台中房價預測

小組成員：劉順勇

報告日期：2020/05/14

資料探索式分析(EDA)

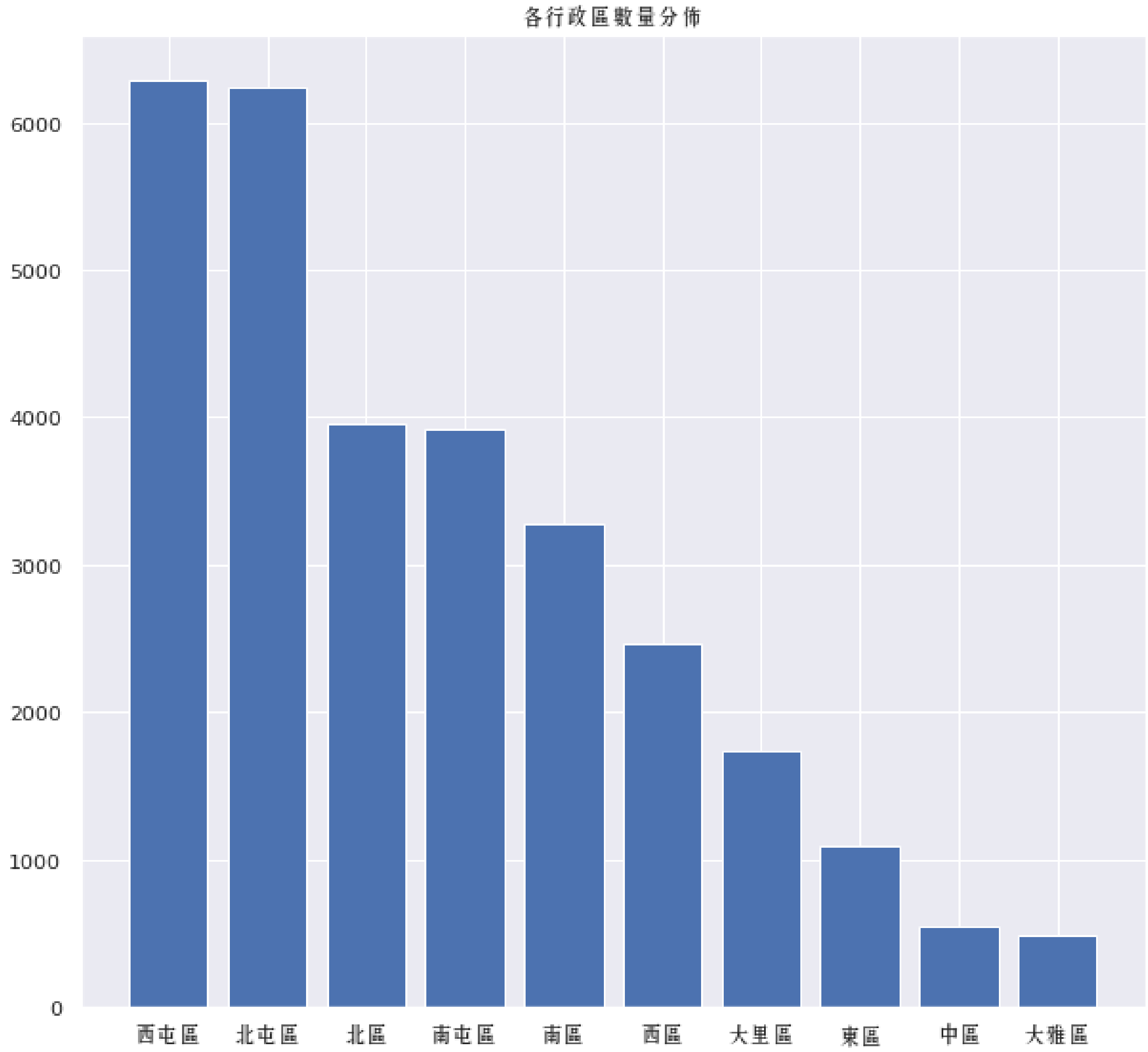
```
df = train.describe()
df.round(2)
```

	土地面積	建物總面積	屋齡	樓層	總樓層	房數	廳數	衛數	電梯	經度	緯度	總價
count	30000.00	30000.00	29999.00	30000.00	30000.00	30000.00	30000.00	30000.00	30000.00	30000.00	30000.00	30000.00
mean	18.73	136.85	16.73	7.46	13.23	2.74	1.68	1.80	0.91	120.67	24.16	841.19
std	13.09	90.88	11.78	4.35	5.79	1.40	0.70	1.19	0.29	0.03	0.03	760.97
min	0.10	0.02	0.00	1.00	1.00	0.00	0.00	0.00	0.00	120.58	23.98	0.00
25%	10.80	86.68	4.04	4.00	9.00	2.00	1.00	1.00	1.00	120.65	24.14	410.00
50%	17.60	127.92	20.47	7.00	13.00	3.00	2.00	2.00	1.00	120.67	24.16	670.00
75%	24.75	169.27	24.92	10.00	15.00	3.00	2.00	2.00	1.00	120.69	24.17	1045.25
max	875.00	6263.64	56.99	30.00	41.00	91.00	33.00	91.00	1.00	120.98	24.24	36000.00

資料探索式分析(EDA)

```python
''' calculate correlation between features to detect colinearity '''
corr = train.corr()
corr = corr.round(2)
```

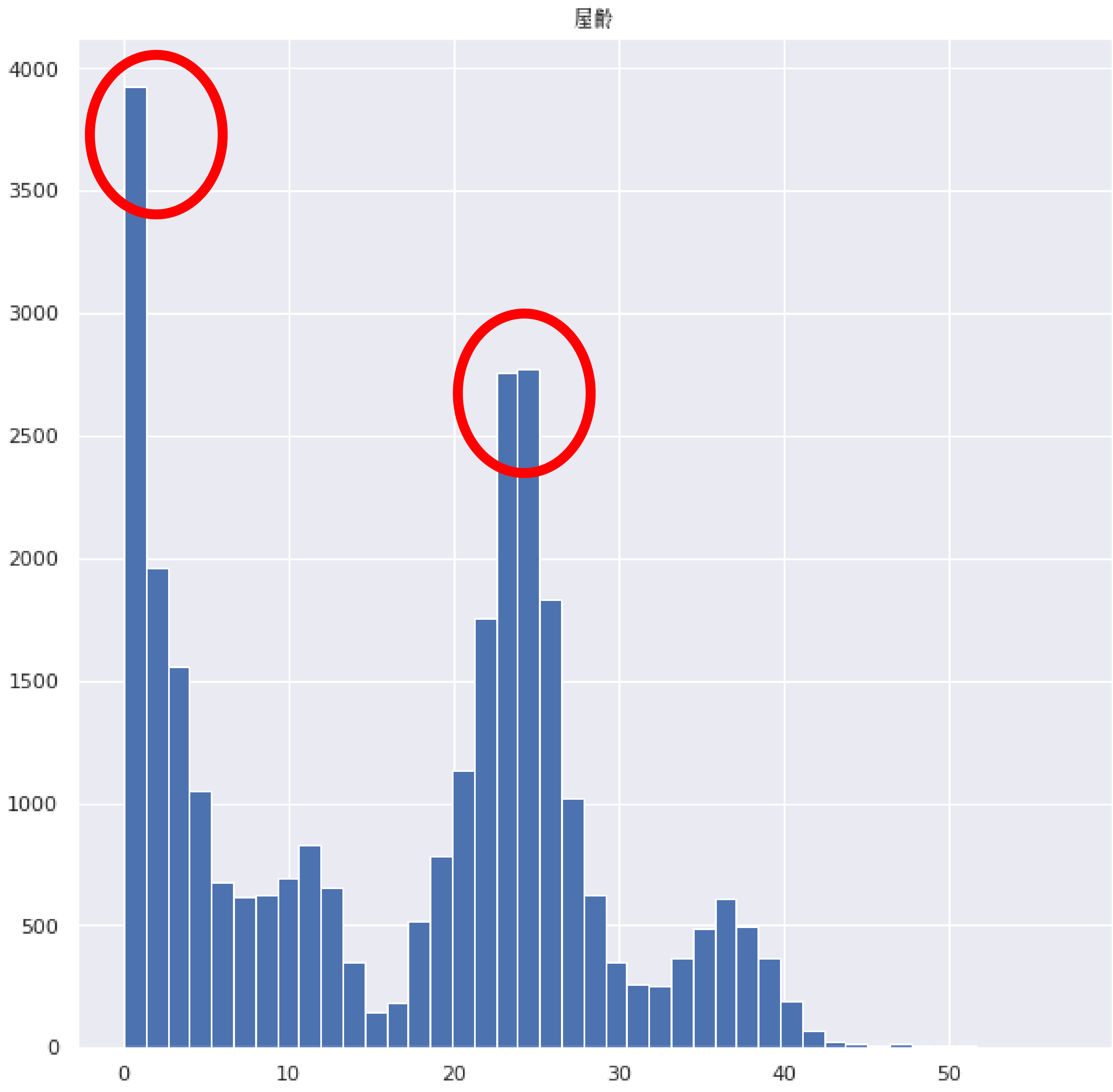
	土地面積	建物總面積	屋齡	樓層	總樓層	房數	廳數	衛數	電梯	經度	緯度	總價
土地面積	1.00	0.74	-0.17	-0.09	-0.11	0.68	0.37	0.60	-0.21	0.00	-0.00	0.59
建物總面積	0.74	1.00	-0.42	0.19	0.36	0.69	0.41	0.62	0.17	-0.08	0.01	0.88
屋齡	-0.17	-0.42	1.00	-0.31	-0.55	-0.08	-0.23	-0.05	-0.48	0.03	-0.00	-0.52
樓層	-0.09	0.19	-0.31	1.00	0.56	-0.00	0.05	0.00	0.29	-0.06	0.01	0.25
總樓層	-0.11	0.36	-0.55	0.56	1.00	0.03	0.11	0.03	0.46	-0.15	0.02	0.46
房數	0.68	0.69	-0.08	-0.00	0.03	1.00	0.41	0.86	-0.11	0.01	-0.03	0.50
廳數	0.37	0.41	-0.23	0.05	0.11	0.41	1.00	0.22	0.02	0.01	-0.02	0.36
衛數	0.60	0.62	-0.05	0.00	0.03	0.86	0.22	1.00	-0.07	-0.01	-0.01	0.46
電梯	-0.21	0.17	-0.48	0.29	0.46	-0.11	0.02	-0.07	1.00	-0.08	0.00	0.18
經度	0.00	-0.08	0.03	-0.06	-0.15	0.01	0.01	-0.01	-0.08	1.00	-0.10	-0.18
緯度	-0.00	0.01	-0.00	0.01	0.02	-0.03	-0.02	-0.01	0.00	-0.10	1.00	0.03
總價	0.59	0.88	-0.52	0.25	0.46	0.50	0.36	0.46	0.18	-0.18	0.03	1.00



