STATS 504 HW4

Statistical Analysis for New Taipei Housing Litigation

March 2022

1 Introduction

For the real estate industry, statistical models are helpful to assess the price of real estate in one particular housing area. However, the price of a real estate unit can be influenced by various factors, like the housing age, geographical location, and house age, which brings many difficulties for the process of price appraisal. Our client tried to refer to some papers in the journal to get help with the data modeling and prediction task, but the statistical models suggested are way more complicated to understand, especially for non-statisticians. Also, though these models may give relatively accurate predictions, they may lose some interpretability. Since it is also important for our client to know how other factors affect the price, compared to giving out good final price estimations, we decided to construct a model that is easy to interpret and gives relatively precise price estimations.

Our final choice for our client is a spline model, which treats variables differently according to their values, allowing a more flexible model fit. Specifically, we developed a general additive model (GAM) with B-splines for some covariates. It performed well in the prediction task, and its results are also easy to interpret, which make it perfectly suit our client's need.

2 Methods

Our statistical analysis aims to build a model on the given data to estimate the expected house prices in a given area of New Taipei City. Specifically, we take housing information such as location, number of grocery stores, number of MRT stations nearby, and house age to predict the housing price in a certain area.

2.1 Splines Model

The most common statistical model to make a prediction with interpretability is a linear regression model. However, the downside of a simple linear regression model is that the model may not include enough complex functional relationships between X and Y. This may lead to the missing of important information or wrong interpretations.

Besides, using polynomial regression may encounter the edge effect because of the high dimensional. We overcome these issues by applying splines to our model. Splines are piecewise smooth functions that can explain different variations within each covariate while avoiding perplexing high-dimensional mathematical formulas.

In particular, we will apply general additive models with B-splines on this given data set. We compare the models with different numbers of knots using simple iterations and choose the best number of knots according to our evaluation metrics.

Assume we choose K knots within the domain of $x : \xi_1, \ldots, \xi_K$ and we choose to use an order M spline, denote:

- ξ_0 and ξ_{K+1} be the boundary knots that to be the scalar domain for the spline function.
- $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_M \leq \xi_0$
- $\tau_{j+M} = \xi_j, j = 1, \ldots, K$
- $\xi_{K+1} \leq \tau_{K+M+1} \leq \tau_{K+M+2} \leq \cdots \leq \tau_{K+2M}$

Denote $B_{i,m}(x)$ to be the i th B-spline basis function of order m for the knot sequence $\tau, m \leq M$, we got

$$B_{i,1}(x) = \begin{cases} 1 & \text{if } \tau_i \leq x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, K + 2M - 1$ and

$$B_{i,m} = \frac{x - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m} - x}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(x)$$

for $i = 1, \ldots, K + 2M - 1$.

2.2 Akaike information criterion (AIC) and mse

In this statistical analysis task, we used the Akaike information criterion (AIC) and Mean squared error (MSE) for model selection. For this report, we first use iterations to find the best-fitted spline model using the AIC method. Next, we use the MSE scores for comparing the general linear regression model and the best-fitted spline model to do the final model selection between them.

The AIC score is a likelihood metric that is easy and fast to compute. It can function as a sanity check for the improvement of model accuracy. However, we cannot use the data twice in model fitting and evaluation. It could potentially deviate the accuracy of model parameters from the true model as repeated information is used. In this scenario, AIC scores might not be the favorable evaluating metric. Therefore, we use MSE to evaluate the final chosen models, which is more robust to avoid overfitting and elevating the model accuracy.

And AIC's mathematical form can be represented as:

$$\text{AIC}(\mathbf{m}) = -2 \log\text{likelihood} + 2p_{\mathbf{m}} \overset{\epsilon_{\mathbf{mi}} \sim NID, LS}{=} n \log\left(\frac{\text{RSS}(\mathbf{m})}{n}\right) + 2p_{\mathbf{m}}$$

with LS means dropping constants. Since AIC contains -2 loglikelihood, the lower AIC, the better is the model. The "best" model using AIC is

$$\hat{\mathbf{m}}_{\text{AIC}} = \text{argmin}_{\mathbf{m} \in \mathcal{A}} \, \text{AIC}(\mathbf{m})$$

As the model with the lowest AIC returns the best model. Our choice of degree of freedom' combination that gives us the lowest AIC will give us the best degree of freedom for variables in our B-spline model, which will lead us to the best-fitted B-spline model.

