '西屯區', -3.84126e+01, 0.    , 0.,  True),
       ('北屯區', '南區',  8.93600e-01, 0.3715, 1., False),
       ('北屯區', '南屯區', -5.16298e+01, 0.    , 0.,  True),
       ('北屯區', '東區',  9.32100e+00, 0.    , 0.,  True),
       ('北屯區', '西區', -1.92545e+01, 0.    , 0.,  True),
       ('北屯區', '西屯區', -5.93534e+01, 0.    , 0.,  True),
       ('南區', '南屯區', -4.42247e+01, 0.    , 0.,  True),
       ('南區', '東區',  9.56840e+00, 0.    , 0.,  True),
       ('南區', '西區', -1.96791e+01, 0.    , 0.,  True),
       ('南區', '西屯區', -4.88629e+01, 0.    , 0.,  True),
       ('南屯區', '東區',  3.47002e+01, 0.    , 0.,  True),
       ('南屯區', '西區',  1.97990e+01, 0.    , 0.,  True),
       ('南屯區', '西屯區', -1.58000e-02, 0.9874, 1., False),
       ('東區', '西區', -2.02562e+01, 0.    , 0.,  True),
       ('東區', '西屯區', -3.79987e+01, 0.    , 0.,  True),
       ('西區', '西屯區', -2.19908e+01, 0.    , 0.,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')]))
```

- build_type vs. unit_price (t-test)

  * levels are not significant, should be in one group

  - 公寓&套房

```
In [455]: comp1 = mc.MultiComparison(df_train_1_scale["unit_price"], df_train_1_scale["build_type"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('住宅大樓', '公寓',  67.9801, 0.    , 0.,  True),
       ('住宅大樓', '套房',  71.4737, 0.    , 0.,  True),
       ('住宅大樓', '華廈',  62.5255, 0.    , 0.,  True),
       ('住宅大樓', '透天厝', 268.0681, 0.    , 0.,  True),
       ('公寓', '套房',   0.3247, 0.7454, 1., False),
       ('公寓', '華廈', -23.3019, 0.    , 0.,  True),
       ('公寓', '透天厝', 259.9044, 0.    , 0.,  True),
       ('套房', '華廈', -23.9088, 0.    , 0.,  True),
       ('套房', '透天厝', 239.2196, 0.    , 0.,  True),
       ('華廈', '透天厝', 263.7997, 0.    , 0.,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')]))
```

- elevator vs. price (t-test)

  * levels are significant

```
In [456]: comp1 = mc.MultiComparison(df_train_1_scale["price_norm"], df_train_1_scale["elevator"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('有', '無', 32.1111, 0., 0.,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')]))
```

- manager vs. price (t-test)

  * levels are significant

```
In [457]: comp1 = mc.MultiComparison(df_train_1_scale["price_norm"], df_train_1_scale["manager"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('有', '無', 20.3536, 0., 0.,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')]))
```

- Model_2
- district vs. price(t-test)

  * levels are not significant, should be in one group

  - 北屯區&西屯區
  - 南區&西區＆南區

```
In [458]: comp1 = mc.MultiComparison(df_train_2_scale["price_norm"], df_train_2_scale["district"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('中區', '北區', -33.987 , 0.    , 0.    ,  True),
       ('中區', '北屯區', -72.2683, 0.    , 0.    ,  True),
       ('中區', '南區', -56.0816, 0.    , 0.    ,  True),
       ('中區', '南屯區', -71.6706, 0.    , 0.    ,  True),
       ('中區', '東區', -48.0902, 0.    , 0.    ,  True),
       ('中區', '西區', -37.2589, 0.    , 0.    ,  True),
       ('中區', '西屯區', -53.8269, 0.    , 0.    ,  True),
       ('北區', '北屯區', -60.6062, 0.    , 0.    ,  True),
       ('北區', '南區', -21.878 , 0.    , 0.    ,  True),
       ('北區', '南屯區', -75.4889, 0.    , 0.    ,  True),
       ('北區', '東區', -18.2836, 0.    , 0.    ,  True),
       ('北區', '西區', -18.5386, 0.    , 0.    ,  True),
       ('北區', '西屯區', -48.676 , 0.    , 0.    ,  True),
       ('北屯區', '南區',  35.9844, 0.    , 0.    ,  True),
       ('北屯區', '南屯區', -40.6479, 0.    , 0.    ,  True),
       ('北屯區', '東區',  16.653 , 0.    , 0.    ,  True),
       ('北屯區', '西區',  22.9164, 0.    , 0.    ,  True),
       ('北屯區', '西屯區',  -2.3816, 0.0172, 0.4828, False),
       ('南區', '南屯區', -60.1647, 0.    , 0.    ,  True),
       ('南區', '東區',  -5.4286, 0.    , 0.    ,  True),
       ('南區', '西區',  -3.0123, 0.0026, 0.0727, False),
       ('南區', '西屯區', -28.9969, 0.    , 0.    ,  True),
       ('南屯區', '東區',  33.556 , 0.    , 0.    ,  True),
       ('南屯區', '西區',  41.0098, 0.    , 0.    ,  True),
       ('南屯區', '西屯區',  29.8651, 0.    , 0.    ,  True),
       ('東區', '西區',   1.6091, 0.1076, 1.    , False),
       ('東區', '西屯區', -13.3902, 0.    , 0.    ,  True),
       ('西區', '西屯區', -19.0221, 0.    , 0.    ,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')]))
```

- district vs. unit_price(t-test)

  * levels are not significant, should be in one group

  - 北屯區&北區&南區

- 南屯&西屯

```
In [459]: comp1 = mc.MultiComparison(df_train_2_scale["unit_price"], df_train_2_scale["district"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('中區', '北區', -18.4661, 0.    , 0.    ,  True),
       ('中區', '北屯區', -18.4125, 0.    , 0.    ,  True),
       ('中區', '南區', -22.5155, 0.    , 0.    ,  True),
       ('中區', '南屯區', -33.2269, 0.    , 0.    ,  True),
       ('中區', '東區',  -6.8792, 0.    , 0.    ,  True),
       ('中區', '西區', -25.0917, 0.    , 0.    ,  True),
       ('中區', '西屯區', -35.5055, 0.    , 0.    ,  True),
       ('北區', '北屯區',   1.9958, 0.046 , 1.    , False),
       ('北區', '南區',   2.4848, 0.013 , 0.3631, False),
       ('北區', '南屯區', -39.8871, 0.    , 0.    ,  True),
       ('北區', '東區',  16.4185, 0.    , 0.    ,  True),
       ('北區', '西區', -13.875 , 0.    , 0.    ,  True),
       ('北區', '西屯區', -43.5929, 0.    , 0.    ,  True),
       ('北屯區', '南區',   0.7391, 0.4599, 1.    , False),
       ('北屯區', '南屯區', -52.9299, 0.    , 0.    ,  True),
       ('北屯區', '東區',  16.6437, 0.    , 0.    ,  True),
       ('北屯區', '西區', -17.7791, 0.    , 0.    ,  True),
       ('北屯區', '西屯區', -59.1399, 0.    , 0.    ,  True),
       ('南區', '南屯區', -44.836 , 0.    , 0.    ,  True),
       ('南區', '東區',  18.1879, 0.    , 0.    ,  True),
       ('南區', '西區', -18.134 , 0.    , 0.    ,  True),
       ('南區', '西屯區', -48.409 , 0.    , 0.    ,  True),
       ('南屯區', '東區',  40.2568, 0.    , 0.    ,  True),
       ('南屯區', '西區',  20.7781, 0.    , 0.    ,  True),
       ('南屯區', '西屯區',   0.9343, 0.3501, 1.    , False),
       ('東區', '西區', -25.4106, 0.    , 0.    ,  True),
       ('東區', '西屯區', -43.3117, 0.    , 0.    ,  True),
       ('西區', '西屯區', -22.2045, 0.    , 0.    ,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')])
```

- elevator vs. price (t-test)

  * levels are significant

```
In [460]: comp1 = mc.MultiComparison(df_train_2_scale["unit_price"], df_train_2_scale["build_type"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('住宅大樓', '公寓',  73.3515, 0., 0.,  True),
       ('住宅大樓', '套房',  71.6285, 0., 0.,  True),
       ('住宅大樓', '華廈',  70.4735, 0., 0.,  True),
       ('住宅大樓', '透天厝', 275.1755, 0., 0.,  True),
       ('公寓', '套房', -12.0756, 0., 0.,  True),
       ('公寓', '華廈', -24.3412, 0., 0.,  True),
       ('公寓', '透天厝', 282.7774, 0., 0.,  True),
       ('套房', '華廈', -11.3118, 0., 0.,  True),
       ('套房', '透天厝', 243.1805, 0., 0.,  True),
       ('華廈', '透天厝', 277.6131, 0., 0.,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')])
```

- manager vs. price (t-test)

  * levels are significant

```
In [461]: comp1 = mc.MultiComparison(df_train_2_scale["price_norm"], df_train_2_scale["manager"])
     ...: comp1.allpairtest(stats.ttest_ind, method= "bonf")
```

```
array([('有', '無', 20.242, 0., 0.,  True)],
      dtype=[('group1', 'O'), ('group2', 'O'), ('stat', '<f8'), ('pval', '<f8'), ('pval_corr', '<f8'), ('reject', '?')])
```

- combine categories based on t-test result for feature reduction purpose
  - Model_1

```
In [462]:
    ...: df_train_1_scale["district_price_norm"] = df_train_1_scale["district"]
    ...: df_train_1_scale.district_price_norm[df_train_1_scale.district_price_norm == "西屯區"] = "北屯區"
    ...: df_train_1_scale.district_price_norm[df_train_1_scale.district_price_norm == "北屯區"] = "西屯北屯區"
    ...: df_train_1_scale.district_price_norm[df_train_1_scale.district_price_norm == "東區"] = "西區"
    ...: df_train_1_scale.district_price_norm[df_train_1_scale.district_price_norm == "南區"] = "西區"
    ...: df_train_1_scale.district_price_norm[df_train_1_scale.district_price_norm == "西區"] = "東西南區"
    ...: df_train_1_scale["district_unitprice"] = df_train_1_scale["district"]
    ...: df_train_1_scale.district_unitprice[df_train_1_scale.district_unitprice == "西屯區"] = "南屯區"
    ...: df_train_1_scale.district_unitprice[df_train_1_scale.district_unitprice == "南屯區"] = "西屯南屯區"
    ...: df_train_1_scale.district_unitprice[df_train_1_scale.district_unitprice == "北屯區"] = "南區"
    ...: df_train_1_scale.district_unitprice[df_train_1_scale.district_unitprice == "南區"] = "北屯南區"
    ...: df_train_1_scale["build_type_unitprice"] = df_train_1_scale["build_type"]
    ...: df_train_1_scale.build_type_unitprice[df_train_1_scale.build_type_unitprice == "公寓"] = "套房"
    ...: df_train_1_scale.build_type_unitprice[df_train_1_scale.build_type_unitprice == "套房"] = "公寓套房"
    ...:
    ...: # Model_1: val
    ...: df_val_1_scale["district_price_norm"] = df_val_1_scale["district"]
    ...: df_val_1_scale.district_price_norm[df_val_1_scale.district_price_norm == "西屯區"] = "北屯區"
    ...: df_val_1_scale.district_price_norm[df_val_1_scale.district_price_norm == "北屯區"] = "西屯北屯區"
    ...: df_val_1_scale.district_price_norm[df_val_1_scale.district_price_norm == "東區"] = "西區"
    ...: df_val_1_scale.district_price_norm[df_val_1_scale.district_price_norm == "南區"] = "西區"
    ...: df_val_1_scale.district_price_norm[df_val_1_scale.district_price_norm == "西區"] = "東西南區"
    ...: df_val_1_scale["district_unitprice"] = df_val_1_scale["district"]
    ...: df_val_1_scale.district_unitprice[df_val_1_scale.district_unitprice == "西屯區"] = "南屯區"
    ...: df_val_1_scale.district_unitprice[df_val_1_scale.district_unitprice == "南屯區"] = "西屯南屯區"
    ...: df_val_1_scale.district_unitprice[df_val_1_scale.district_unitprice == "北屯區"] = "南區"
    ...: df_val_1_scale.district_unitprice[df_val_1_scale.district_unitprice == "南區"] = "北屯南區"
    ...: df_val_1_scale["build_type_unitprice"] = df_val_1_scale["build_type"]
    ...: df_val_1_scale.build_type_unitprice[df_val_1_scale.build_type_unitprice == "公寓"] = "套房"
    ...: df_val_1_scale.build_type_unitprice[df_val_1_scale.build_type_unitprice == "套房"] = "公寓套房"
    ...:
```

```
...: # Model_1: test
...: df_test_1_scale["district_price_norm"] = df_test_1_scale["district"]
...: df_test_1_scale.district_price_norm[df_test_1_scale.district_price_norm == "西屯區"] = "北屯區"
...: df_test_1_scale.district_price_norm[df_test_1_scale.district_price_norm == "北屯區"] = "西屯北屯區"
...: df_test_1_scale.district_price_norm[df_test_1_scale.district_price_norm == "東區"] = "西區"
...: df_test_1_scale.district_price_norm[df_test_1_scale.district_price_norm == "南區"] = "西區"
...: df_test_1_scale.district_price_norm[df_test_1_scale.district_price_norm == "西區"] = "東西南區"
...: df_test_1_scale["district_unitprice"] = df_test_1_scale["district"]
...: df_test_1_scale.district_unitprice[df_test_1_scale.district_unitprice == "西屯區"] = "南屯區"
...: df_test_1_scale.district_unitprice[df_test_1_scale.district_unitprice == "南屯區"] = "西屯南屯區"
...: df_test_1_scale.district_unitprice[df_test_1_scale.district_unitprice == "北屯區"] = "南區"
...: df_test_1_scale.district_unitprice[df_test_1_scale.district_unitprice == "南區"] = "北屯南區"
...: df_test_1_scale["build_type_unitprice"] = df_test_1_scale["build_type"]
...: df_test_1_scale.build_type_unitprice[df_test_1_scale.build_type_unitprice == "公寓"] = "套房"
...: df_test_1_scale.build_type_unitprice[df_test_1_scale.build_type_unitprice == "套房"] = "公寓套房"
```

- Model_2