各行政區數量分佈

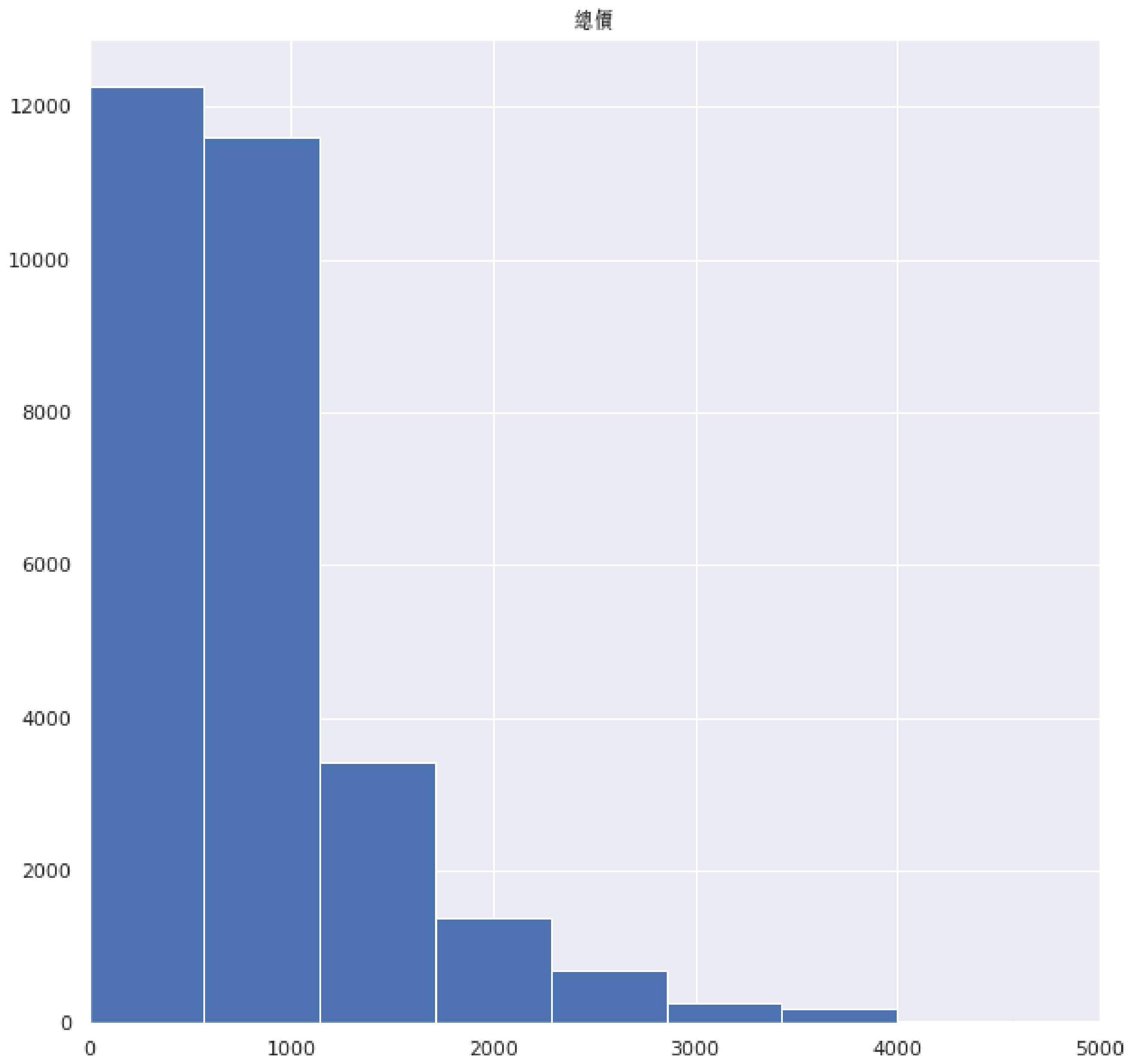
資料多集中在西屯區、北屯區、北區以及南屯區。

房子屋齡分佈



房子房齡介於 0 – 50 歲內，

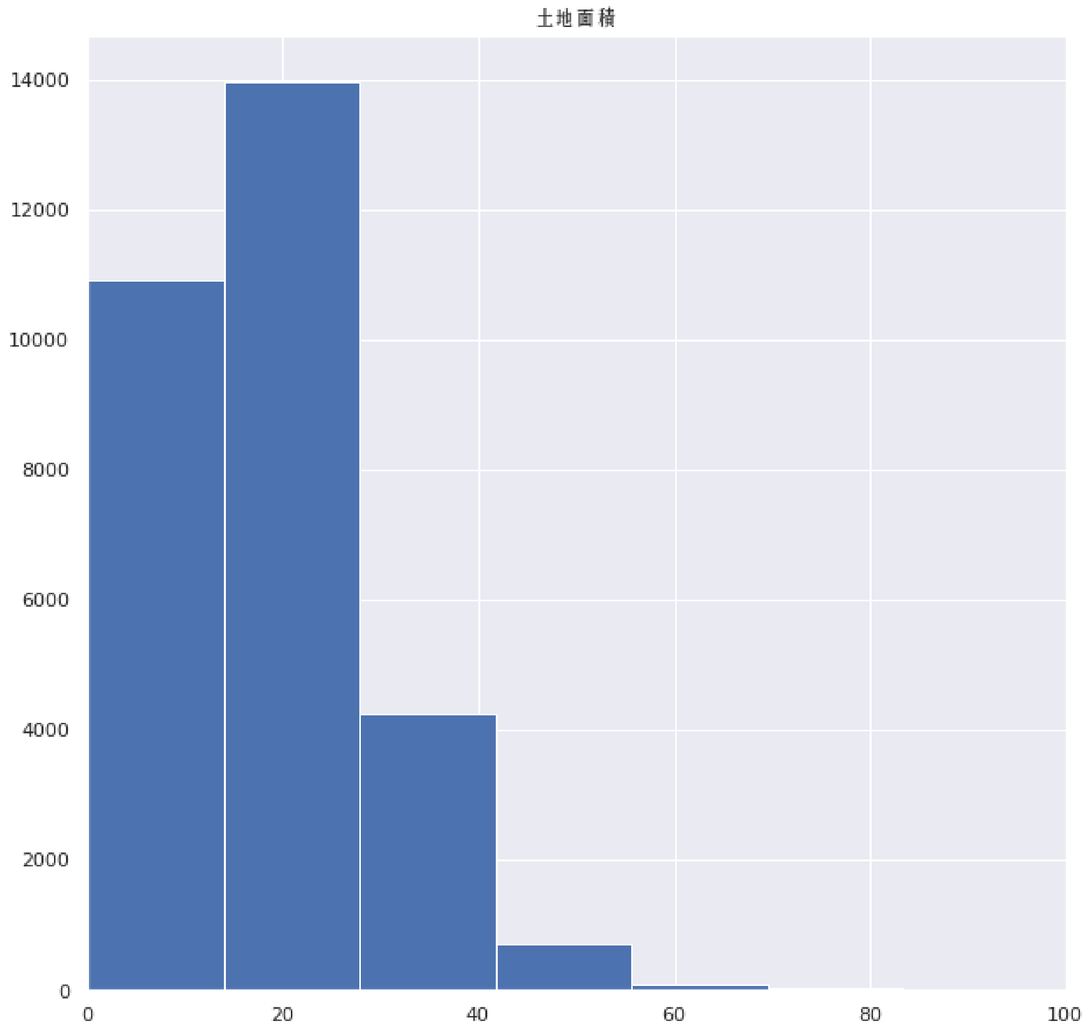
主要集中在0-5歲和 20-30歲。



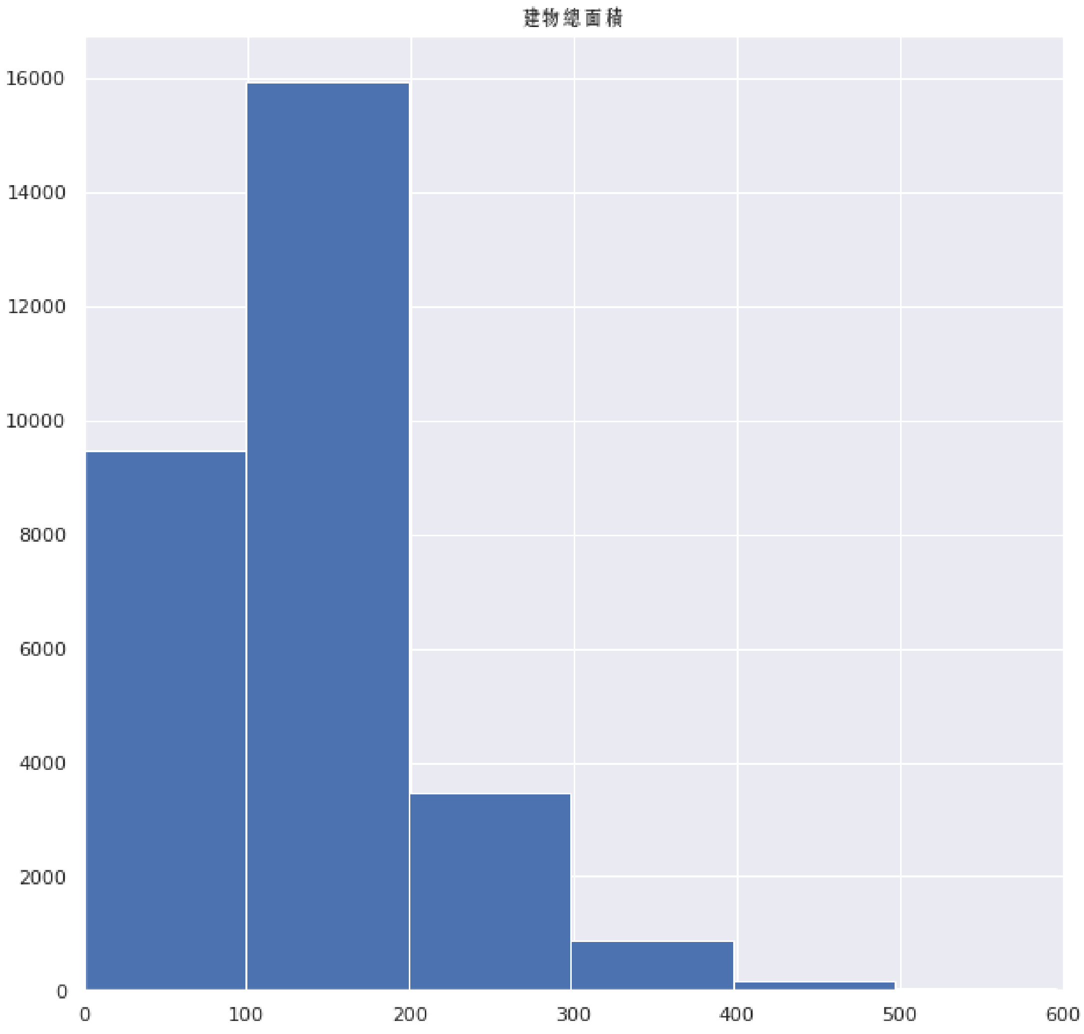
房子總價分佈

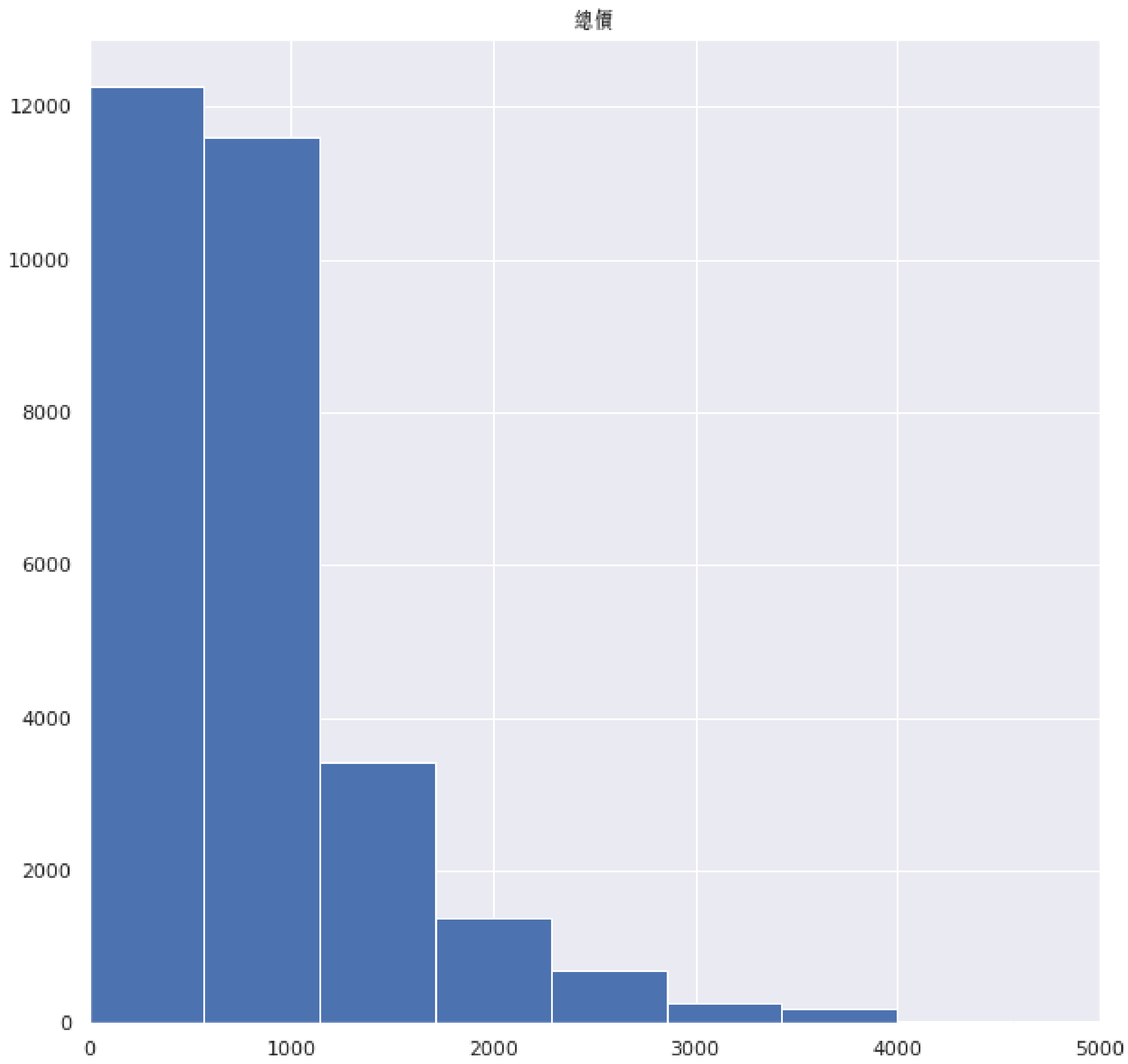
房子總價介於 0 – 1000 內。

土地面積分佈



建物總面積分佈

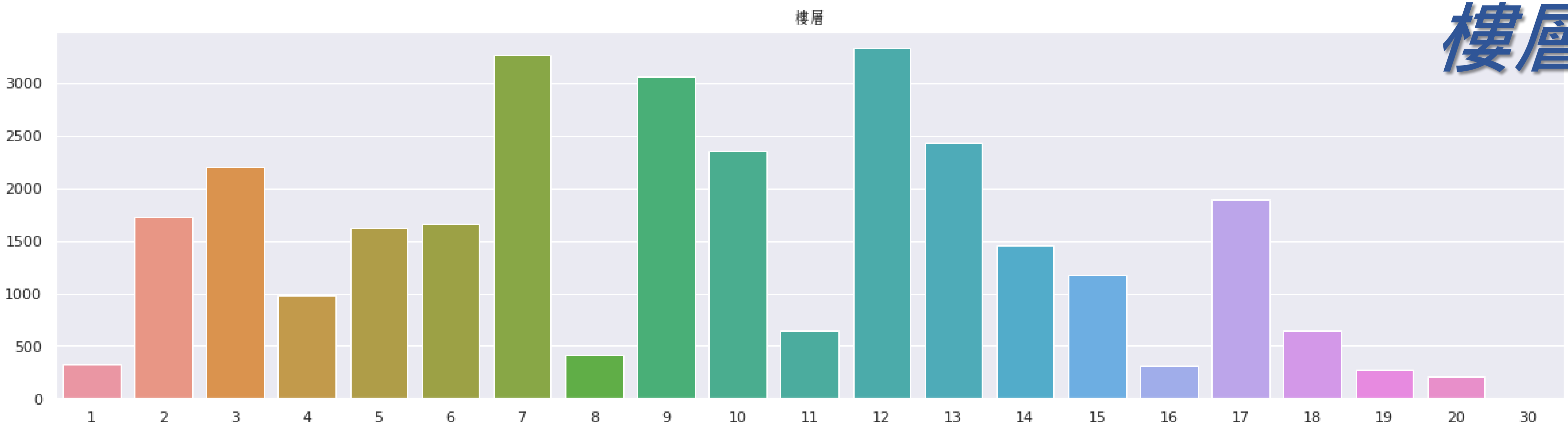




房子總價分佈

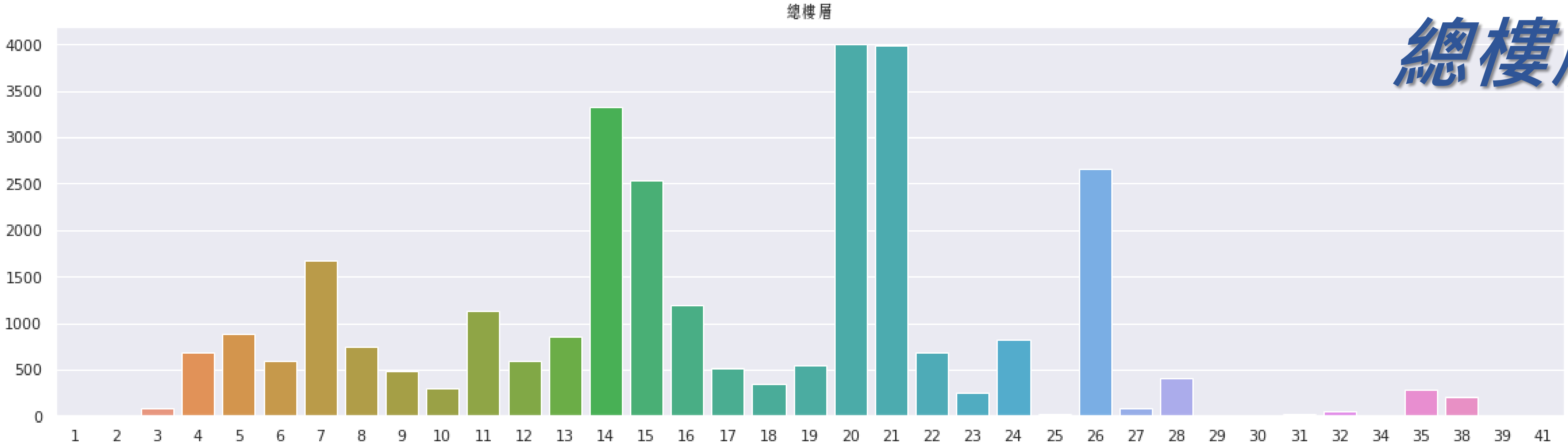
房子總價介於 0 – 1000 內。

樓層



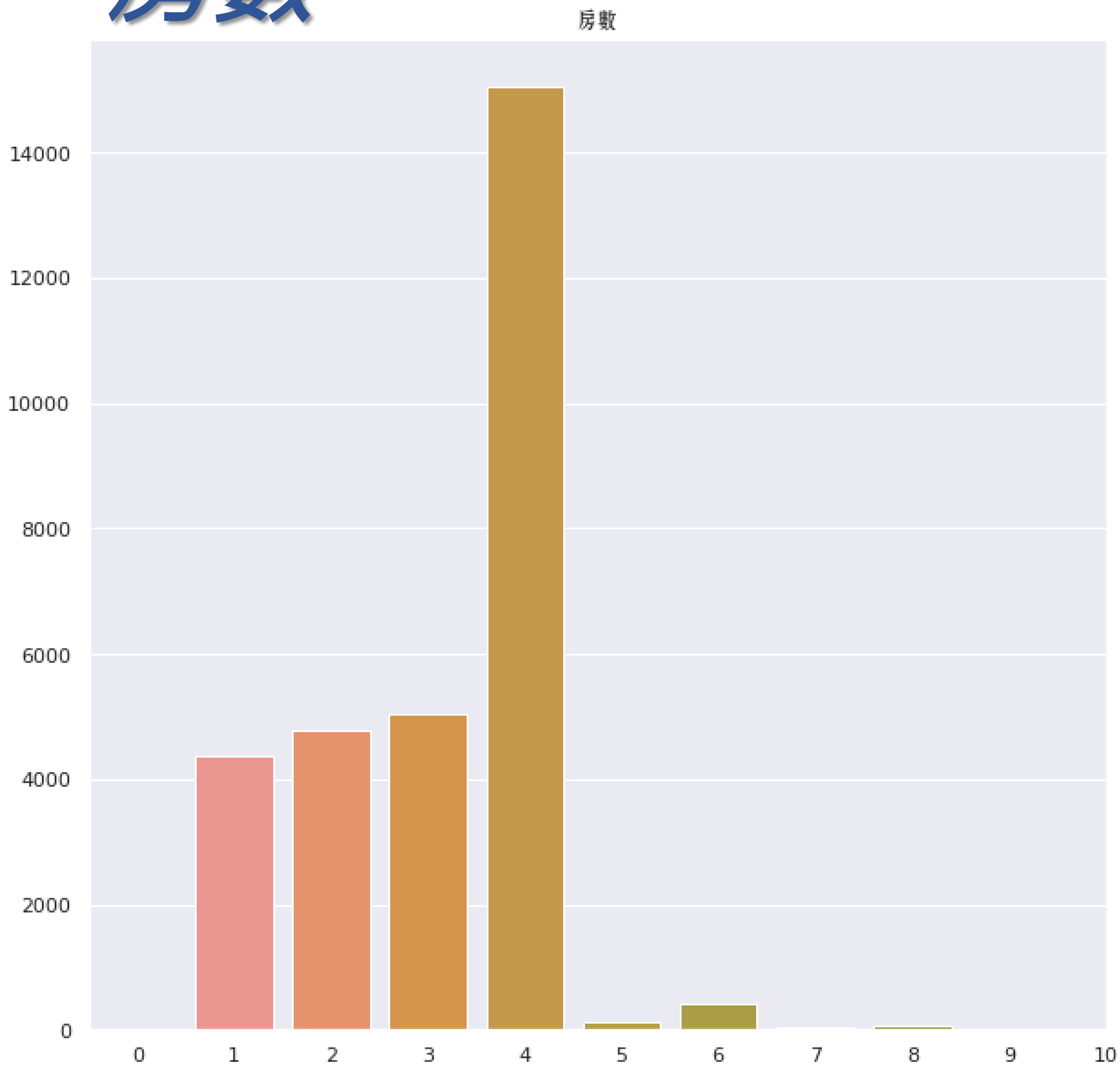
樓層
3000
2500
2000
1500
1000
500
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 30