And for mean squared error (MSE):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2$$

where:

- n = number of data points
- Y_i = observed values
- $\hat{Y}_i$ = predicted values

3 Results

In this statistical analysis task, we are provided with the historical market data of real estate valuation from Sindian Dist., New Taipei City, Taiwan. This data set includes the housing prices with six numerical predictors representing different attributes of each house. We include all explanatory variables in our model without selection because of the relatively large sample size.

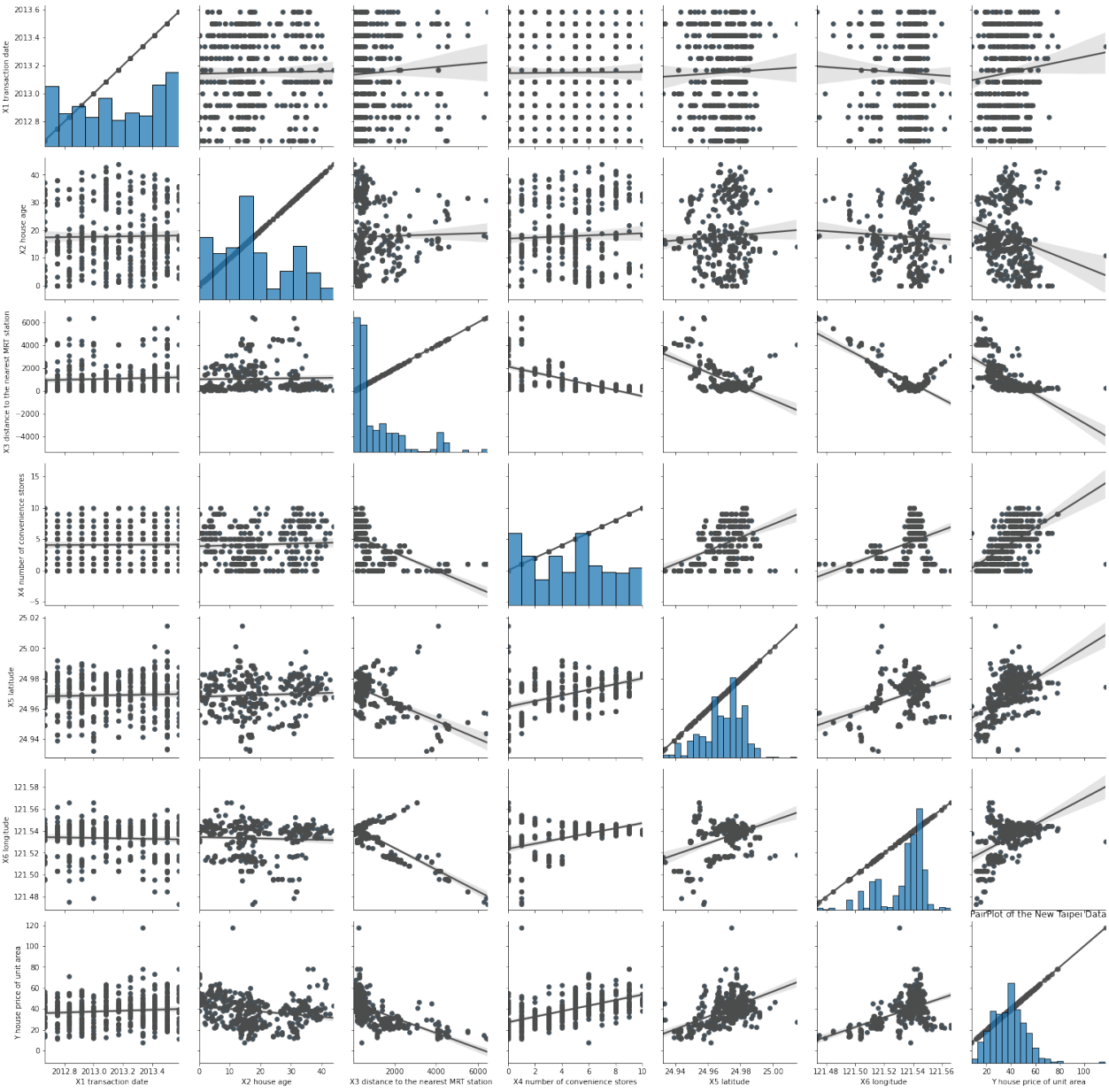


Figure 1: Pariplor of Different Variables

The data includes 414 observations and six predictors. And the basic statistics of the data are shown in Table 1. The response variable is the house price of the unit area. It shows that the response variable's mean(standard deviation) is 37.98(13.61). In addition, there are some interesting statistics for some predicting variables. Transaction date, latitude, and longitude have a very small standard deviation, while the distance to the nearest MRT station has an extremely large standard deviation.

Covariates	Min	25%	Median	75%	Max	Mean	Std
X1: transaction date	2012.67	2012.91	2013.16	2013.42	2013.58	2013.15	0.28
X2: house age	0.00	9.03	16.10	28.15	43.80	17.71	11.39
X3: distance to the nearest MRT station	23.38	289.32	492.23	1454.28	6488.02	1083.89	1262.11
X4: number of convenience stores	0.00	1.00	4.00	6.00	10.00	4.09	2.95
X5: latitude	24.93	24.96	24.97	24.98	25.01	24.97	0.01