```
In [463]:
    ...: df_train_2_scale["district_price_norm"] = df_train_2_scale["district"]
    ...: df_train_2_scale.district_price_norm[df_train_2_scale.district_price_norm == "西屯區"] = "北屯區"
    ...: df_train_2_scale.district_price_norm[df_train_2_scale.district_price_norm == "北屯區"] = "西屯北屯區"
    ...: df_train_2_scale.district_price_norm[df_train_2_scale.district_price_norm == "東區"] = "西區"
    ...: df_train_2_scale.district_price_norm[df_train_2_scale.district_price_norm == "南區"] = "西區"
    ...: df_train_2_scale.district_price_norm[df_train_2_scale.district_price_norm == "西區"] = "東西南區"
    ...: df_train_2_scale["district_unitprice"] = df_train_2_scale["district"]
    ...: df_train_2_scale.district_unitprice[df_train_2_scale.district_unitprice == "北區"] = "北屯區"
    ...: df_train_2_scale.district_unitprice[df_train_2_scale.district_unitprice == "南區"] = "北屯區"
    ...: df_train_2_scale.district_unitprice[df_train_2_scale.district_unitprice == "北屯區"] = "北屯北南區"
    ...: df_train_2_scale.district_unitprice[df_train_2_scale.district_unitprice == "南屯區"] = "西屯區"
    ...: df_train_2_scale.district_unitprice[df_train_2_scale.district_unitprice == "西屯區"] = "西屯南屯區"
    ...:
    ...: # Model_2: val
    ...: df_val_2_scale["district_price_norm"] = df_val_2_scale["district"]
    ...: df_val_2_scale.district_price_norm[df_val_2_scale.district_price_norm == "西屯區"] = "北屯區"
    ...: df_val_2_scale.district_price_norm[df_val_2_scale.district_price_norm == "北屯區"] = "西屯北屯區"
    ...: df_val_2_scale.district_price_norm[df_val_2_scale.district_price_norm == "東區"] = "西區"
    ...: df_val_2_scale.district_price_norm[df_val_2_scale.district_price_norm == "南區"] = "西區"
    ...: df_val_2_scale.district_price_norm[df_val_2_scale.district_price_norm == "西區"] = "東西南區"
    ...: df_val_2_scale["district_unitprice"] = df_val_2_scale["district"]
    ...: df_val_2_scale.district_unitprice[df_val_2_scale.district_unitprice == "北區"] = "北屯區"
    ...: df_val_2_scale.district_unitprice[df_val_2_scale.district_unitprice == "南區"] = "北屯區"
    ...: df_val_2_scale.district_unitprice[df_val_2_scale.district_unitprice == "北屯區"] = "北屯北南區"
    ...: df_val_2_scale.district_unitprice[df_val_2_scale.district_unitprice == "南屯區"] = "西屯區"
    ...: df_val_2_scale.district_unitprice[df_val_2_scale.district_unitprice == "西屯區"] = "西屯南屯區"
```

```
...: # Model_2: test
...: df_test_2_scale["district_price_norm"] = df_test_2_scale["district"]
...: df_test_2_scale.district_price_norm[df_test_2_scale.district_price_norm == "西屯區"] = "北屯區"
...: df_test_2_scale.district_price_norm[df_test_2_scale.district_price_norm == "北屯區"] = "西屯北屯區"
...: df_test_2_scale.district_price_norm[df_test_2_scale.district_price_norm == "東區"] = "西區"
...: df_test_2_scale.district_price_norm[df_test_2_scale.district_price_norm == "南區"] = "西區"
...: df_test_2_scale.district_price_norm[df_test_2_scale.district_price_norm == "西區"] = "東西南區"
...: df_test_2_scale["district_unitprice"] = df_test_2_scale["district"]
...: df_test_2_scale.district_unitprice[df_test_2_scale.district_unitprice == "北區"] = "北屯區"
...: df_test_2_scale.district_unitprice[df_test_2_scale.district_unitprice == "南區"] = "北屯區"
...: df_test_2_scale.district_unitprice[df_test_2_scale.district_unitprice == "北屯區"] = "北屯北南區"
...: df_test_2_scale.district_unitprice[df_test_2_scale.district_unitprice == "南屯區"] = "西屯區"
...: df_test_2_scale.district_unitprice[df_test_2_scale.district_unitprice == "西屯區"] = "西屯南屯區"
```

- **Encoding (one-hot encoding)**
  - Model_1 train
  - Manager and elevator

```
In [464]: d_map={"無" : 0, "有" : 1}
     ...: df_train_1_scale["elevator"] = df_train_1_scale["elevator"].map(d_map)
     ...: df_train_1_scale["manager"] = df_train_1_scale["manager"].map(d_map)
```

  - district_price_norm(dummy base: 透天厝_price_norm)

```
In [465]: dummy = pd.get_dummies(df_train_1_scale["district_price_norm"])
     ...: df_train_1_scale = pd.concat((df_train_1_scale, dummy), axis=1)
     ...: df_train_1_scale.isna().sum()
     ...: df_train_1_scale = df_train_1_scale.drop(["中區"], axis=1)
```

- district_unitprice(dummy base: 中區_unitprice)

```
In [466]: dummy = pd.get_dummies(df_train_1_scale["district_unitprice"])
     ...: df_train_1_scale = pd.concat((df_train_1_scale, dummy), axis=1)
     ...: df_train_1_scale.isna().sum()
     ...: df_train_1_scale = df_train_1_scale.drop(["中區"], axis=1)
```

- build_type(dummy base: 透天厝_price_norm)

```
In [467]: dummy = pd.get_dummies(df_train_1_scale["build_type"])
     ...: df_train_1_scale = pd.concat((df_train_1_scale, dummy), axis=1)
     ...: df_train_1_scale.isna().sum()
     ...: df_train_1_scale = df_train_1_scale.drop(["透天厝"], axis=1)
```

- build_type_unitprice(dummy base: 透天厝_unitprice)

```
In [468]: dummy = pd.get_dummies(df_train_1_scale["build_type_unitprice"])
     ...: df_train_1_scale = pd.concat((df_train_1_scale, dummy), axis=1)
     ...: df_train_1_scale.isna().sum()
     ...: df_train_1_scale = df_train_1_scale.drop(["透天厝"], axis=1)
```

- Drop redundant columns

```
In [469]: df_train_1_scale.columns = ['CPI%', 'GDP%', 'M1b', 'age', 'area', 'bathroom_num', 'bedroom_num',
     ...:        'build_share1', 'floor_sold', 'hall_num', 'lat', 'lon',
     ...:        'mortgage_rate', 'parking_num', 'parking_num_norm', 'price',
     ...:        'price_norm', 'story', 'total_deal_floor', 'unit_price', 'district',
     ...:        'road_name', 'build_type', 'elevator', 'manager',
     ...:        'deal_date', 'year', 'month', 'district_price_norm', 'district_unitprice',
     ...:        'build_type_unitprice', '北區_price_norm', '南屯區_price_norm',
     ...:        '東西南區_price_norm', '西屯北屯區_price_norm', '北區_unitprice', '北屯南區_unitprice',
     ...:        '東區_unitprice', '西區_unitprice', '西屯南屯區_unitprice', '住宅大樓_price_norm',
     ...:        '公寓_price_norm', '套房_price_norm', '華廈_price_norm', '住宅大樓_unitprice',
     ...:        '公寓套房_unitprice', '華廈_unitprice']
     ...: df_train_1_scale.drop(["district", "road_name", "build_type", "district_price_norm",
     ...:                        "district_unitprice", "build_type_unitprice"], axis=1, inplace = True)
```

- Model_1 validation

  * Same process with training dataset

```
In [470]:
     ...: d_map={"無" : 0, "有" : 1}
     ...: df_val_1_scale["elevator"] = df_val_1_scale["elevator"].map(d_map)
     ...: df_val_1_scale["manager"] = df_val_1_scale["manager"].map(d_map)
     ...: # one-hot encoding: district_price_norm(dummy base: 中區_price_norm)
     ...: dummy = pd.get_dummies(df_val_1_scale["district_price_norm"])
     ...: df_val_1_scale = pd.concat((df_val_1_scale, dummy), axis=1)
     ...: df_val_1_scale.isna().sum()
     ...: df_val_1_scale = df_val_1_scale.drop(["中區"], axis=1)
     ...: # one-hot encoding: district_unitprice(dummy base: 中區_unitprice)
     ...: dummy = pd.get_dummies(df_val_1_scale["district_unitprice"])
     ...: df_val_1_scale = pd.concat((df_val_1_scale, dummy), axis=1)
     ...: df_val_1_scale.isna().sum()
     ...: df_val_1_scale = df_val_1_scale.drop(["中區"], axis=1)
     ...: # one-hot encoding: build_type(dummy base: 透天厝_price_norm)
     ...: dummy = pd.get_dummies(df_val_1_scale["build_type"])
     ...: df_val_1_scale = pd.concat((df_val_1_scale, dummy), axis=1)
     ...: df_val_1_scale.isna().sum()
     ...: df_val_1_scale = df_val_1_scale.drop(["透天厝"], axis=1)
     ...: # one-hot encoding: build_type_unitprice(dummy base: 透天厝_unitprice)
     ...: dummy = pd.get_dummies(df_val_1_scale["build_type_unitprice"])
     ...: df_val_1_scale = pd.concat((df_val_1_scale, dummy), axis=1)
     ...: df_val_1_scale.isna().sum()
     ...: df_val_1_scale = df_val_1_scale.drop(["透天厝"], axis=1)
     ...: df_val_1_scale.columns = ['CPI%', 'GDP%', 'M1b', 'age', 'area', 'bathroom_num', 'bedroom_num',
     ...:        'build_share1', 'floor_sold', 'hall_num', 'lat', 'lon',
     ...:        'mortgage_rate', 'parking_num', 'parking_num_norm', 'price',
     ...:        'price_norm', 'story', 'total_deal_floor', 'unit_price', 'district',
     ...:        'road_name', 'build_type', 'elevator', 'manager',
     ...:        'deal_date', 'year', 'month', 'district_price_norm', 'district_unitprice',
     ...:        'build_type_unitprice', '北區_price_norm', '南屯區_price_norm',
     ...:        '東西南區_price_norm', '西屯北屯區_price_norm', '北區_unitprice', '北屯南區_unitprice',
     ...:        '東區_unitprice', '西區_unitprice', '西屯南屯區_unitprice', '住宅大樓_price_norm',
     ...:        '公寓_price_norm', '套房_price_norm', '華廈_price_norm', '住宅大樓_unitprice',
     ...:        '公寓套房_unitprice', '華廈_unitprice']
     ...: df_val_1_scale.drop(["district", "road_name", "build_type", "district_price_norm",
     ...:                      "district_unitprice", "build_type_unitprice"], axis=1, inplace = True)
```

- Model_1 test

  * Same process with training dataset

```
In [471]:
    ...: d_map={"無" : 0, "有" : 1}
    ...: df_test_1_scale["elevator"] = df_test_1_scale["elevator"].map(d_map)
    ...: df_test_1_scale["manager"] = df_test_1_scale["manager"].map(d_map)
    ...: # one-hot encoding: district_price_norm(dummy base: 中區_price_norm)
    ...: dummy = pd.get_dummies(df_test_1_scale["district_price_norm"])
    ...: df_test_1_scale = pd.concat((df_test_1_scale, dummy), axis=1)
    ...: df_test_1_scale.isna().sum()
    ...: df_test_1_scale = df_test_1_scale.drop(["中區"], axis=1)
    ...: # one-hot encoding: district_unitprice(dummy base: 中區_unitprice)
    ...: dummy = pd.get_dummies(df_test_1_scale["district_unitprice"])
    ...: df_test_1_scale = pd.concat((df_test_1_scale, dummy), axis=1)
    ...: df_test_1_scale.isna().sum()
    ...: df_test_1_scale = df_test_1_scale.drop(["中區"], axis=1)
    ...: # one-hot encoding: build_type(dummy base: 透天厝_price_norm)
    ...: dummy = pd.get_dummies(df_test_1_scale["build_type"])
    ...: df_test_1_scale = pd.concat((df_test_1_scale, dummy), axis=1)
    ...: df_test_1_scale.isna().sum()
    ...: df_test_1_scale = df_test_1_scale.drop(["透天厝"], axis=1)
    ...: # one-hot encoding: build_type_unitprice(dummy base: 透天厝_unitprice)
    ...: dummy = pd.get_dummies(df_test_1_scale["build_type_unitprice"])
    ...: df_test_1_scale = pd.concat((df_test_1_scale, dummy), axis=1)
    ...: df_test_1_scale.isna().sum()
    ...: df_test_1_scale = df_test_1_scale.drop(["透天厝"], axis=1)
    ...: df_test_1_scale.columns = ['CPI%', 'GDP%', 'M1b', 'age', 'area', 'bathroom_num', 'bedroom_num',
    ...:        'build_share1', 'floor_sold', 'hall_num', 'lat', 'lon',
    ...:        'mortgage_rate', 'parking_num', 'parking_num_norm', 'price',
    ...:        'price_norm', 'story', 'total_deal_floor', 'unit_price', 'district',
    ...:        'road_name', 'build_type', 'elevator', 'manager',
    ...:        'deal_date', 'year', 'month', 'district_price_norm', 'district_unitprice',
    ...:        'build_type_unitprice', '北區_price_norm', '南屯區_price_norm',
    ...:        '東西南區_price_norm', '西屯北屯區_price_norm', '北區_unitprice', '北屯南區_unitprice',
    ...:        '東區_unitprice', '西區_unitprice', '西屯南屯區_unitprice', '住宅大樓_price_norm',
    ...:        '公寓_price_norm', '套房_price_norm', '華廈_price_norm', '住宅大樓_unitprice',
    ...:        '公寓套房_unitprice', '華廈_unitprice']
    ...: df_test_1_scale.drop(["district", "road_name", "build_type", "district_price_norm",
    ...:                       "district_unitprice", "build_type_unitprice"], axis=1, inplace = True)
```

- Model_2 train
- Manager and elevator

```
In [472]: d_map={"無" : 0, "有" : 1}
     ...: df_train_2_scale["elevator"] = df_train_2_scale["elevator"].map(d_map)
     ...: df_train_2_scale["manager"] = df_train_2_scale["manager"].map(d_map)
```

- district_price_norm(dummy base: 中區_price_norm)

```
In [473]: dummy = pd.get_dummies(df_train_2_scale["district_price_norm"])
     ...: df_train_2_scale = pd.concat((df_train_2_scale, dummy), axis=1)
     ...: df_train_2_scale.isna().sum()
     ...: df_train_2_scale = df_train_2_scale.drop(["中區"], axis=1)
```

- district_unitprice(dummy base: 中區_unitprice)

```
In [474]: dummy = pd.get_dummies(df_train_2_scale["district_unitprice"])
     ...: df_train_2_scale = pd.concat((df_train_2_scale, dummy), axis=1)
     ...: df_train_2_scale.isna().sum()
     ...: df_train_2_scale = df_train_2_scale.drop(["中區"], axis=1)
```

- build_type(dummy base: 透天厝)

```
In [475]: dummy = pd.get_dummies(df_train_2_scale["build_type"])
     ...: df_train_2_scale = pd.concat((df_train_2_scale, dummy), axis=1)
     ...: df_train_2_scale.isna().sum()
     ...: df_train_2_scale = df_train_2_scale.drop(["透天厝"], axis=1)
```

- Drop redundant columns

```
In [476]: df_train_2_scale.columns = ['CPI%', 'GDP%', 'M1b', 'age', 'area', 'bathroom_num', 'bedroom_num',
     ...:        'build_share1', 'floor_sold', 'hall_num', 'lat', 'lon', 'mortgage_rate',
     ...:        'parking_num', 'parking_num_norm', 'price', 'price_norm', 'story',
     ...:        'total_deal_floor', 'unit_price', 'district', 'road_name', 'build_type',
     ...:        'elevator', 'manager', 'deal_date', 'year', 'month',
     ...:        'district_price_norm', 'district_unitprice', '北區_price_norm', '南屯區_price_norm',
     ...:        '東西南區_price_norm', '西屯北屯區_price_norm', '北屯北南區_unitprice', '東區_unitprice',
     ...:        '西區_unitprice', '西屯南屯區_unitprice', '住宅大樓', '公寓', '套房', '華廈']
     ...: df_train_2_scale.drop(["district", "road_name", "build_type", "district_price_norm",
     ...:                        "district_unitprice"], axis=1, inplace = True)
```

- Model_2 validation

  * Same process with training dataset