總樓層



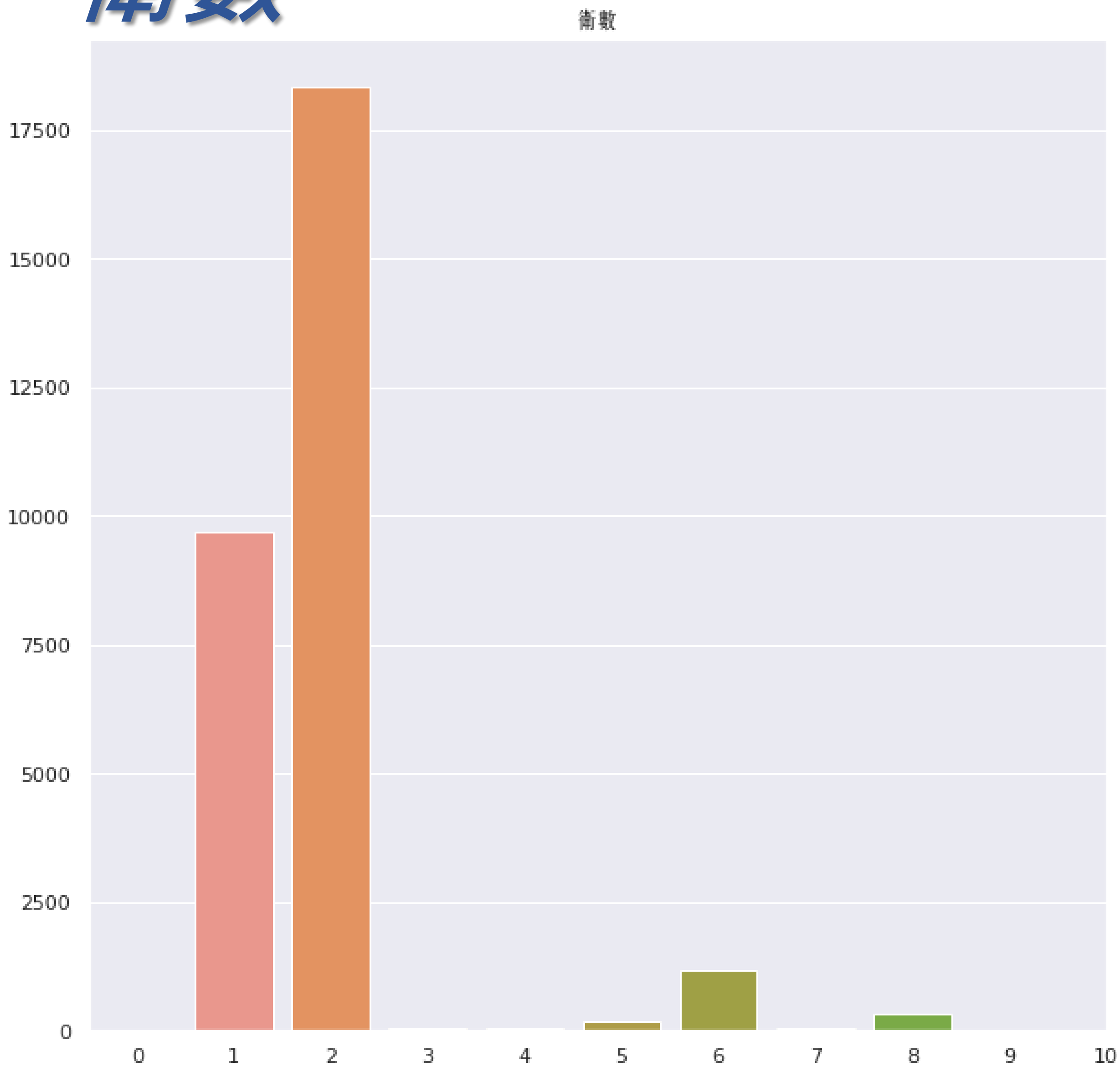
總樓層
4000
3500
3000
2500
2000
1500
1000
500
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 34 35 38 39 41

房數



房數
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
6
7
8
9
10

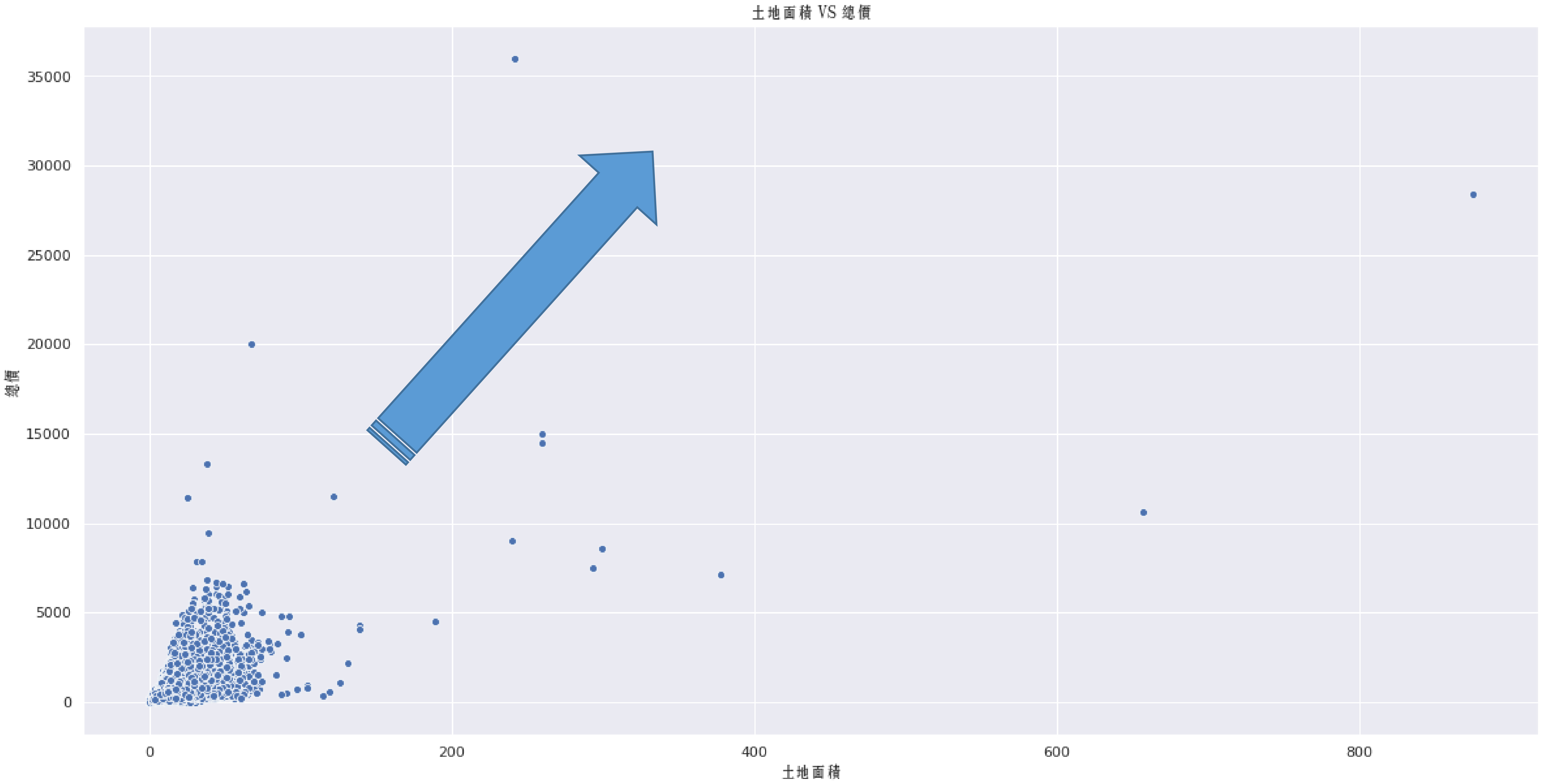
衛數



衛數
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
6
7
8
9
10

Bivariate analysis

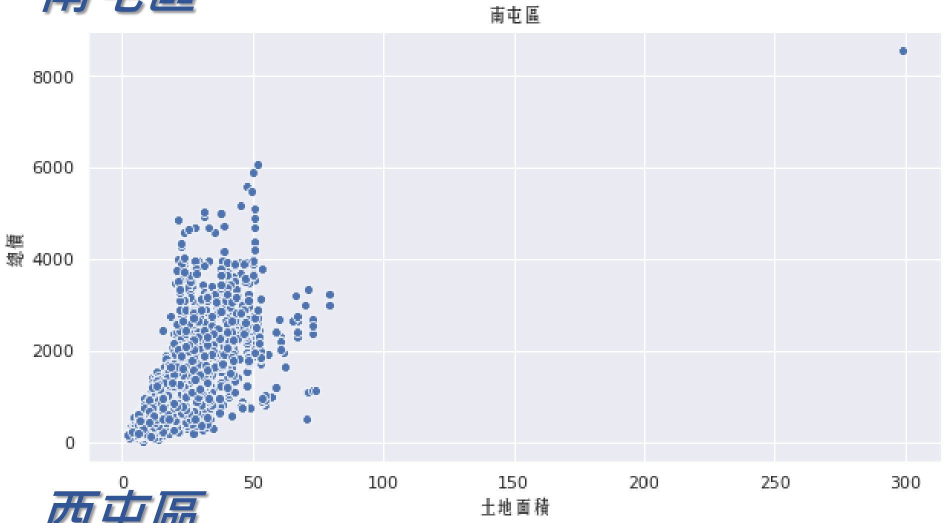
土地面積 & 總價



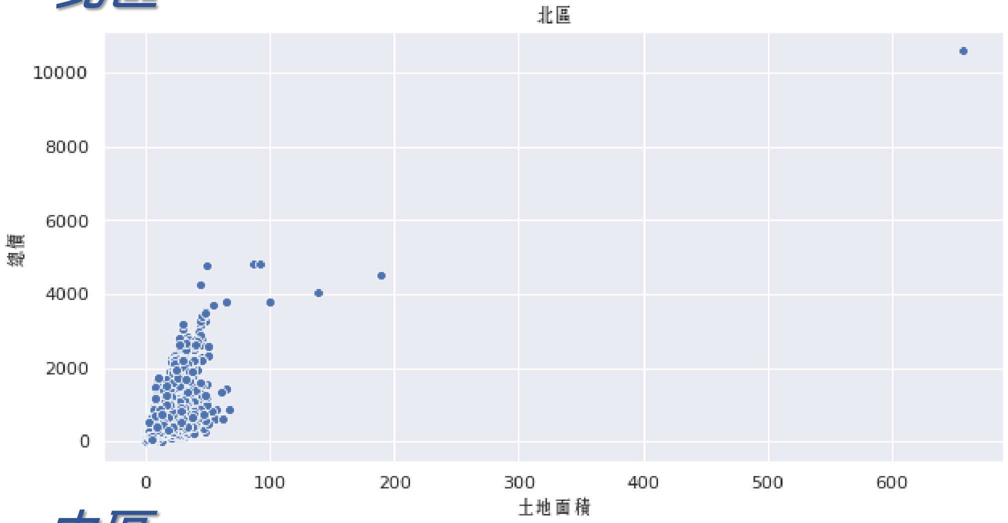
Bivariate analysis

土地面積 & 總價 (依行政區)