X6: longitude	121.47	121.53	121.54	121.54	121.57	121.53	0.02
Y: house price of unit area	7.60	27.70	38.45	46.60	117.50	37.98	13.61

Table 1: Variable Summary

The distribution of the response variable price is shown in Figure 1. We noticed that the value of the response variable is mostly between 20 and 55 and is approximately normally distributed. On the other hand, the scatter plots between the response variable and the predictor are also shown in Figure 1. The plots indicated that for some predictors like house age, transaction date, and distance to the nearest MRT station, the simple linear regression might not capture the association between the response variable and predictor. However, for the number of convenience stores, there is an approximately linear association. These figures may indicate the necessity of using splines to capture non-linear associations.

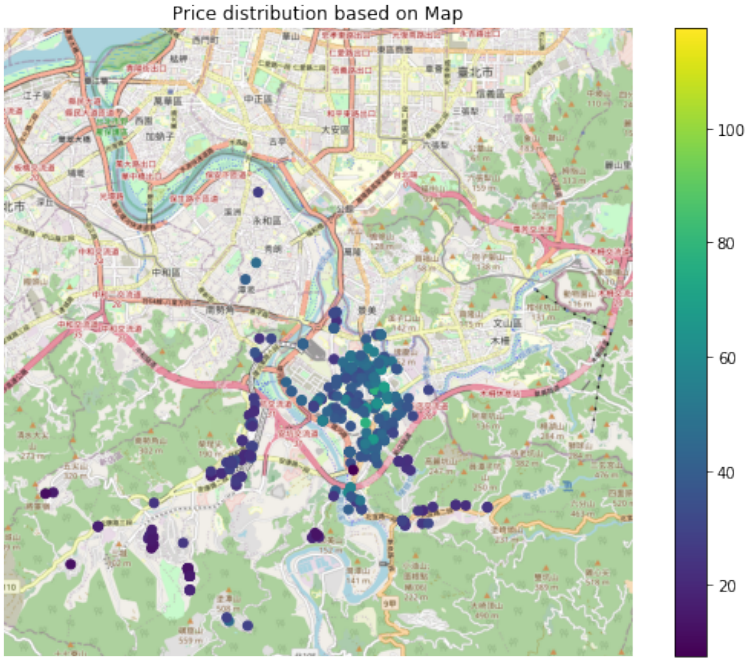


Figure 2: Real map plot

Next, we check the distribution of variable house prices of the unit area on the real map. Here we can see that the house price distribution is highly correlated with the house's location (can be get by variable longitude and latitude). The closer the house is to the city's center, the higher the house price of the unit area is.

Next, we start to build the B-spline model and use the iteration to determine the best degree of freedom of

each variable in the model. The model selection criterion is AIC. And the results of the best B-spline model we got from iteration are shown in Table 2.