```
In [477]: d_map={"無" : 0, "有" : 1}
     ...: df_val_2_scale["elevator"] = df_val_2_scale["elevator"].map(d_map)
     ...: df_val_2_scale["manager"] = df_val_2_scale["manager"].map(d_map)
     ...: # one-hot encoding: district_price_norm(dummy base: 中區_price_norm)
     ...: dummy = pd.get_dummies(df_val_2_scale["district_price_norm"])
     ...: df_val_2_scale = pd.concat((df_val_2_scale, dummy), axis=1)
     ...: df_val_2_scale.isna().sum()
     ...: df_val_2_scale = df_val_2_scale.drop(["中區"], axis=1)
     ...: # one-hot encoding: district_unitprice(dummy base: 中區_unitprice)
     ...: dummy = pd.get_dummies(df_val_2_scale["district_unitprice"])
     ...: df_val_2_scale = pd.concat((df_val_2_scale, dummy), axis=1)
     ...: df_val_2_scale.isna().sum()
     ...: df_val_2_scale = df_val_2_scale.drop(["中區"], axis=1)
     ...: # one-hot encoding: build_type(dummy base: 透天厝)
     ...: dummy = pd.get_dummies(df_val_2_scale["build_type"])
     ...: df_val_2_scale = pd.concat((df_val_2_scale, dummy), axis=1)
     ...: df_val_2_scale.isna().sum()
     ...: df_val_2_scale = df_val_2_scale.drop(["透天厝"], axis=1)
     ...: df_val_2_scale.columns = ['CPI%', 'GDP%', 'M1b', 'age', 'area', 'bathroom_num', 'bedroom_num',
     ...:        'build_share1', 'floor_sold', 'hall_num', 'lat', 'lon', 'mortgage_rate',
     ...:        'parking_num', 'parking_num_norm', 'price', 'price_norm', 'story',
     ...:        'total_deal_floor', 'unit_price', 'district', 'road_name', 'build_type',
     ...:        'elevator', 'manager', 'deal_date', 'year', 'month',
     ...:        'district_price_norm', 'district_unitprice', '北區_price_norm', '南屯區_price_norm',
     ...:        '東西南區_price_norm', '西屯北屯區_price_norm', '北屯北南區_unitprice', '東區_unitprice',
     ...:        '西區_unitprice', '西屯南屯區_unitprice', '住宅大樓', '公寓', '套房', '華廈']
     ...: df_val_2_scale.drop(["district", "road_name", "build_type", "district_price_norm",
     ...:                      "district_unitprice"], axis=1, inplace = True)
```

- Model_2 test

  * Same process with training dataset

```
In [478]:
     ...:
     ...: d_map={"無" : 0, "有" : 1}
     ...: df_test_2_scale["elevator"] = df_test_2_scale["elevator"].map(d_map)
     ...: df_test_2_scale["manager"] = df_test_2_scale["manager"].map(d_map)
     ...: # one-hot encoding: district_price_norm(dummy base: 中區_price_norm)
     ...: dummy = pd.get_dummies(df_test_2_scale["district_price_norm"])
     ...: df_test_2_scale = pd.concat((df_test_2_scale, dummy), axis=1)
     ...: df_test_2_scale.isna().sum()
     ...: df_test_2_scale = df_test_2_scale.drop(["中區"], axis=1)
     ...: # one-hot encoding: district_unitprice(dummy base: 中區_unitprice)
     ...: dummy = pd.get_dummies(df_test_2_scale["district_unitprice"])
     ...: df_test_2_scale = pd.concat((df_test_2_scale, dummy), axis=1)
     ...: df_test_2_scale.isna().sum()
     ...: df_test_2_scale = df_test_2_scale.drop(["中區"], axis=1)
     ...: # one-hot encoding: build_type(dummy base: 透天厝)
     ...: dummy = pd.get_dummies(df_test_2_scale["build_type"])
     ...: df_test_2_scale = pd.concat((df_test_2_scale, dummy), axis=1)
     ...: df_test_2_scale.isna().sum()
     ...: df_test_2_scale = df_test_2_scale.drop(["透天厝"], axis=1)
     ...: df_test_2_scale.columns = ['CPI%', 'GDP%', 'M1b', 'age', 'area', 'bathroom_num', 'bedroom_num',
     ...:        'build_share1', 'floor_sold', 'hall_num', 'lat', 'lon', 'mortgage_rate',
     ...:        'parking_num', 'parking_num_norm', 'price', 'price_norm', 'story',
     ...:        'total_deal_floor', 'unit_price', 'district', 'road_name', 'build_type',
     ...:        'elevator', 'manager', 'deal_date', 'year', 'month',
     ...:        'district_price_norm', 'district_unitprice', '北區_price_norm', '南屯區_price_norm',
     ...:        '東西南區_price_norm', '西屯北屯區_price_norm', '北屯北南區_unitprice', '東區_unitprice',
     ...:        '西區_unitprice', '西屯南屯區_unitprice', '住宅大樓', '公寓', '套房', '華廈']
     ...: df_test_2_scale.drop(["district", "road_name", "build_type", "district_price_norm",
     ...:                       "district_unitprice"], axis=1, inplace = True)
```

Autocorrelation check

*** Other features have been manipulated except for time related data, How to deal with them? Is time series analysis applicable to this data set? We need autocorrelation check!

- splitting the data and set deal_date to index
  - Model_1_1

```
In [480]: X_train_1_1_scale1 = df_train_1_scale.drop(['parking_num', 'price', 'price_norm', 'unit_price', '北區_unitprice',
     ...:                                             '北屯南區_unitprice', '東區_unitprice','西區_unitprice',
     ...:                                             '西屯南屯區_unitprice', '住宅大樓_unitprice', '公寓套房_unitprice',
     ...:                                             '華廈_unitprice', 'year', 'month'], axis = 1)
     ...: y_train_1_1_scale1 = df_train_1_scale.price_norm
     ...: X_train_1_1_scale1.set_index('deal_date', inplace=True)
     ...: y_train_1_1_scale1.index = df_train_1_scale['deal_date']
     ...:
     ...: X_val_1_1_scale1 = df_val_1_scale.drop(['parking_num', 'price', 'price_norm', 'unit_price', '北區_unitprice',
     ...:                                         '北屯南區_unitprice', '東區_unitprice','西區_unitprice',
     ...:                                         '西屯南屯區_unitprice', '住宅大樓_unitprice', '公寓套房_unitprice',
     ...:                                         '華廈_unitprice', 'year', 'month'], axis = 1)
     ...: y_val_1_1_scale1 = df_val_1_scale.price_norm
     ...: X_val_1_1_scale1.set_index('deal_date', inplace=True)
     ...: y_val_1_1_scale1.index = df_val_1_scale['deal_date']
     ...:
     ...: X_test_1_1_scale1 = df_test_1_scale.drop(['parking_num', 'price', 'price_norm', 'unit_price', '北區_unitprice',
     ...:                                           '北屯南區_unitprice', '東區_unitprice','西區_unitprice',
     ...:                                           '西屯南屯區_unitprice', '住宅大樓_unitprice', '公寓套房_unitprice',
     ...:                                           '華廈_unitprice', 'year', 'month'], axis = 1)
     ...: y_test_1_1_scale1 = df_test_1_scale.price_norm
     ...: X_test_1_1_scale1.set_index('deal_date', inplace=True)
     ...: y_test_1_1_scale1.index = df_test_1_scale['deal_date']
```

- Model_1_2

```
In [482]:
...: X_train_1_2_scale1 = df_train_1_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                             '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm',
...:                                            '西屯北屯區_price_norm', '住宅大樓_price_norm', '公寓_price_norm',
...:                                            '套房_price_norm', '華廈_price_norm', 'year',
...:                                            'month'], axis = 1)
...: y_train_1_2_scale1 = df_train_1_scale.unit_price
...: X_train_1_2_scale1.set_index('deal_date', inplace=True)
...: y_train_1_2_scale1.index = df_train_1_scale['deal_date']
...:
...: X_val_1_2_scale1 = df_val_1_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                         '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm',
...:                                        '西屯北屯區_price_norm', '住宅大樓_price_norm', '公寓_price_norm',
...:                                        '套房_price_norm', '華廈_price_norm', 'year',
...:                                        'month'], axis = 1)
...: y_val_1_2_scale1 = df_val_1_scale.unit_price
...: X_val_1_2_scale1.set_index('deal_date', inplace=True)
...: y_val_1_2_scale1.index = df_val_1_scale['deal_date']
...:
...: X_test_1_2_scale1 = df_test_1_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                           '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm',
...:                                          '西屯北屯區_price_norm', '住宅大樓_price_norm', '公寓_price_norm',
...:                                          '套房_price_norm', '華廈_price_norm', 'year',
...:                                          'month'], axis = 1)
...: y_test_1_2_scale1 = df_test_1_scale.unit_price
...: X_test_1_2_scale1.set_index('deal_date', inplace=True)
...: y_test_1_2_scale1.index = df_test_1_scale['deal_date']
```

- Model_2_1

```
In [483]:
...: X_train_2_1_scale1 = df_train_2_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                             '北屯北南區_unitprice', '東區_unitprice', '西區_unitprice',
...:                                            '西屯南屯區_unitprice', 'year', 'month'], axis = 1)
...: y_train_2_1_scale1 = df_train_2_scale.price_norm
...: X_train_2_1_scale1.set_index('deal_date', inplace=True)
...: y_train_2_1_scale1.index = df_train_2_scale['deal_date']
...:
...: X_val_2_1_scale1 = df_val_2_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                         '北屯北南區_unitprice', '東區_unitprice', '西區_unitprice',
...:                                        '西屯南屯區_unitprice', 'year', 'month'], axis = 1)
...: y_val_2_1_scale1 = df_val_2_scale.price_norm
...: X_val_2_1_scale1.set_index('deal_date', inplace=True)
...: y_val_2_1_scale1.index = df_val_2_scale['deal_date']
...:
...: X_test_2_1_scale1 = df_test_2_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                           '北屯北南區_unitprice', '東區_unitprice', '西區_unitprice',
...:                                          '西屯南屯區_unitprice', 'year', 'month'], axis = 1)
...: y_test_2_1_scale1 = df_test_2_scale.price_norm
...: X_test_2_1_scale1.set_index('deal_date', inplace=True)
...: y_test_2_1_scale1.index = df_test_2_scale['deal_date']
```

- Model_2_2

```
In [484]:
...: X_train_2_2_scale1 = df_train_2_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                             '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm',
...:                                            '西屯北屯區_price_norm', 'year', 'month'], axis = 1)
...: y_train_2_2_scale1 = df_train_2_scale.unit_price
...: X_train_2_2_scale1.set_index('deal_date', inplace=True)
...: y_train_2_2_scale1.index = df_train_2_scale['deal_date']
...:
...: X_val_2_2_scale1 = df_val_2_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                         '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm',
...:                                        '西屯北屯區_price_norm', 'year', 'month'], axis = 1)
...: y_val_2_2_scale1 = df_val_2_scale.unit_price
...: X_val_2_2_scale1.set_index('deal_date', inplace=True)
...: y_val_2_2_scale1.index = df_val_2_scale['deal_date']
...:
...: X_test_2_2_scale1 = df_test_2_scale.drop(['parking_num', 'price_norm', 'unit_price', 'price',
...:                                           '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm',
...:                                          '西屯北屯區_price_norm', 'year', 'month'], axis = 1)
...: y_test_2_2_scale1 = df_test_2_scale.unit_price
...: X_test_2_2_scale1.set_index('deal_date', inplace=True)
...: y_test_2_2_scale1.index = df_test_2_scale['deal_date']
```

- Residuals plot and Durbin-Watson test are used to check for the presence of autocorrelation.

  *** Durbin-Watson statistic will always have a value ranging between 0 and 4. A value of 2.0 indicates there is no autocorrelation detected in the sample.

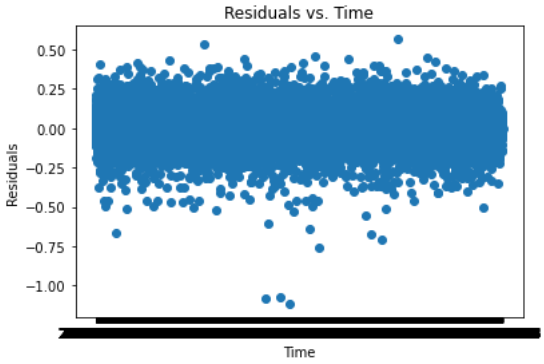
  * Autocorrelation is NOT significant. I'm not going to use time series model.

- Model_1_1

  Durbin-Watson

```
Durbin-Watson test statistic: 1.9961351077901701
```

Residuals vs. Time

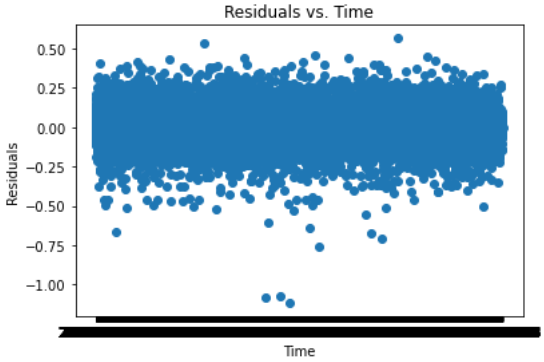


- Model_1_2

  Durbin-Watson

  ```
  Durbin-Watson test statistic: 1.9956873587193238
  ```

  Residuals vs. Time

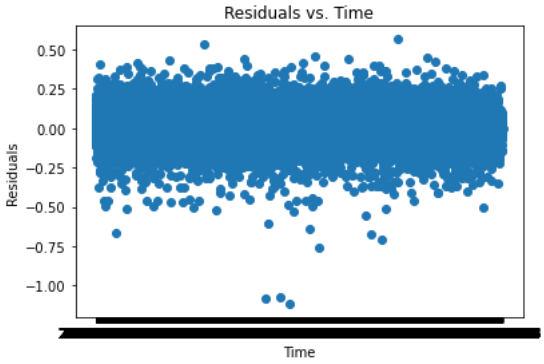


- Model_2_1

  Durbin-Watson

  ```
  Durbin-Watson test statistic: 1.9956873587193238
  ```

  Residuals vs. Time

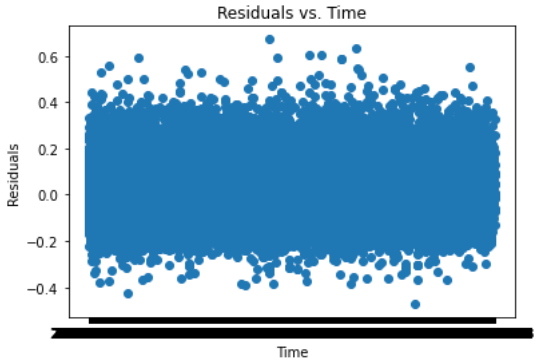


- Model_2_2

  Durbin-Watson

  ```
  Durbin-Watson test statistic: 1.9928693606097327
  ```

Residuals vs. Time



Categorize time feature

- Since there is no significant autocorrelation, try to remove the **deal_date** feature from the dataset. Additionally, remove the **year** feature and replace it with a **quarter** feature to reduce the number of features in the dataset.