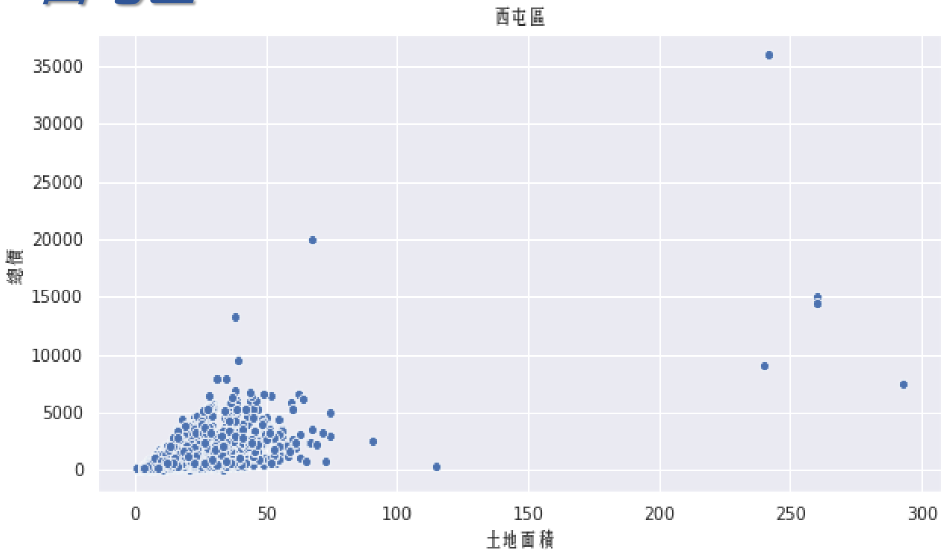
南屯區



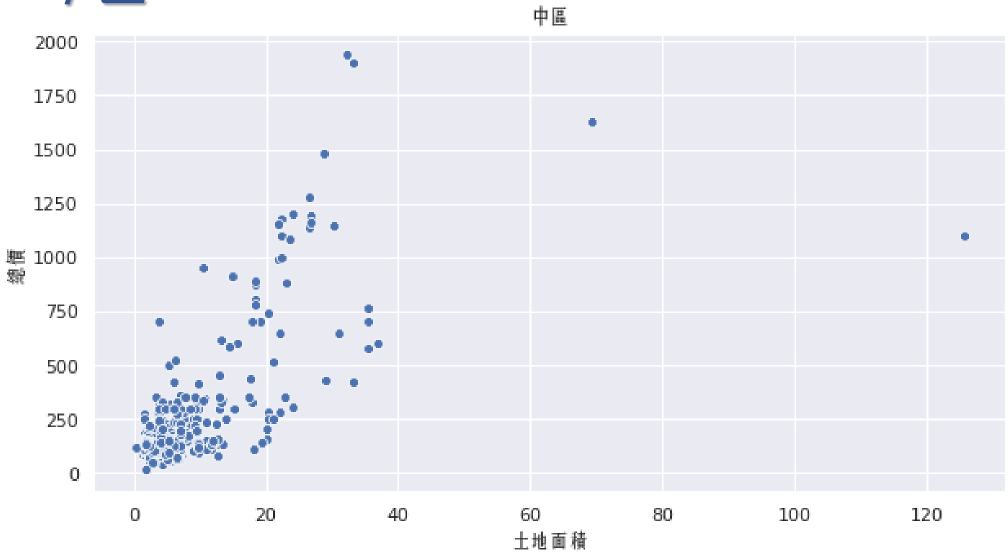
北區



西屯區

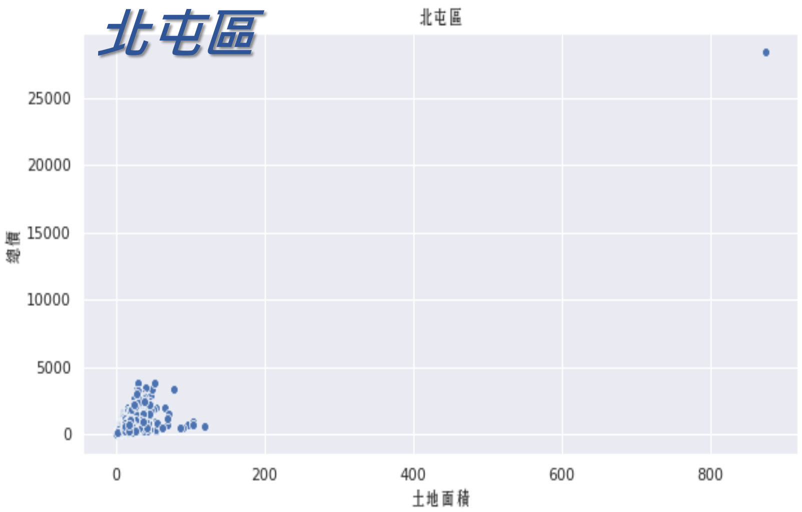


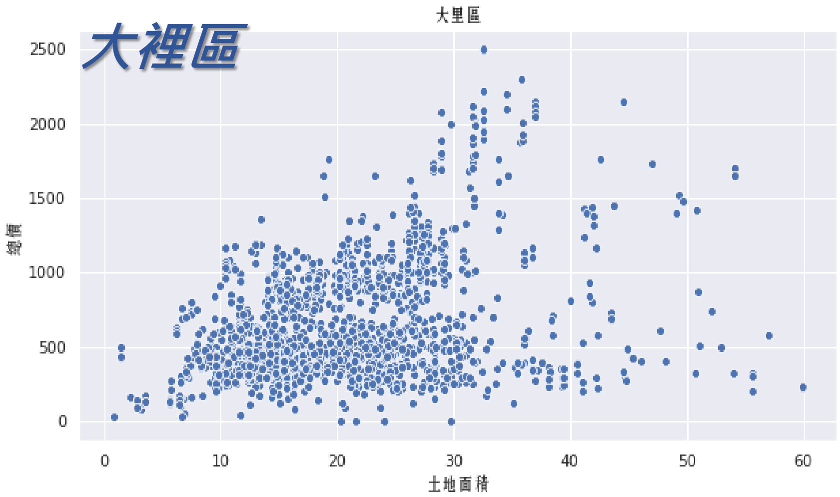
中區

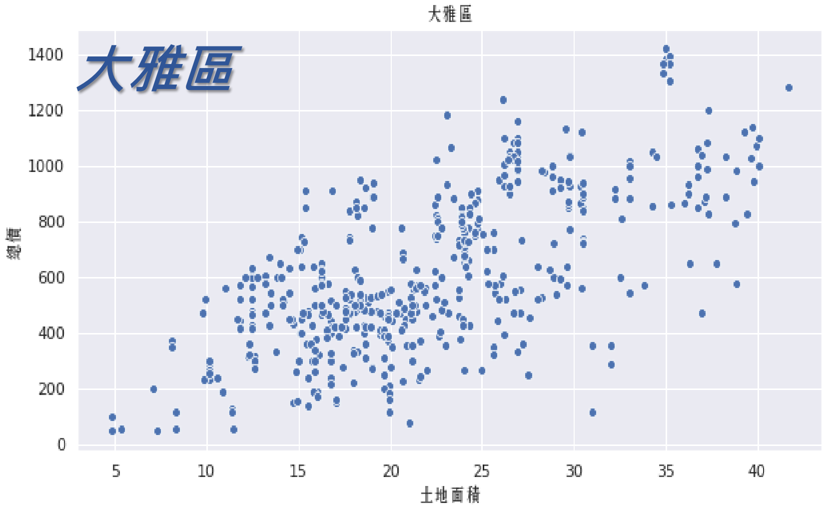


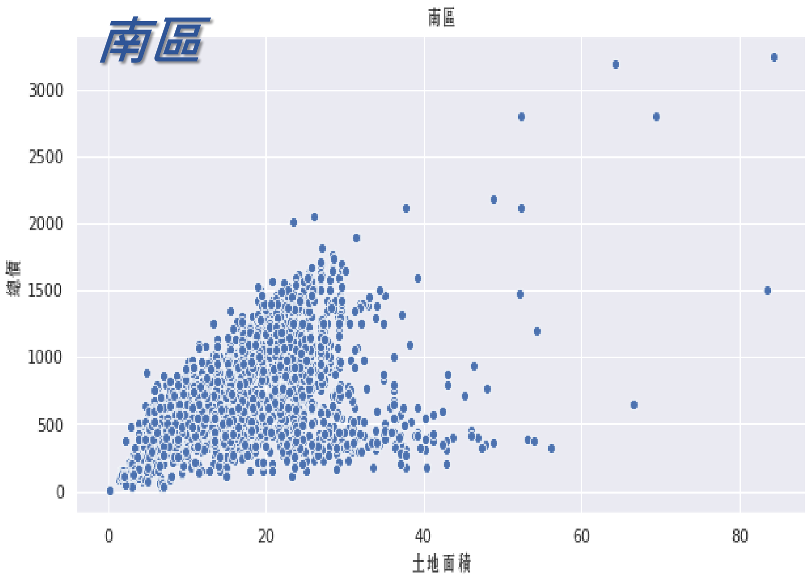
Bivariate analysis

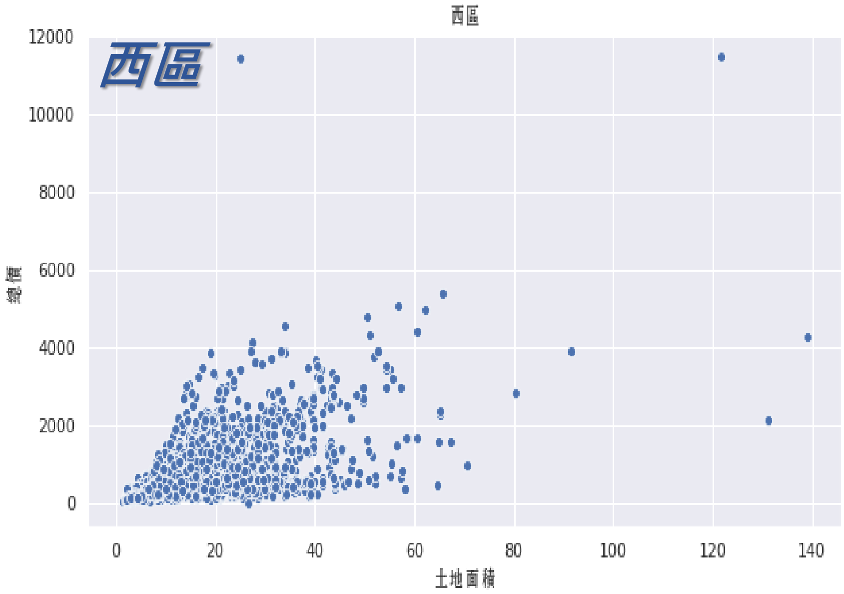
土地面積 & 總價（依行政區）

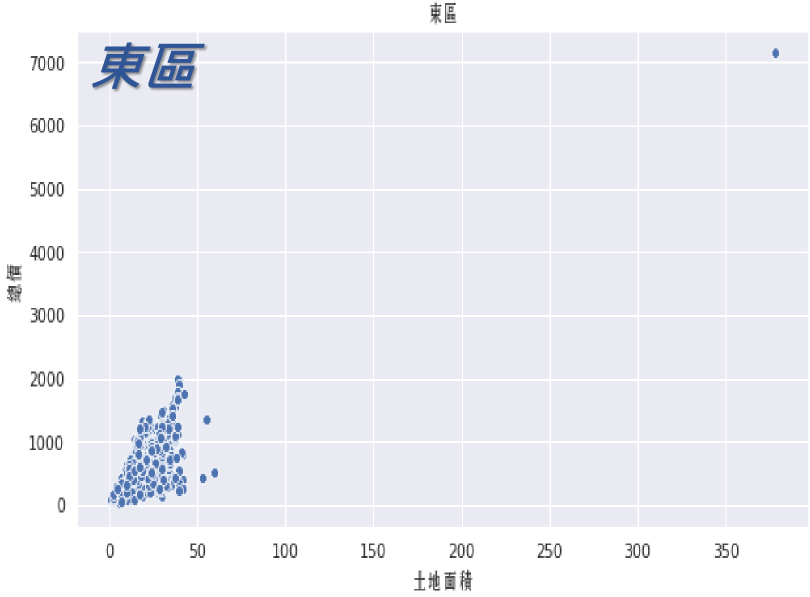








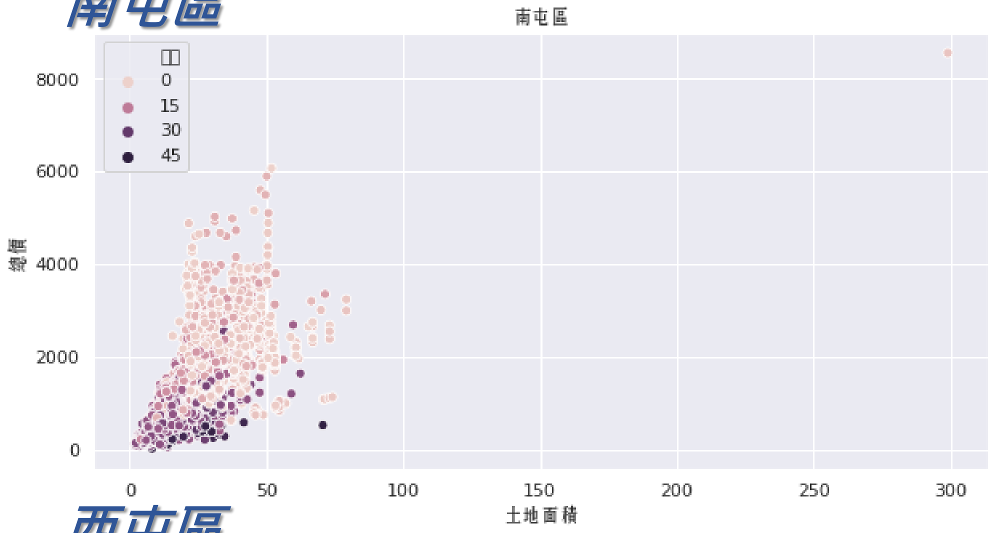




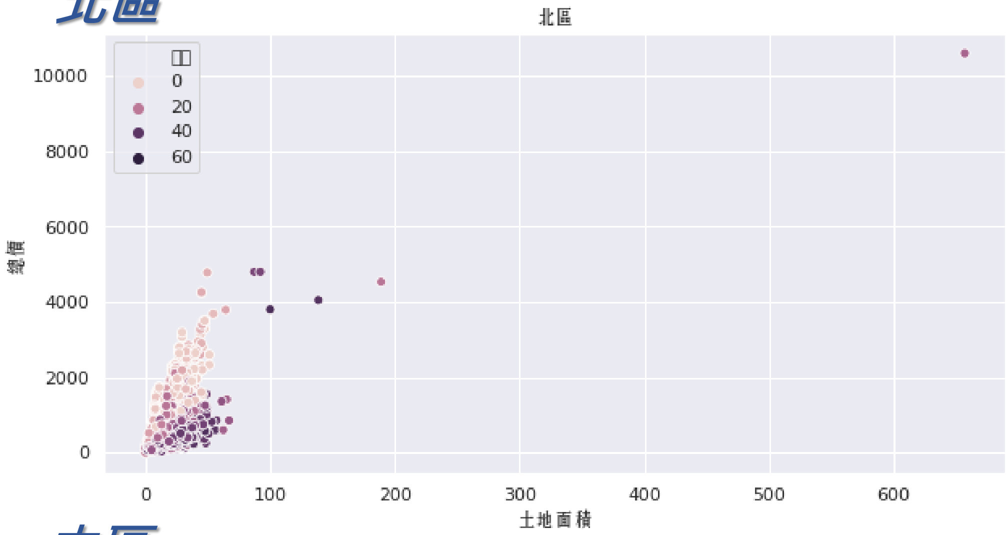
Bivariate analysis

土地面積 & 總價（依行政區）

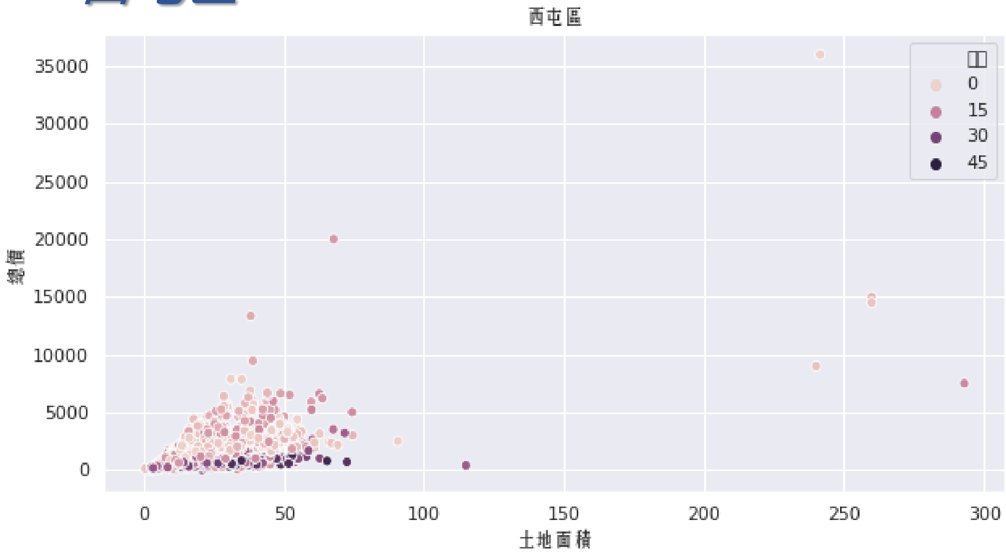
南屯區



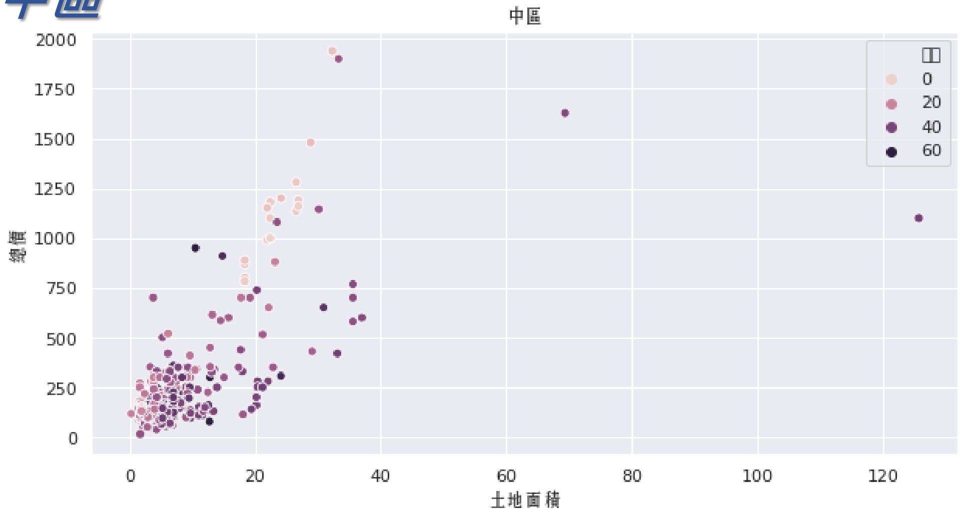
北區



西屯區



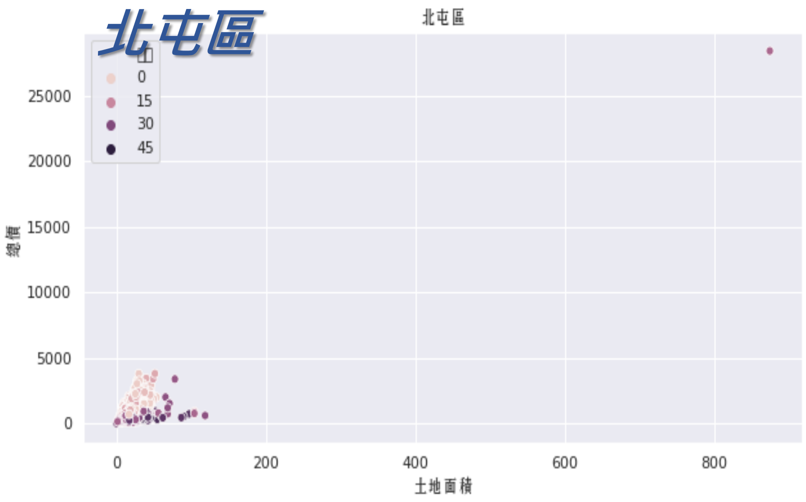
中區



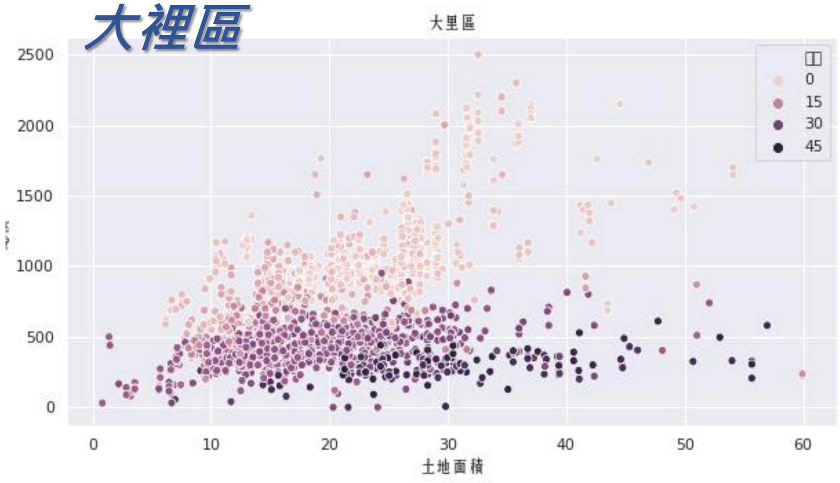
Bivariate analysis

土地面積 & 總價（依行政區）

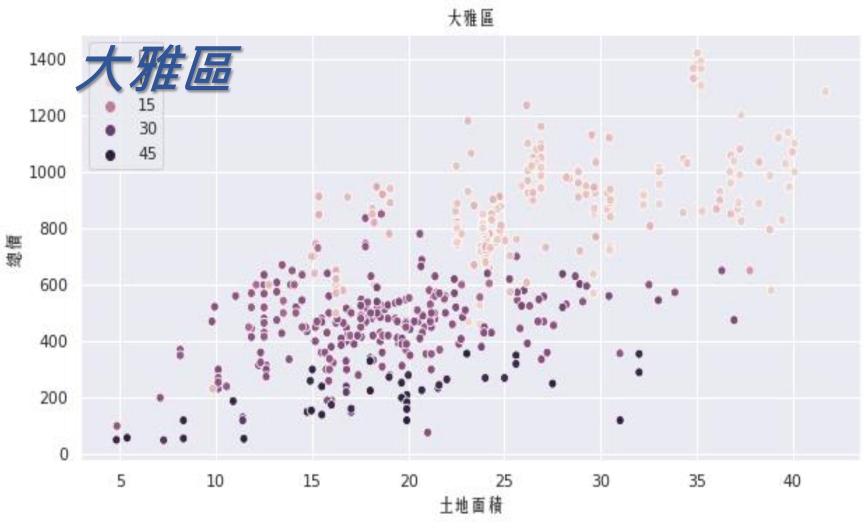
北屯區



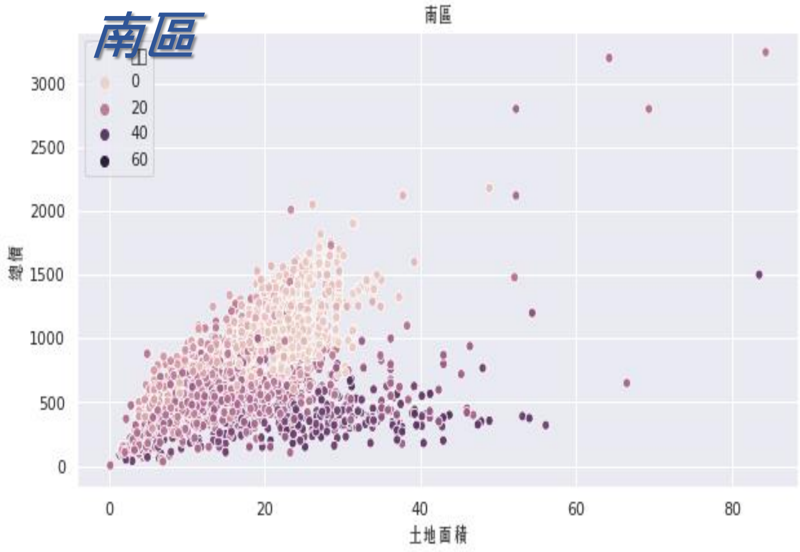
大裡區



大雅區

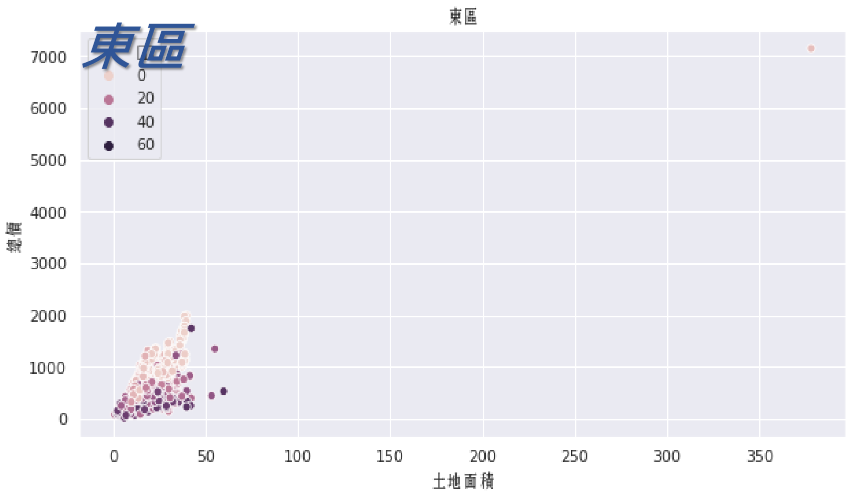


南區



西區

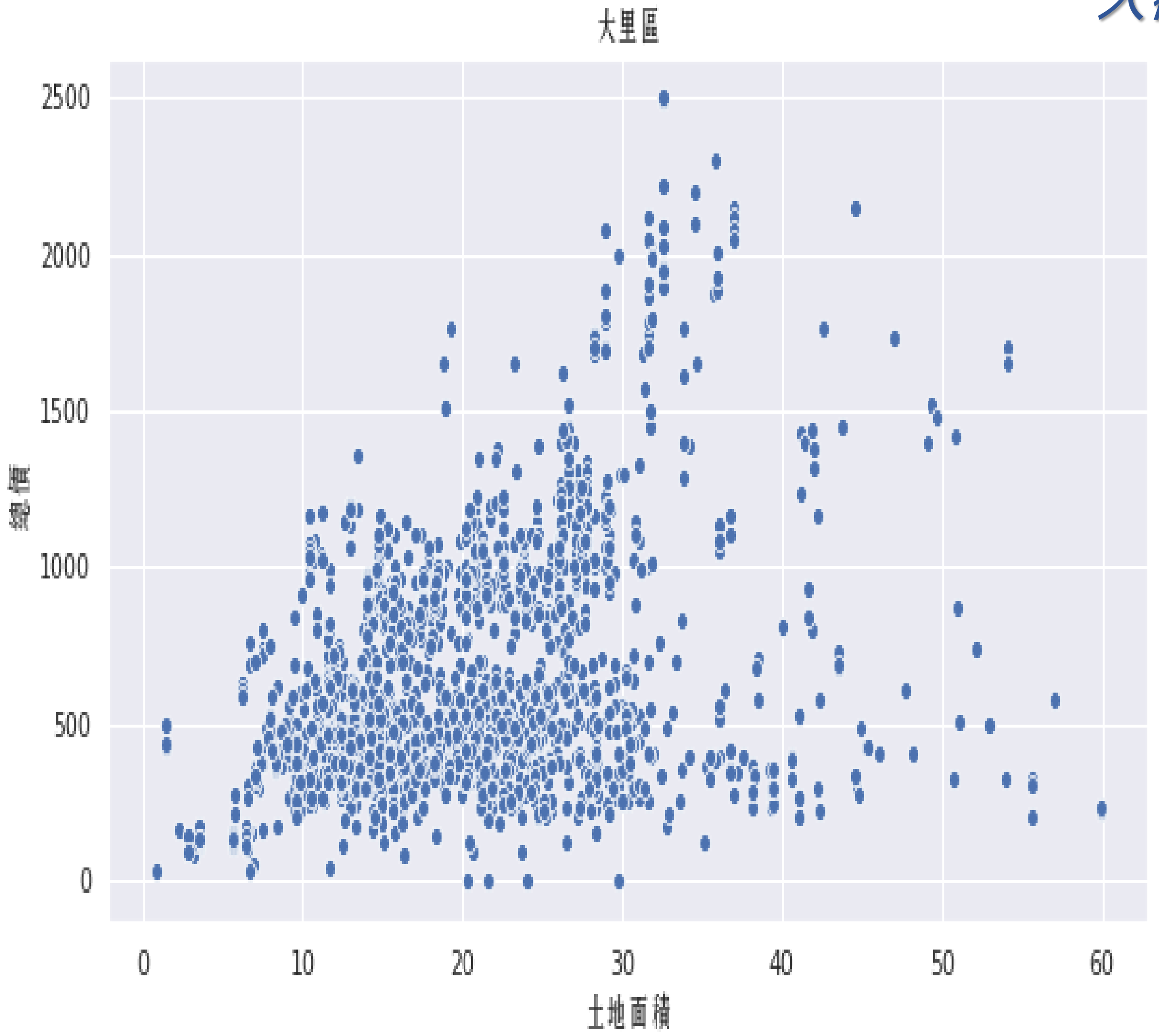
東區



Bivariate analysis

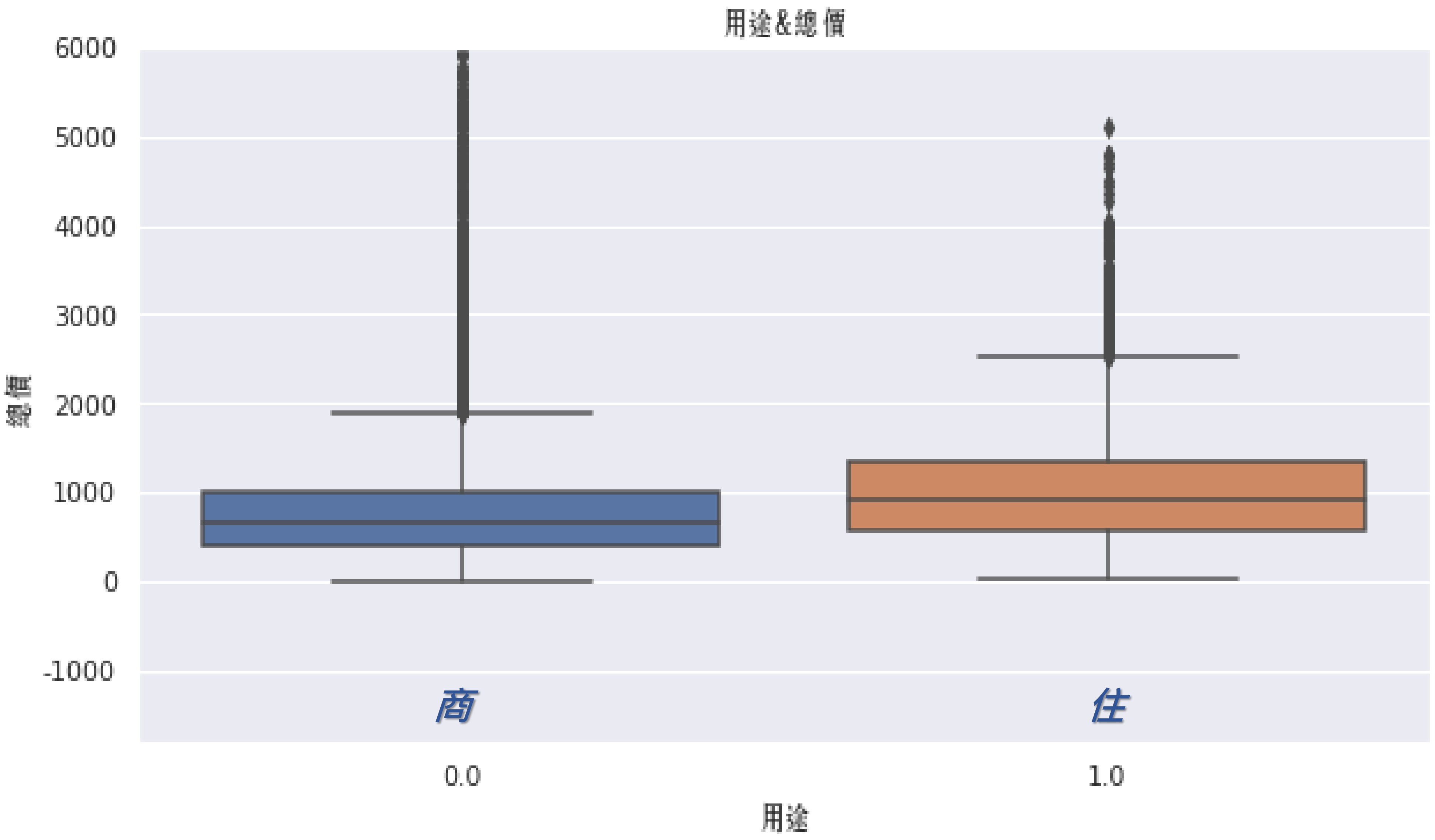
土地面積 & 總價 (依行政區)

大裡區



Bivariate analysis

用途 & 總價



#1 遺漏值 Imputation

```
''' Step 1: impute missing value or invalid value '''
train[train.isnull().values]
```

	行政區	土地面積	建物總面積	屋齡	樓層	總樓層	用途	房數	廳數	衛數	電梯	車位類別	交易日期	經度	緯度	總價