Covariates	df
X1: transaction date	1
X2: house age	5
X3: distance to the nearest MRT station	7
X4: number of convenience stores	1
X5: latitude	5
X6: longitude	4

Table 2: Degree of freedom for each variable

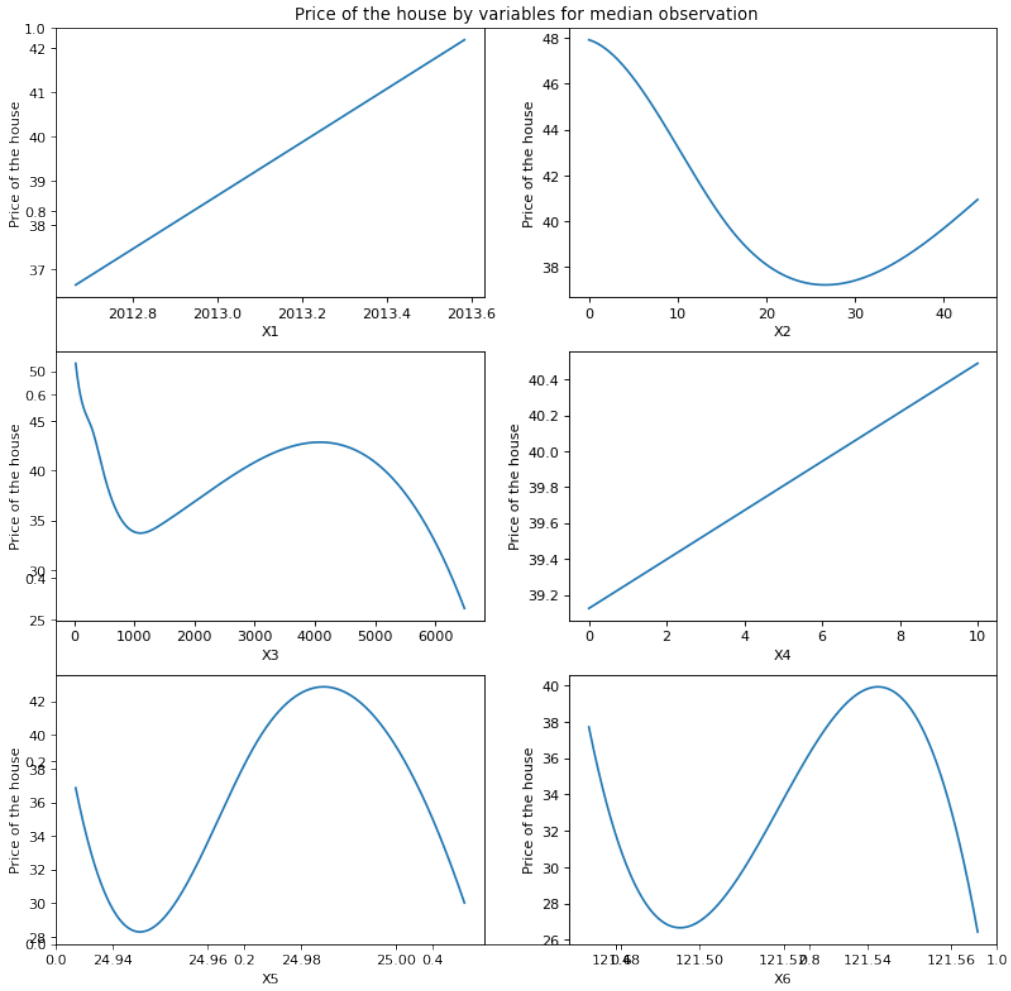


Figure 3: Result Between House Price of Unit Area and Predictors

The visual representation of our best spline model found in the above step is shown in the Figure 3. From here, we can have some interpolation on the variables.

- Both longitude(X6) and latitude(X5) have a shape of convex in the middle, this means in the middle of the area include in the data, the price is higher. This result consist with the result from the map plot.
- Both transection date (X1) and number of conveniece stores (X4) are positively linearly correlated with the price of the house.

- For house age(X2), the U-shape can be intepreted as the linear shape. So the Price of the house will decrease with the age of the house.
- It seems that the price of the house are negatively correlated with the distance to the MRT station(X3) in general.

In order to find an easy model to explain and with more appropriate model, we compare different models including general linear regression and general additive model (GAM) with B-splines. The performance of the model on the test data set is summarized in Table 3. Based on the MSEs of our two models, $47.01 < 69.49$, we have the result that our GAM with B-spline model is better than the GLM model.