```
In [488]: quarter_dict = {1:"Q1", 2:"Q1", 3:"Q1", 4:"Q2", 5:"Q2", 6:"Q2", 7:"Q3", 8:"Q3",
     ...:                 9:"Q3", 10:"Q4", 11:"Q4", 12:"Q4"}
     ...: df_train_1_scale['quarter'] = df_train_1_scale['month'].map(quarter_dict)
     ...: df_val_1_scale['quarter'] = df_val_1_scale['month'].map(quarter_dict)
     ...: df_test_1_scale['quarter'] = df_test_1_scale['month'].map(quarter_dict)
     ...: df_train_2_scale['quarter'] = df_train_2_scale['month'].map(quarter_dict)
     ...: df_val_2_scale['quarter'] = df_val_2_scale['month'].map(quarter_dict)
     ...: df_test_2_scale['quarter'] = df_test_2_scale['month'].map(quarter_dict)
```

- Encoding for quarter
  - Model_1 (dummy base: Q1)

```
In [489]: dummy = pd.get_dummies(df_train_1_scale["quarter"])
     ...: df_train_1_scale = pd.concat((df_train_1_scale, dummy), axis=1)
     ...: df_train_1_scale.isna().sum()
     ...: df_train_1_scale = df_train_1_scale.drop(["Q1"], axis=1)
     ...:
     ...: dummy = pd.get_dummies(df_val_1_scale["quarter"])
     ...: df_val_1_scale = pd.concat((df_val_1_scale, dummy), axis=1)
     ...: df_val_1_scale.isna().sum()
     ...: df_val_1_scale = df_val_1_scale.drop(["Q1"], axis=1)
     ...:
     ...: dummy = pd.get_dummies(df_test_1_scale["quarter"])
     ...: df_test_1_scale = pd.concat((df_test_1_scale, dummy), axis=1)
     ...: df_test_1_scale.isna().sum()
     ...: df_test_1_scale = df_test_1_scale.drop(["Q1"], axis=1)
```

  - Model_2 (dummy base: Q1)

```
In [490]: dummy = pd.get_dummies(df_train_2_scale["quarter"])
     ...: df_train_2_scale = pd.concat((df_train_2_scale, dummy), axis=1)
     ...: df_train_2_scale.isna().sum()
     ...: df_train_2_scale = df_train_2_scale.drop(["Q1"], axis=1)
     ...:
     ...: dummy = pd.get_dummies(df_val_2_scale["quarter"])
     ...: df_val_2_scale = pd.concat((df_val_2_scale, dummy), axis=1)
     ...: df_val_2_scale.isna().sum()
     ...: df_val_2_scale = df_val_2_scale.drop(["Q1"], axis=1)
     ...:
     ...: dummy = pd.get_dummies(df_test_2_scale["quarter"])
     ...: df_test_2_scale = pd.concat((df_test_2_scale, dummy), axis=1)
     ...: df_test_2_scale.isna().sum()
     ...: df_test_2_scale = df_test_2_scale.drop(["Q1"], axis=1)
```

Re-splitting Data

- Model_1_1

```
In [491]:
    ...: X_train_1_1_scale = df_train_1_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_unitprice',
    ...:                                            '北屯南區_unitprice', '東區_unitprice','西區_unitprice',
    ...:                                            '西屯南屯區_unitprice', '住宅大樓_unitprice',
    ...:                                            '公寓套房_unitprice', '華廈_unitprice', 'year',
    ...:                                            'deal_date', 'month', 'quarter', 'price'], axis = 1)
    ...: y_train_1_1_scale = df_train_1_scale.price_norm
    ...: X_val_1_1_scale = df_val_1_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_unitprice',
    ...:                                        '北屯南區_unitprice', '東區_unitprice','西區_unitprice',
    ...:                                        '西屯南屯區_unitprice', '住宅大樓_unitprice',
    ...:                                        '公寓套房_unitprice', '華廈_unitprice', 'year',
    ...:                                        'deal_date', 'month', 'quarter', 'price'], axis = 1)
    ...: y_val_1_1_scale = df_val_1_scale.price_norm
    ...: X_test_1_1_scale = df_test_1_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_unitprice',
    ...:                                          '北屯南區_unitprice', '東區_unitprice','西區_unitprice',
    ...:                                          '西屯南屯區_unitprice', '住宅大樓_unitprice',
    ...:                                          '公寓套房_unitprice', '華廈_unitprice', 'year',
    ...:                                          'deal_date', 'month', 'quarter', 'price'], axis = 1)
    ...: y_test_1_1_scale = df_test_1_scale.price_norm
```

- Model_1_2

```
In [492]:
    ...: X_train_1_2_scale = df_train_1_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_price_norm',
    ...:                                            '南屯區_price_norm', '東西南區_price_norm',
    ...:                                            '西屯北屯區_price_norm', '住宅大樓_price_norm',
    ...:                                            '公寓_price_norm', '套房_price_norm',
    ...:                                            '華廈_price_norm', 'year', 'deal_date', 'month',
    ...:                                            'quarter', 'price'], axis = 1)
    ...: y_train_1_2_scale = df_train_1_scale.unit_price
    ...: X_val_1_2_scale = df_val_1_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_price_norm',
    ...:                                        '南屯區_price_norm', '東西南區_price_norm',
    ...:                                        '西屯北屯區_price_norm', '住宅大樓_price_norm',
    ...:                                        '公寓_price_norm', '套房_price_norm',
    ...:                                        '華廈_price_norm', 'year', 'deal_date', 'month',
    ...:                                        'quarter', 'price'], axis = 1)
    ...: y_val_1_2_scale = df_val_1_scale.unit_price
    ...: X_test_1_2_scale = df_test_1_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_price_norm',
    ...:                                          '南屯區_price_norm', '東西南區_price_norm',
    ...:                                          '西屯北屯區_price_norm', '住宅大樓_price_norm',
    ...:                                          '公寓_price_norm', '套房_price_norm',
    ...:                                          '華廈_price_norm', 'year', 'deal_date', 'month',
    ...:                                          'quarter', 'price'], axis = 1)
    ...: y_test_1_2_scale = df_test_1_scale.unit_price
```

- Model_2_1

```
In [493]:
    ...: X_train_2_1_scale = df_train_2_scale.drop(['parking_num', 'price_norm', 'unit_price', '北屯北南區_unitprice',
    ...:                                            '東區_unitprice', '西區_unitprice', '西屯南屯區_unitprice',
    ...:                                            'year', 'deal_date', 'month', 'quarter', 'price'], axis = 1)
    ...: y_train_2_1_scale = df_train_2_scale.price_norm
    ...: X_val_2_1_scale = df_val_2_scale.drop(['parking_num', 'price_norm', 'unit_price', '北屯北南區_unitprice',
    ...:                                        '東區_unitprice', '西區_unitprice', '西屯南屯區_unitprice',
    ...:                                        'year', 'deal_date', 'month', 'quarter', 'price'], axis = 1)
    ...: y_val_2_1_scale = df_val_2_scale.price_norm
    ...: X_test_2_1_scale = df_test_2_scale.drop(['parking_num', 'price_norm', 'unit_price', '北屯北南區_unitprice',
    ...:                                          '東區_unitprice', '西區_unitprice', '西屯南屯區_unitprice',
    ...:                                          'year', 'deal_date', 'month', 'quarter', 'price'], axis = 1)
    ...: y_test_2_1_scale = df_test_2_scale.price_norm
```

- Model_2_2