28986	北區	3.96	33.79	NaN	8	16	1.0	1	0	1	1	無	2017/5/13	120.665374	24.159243	150

```
train['屋齡'].fillna(value=train['屋齡'].mean(), inplace=True)
```

資料清理

#2 Scale numerical data

```
cols = ['土地面積', '建物總面積', '屋齡']
transformer = RobustScaler()
for col in cols:
  train[col] = transformer.fit_transform(np.array(train[col]).reshape(30000, -1)).flatten()
```

#3 Discretize numerical data

```
for col in cols:
  X = np.array(train[col]).reshape(-1,1)
  est = KBinsDiscretizer(n_bins=8, encode='ordinal', strategy='kmeans').fit(X)
  train[str(col)+'組別'] = est.transform(X)
```

#4 Deal with observations of which house price=0

```
''' Step 4: deal with observations with house price = 0 '''
train[train['總價'] == 0] #6 observations
```

	行政區	土地面積	建物總面積	屋齡	樓層	總樓層	用途	房數	廳數	衛數	電梯	車位類別	交易日期	經度	緯度	總價
4386	大里區	24.09	148.49	25.919766	5	7	1.0	3	2	2	1	無	2018/1/11	120.679558	24.095737	0
5798	大里區	20.33	123.95	26.946481	6	7	1.0	3	2	2	1	無	2019/11/13	120.688712	24.107648	0
15470	大里區	21.60	84.12	35.217698	4	5	1.0	3	2	2	0	無	2018/1/18	120.683116	24.114160	0
22036	西屯區	10.96	94.84	21.708865	9	15	1.0	2	2	1	1	坡道機械	2017/1/3	120.634910	24.176418	0
23785	西屯區	20.58	129.33	26.639835	6	15	1.0	3	2	2	1	無	2019/8/12	120.614063	24.182148	0
29228	西區	26.49	77.76	39.171236	5	5	1.0	3	2	2	0	無	2017/9/20	120.661001	24.150425	0