Model	Mean Square Error on Test Data
Linear Regression (Base Model)	69.49
GAM with Bspline	47.07

Table 3: MSE of Different Models

Finally, we check the prediction function of our model using the Q-Q plot. Based on the Q-Q plot, we can have the prediction v.s. the real value in our test data set basically lies on the 45-degree line of the QQ plot, so our GAM with B-spline model did a great job in the prediction task.

Figure 4: Q-Q plot of the predicted price v.s. real price

4 Conclusion

This study aims to provide a statistical model for our client to assess the house price of unit area in Taipei, considering other variables. However, to keep the model interpretability, which can be informative to our client, We finally came up with a B-spline model, which treats variables differently concerning their value and successfully fulfilled our client's requirement.

However, since our model still takes the linear form, the prediction result may not be as accurate as other complicated models. This is the limitation of our model. However, our model gives sufficient information for

the underlying relationship between the prices and other variables. Our client can take a closer look at our model to investigate some influential variables. If a new case is encountered, our model can give a relatively accurate price estimation, and it can also tell how these variables affect the predicted price.

Also, our model is easy for future modifications. Certainly, the house price depends not only on the variables we have, but if we are provided with some new features that can be captured in the future, we can add them into our model and retrain it. Our model can use these additional features with little adjustments for more accurate results. For now, the example prediction process is shown in the end in the appendix.

HW4_Appendix

March 18, 2022

1 Appendix

First, we load the python packages that we are going to use.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from patsy import dmatrix
import seaborn as sns
import os
import smopy
import random
from sklearn.model_selection import train_test_split
from statsmodels.gam.api import GLMGam
import warnings
warnings.filterwarnings('ignore')
random.seed(615201)
```

Next, we load the data that we downloaded from the website.

[2]:
```
data = pd.read_excel("data.xlsx")
data.head()
```

[2]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores
0	1	2012.916667	32.0	84.87882	10
1	2	2012.916667	19.5	306.59470	9
2	3	2013.583333	13.3	561.98450	5
3	4	2013.500000	13.3	561.98450	5
4	5	2012.833333	5.0	390.56840	5

```
   X5 latitude  X6 longitude  Y house price of unit area
0     24.98298     121.54024                        37.9
1     24.98034     121.53951                        42.2
2     24.98746     121.54391                        47.3
3     24.98746     121.54391                        54.8
4     24.97937     121.54245                        43.1
```

Now, we start the basic data manipulation and analysis.

[3]:
```
# drop the column "No"
data = data.drop("No", axis=1)
# check if there is NaN
data.isnull().values.any()
```

[3]:
```
False
```

So, there is no NaN values in the dataframe

[4]:
```
print(f"The dimensions of the data is {data.shape}.")
data.describe()
```