```
In [494]:
    ...: X_train_2_2_scale = df_train_2_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_price_norm',
    ...:                                            '南屯區_price_norm', '東西南區_price_norm',
    ...:                                            '西屯北屯區_price_norm', 'year', 'deal_date',
    ...:                                            'month', 'quarter', 'price'], axis = 1)
    ...: y_train_2_2_scale = df_train_2_scale.unit_price
    ...: X_val_2_2_scale = df_val_2_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_price_norm',
    ...:                                        '南屯區_price_norm', '東西南區_price_norm',
    ...:                                        '西屯北屯區_price_norm', 'year', 'deal_date',
    ...:                                        'month', 'quarter', 'price'], axis = 1)
    ...: y_val_2_2_scale = df_val_2_scale.unit_price
    ...: X_test_2_2_scale = df_test_2_scale.drop(['parking_num', 'price_norm', 'unit_price', '北區_price_norm',
    ...:                                          '南屯區_price_norm', '東西南區_price_norm',
    ...:                                          '西屯北屯區_price_norm', 'year', 'deal_date',
    ...:                                          'month', 'quarter', 'price'], axis = 1)
    ...: y_test_2_2_scale = df_test_2_scale.unit_price
```

Feature Selection

SelectFpr

* alpha = 0.05

* store it in v1

- Model_1_1
  - Dropped: ['CPI%', 'Q2', 'Q3', 'Q4']

features	p value
CPI%	0.4724
GDP%	0.0000
M1b	0.0000
age	0.0000
area	0.0000
bathroom_num	0.0000
bedroom_num	0.0000
build_share1	0.0000
floor_sold	0.0000
hall_num	0.0000
lat	0.0000
lon	0.0000
mortgage_rate	0.0000
parking_num_norm	0.0000
story	0.0000
total_deal_floor	0.0000
elevator	0.0000
manager	0.0000
北區_price_norm	0.0000
南屯區_price_norm	0.0000
東西南區_price_norm	0.0000
西屯北屯區_price_norm	0.0000
住宅大樓_price_norm	0.0000
公寓_price_norm	0.0000
套房_price_norm	0.0000
華廈_price_norm	0.0000
Q2	0.2241
Q3	0.5004
Q4	0.7449

features	f value
CPI%	0.5163
GDP%	82.1755
M1b	1075.3070
age	28999.8162
area	314839.4618
bathroom_num	22524.4089
bedroom_num	37625.4821
build_share1	33653.4615
floor_sold	1400.6664
hall_num	36549.4325
lat	234.3988
lon	598.6926
mortgage_rate	860.4840
parking_num_norm	54105.4464
story	7495.7210
total_deal_floor	5445.6692
elevator	1031.1231
manager	414.2707
北區_price_norm	2981.2477
南屯區_price_norm	3696.9827
東西南區_price_norm	1012.4202
西屯北屯區_price_norm	1168.1587
住宅大樓_price_norm	12515.4685
公寓_price_norm	5119.5903
套房_price_norm	25402.4673
華廈_price_norm	3751.0161
Q2	1.4780
Q3	0.4540
Q4	0.1059

- Model_1_2
  - Dropped: []

features	p value
CPI%	0.0000
GDP%	0.0000
M1b	0.0000
age	0.0000
area	0.0000
bathroom_num	0.0000
bedroom_num	0.0000
build_share1	0.0000
floor_sold	0.0000
hall_num	0.0000
lat	0.0000
lon	0.0000
mortgage_rate	0.0000
parking_num_norm	0.0000
story	0.0000
total_deal_floor	0.0000
elevator	0.0000
manager	0.0000
北區_unitprice	0.0000
北屯南區_unitprice	0.0000
東區_unitprice	0.0000
西區_unitprice	0.0205
西屯南屯區_unitprice	0.0000
住宅大樓_unitprice	0.0000
公寓套房_unitprice	0.0000
華廈_unitprice	0.0000
Q2	0.0251
Q3	0.0009
Q4	0.0000

features	f value
CPI%	548.3942
GDP%	457.3415
M1b	11250.1964
age	18043.0525
area	2238.1632
bathroom_num	5808.4235
bedroom_num	3251.2141
build_share1	6864.3930
floor_sold	25625.5583
hall_num	69.7000
lat	72.0328
lon	4972.7601
mortgage_rate	6493.8958
parking_num_norm	33036.9009
story	48175.9120
total_deal_floor	42959.5579
elevator	12623.2384
manager	21188.2004
北區_unitprice	186.1864
北屯南區_unitprice	2822.8201
東區_unitprice	516.2899
西區_unitprice	5.3707
西屯南屯區_unitprice	6202.4444
住宅大樓_unitprice	30903.0772
公寓套房_unitprice	2051.4280
華廈_unitprice	228.1121
Q2	5.0173
Q3	11.1222
Q4	17.8620

- Model_2_1
  - Dropped: ['Q3', 'Q4']

features	p value
CPI%	0.0000
GDP%	0.0224
M1b	0.0000
age	0.0000
area	0.0000
bathroom_num	0.0000
bedroom_num	0.0000
build_share1	0.0000
floor_sold	0.0000
hall_num	0.0000
lat	0.0000
lon	0.0000
mortgage_rate	0.0000
parking_num_norm	0.0000
story	0.0000
total_deal_floor	0.0000
elevator	0.0000
manager	0.0000
北區_price_norm	0.0000
南屯區_price_norm	0.0000
東西南區_price_norm	0.0000
西屯北屯區_price_norm	0.0000
住宅大樓	0.0000
公寓	0.0000
套房	0.0000
華廈	0.0000
Q2	0.0042
Q3	0.6026
Q4	0.6377

features	f value
CPI%	134.1475
GDP%	5.2126
M1b	928.9874
age	30495.3488
area	342251.6612
bathroom_num	22760.8903
bedroom_num	38665.4585
build_share1	33710.6166
floor_sold	1570.8713
hall_num	35574.4834
lat	198.1597
lon	686.3140
mortgage_rate	587.4876
parking_num_norm	53553.1297
story	7649.9851
total_deal_floor	5755.5398
elevator	913.7317
manager	409.7390
北區_price_norm	3119.5741
南屯區_price_norm	3619.0493
東西南區_price_norm	999.2391
西屯北屯區_price_norm	1127.4834
住宅大樓	14503.7429
公寓	5277.3876
套房	30229.7476
華廈	3347.0252
Q2	8.2150
Q3	0.2711
Q4	0.2217

- Model_2_2
  - Dropped: ['GDP%', 'Q3']

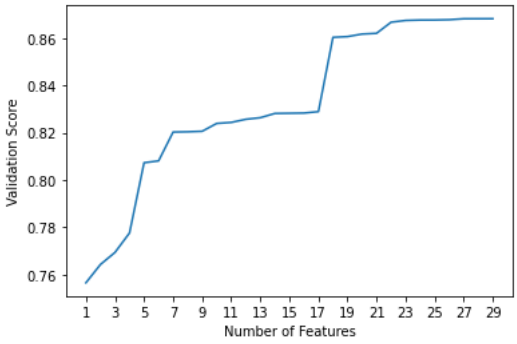
features	p value
CPI%	0.0000
GDP%	0.6357
M1b	0.0000
age	0.0000
area	0.0000
bathroom_num	0.0000
bedroom_num	0.0000
build_share1	0.0000
floor_sold	0.0000
hall_num	0.0037
lat	0.0000
lon	0.0000
mortgage_rate	0.0000
parking_num_norm	0.0000
story	0.0000
total_deal_floor	0.0000
elevator	0.0000
manager	0.0000
北屯北南區_unitprice	0.0000
東區_unitprice	0.0000
西區_unitprice	0.0238
西屯南屯區_unitprice	0.0000
住宅大樓	0.0000
公寓	0.0000
套房	0.0000
華廈	0.0000
Q2	0.0000
Q3	0.1101
Q4	0.0000

features	f value
CPI%	491.5151
GDP%	0.2244
M1b	4921.2546
age	24312.3061
area	4101.4183
bathroom_num	4989.7007
bedroom_num	2333.1285
build_share1	7218.0596
floor_sold	29378.4401
hall_num	8.4060
lat	63.7368
lon	5538.4998
mortgage_rate	3377.0930
parking_num_norm	41751.8255
story	56645.5478
total_deal_floor	45735.4455
elevator	2796.3470
manager	22381.6180
北屯北南區_unitprice	3905.5341
東區_unitprice	807.4649
西區_unitprice	5.1073
西屯南屯區_unitprice	6830.0985
住宅大樓	35090.3462
公寓	1091.7831
套房	764.0023
華廈	445.4237
Q2	46.1538
Q3	2.5528
Q4	29.3951

SelectKBest

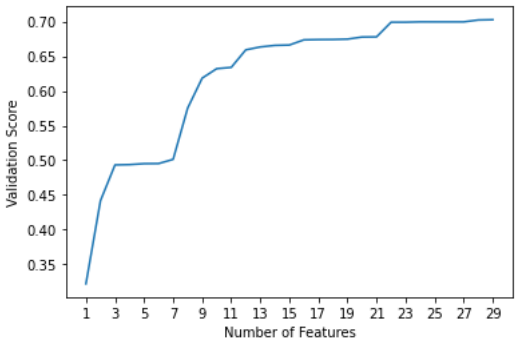
* store it in v2

- Model_1_1
  - K = 18



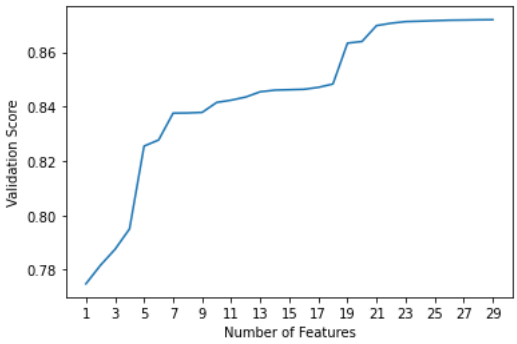
- Dropped: ['CPI%', 'GDP%', 'lat', 'lon', 'mortgage_rate', 'elevator', 'manager', '東西南區_price_norm', 'Q2', 'Q3', 'Q4']

- Model_1_2
  - K = 22



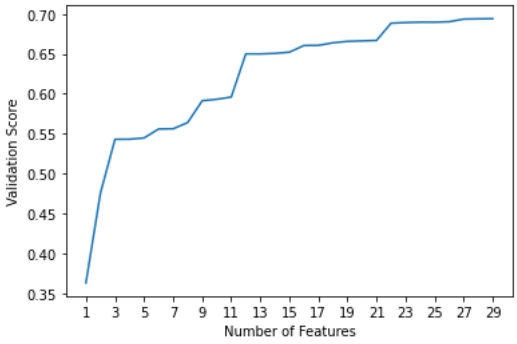
  - Dropped: ['hall_num', 'lat', '北區_unitprice', '西區_unitprice', 'Q2', 'Q3', 'Q4']

- Model_2_1
  - K = 21



  - Dropped: ['CPI%', 'GDP%', 'lat', 'mortgage_rate', 'manager', 'Q2', 'Q3', 'Q4]

- Model_2_2
  - K = 22



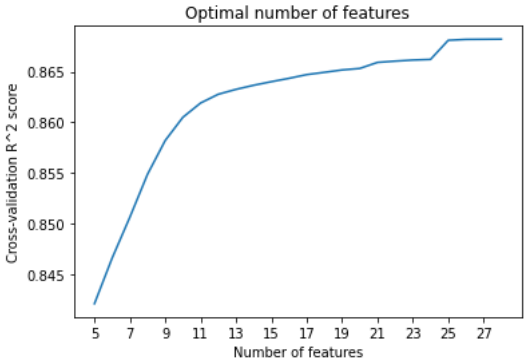
- Dropped: ['GDP%', 'hall_num', 'lat', '西區_unitprice', 'Q2', 'Q3', 'Q4']

Forward Selection

* store it in v3

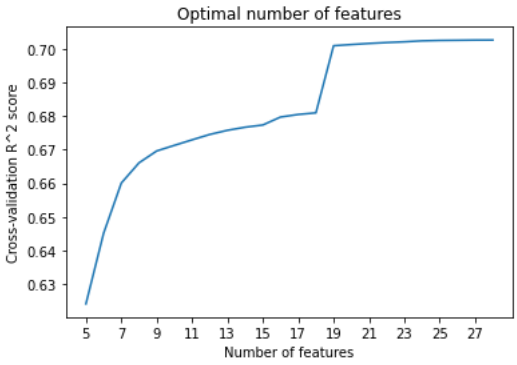
* (from sklearn.feature_selection import SequentialFeatureSelector as SFS)

- Model_1_1
  - n_features_to_select = 11



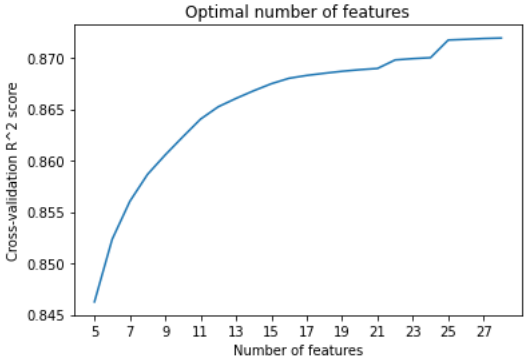
  - features: ['M1b', 'age', 'area', 'bedroom_num', 'build_share1', 'hall_num', 'lon', 'parking_num_norm', 'story', 'manager', '套房_price_norm']

- Model_1_2
  - n_features_to_select = 19



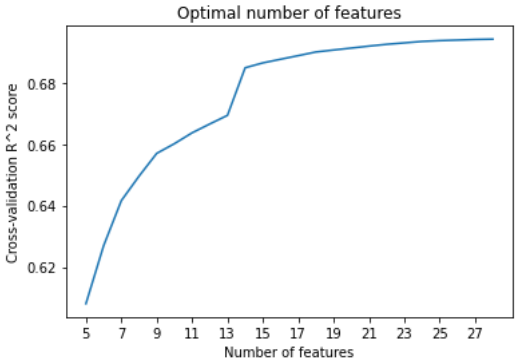
- features: ['M1b', 'age', 'area', 'bathroom_num', 'build_share1', 'floor_sold', 'mortgage_rate', 'parking_num_norm', 'story', 'total_deal_floor', 'manager', '北區_unitprice', '北屯南區_unitprice', '東區_unitprice', '西區_unitprice', '西屯南屯區_unitprice', '住宅大樓_unitprice', '公寓套房_unitprice', '_unitprice']

- Model_2_1
  - n_features_to_select = 12



  - features: ['CPI%', 'GDP%', 'bathroom_num', 'floor_sold', 'lat', 'mortgage_rate', 'elevator', '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm', '西屯北屯區_price_norm', '住宅大樓', '公寓', '華廈', 'Q2', 'Q3', 'Q4']

- Model_2_2
  - n_features_to_select = 14



  - features: ['M1b', 'age', 'bedroom_num', 'build_share1', 'parking_num_norm', 'story','total_deal_floor', 'elevator', '西區_unitprice', '西屯南屯區_unitprice', '住宅大樓','公寓', '套房', '華廈']

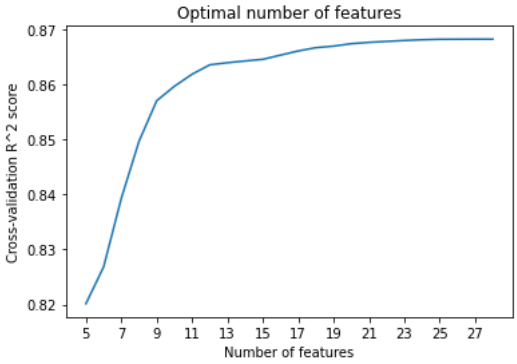
Backward Selection

* store it in v4

* (from sklearn.feature_selection import SequentialFeatureSelector as SFS)

- Model_1_1

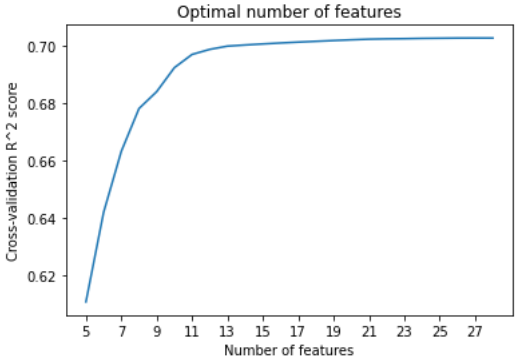
- n_features_to_select = 12



- Dropped: ['CPI%', 'GDP%', 'bathroom_num', 'build_share1', 'floor_sold', 'lat', 'mortgage_rate', 'total_deal_floor', 'elevator', 'manager', '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm', '西屯北屯區_price_norm', 'Q2', 'Q3', 'Q4']

○ Model_1_2

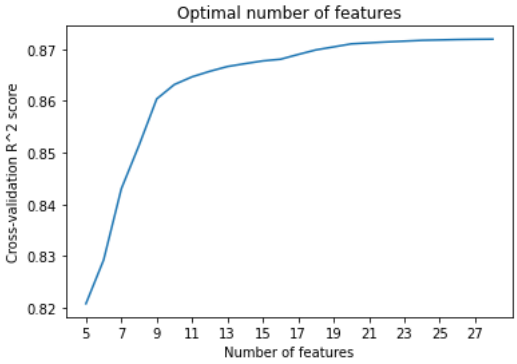
- n_features_to_select = 11



- Dropped: ['CPI%', 'GDP%', 'area', 'bathroom_num', 'bedroom_num', 'build_share1', 'floor_sold', 'hall_num', 'lat', 'lon', 'mortgage_rate', 'total_deal_floor', 'elevator', '北屯南區_unitprice', '東區_unitprice', 'Q2', 'Q3', 'Q4']

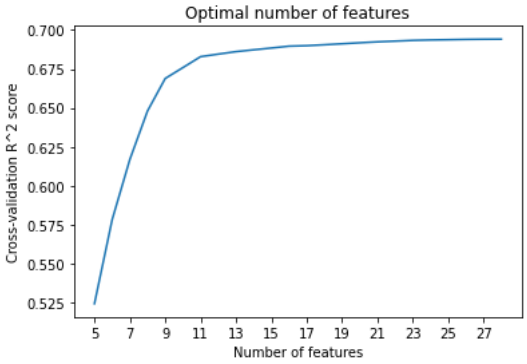
○ Model_2_1

- n_features_to_select = 9



- Dropped: ['CPI%', 'GDP%', 'bathroom_num', 'bedroom_num', 'build_share1', 'floor_sold', 'hall_num', 'lat', 'mortgage_rate', 'story', 'total_deal_floor', 'elevator', 'manager', '北區_price_norm', '南屯區_price_norm', '東西南區_price_norm', '西屯北屯區_price_norm', 'Q2', 'Q3', 'Q4']

- Model_2_2
  - n_features_to_select = 11



  - Dropped: : ['CPI%', 'GDP%', 'area', 'bathroom_num', 'bedroom_num', 'build_share1', 'floor_sold', 'hall_num', 'lat', 'lon', 'mortgage_rate', 'total_deal_floor', 'manager', '北屯北南區_unitprice', '東區_unitprice', 'Q2', 'Q3', 'Q4']

Regression Coefficients

* store it in v5

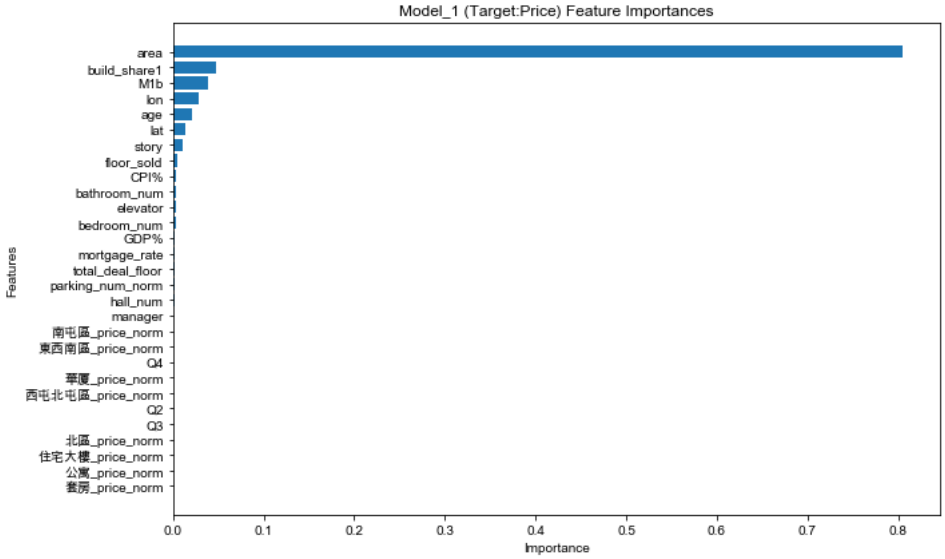
- Model_1_1
  - Features: ['M1b', 'age', 'area', 'bathroom_num', 'bedroom_num', 'hall_num', 'parking_num_norm', '套房_price_norm']
- Model_1_2
  - Features: ['M1b', 'age', 'parking_num_norm', 'story', '住宅大樓_unitprice', '公寓套房_unitprice', '華廈_unitprice']
- Model_2_1
  - Features: ['M1b', 'age', 'area', 'bedroom_num', 'hall_num', 'parking_num_norm', '住宅大樓', '套房', '華廈']
- Model_2_2
  - Features: ['M1b', 'age', 'bedroom_num', 'hall_num', 'parking_num_norm', 'story', '住宅大樓', '公寓', '套房', '華廈']

Feature importance

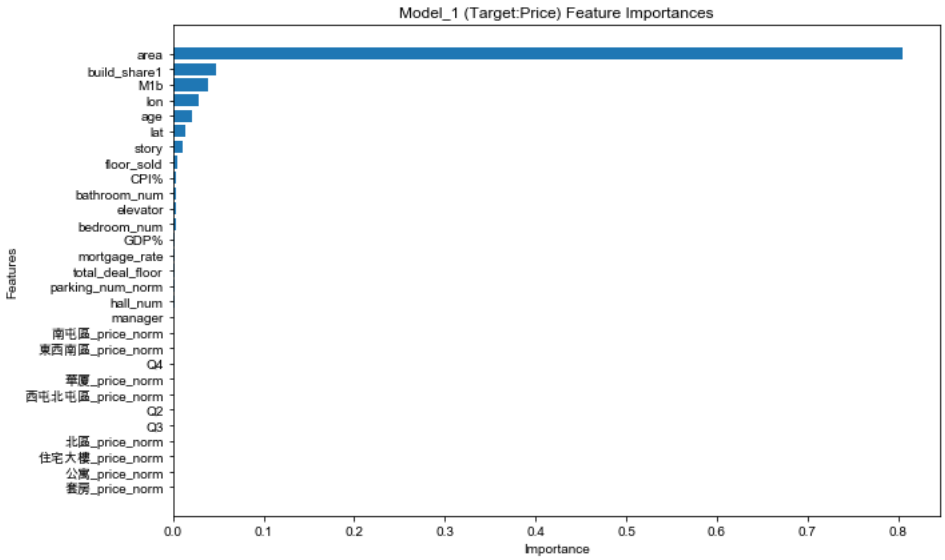
* store it in v6

* model = SelectFromModel(RandomForestRegressor(random_state = 42)

- Model_1_1
  - Features: ['area', 'build_share1', 'age', 'lon', 'M1b', 'lat', 'story', 'floor_sold', 'CPI%', 'bathroom_num']

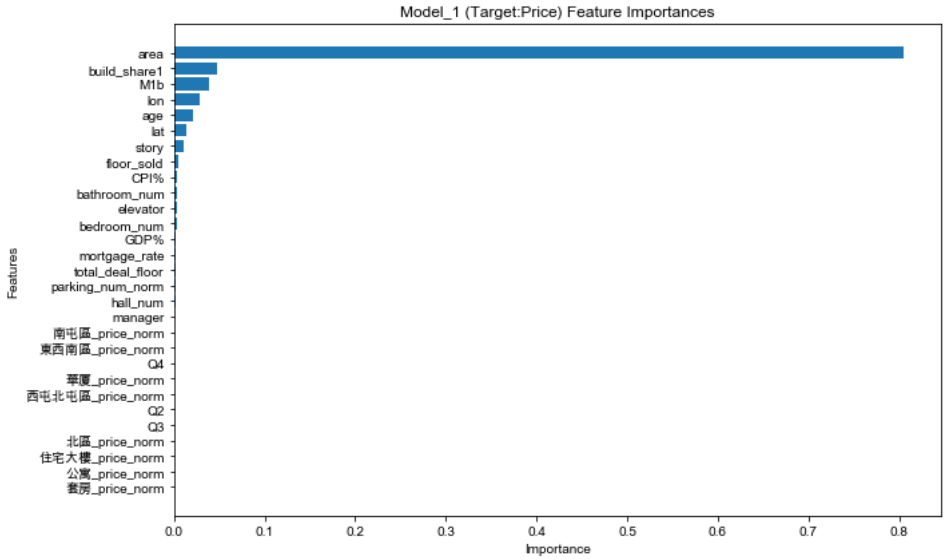


- Model_1_2
  - Features: ['build_share1', 'age', 'lon', 'M1b', 'elevator', 'story', 'lat', 'area','floor_sold', '北屯南區_unitprice']

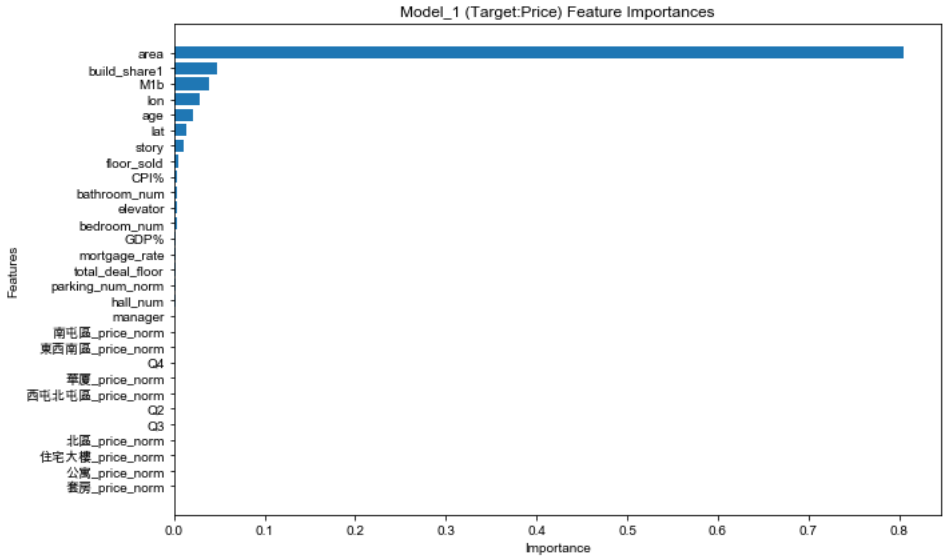


- Model_2_1

- Features: ['area', 'build_share1', 'age', 'lon', 'M1b', 'lat', 'story', 'floor_sold', 'CPI%', 'bathroom_num']



- Model_2_2
  - Features: ['build_share1', 'age', 'lon', 'M1b', 'elevator', 'story', 'lat', 'area','floor_sold', '西屯南屯區_unitprice']



VIF test

- V1_1_1
  - dropped: ['住宅大樓_price_norm', 'total_deal_floor', '西屯北屯區_price_norm', 'bedroom_num']

	feature	VIF
0	GDP%	6.07438
1	M1b	11.573878
2	age	5.808849
3	area	14.122726
4	bathroom_num	13.654846
5	bedroom_num	24.627436
6	build_share1	58.181108
7	floor_sold	5.693189
8	hall_num	11.195968
9	lat	28.309014
10	lon	9.138911
11	mortgage_rate	11.589896
12	parking_num_norm	7.484091
13	story	15.538349
14	total_deal_floor	51.464491
15	elevator	3.221676
16	manager	14.617293
17	北區_price_norm	7.521095
18	南屯區_price_norm	7.456627
19	東西南區_price_norm	10.430883
20	西屯北屯區_price_norm	32.041842
21	住宅大樓_price_norm	114.812764
22	公寓_price_norm	13.87563
23	套房_price_norm	13.218226
24	華廈_price_norm	23.220813

→

	feature	VIF
0	GDP%	5.921956
1	M1b	10.789551
2	age	4.99423
3	area	12.817344
4	bathroom_num	5.094588
5	build_share1	18.443175
6	floor_sold	5.644925
7	hall_num	10.55683
8	lat	17.809916
9	lon	8.674423
10	mortgage_rate	10.709478
11	parking_num_norm	5.714458
12	story	14.160835
13	elevator	3.188642
14	manager	13.221744
15	北區_price_norm	1.487146
16	南屯區_price_norm	2.333086
17	東西南區_price_norm	3.728789
18	公寓_price_norm	2.762315
19	套房_price_norm	1.505674
20	華廈_price_norm	1.821978

- V2_1_1
  - dropped: ['住宅大樓_price_norm', 'total_deal_floor', 'bedroom_num']

	feature	VIF
0	M1b	2.841056