```
''' method 1: treat them as outliers and remove '''
train.drop(train[train['總價'] == 0].index, axis=0, inplace=True)
```

資料清理

#4 Deal with observations of which house price=0

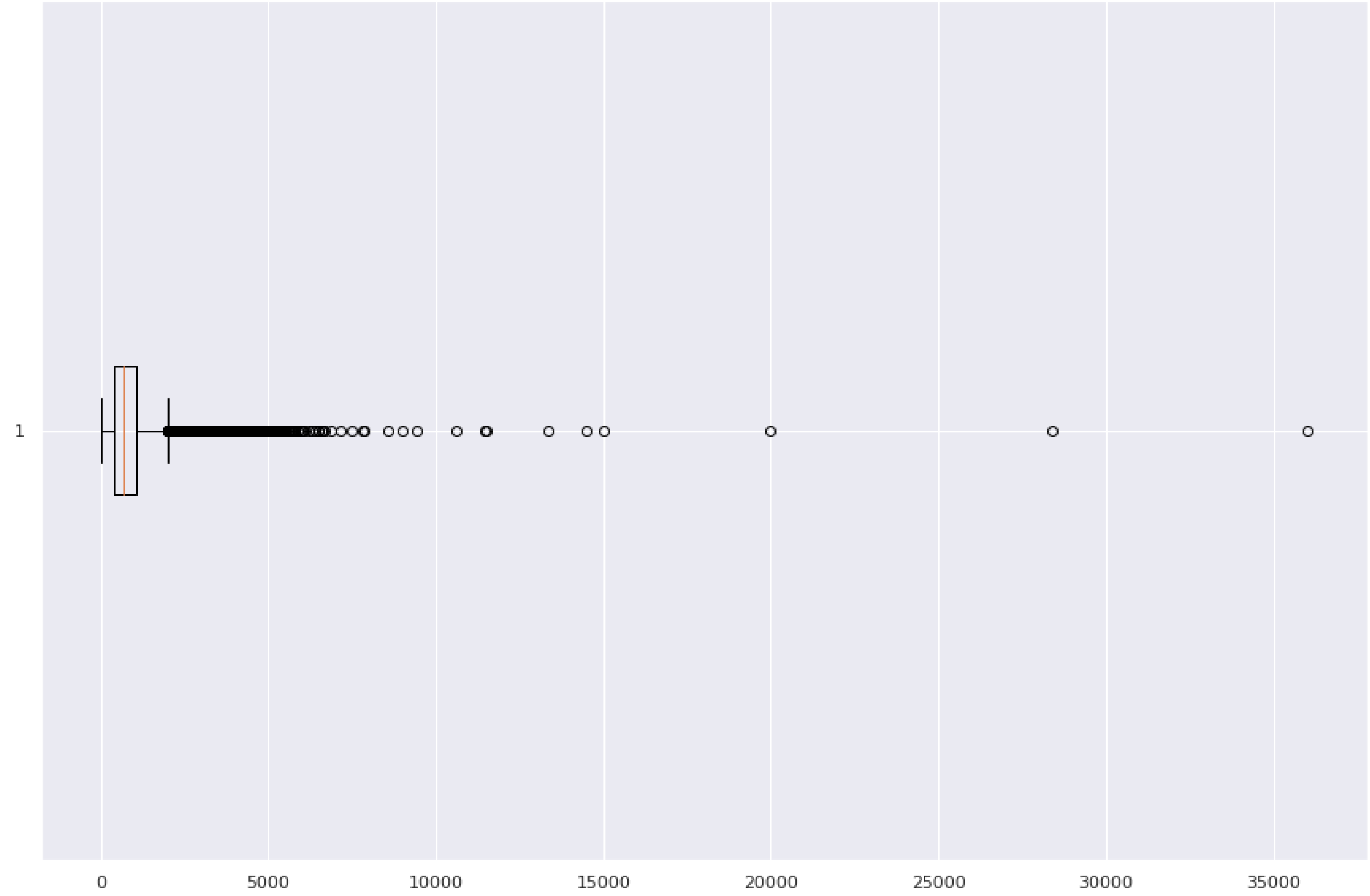
```python
''' method 2: treat them as missing values and impute using KNN '''
district = pd.get_dummies(train['行政區'], drop_first=True)
df_knn = pd.concat([train, district], axis=1)
df_knn.drop('行政區', inplace=True, axis=1)

enc = OneHotEncoder()
df_knn['用途'] = enc.fit_transform(np.array(df_knn['用途']).reshape(-1,1)).toarray()

date = df_knn['交易日期'].str.split('/')
df_knn['year'] = [i[0] for i in date]
df_knn.drop('交易日期', inplace=True, axis=1)

data = df_knn.loc[df_knn['總價']!=0, :]
X = data.drop(['車位類別', '總價'], axis=1)
y = data['總價']

neigh = KNeighborsRegressor(n_neighbors=5)
neigh.fit(X, y)

X_test = df_knn.loc[df_knn['總價']==0,:]
y_pred = neigh.predict(X_test.drop(['車位類別', '總價'], axis=1))

index = train.loc[train['總價']==0,:].index
for i,j in zip(index, range(len(y_pred))):
  train.loc[i,'總價'] = y_pred[j]
train.iloc[index,:]
```

總價
511.2
376.2
320.4
510.0
811.6
316.2

#5 Deal with univariate outliers

資料清理

```
''' Step 5: deal with univariate outliers '''
plt.boxplot(train['總價'], vert=False)
plt.show()
```