```
The dimensions of the data is (414, 7).
```

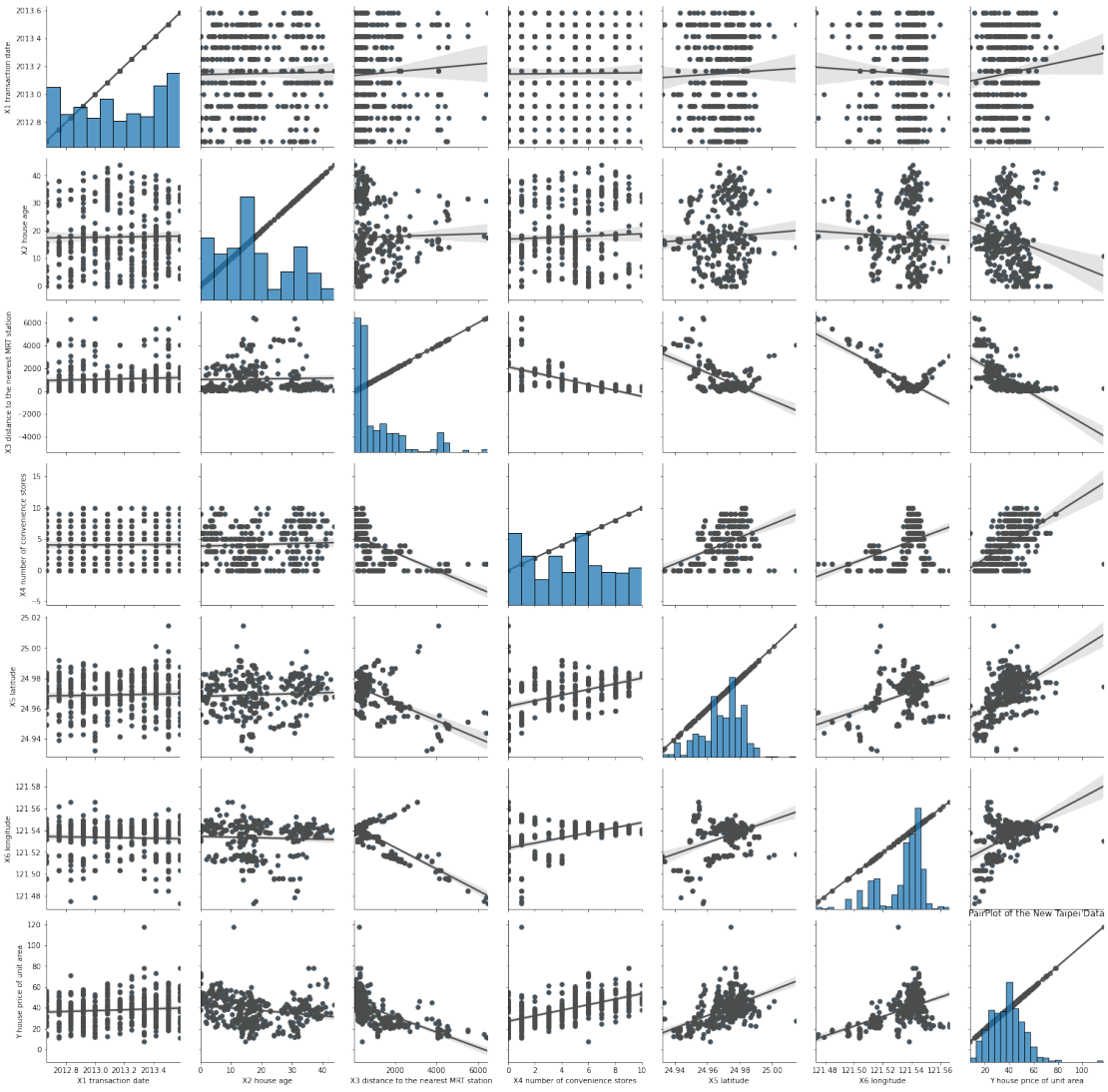
[4]:
```
       X1 transaction date  X2 house age  \
count           414.000000    414.000000
mean           2013.148953     17.712560
std               0.281995     11.392485
min            2012.666667      0.000000
25%            2012.916667      9.025000
50%            2013.166667     16.100000
75%            2013.416667     28.150000
max            2013.583333     43.800000

       X3 distance to the nearest MRT station  \
count                              414.000000
mean                              1083.885689
std                               1262.109595
min                                 23.382840
25%                                289.324800
50%                                492.231300
75%                               1454.279000
max                               6488.021000

       X4 number of convenience stores  X5 latitude  X6 longitude  \
count                       414.000000   414.000000    414.000000
mean                          4.094203    24.969030    121.533361
std                           2.945562     0.012410      0.015347
min                           0.000000    24.932070    121.473530
25%                           1.000000    24.963000    121.528085
```

```
50%                          4.000000    24.971100    121.538630
75%                          6.000000    24.977455    121.543305
max                         10.000000    25.014590    121.566270
```