1	age	4.835561
2	area	13.32393
3	bathroom_num	13.423704
4	bedroom_num	24.346189
5	build_share1	54.505115
6	floor_sold	5.674624
7	hall_num	9.595506
8	parking_num_norm	7.127052
9	story	15.210455
10	total_deal_floor	51.254799
11	北區_price_norm	1.519512
12	南屯區_price_norm	1.657126
13	西屯北屯區_price_norm	2.955962
14	住宅大樓_price_norm	87.12067
15	公寓_price_norm	11.791929
16	套房_price_norm	9.544188
17	華廈_price_norm	17.920129

→

	feature	VIF
0	M1b	2.705086
1	age	4.354276
2	area	12.486814
3	bathroom_num	4.822628
4	build_share1	13.599692
5	floor_sold	5.644013
6	hall_num	8.931403
7	parking_num_norm	5.558887
8	story	13.307229
9	北區_price_norm	1.498889
10	南屯區_price_norm	1.622504
11	西屯北屯區_price_norm	2.82808
12	公寓_price_norm	2.175903
13	套房_price_norm	1.381142
14	華廈_price_norm	1.810613

- V3_1_1
  - dropped: []

	feature	VIF
0	M1b	2.763559
1	age	4.457454
2	area	12.402969
3	bedroom_num	8.933759
4	build_share1	9.746669
5	hall_num	9.732688
6	lon	6.264917
7	parking_num_norm	5.424639
8	story	7.820625
9	manager	9.191706
10	套房_price_norm	1.357801

- V4_1_1
  - dropped: [住宅大樓_price_norm]

	feature	VIF
0	M1b	2.741962
1	age	4.016146
2	area	12.767931
3	bedroom_num	8.930926
4	hall_num	9.733486
5	lon	6.447661
6	parking_num_norm	6.046684
7	story	11.990006
8	住宅大樓_price_norm	15.031083
9	公寓_price_norm	1.834125
10	套房_price_norm	2.209848
11	華廈_price_norm	2.532186

→

	feature	VIF
0	M1b	2.722056
1	age	3.885707
2	area	11.958776
3	bedroom_num	8.925847
4	hall_num	9.421956
5	lon	5.849983
6	parking_num_norm	5.183508
7	story	5.682768
8	公寓_price_norm	1.469641
9	套房_price_norm	1.350194
10	華廈_price_norm	1.392175

- V5_1_1
  - dropped: [' bedroom_num ']

	feature	VIF
0	M1b	2.570217
1	age	3.147826
2	area	12.038573
3	bathroom_num	13.005783
4	bedroom_num	23.51971
5	hall_num	8.52852
6	parking_num_norm	4.72216
7	套房_price_norm	1.15783

→

	feature	VIF
0	M1b	2.547472
1	age	2.791024
2	area	11.608475
3	bathroom_num	4.749831
4	hall_num	7.801567
5	parking_num_norm	4.593767
6	套房_price_norm	1.113141

- V6_1_1
  - dropped: []

	feature	VIF
0	area	7.111007
1	build_share1	8.983205
2	age	3.812632
3	lon	6.38689
4	M1b	2.902664
5	lat	6.124345
6	story	10.129352
7	floor_sold	5.600771
8	CPI%	5.240819
9	bathroom_num	4.693434

- V1_1_2
  - dropped: []

	feature	VIF
0	GDP%	5.921956
1	M1b	10.789551
2	age	4.99423
3	area	12.817344
4	bathroom_num	5.094588
5	build_share1	18.443175
6	floor_sold	5.644925
7	hall_num	10.55683
8	lat	17.809916
9	lon	8.674423
10	mortgage_rate	10.709478
11	parking_num_norm	5.714458
12	story	14.160835
13	elevator	3.188642
14	manager	13.221744
15	北區_price_norm	1.487146
16	南屯區_price_norm	2.333086
17	東西南區_price_norm	3.728789
18	公寓_price_norm	2.762315
19	套房_price_norm	1.505674
20	華廈_price_norm	1.821978

- V2_1_2
  - dropped: ['story']

	feature	VIF
0	M1b	2.705086
1	age	4.354276
2	area	12.486814
3	bathroom_num	4.822628
4	build_share1	13.599692
5	floor_sold	5.644013
6	hall_num	8.931403
7	parking_num_norm	5.558887
8	story	13.307229
9	北區_price_norm	1.498889
10	南屯區_price_norm	1.622504
11	西屯北屯區_price_norm	2.82808
12	公寓_price_norm	2.175903
13	套房_price_norm	1.381142
14	華廈_price_norm	1.810613

- V3_1_2
  - dropped: []

	feature	VIF
0	M1b	2.763559
1	age	4.457454
2	area	12.402969
3	bedroom_num	8.933759
4	build_share1	9.746669
5	hall_num	9.732688
6	lon	6.264917
7	parking_num_norm	5.424639
8	story	7.820625
9	manager	9.191706
10	套房_price_norm	1.357801

- V4_1_2
  - dropped: []

	feature	VIF
0	M1b	2.722056
1	age	3.885707
2	area	11.958776
3	bedroom_num	8.925847
4	hall_num	9.421956
5	lon	5.849983
6	parking_num_norm	5.183508
7	story	5.682768
8	公寓_price_norm	1.469641
9	套房_price_norm	1.350194
10	華廈_price_norm	1.392175

- V5_1_2
  - dropped: []

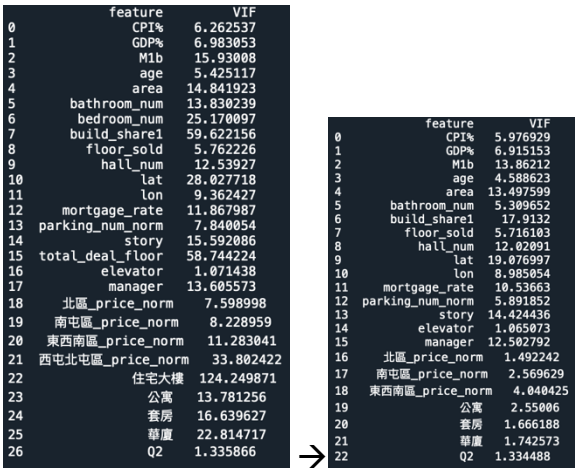
	feature	VIF
0	M1b	2.547472
1	age	2.791024
2	area	11.608475
3	bathroom_num	4.749831
4	hall_num	7.801567
5	parking_num_norm	4.593767
6	套房_price_norm	1.113141

- V6_1_2
  - dropped: []

	feature	VIF
0	area	7.111007
1	build_share1	8.983205
2	age	3.812632
3	lon	6.38689
4	M1b	2.902664
5	lat	6.124345
6	story	10.129352
7	floor_sold	5.600771
8	CPI%	5.240819
9	bathroom_num	4.693434

- V1_2_1

- dropped: ['住宅大樓', 'total_deal_floor', '西屯北屯區_price_norm', 'bedroom_num']



	feature	VIF
0	CPI%	6.262537
1	GDP%	6.983053
2	M1b	15.93008
3	age	5.425117
4	area	14.841923
5	bathroom_num	13.830239
6	bedroom_num	25.170097
7	build_share1	59.622156
8	floor_sold	5.762226
9	hall_num	12.53927
10	lat	28.027718
11	lon	9.362427
12	mortgage_rate	11.867987
13	parking_num_norm	7.840054
14	story	15.592086
15	total_deal_floor	58.744224
16	elevator	1.071438
17	manager	13.605573
18	北區_price_norm	7.598998
19	南屯區_price_norm	8.228959
20	東西南區_price_norm	11.283041
21	西屯北屯區_price_norm	33.802422
22	住宅大樓	124.249871
23	公寓	13.781256
24	套房	16.639627
25	華廈	22.814717
26	Q2	1.335866

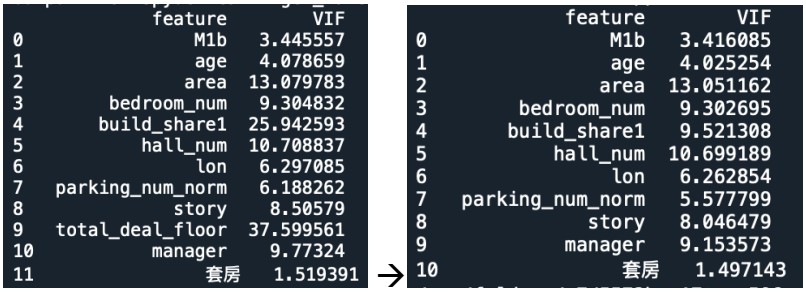
	feature	VIF
0	CPI%	5.976929
1	GDP%	6.915153
2	M1b	13.86212
3	age	4.588623
4	area	13.497599
5	bathroom_num	5.309652
6	build_share1	17.9132
7	floor_sold	5.716103
8	hall_num	12.02091
9	lat	19.076997
10	lon	8.985054
11	mortgage_rate	10.53663
12	parking_num_norm	5.891852
13	story	14.424436
14	elevator	1.065073
15	manager	12.502792
16	北區_price_norm	1.492242
17	南屯區_price_norm	2.569629
18	東西南區_price_norm	4.040425
19	公寓	2.55006
20	套房	1.666188
21	華廈	1.742573
22	Q2	1.334488

- V2_2_1
  - dropped: ['住宅大樓', 'total_deal_floor', 'bedroom_num']

	feature	VIF
0	M1b	3.653431
1	age	4.883882
2	area	14.528778
3	bathroom_num	13.688309
4	bedroom_num	25.097555
5	build_share1	58.944842
6	floor_sold	5.760221
7	hall_num	12.178864
8	lon	8.825787
9	parking_num_norm	7.72381
10	story	15.497713
11	total_deal_floor	58.618311
12	elevator	1.069327
13	北區_price_norm	5.336908
14	南屯區_price_norm	5.991261
15	東西南區_price_norm	8.774589
16	西屯北屯區_price_norm	18.339264
17	住宅大樓	113.47192
18	公寓	13.381598
19	套房	14.88696
20	華廈	20.911574

	feature	VIF
0	M1b	3.558945
1	age	4.128803
2	area	13.336775
3	bathroom_num	5.290379
4	build_share1	15.430929
5	floor_sold	5.718707
6	hall_num	11.94598
7	lon	8.751602
8	parking_num_norm	5.737616
9	story	14.013822
10	elevator	1.065131
11	北區_price_norm	4.604507
12	南屯區_price_norm	4.852554
13	東西南區_price_norm	7.311696
14	西屯北屯區_price_norm	14.978675
15	公寓	2.143575
16	套房	1.599377
17	華廈	1.738792

- V3_2_1
  - dropped: ['total_deal_floor']



	feature	VIF
0	M1b	3.445557
1	age	4.078659
2	area	13.079783
3	bedroom_num	9.304832
4	build_share1	25.942593
5	hall_num	10.708837
6	lon	6.297085
7	parking_num_norm	6.188262
8	story	8.50579
9	total_deal_floor	37.599561
10	manager	9.77324
11	套房	1.519391

	feature	VIF
0	M1b	3.416085
1	age	4.025254
2	area	13.051162
3	bedroom_num	9.302695
4	build_share1	9.521308
5	hall_num	10.699189
6	lon	6.262854
7	parking_num_norm	5.577799
8	story	8.046479
9	manager	9.153573
10	套房	1.497143

- V4_2_1
  - dropped: []

	feature	VIF
0	M1b	3.398971
1	age	3.268833
2	area	6.357964
3	lon	5.687675
4	parking_num_norm	6.138024
5	住宅大樓	6.811437
6	公寓	1.652222
7	套房	1.578436
8	華廈	1.986159

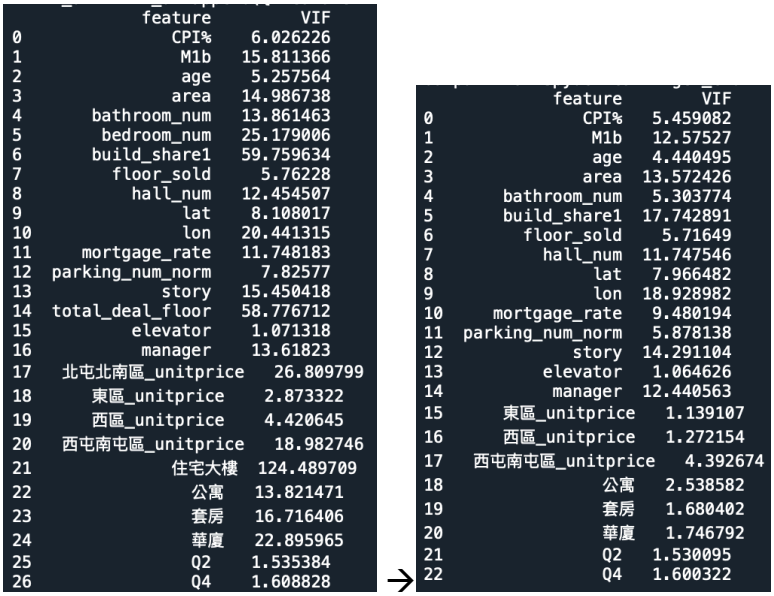
- V5_2_1
  - dropped: []

	feature	VIF
0	M1b	3.380486
1	age	3.066764
2	area	12.40963
3	bedroom_num	8.912653
4	hall_num	9.90941
5	parking_num_norm	6.072054
6	住宅大樓	5.745798
7	套房	1.294854
8	華廈	1.635169

- V6_2_1
  - dropped: ['住宅大樓_price_norm', 'total_deal_floor', '西屯北屯區_price_norm', 'bedroom_num']

	feature	VIF
0	area	7.474267
1	build_share1	8.814222
2	age	3.426515
3	lon	6.410615
4	M1b	3.362696
5	lat	6.027566
6	story	10.500132
7	floor_sold	5.665719
8	CPI%	4.833168
9	bathroom_num	4.854094

- V1_2_2
  - dropped: ['住宅大樓', 'total_deal_floor', '北屯北南區_unitprice', 'bedroom_num']



	feature	VIF
0	CPI%	6.026226
1	M1b	15.811366
2	age	5.257564
3	area	14.986738
4	bathroom_num	13.861463
5	bedroom_num	25.179006
6	build_share1	59.759634
7	floor_sold	5.76228
8	hall_num	12.454507
9	lat	8.108017
10	lon	20.441315
11	mortgage_rate	11.748183
12	parking_num_norm	7.82577
13	story	15.450418
14	total_deal_floor	58.776712
15	elevator	1.071318
16	manager	13.61823
17	北屯北南區_unitprice	26.809799
18	東區_unitprice	2.873322
19	西區_unitprice	4.420645
20	西屯南屯區_unitprice	18.982746
21	住宅大樓	124.489709
22	公寓	13.821471
23	套房	16.716406
24	華廈	22.895965
25	Q2	1.535384
26	Q4	1.608828

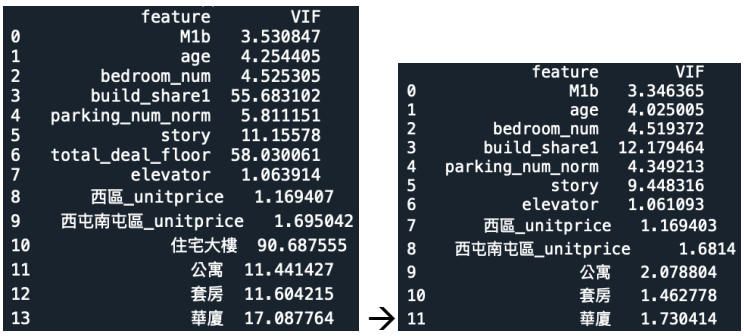
	feature	VIF
0	CPI%	5.459082
1	M1b	12.57527
2	age	4.440495
3	area	13.572426
4	bathroom_num	5.303774
5	build_share1	17.742891
6	floor_sold	5.71649
7	hall_num	11.747546
8	lat	7.966482
9	lon	18.928982
10	mortgage_rate	9.480194
11	parking_num_norm	5.878138
12	story	14.291104
13	elevator	1.064626
14	manager	12.440563
15	東區_unitprice	1.139107
16	西區_unitprice	1.272154
17	西屯南屯區_unitprice	4.392674
18	公寓	2.538582
19	套房	1.680402
20	華廈	1.746792
21	Q2	1.530095
22	Q4	1.600322

- V2_2_2
  - dropped:

	feature	VIF
0	CPI%	5.50762
1	M1b	13.273574
2	age	5.075024
3	area	13.817375
4	bathroom_num	13.592348
5	bedroom_num	24.537385
6	build_share1	58.58427
7	floor_sold	5.75369
8	lon	17.456805
9	mortgage_rate	9.697731
10	parking_num_norm	7.770621
11	story	15.394525
12	total_deal_floor	58.733586
13	elevator	1.068901
14	manager	13.531889
15	北屯北南區_unitprice	7.560068
16	東區_unitprice	1.527114
17	西屯南屯區_unitprice	5.376958
18	住宅大樓	120.175696
19	公寓	13.294998
20	套房	15.790911
21	華廈	22.056806

	feature	VIF
0	CPI%	5.365475
1	M1b	12.12414
2	age	4.343694
3	area	12.13989
4	bathroom_num	5.322017
5	build_share1	17.664802
6	floor_sold	5.716711
7	lon	17.25794
8	mortgage_rate	9.076963
9	parking_num_norm	5.873544
10	story	14.19726
11	elevator	1.06485
12	manager	12.435057
13	北屯北南區_unitprice	7.483527
14	東區_unitprice	1.519278
15	西屯南屯區_unitprice	5.226002
16	公寓	2.527636
17	套房	1.495017
18	華廈	1.733947

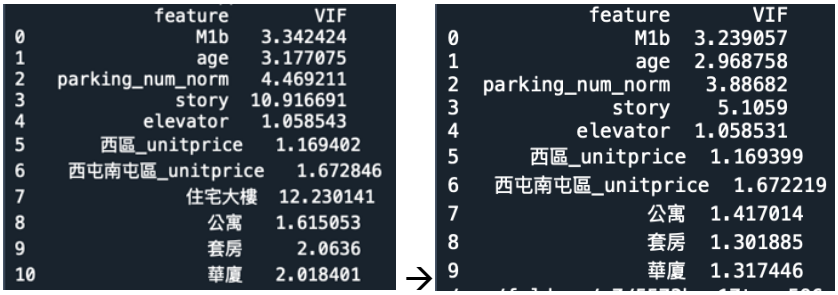
- V3_2_2
  - dropped: ['住宅大樓', 'total_deal_floor']



	feature	VIF
0	M1b	3.530847
1	age	4.254405
2	bedroom_num	4.525305
3	build_share1	55.683102
4	parking_num_norm	5.811151
5	story	11.15578
6	total_deal_floor	58.030061
7	elevator	1.063914
8	西區_unitprice	1.169407
9	西屯南屯區_unitprice	1.695042
10	住宅大樓	90.687555
11	公寓	11.441427
12	套房	11.604215
13	華廈	17.087764

	feature	VIF
0	M1b	3.346365
1	age	4.025005
2	bedroom_num	4.519372
3	build_share1	12.179464
4	parking_num_norm	4.349213
5	story	9.448316
6	elevator	1.061093
7	西區_unitprice	1.169403
8	西屯南屯區_unitprice	1.6814
9	公寓	2.078804
10	套房	1.462778
11	華廈	1.730414

- V4_2_2
  - dropped: [‘住宅大樓’]

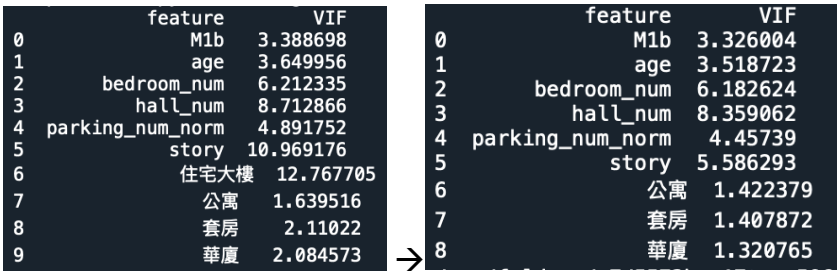


	feature	VIF
0	M1b	3.342424
1	age	3.177075
2	parking_num_norm	4.469211
3	story	10.916691
4	elevator	1.058543
5	西區_unitprice	1.169402
6	西屯南屯區_unitprice	1.672846
7	住宅大樓	12.230141
8	公寓	1.615053
9	套房	2.0636
10	華廈	2.018401

→

	feature	VIF
0	M1b	3.239057
1	age	2.968758
2	parking_num_norm	3.88682
3	story	5.1059
4	elevator	1.058531
5	西區_unitprice	1.169399
6	西屯南屯區_unitprice	1.672219
7	公寓	1.417014
8	套房	1.301885
9	華廈	1.317446

- V5_2_2
  - dropped: [‘住宅大樓’]



	feature	VIF
0	M1b	3.388698
1	age	3.649956
2	bedroom_num	6.212335
3	hall_num	8.712866
4	parking_num_norm	4.891752
5	story	10.969176
6	住宅大樓	12.767705
7	公寓	1.639516
8	套房	2.11022
9	華廈	2.084573

→

	feature	VIF
0	M1b	3.326004
1	age	3.518723
2	bedroom_num	6.182624
3	hall_num	8.359062
4	parking_num_norm	4.45739
5	story	5.586293
6	公寓	1.422379
7	套房	1.407872
8	華廈	1.320765

- V6_2_2
  - dropped: []

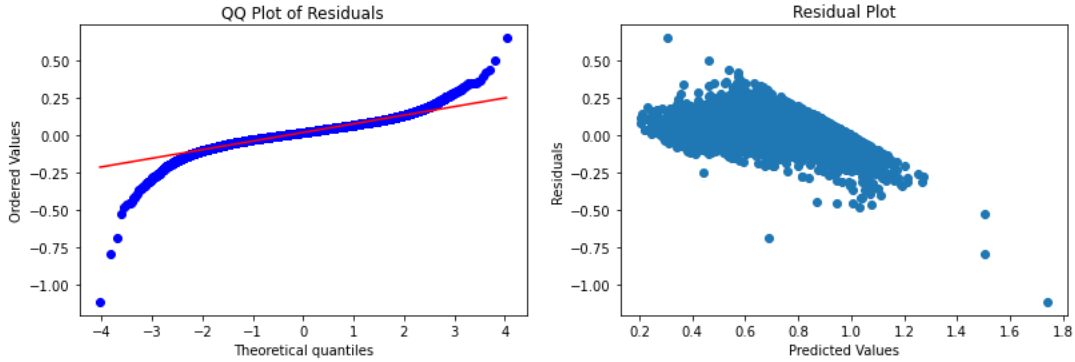
	feature	VIF
0	build_share1	8.928535
1	age	3.261733
2	lon	11.803139
3	M1b	3.478222
4	elevator	1.056504
5	story	10.474927
6	lat	7.528217
7	area	5.60957
8	floor_sold	5.659021
9	西屯南屯區_unitprice	3.199176

Normality test(Residual plot & Shapiro-West test)

*** The assumption of normality of residuals and the random scatter of residuals around zero, which could **indicate the presence of non-linearity or heteroscedasticity**.

* All models are non-linear. All residual plots tend to have some patterns and do not scatter randomly around 0.

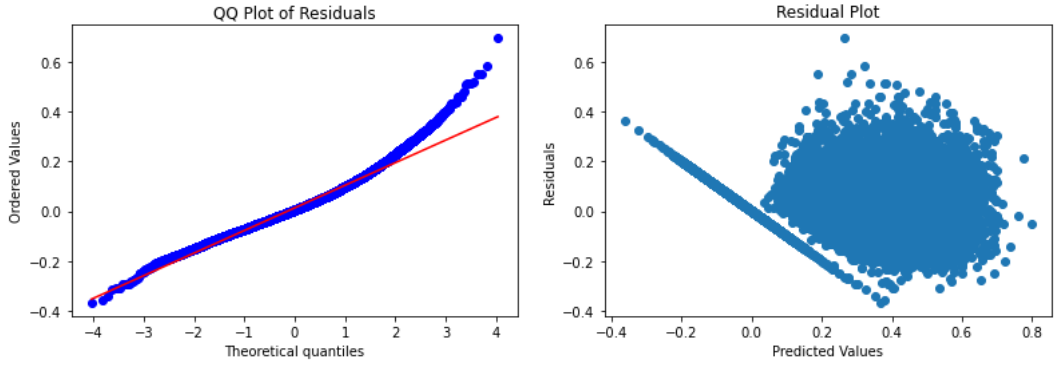
- Linear Regression (Original)
  - Model_1: total_price