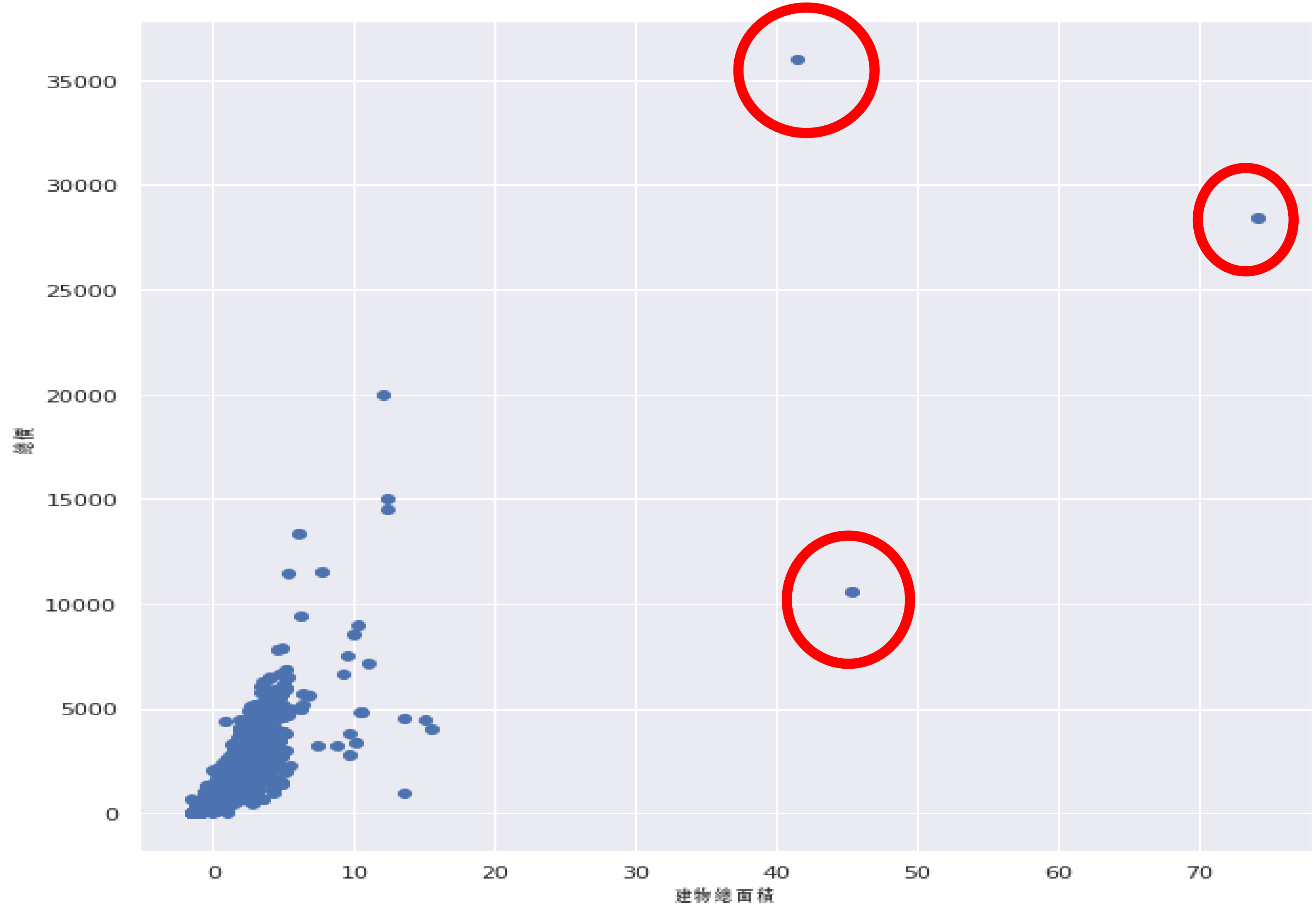


以IQR計算，

Outliers 多達

1924 rows

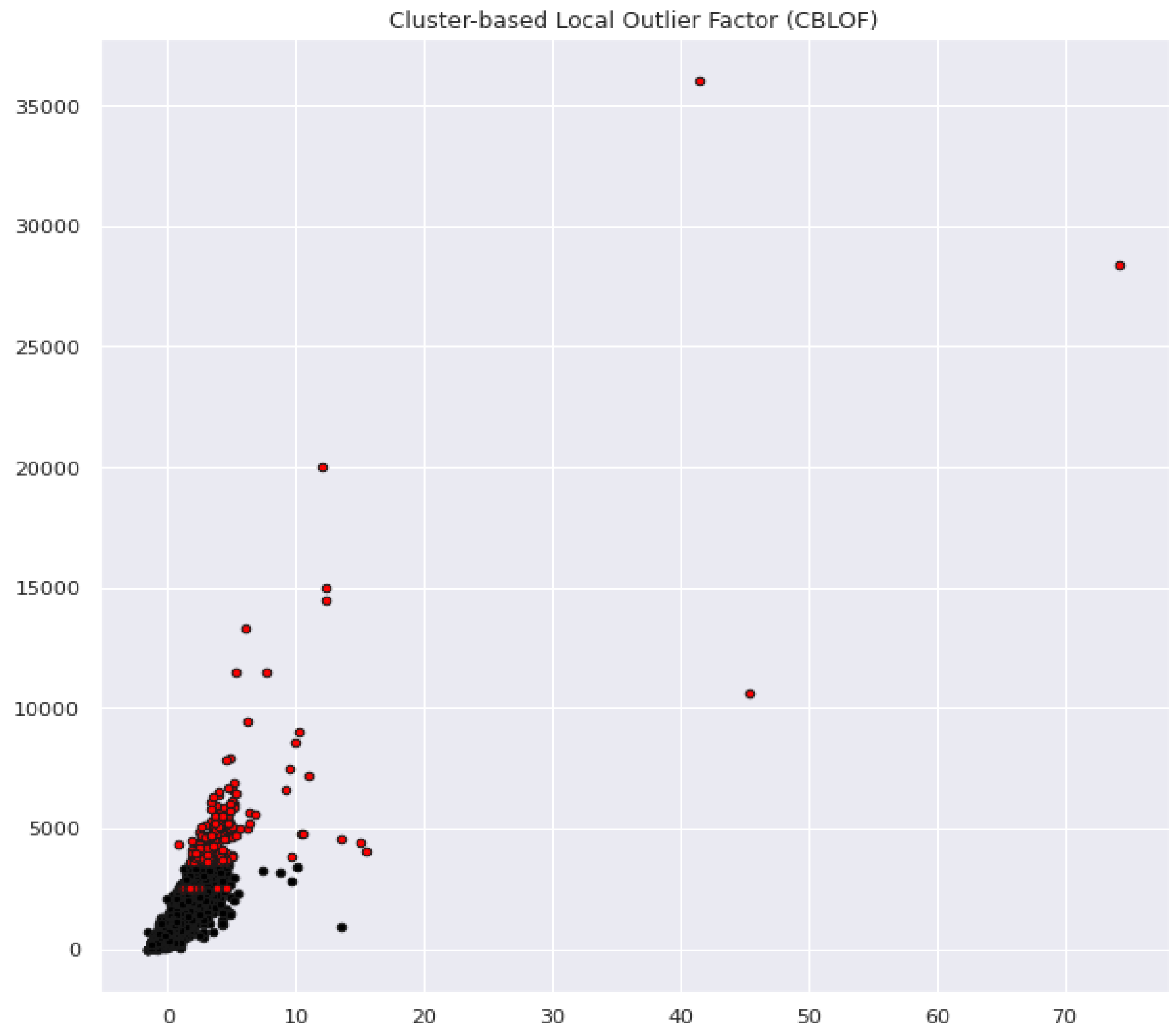
#6 Deal with multivariate outliers



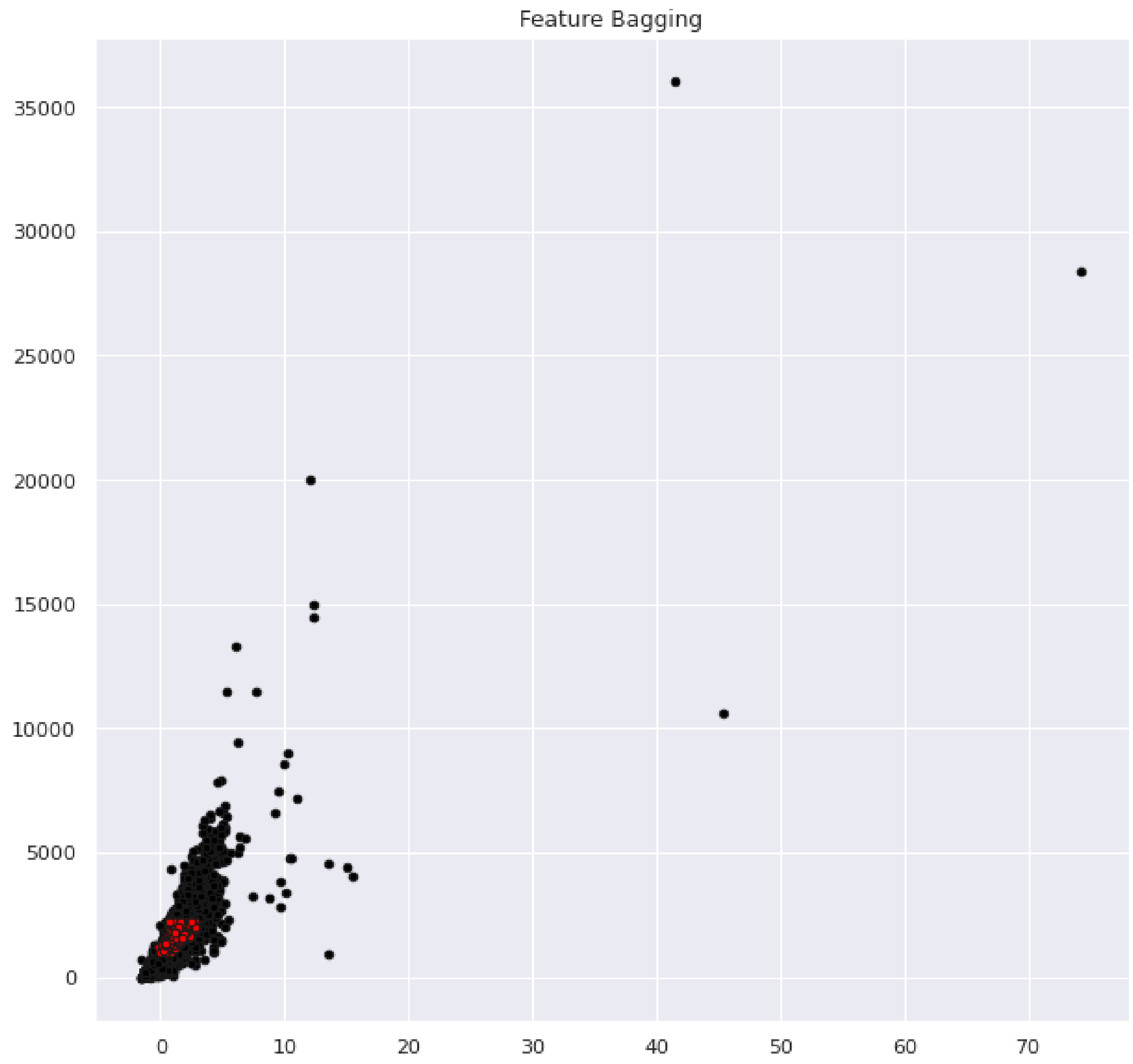
1. Cluster-based Local Outlier Factor
2. Feature Bagging
3. Histogram-base Outlier Detection (HBOS)
4. Isolation Forest
5. K Nearest Neighbors (KNN)
6. Average KNN