```
       Y house price of unit area
count                   414.000000
mean                     37.980193
std                      13.606488
min                       7.600000
25%                      27.700000
50%                      38.450000
75%                      46.600000
max                     117.500000
```

Now, let's check the correlation between the variables.

[5]:
```
g = sns.PairGrid(data, height=3)
g.map_diag(sns.histplot)
g.map_offdiag(sns.scatterplot)
g.map(sns.regplot, color="0.3")
g.add_legend()
plt.title("PairPlot of the New Taipei Data")
plt.show()
```



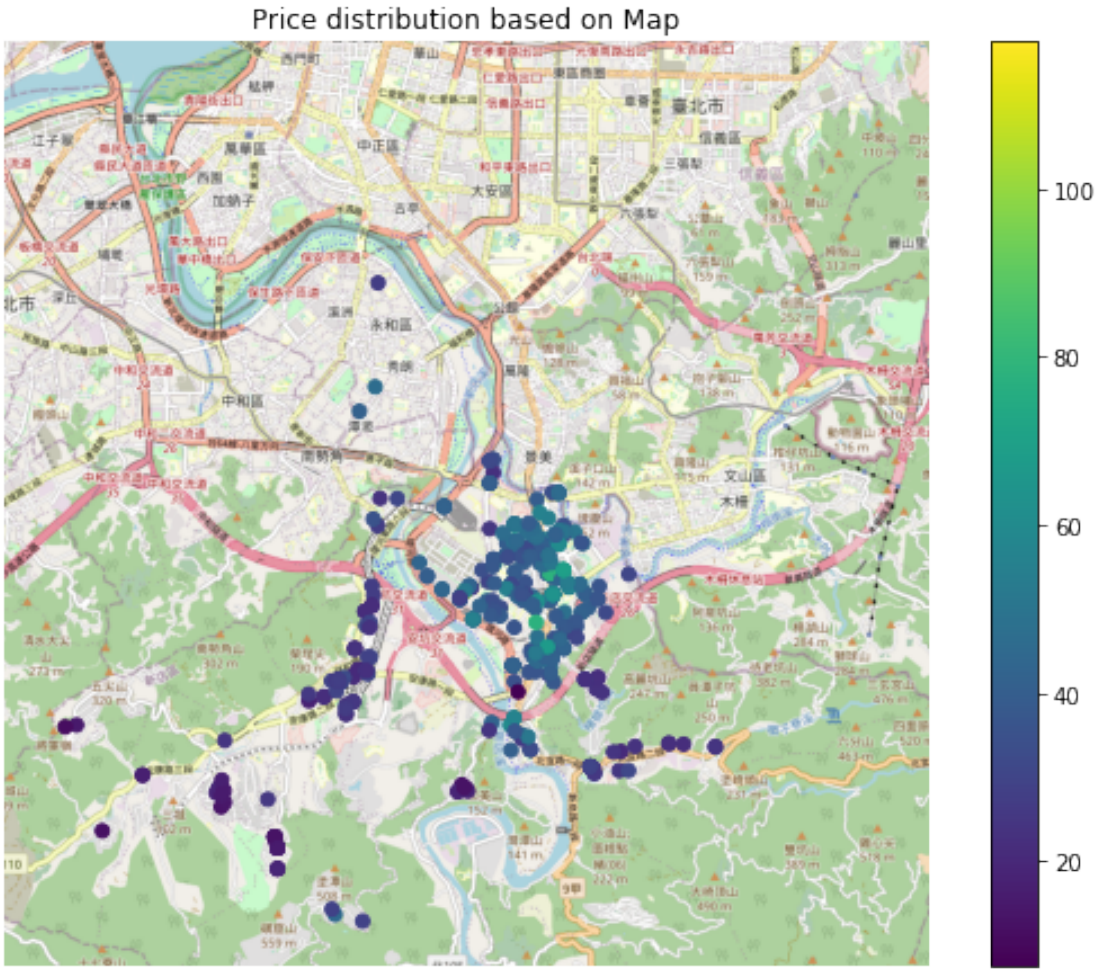
Next, lets check the price distribution based on the map in the real life.

[6]:
```
lat_max = data["X5 latitude"].max()
lon_max = data["X6 longitude"].max()
lat_min = data["X5 latitude"].min()
lon_min = data["X6 longitude"].min()
```

[7]:
```
base = smopy.Map((lat_min+0.005, lon_min+0.005, lat_max+0.005, lon_max+0.
↪005),z=13)
ax = base.show_mpl(figsize=(8, 6))
xl = []
yl = []
for i in range(len(data)):
```

```
    x, y = base.to_pixels(data["X5 latitude"].loc[i], data["X6 longitude"].
 ↪loc[i])
    xl.append(x)
    yl.append(y)
plt.title("Price distribution based on Map")
plt.scatter(xl, yl, c=data["Y house price of unit area"])
plt.colorbar()
```

[7]: <matplotlib.colorbar.Colorbar at 0x1ab78ed0940>



Now, we start the train-test split and prepare them further works.

[8]:
```
# reset column names
data = data.rename(columns = {"X1 transaction date":"X1", "X2 house age":"X2",
       "X3 distance to the nearest MRT station":"X3", "X4 number of convenience␣
 ↪stores":"X4",
```