```
adj_r2: 0.8067
MAE: 0.0454
MSE: 0.0039
RMSE: 0.0623
```

```
Shapiro-Wilk test:
Test statistic: 0.9363, p-value: 0.0000
```

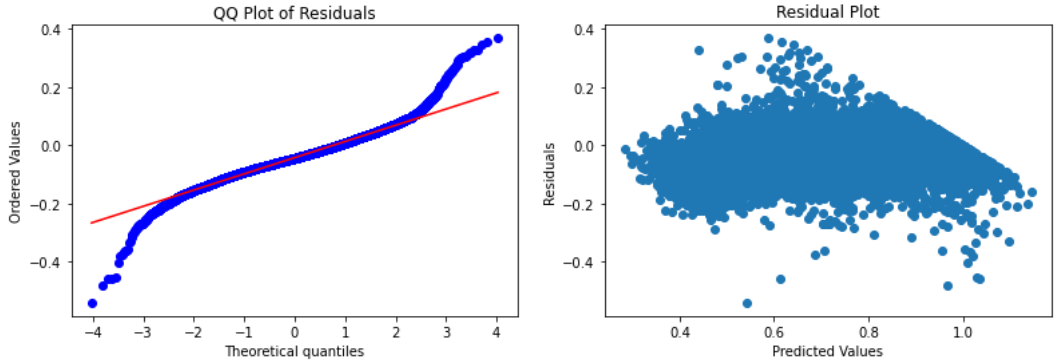
- Model_1: unit_price