資料清理

#6 Deal with multivariate outliers

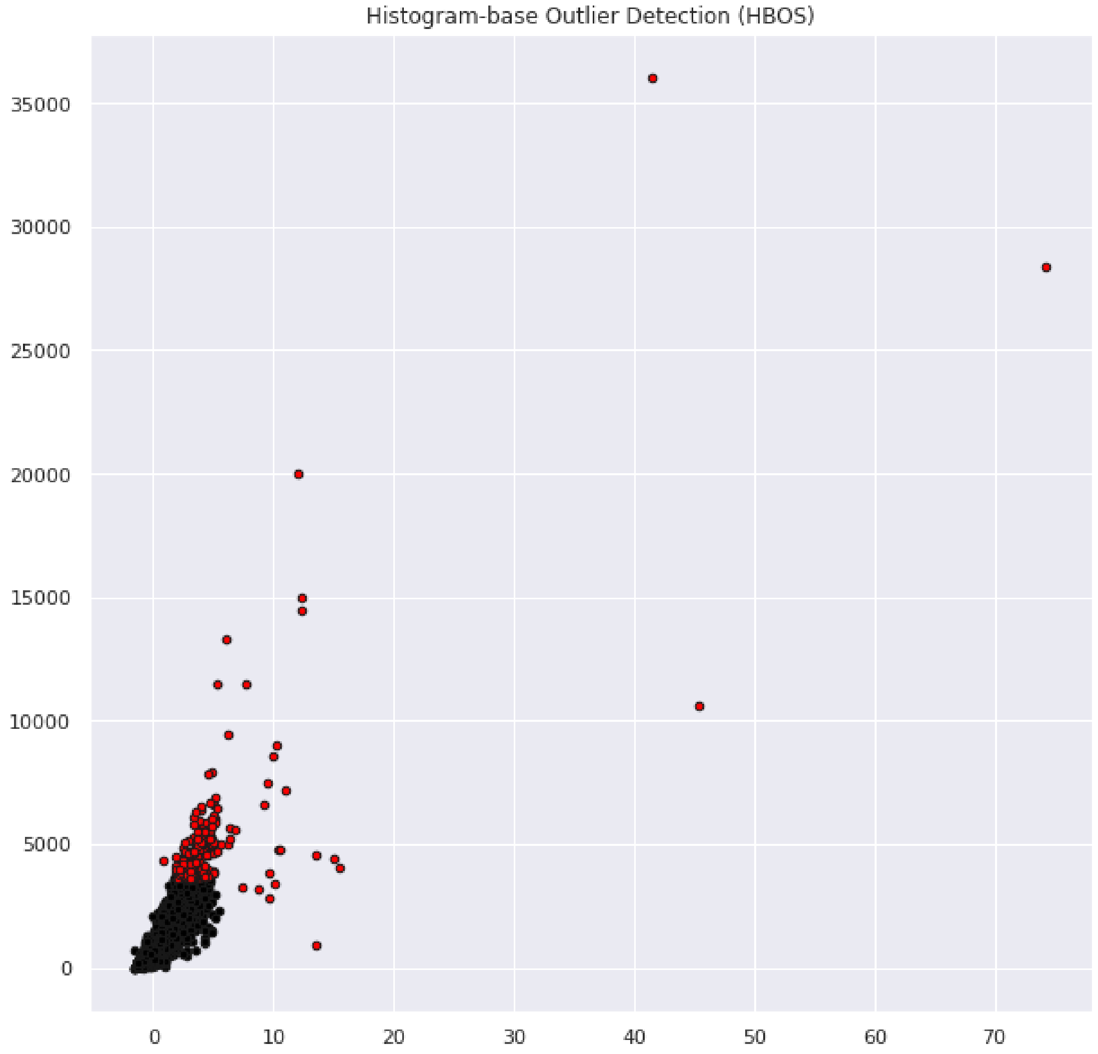


OUTLIERS : 300 INLIERS : 29700

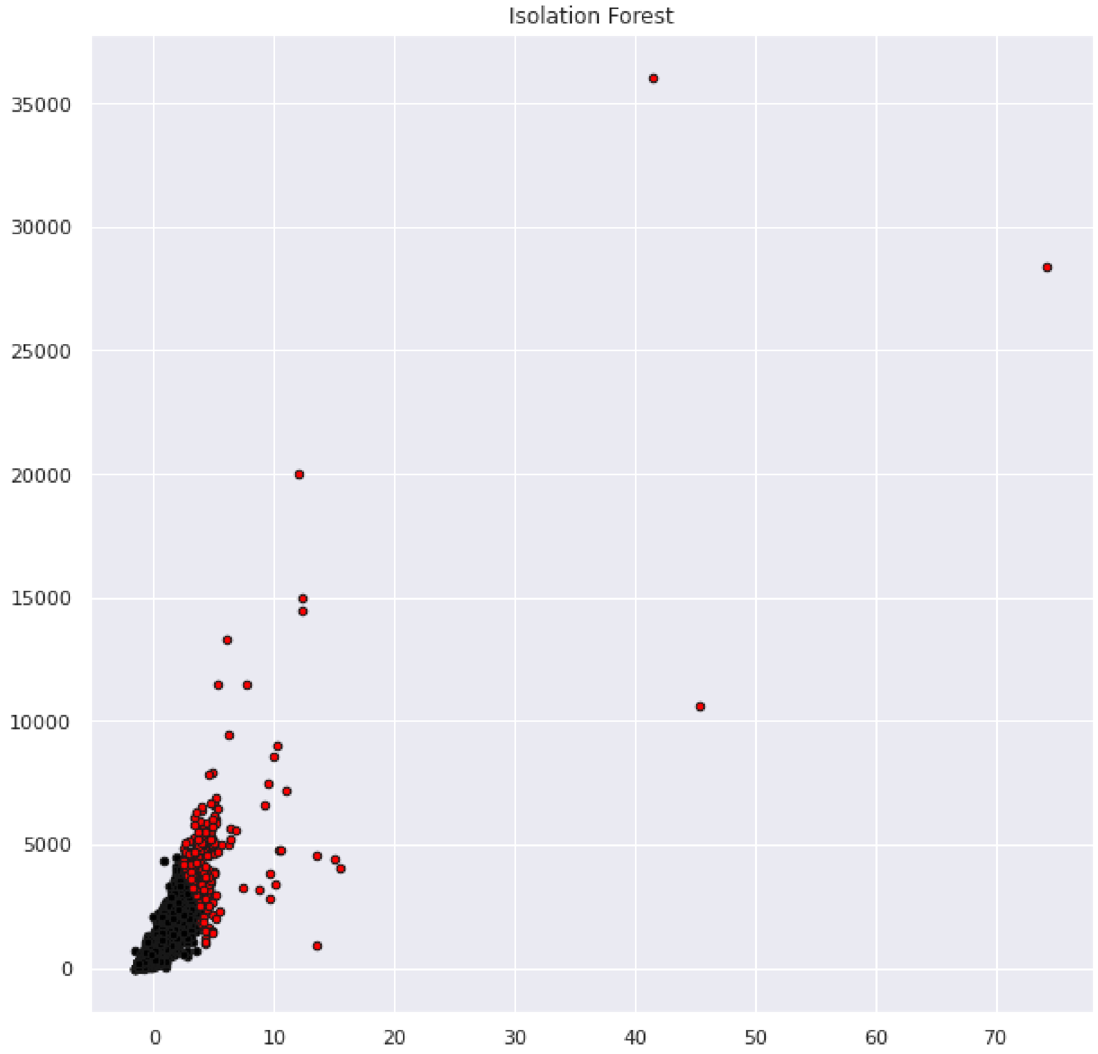


OUTLIERS : 258 INLIERS : 29742

#6 Deal with multivariate outliers



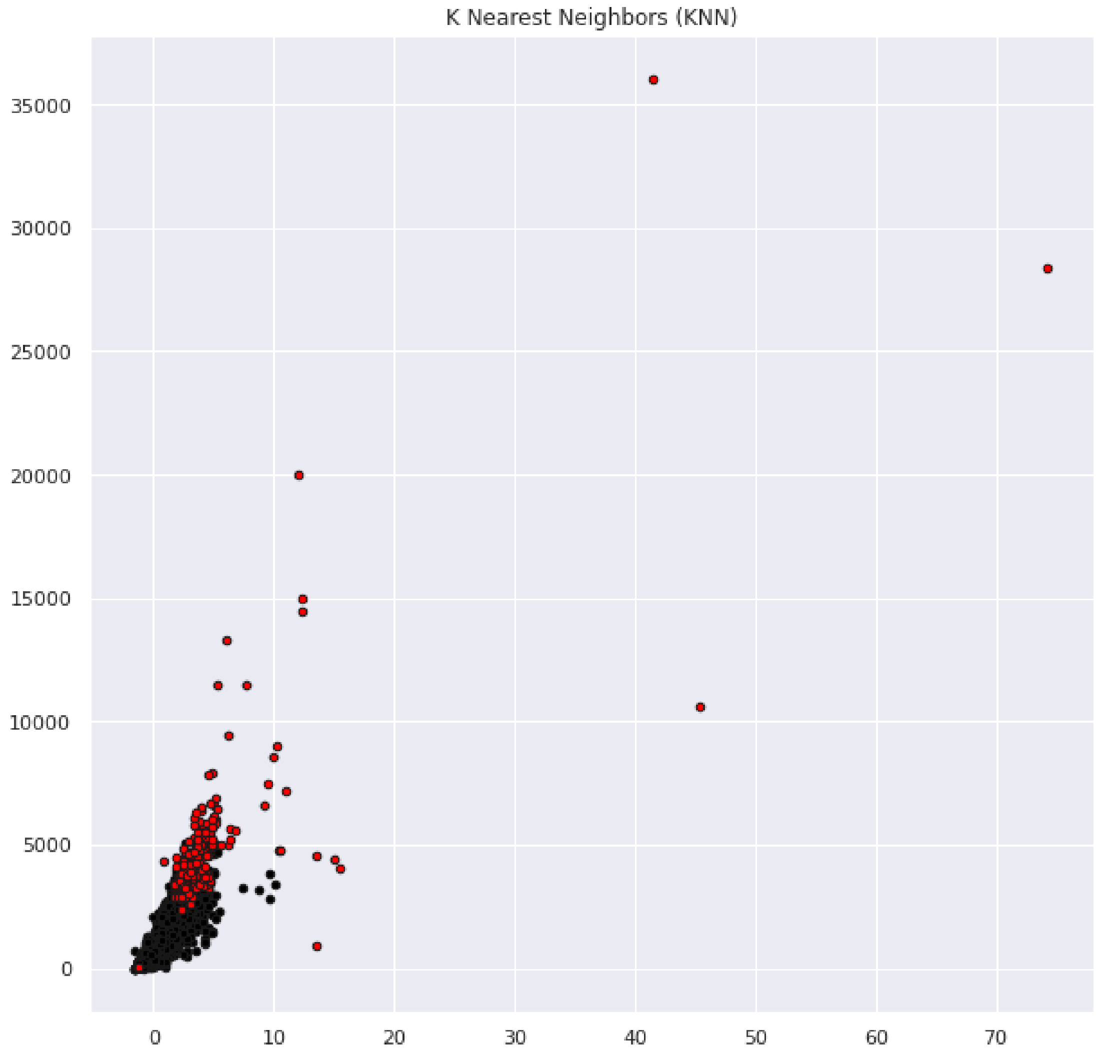
OUTLIERS : 258 INLIERS : 29742



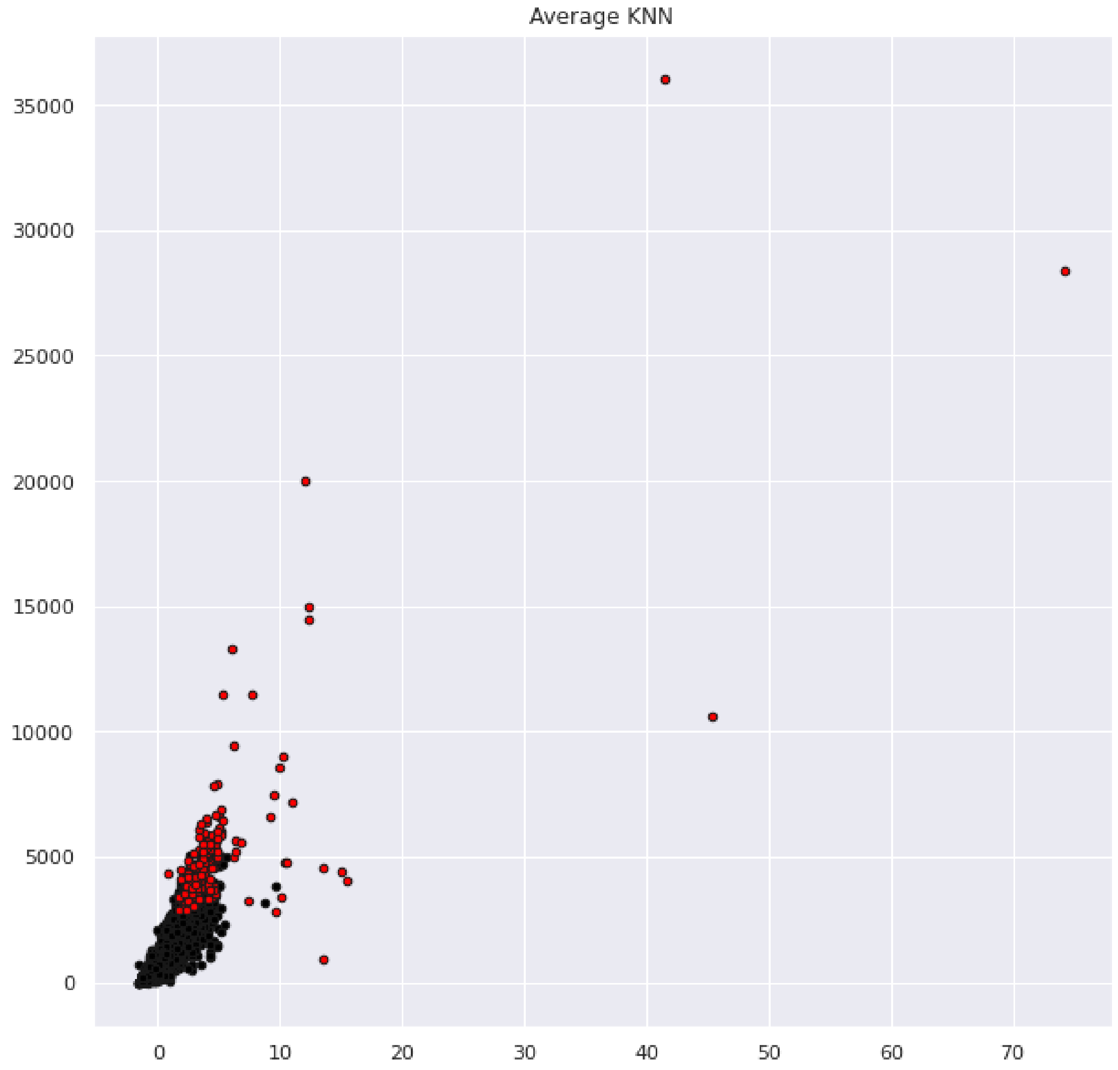
OUTLIERS : 300 INLIERS : 29700

#6 Deal with multivariate outliers

資料清理



OUTLIERS : 225 INLIERS : 29775



OUTLIERS : 182 INLIERS : 29818

Feature Extraction

Dummy variables / One-hot encoding

```
district = pd.get_dummies(train['行政區'], drop_first=True)
train = pd.concat([train, district], axis=1)
train.drop('行政區', inplace=True, axis=1)
train.head(5)
```

```
enc = OneHotEncoder()
train['用途'] = enc.fit_transform(np.array(train['用途']).reshape(-1,1)).toarray()
train.head(5)
```

北區	北屯區	南區	南屯區	大里區	大雅區	東區	西區	西屯區
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1

用途	房數	廳數	衛數	電梯
1.0	4	2	2	1
1.0	3	2	2	1
1.0	4	2	2	1
1.0	2	1	2	1
1.0	3	2	2	1

Feature Extraction

Date → Year / Month / Day

```python
date = train['交易日期'].str.split('/')
train['year'] = [i[0] for i in date]
train['month'] = [i[1] for i in date]
#train['day'] = [i[2] for i in date]
train.drop('交易日期', inplace=True, axis=1)
train.head()
```

交易日期
2019/3/22
2019/11/18
2018/12/11
2017/9/20
2019/5/26

year	month
2019	3
2019	11
2018	12
2017	9
2019	5

Feature Selection

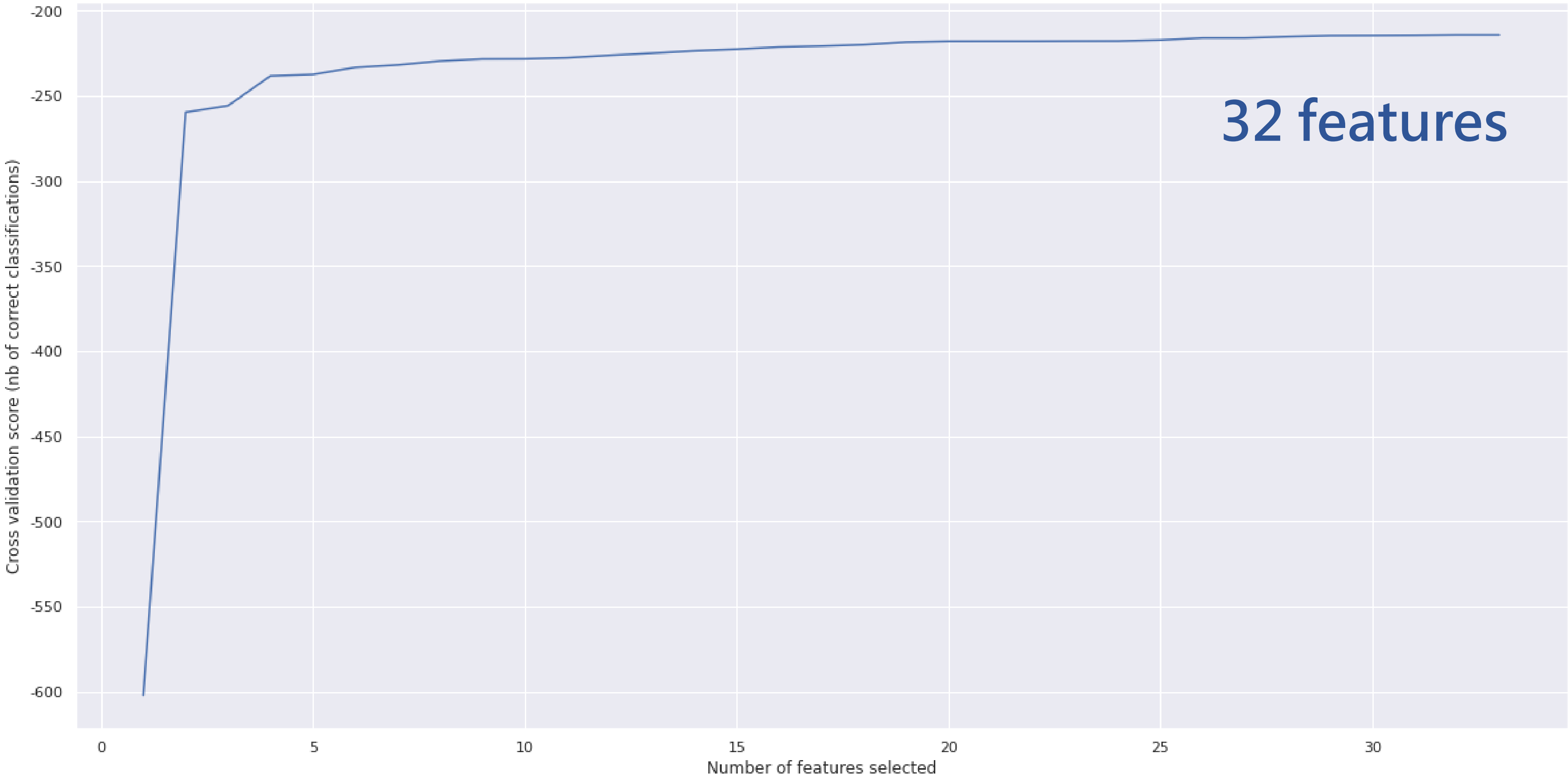
1. Select according to p-value
2. Backward Elimination (BE)
3. Recursive Feature Elimination (RFE) ✓
4. L1-based feature selection
    - Lasso selection (LSO)
    - Lasso CV selection (LSOCV)

5. Tree-based Feature Elimination

- Random Forest Regressor (RF)

Feature Selection

Recursive Feature Elimination (RFE)



32 features

Model Selection

Model Sets (11)

```python
regressor = []
model_0 = linear_model.LinearRegression()
regressor.append(model_0)
model_1 = linear_model.Ridge()
regressor.append(model_1)
model_2 = linear_model.Lasso()
regressor.append(model_2)
model_3 = linear_model.LassoCV()
regressor.append(model_3)
model_4 = linear_model.LassoLars()
regressor.append(model_4)
model_5 = linear_model.LassoLarsCV()
regressor.append(model_5)
model_6 = linear_model.LassoLarsIC(criterion='bic')
regressor.append(model_6)
model_7 = linear_model.ElasticNet()
regressor.append(model_7)
model_8 = linear_model.BayesianRidge()
regressor.append(model_8)
#model_9 = linear_model.ARDRegression()
#regressor.append(model_9)
#model_10 = linear_model.SGDRegressor()
#regressor.append(model_10)
model_9 = linear_model.RANSACRegressor()
regressor.append(model_9)
model_10 = linear_model.TheilSenRegressor()
regressor.append(model_10)
model_11 = linear_model.HuberRegressor()
regressor.append(model_11)
```

×

Feature Sets (6)

```python
features_set = {'All': X.columns,
                'BE': features_BE,
                'RFE': features_RFE,
                'LSO': features_LSO,
                'LSOCV': features_LSOCV,
                'RF': features_RF}
```

66 models

Model Selection

Linear Regression with features from RFE ✓

```python
from sklearn.linear_model import LinearRegression

X_train = train.drop('總價', axis=1)
X_train = X_train.loc[:, features_RFE]
y_train = train['總價']
reg = LinearRegression().fit(X_train, y_train)

test = test.loc[:, features_RFE]
y_pred = reg.predict(test)
```

The finishing touch

Negative predictions (KNN)

```
''' use knn to deal with negative predictions '''
X = test.loc[(y_pred>0), :]
y = y_pred[y_pred>0]

neigh = KNeighborsRegressor(n_neighbors=5)
neigh.fit(X, y)

X_test = test.loc[(y_pred<=0), :]
y_test = neigh.predict(X_test)
```

	id	總價
56	56	96.555849
68	68	187.728518
96	96	64.680380
128	128	235.701325
170	170	180.480372
...	...	...
4680	4680	191.008947
4726	4726	85.189675
4749	4749	129.566624
4829	4829	155.537113
4981	4981	117.047040

Q&A