```
adj_r2: 0.6859
MAE: 0.07
MSE: 0.0086
RMSE: 0.0927
```

```
Shapiro-Wilk test:
Test statistic: 0.9766, p-value: 0.0000
```

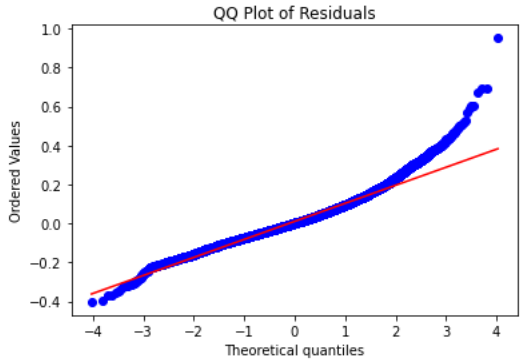
- Model_2: total_price

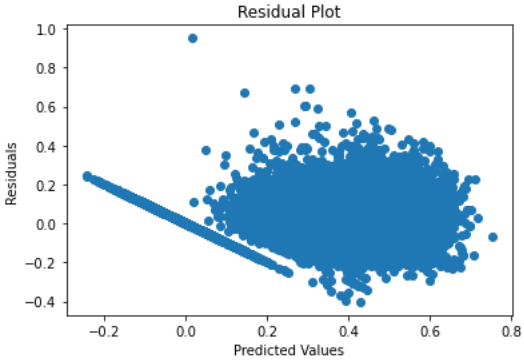



```
adj_r2: 0.7919
MAE: 0.0568
MSE: 0.005
RMSE: 0.0708
```

```
Shapiro-Wilk test:
Test statistic: 0.9730, p-value: 0.0000
```

- Model_2: unit_price






```
adj_r2: 0.68
MAE: 0.0706
MSE: 0.0089
RMSE: 0.0946
```

```
Shapiro-Wilk test:
Test statistic: 0.9682, p-value: 0.0000
```

Modeling and evaluation

* Since data is non-linear, it may be worth exploring other regression models that can handle non-linearity, such as **decision tree-based** models or **non-linear regression models.** 1st 2nd

Models	Feature selection Method	Linear Regression		Polynomial Regression		Support Vector Regression	
		R squared	MSE	R squared	MSE	R squared	MSE
1_1	N / A (n = 29)	0.8067	0.0039	0.7391	0.0052	0.8408	0.0032
	SelectFpr (n = 21)	0.802	0.004	0.8161	0.0037	0.8331	0.0033
	SelectKBest (n = 15)	0.7995	0.004	0.7877	0.0043	0.8138	0.0037
	Forward (n = 11)	0.808	0.0039	0.7571	0.0049	0.8174	0.0037
	Backward (n = 11)	0.7791	0.0044	0.7652	0.0047	0.8324	0.0034
	Regression coefficients (n = 7)	0.7598	0.0048	0.78	0.0044	0.8214	0.0036
	Feature importance (n = 10)	0.7374	0.0053	0.7773	0.0045	0.7952	0.0041
		Deecision Tree		Random Forest		XGBoost	
		R squared	MSE	R squared	MSE	R squared	MSE
	N / A (n = 29)	0.7455	0.0051	0.7624	0.0048	0.5431	0.0092
	SelectFpr (n = 21)	0.7456	0.0051	0.7625	0.0048	0.5514	0.009
	SelectKBest (n = 15)	0.7461	0.0051	0.7628	0.0048	0.573	0.0086
	Forward (n = 11)	0.7465	0.0051	0.7631	0.0048	0.5149	0.0097
	Backward (n = 11)	0.7293	0.0054	0.7464	0.0051	0.4771	0.0105
	Regression coefficients (n = 7)	0.7197	0.0056	0.7339	0.0053	0.5304	0.0094
	Feature importance (n = 10)	0.7455	0.0051	0.7615	0.0048	0.249	0.0151

Models	Feature selection Method	Linear Regression		Polynomial Regression		Support Vector Regression	
		R squared	MSE	R squared	MSE	R squared	MSE
2_1	N / A (n = 29)	0.7919	0.005	0.8769	0.003	0.8135	0.0045
	SelectFpr (n = 23)	0.7797	0.0053	0.8621	0.0033	0.8109	0.0046
	SelectKBest (n = 18)	0.7795	0.0053	0.8557	0.0035	0.8047	0.0047
	Forward (n = 11)	0.7775	0.0054	0.8498	0.0036	0.8123	0.0045
	Backward (n = 9)	0.7912	0.005	0.8433	0.0038	0.8156	0.0055
	Regression coefficients (n = 9)	0.7496	0.006	0.8047	0.0047	0.757	0.0059
	Feature importance (n = 10)	0.7767	0.0054	0.8585	0.0034	0.8403	0.0038
		Deecision Tree		Random Forest		XGBoost	
		R squared	MSE	R squared	MSE	R squared	MSE
	N / A (n = 29)	0.8058	0.0047	0.8113	0.0045	0.6268	0.009
	SelectFpr (n = 23)	0.8059	0.0047	0.8114	0.0045	0.594	0.0098
	SelectKBest (n = 18)	0.8059	0.0047	0.8114	0.0045	0.6216	0.0091
	Forward (n = 11)	0.8061	0.0047	0.8114	0.0045	0.6402	0.0087
	Backward (n = 9)	0.7902	0.0051	0.7951	0.0049	0.4864	0.0124
	Regression coefficients (n = 9)	0.7889	0.0051	0.7924	0.005	0.526	0.0114
	Feature importance (n = 10)	0.806	0.0047	0.8115	0.0045	0.4993	0.0121

Models	Feature selection Method	Linear Regression		Polynomial Regression		Support Vector Regression	
		R squared	MSE	R squared	MSE	R squared	MSE
1_2	N / A (n = 29)	0.6859	0.0086	0.6925	0.0084	0.7891	0.0058
	SelectFpr (n = 29)	-	-	-	-	-	-
	SelectKBest (n = 22)	0.6809	0.0087	0.6785	0.0088	0.7643	0.0065
	Forward (n = 19)	0.6846	0.0086	0.6993	0.0082	0.7504	0.0068
	Backward (n = 11)	0.681	0.0087	0.7228	0.0076	0.7264	0.0075
	Regression coefficients (n = 7)	0.6403	0.0099	0.6709	0.009	0.6792	0.0088
	Feature importance (n = 10)	0.576	0.0116	0.7429	0.007	0.7563	0.0067
		Decision Tree		Random Forest		XGBoost	
		R squared	MSE	R squared	MSE	R squared	MSE
	N / A (n = 29)	0.6886	0.0085	0.709	0.008	0.4906	0.0139
	SelectFpr (n = 29)	-	-	-	-	-	-
	SelectKBest (n = 22)	0.6888	0.0085	0.709	0.008	0.4554	0.0149
	Forward (n = 19)	0.6967	0.0083	0.7023	0.0082	0.4462	0.0152
	Backward (n = 11)	0.5956	0.0111	0.647	0.0097	0.1509	0.0233
	Regression coefficients (n = 7)	0.5972	0.011	0.6479	0.0096	0.2434	0.0207
	Feature importance (n = 10)	0.7023	0.0082	0.7088	0.008	0.4062	0.0163

Models	Feature selection Method	Linear Regression		Polynomial Regression		Support Vector Regression	
		R squared	MSE	R squared	MSE	R squared	MSE
2_2	N / A (n = 29)	0.68	0.0089	0.7759	0.0063	0.8079	0.0054
	SelectFpr (n = 23)	0.6537	0.0097	0.7919	0.0058	0.7971	0.0057
	SelectKBest (n = 19)	0.6469	0.0099	0.7842	0.006	0.7838	0.006
	Forward (n = 12)	0.6447	0.0099	0.7279	0.0076	0.7292	0.0076
	Backward (n = 10)	0.5864	0.0116	0.7001	0.0084	0.6995	0.0084
	Regression coefficients (n = 9)	0.5605	0.0123	0.6506	0.0098	0.6551	0.0096
	Feature importance (n = 10)	0.5866	0.0116	0.7657	0.0066	0.773	0.0064
		Decision Tree		Random Forest		XGBoost	
		R squared	MSE	R squared	MSE	R squared	MSE
	N / A (n = 29)	0.7238	0.0077	0.7355	0.0074	0.5024	0.0139
	SelectFpr (n = 23)	0.724	0.0077	0.7355	0.0074	0.4866	0.0144
	SelectKBest (n = 19)	0.7237	0.0077	0.7345	0.0074	0.4777	0.0146
	Forward (n = 12)	0.6955	0.0085	0.7015	0.0083	0.2852	0.02
	Backward (n = 10)	0.6565	0.0096	0.6664	0.0093	0.2439	0.0212
	Regression coefficients (n = 9)	0.6468	0.0099	0.6578	0.0096	0.2774	0.0202
	Feature importance (n = 10)	0.7241	0.0077	0.7351	0.0074	0.4406	0.0156

1. Model performance: model 2 better than model 1
2. Target: total_price better than target unit_price
3. Top 3 algo: Polynomial Regression, SVM, Random forest
4. High performance (in top 3 algo) feature selection method: SelectFpr, feature importance

→ Consider model complexity, choose **top 3 algo** with **the result of feature importance**(v6) to perform hyperparameter search

Hyperparameter Tuning

* Most of the models showed a significant improvement in performance after tuning the hyperparameters. Among them, the best performing model is Random Forest. I decide to use Random Forest as the final model for model_1_1, model_1_2, model_2_1, model_2_2.

Models		Polynomial Regression		Support Vector Regression		Random Forest	
		R squared	MSE	R squared	MSE	R squared	MSE
Total Price	1_1	0.7649 (0.7773)	0.0047 (0.0045)	0.8213 (0.7952)	0.0036 (0.0041)	0.8863 (0.7615)	0.0023 (0.0048)
	2_1	0.8061 (0.8585)	0.0053 (0.0034)	0.8479 (0.8403)	0.0038 (0.0038)	0.8636 (0.8115)	0.0037 (0.0045)
Unit Price	1_2	0.8745 (0.7429)	0.003 (0.007)	0.8157 (0.7563)	0.0059 (0.0067)	0.8899 (0.7088)	0.0027 (0.008)
	2_2	0.8216 (0.7657)	0.005 (0.0066)	0.8328 (0.773)	0.0056 (0.0064)	0.8998 (0.7351)	0.0028 (0.0074)

Testing

Models		Random Forest	
		R squared	MSE
Total Price	1_1	0.9426	0.0014
	2_1	0.8813	0.0032
Unit Price	1_2	0.8903	0.003
	2_2	0.5839	0.011

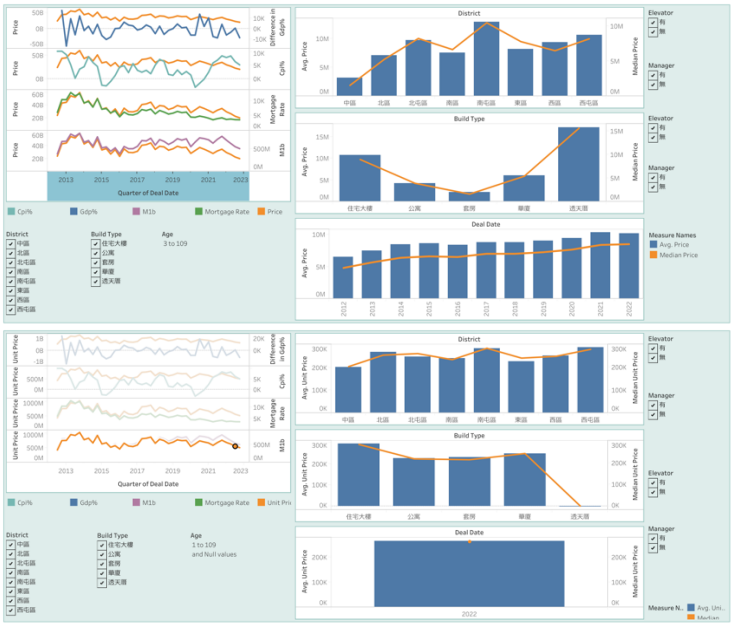
Conclusion

Based on the test dataset performance, it appears that the first model (trained on all available data from 2012 to 2022) consistently outperformed the second model (trained on data from 2012 to 2019, before the pandemic). The random forest model for predicting total price (Model 1_1) achieved a higher R-squared value of 0.9426 and a lower MSE of 0.0014 compared to Model 2_1, which achieved an R-squared value of 0.8813 and a higher MSE of 0.0032. For predicting unit price, Model 1_2 also outperformed Model 2_2, with a higher R-squared value of 0.8903 and a lower MSE of 0.003 compared to Model 2_2, which had a lower R-squared value of 0.5839 and a higher MSE of 0.011.

This suggests that using all available data to train the model can result in better predictions for housing prices. Alternatively, it is possible that the second model did not capture the changes in the housing market brought about by the pandemic, leading to poorer performance. Besides, model 1 performs better than model 2 may indicate that **the pandemic did not have a significant impact** on the data, or that the model is able to adapt to the changes in the data caused by the pandemic.

Data Dashboard

*** I've created interactive data dashboards for this dataset with tableau. Please check the following website for more information.
https://public.tableau.com/app/profile/axi3708/viz/shared/YWP77MPX2



Bibliography

黃惠芬, Development of Reality Pricing Models by Applying Aritficial Neural Network Based on Actual Price Registration Data in Kaohsiung City(以類神經網路方法建構房價估價模型-以高雄市實價登錄資料為例)

劉富容, 游璿達, 黃孝雲, 劉正夫, Exploring the Effect of Environmental Dactors and Residential Characteristics on Housing Prices in Taipei City Using Open(利用政府開放資料探討影響台北市房價之主要房屋特性及周邊設施), 2019/10

黃士峰, 廖育苹, 基於廣義加成模型之房屋價格預測方法, 2022

陳聖元, House Price Prediction Based on Deep Learning Combined with Economic Indicators(基於深度學習結合經濟指標之房價預測), 2022/06

李宗澤, The Impact of Consolidated Housing and Land Tax on the Local Real Market: A Case Study of Taipei City(房地合一稅實施前後對不動產價格與交易量之影響--以臺北市為例), 2018/06

梁仁旭, 詹進發, 陳奉瑤, 大數據於實價登錄資料的應用-備註欄的處理