```
      "X5 latitude":"X5", "X6 longitude":"X6",       "Y house price of unit␣
 ↪area":"Y"})
train, test = train_test_split(data, test_size=0.1, random_state=6150201)
```

Start the fitting of the forward stepwise spline and use AIC as the model selection criterion.

[9]:
```
def fstep_aic(dat, outcome, fixed):
    '''
    Forward stepwise spline fitting with AIC
    input:
        dat = pandas data frame
        outcome = dependent variable name
        fixed = list of tuples to keep fixed with (var_name, df)
    output: list of models with AIC
    '''
    model_data = []
    fmla = f'{outcome}~'
    vrbls = dat.columns.to_list()
    vrbls.remove(outcome)
    fixed_var = []
    fixed_df = []
    for var in fixed:
        if var[0] in vrbls:
            vrbls.remove(var[0])
            fmla = fmla + f'{var[0]}+'
        if var[1]>1:
            fixed_var.append(var[0])
            fixed_df.append(var[1])
    for var in vrbls:
        c_fmla = fmla + var
        if np.issubdtype(dat.dtypes[var], np.number):
            deg_free = [1,4,5,6,7]
        else:
            deg_free = [1]
        for free in deg_free:
            if any([x>1 for x in [free]+fixed_df]):
                if free>1:
                    c_vars = [var]+fixed_var
                    dfs = [free]+fixed_df
                else:
                    c_vars = fixed_var
                    dfs = fixed_df
                smoothing = sm.gam.BSplines(
                    dat[c_vars], df=dfs,
                    degree=(len(dfs))*[3])
                model = GLMGam.from_formula(
                    c_fmla, data=dat, smoother=smoothing,
```

```
                    family=sm.families.Gaussian())
            else:
                model = sm.GLM.from_formula(
                    c_fmla, data=dat,
                    family=sm.families.Gaussian())
            results = model.fit()
            c_model = {'Spline': var, 'DF': free,
                       'AIC': round(results.aic, 2)}
            model_data.append(c_model)
        model_aic = pd.DataFrame.from_records(model_data)
    return(model_aic.sort_values(by='AIC'))
```

[10]:
```
fixed_l = []
i = 0
while i < 6:
    data_choose = fstep_aic(train, 'Y', fixed=fixed_l)
    v = data_choose["Spline"].iloc[0]
    df = data_choose["DF"].iloc[0]
    d = (v, df)
    fixed_l.append(d)
    i += 1
data_choose
```

[10]:

	Spline	DF	AIC
0	X4	1	2581.40
1	X4	4	2584.45
2	X4	5	2586.44
3	X4	6	2588.43
4	X4	7	2589.96

Since there is no more variable, we can stop here, and the best AIC that comes from the spline model is 2581.40.

Now, we check the GLM model.

[11]:
```
fml = 'Y ~ X1 + X2 + X3 + X4 + X5 + X6'
model_base = sm.GLM.from_formula(fml, data=train, family=sm.families.
 ↪Gaussian()).fit()
print(f"The GLM model's AIC is {model_base.aic}")
model_base.summary()
```

```
The GLM model's AIC is 2690.646442599963
```

[11]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      Y   No. Observations:                  372
Model:                            GLM   Df Residuals:                      365
```

```
Model Family:                Gaussian   Df Model:                            6
Link Function:               identity   Scale:                          79.552
Method:                          IRLS   Log-Likelihood:                -1338.3
Date:               Fri, 18 Mar 2022   Deviance:                       29036.
Time:                       21:03:19   Pearson chi2:                 2.90e+04
No. Iterations:                    3
Covariance Type:           nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -1.369e+04   7167.811     -1.910      0.056   -2.77e+04     358.796
X1              5.2383      1.650      3.175      0.001       2.005       8.472
X2             -0.2789      0.041     -6.827      0.000      -0.359      -0.199
X3             -0.0046      0.001     -6.021      0.000      -0.006      -0.003
X4              1.1466      0.200      5.742      0.000       0.755       1.538
X5            212.4165     48.280      4.400      0.000     117.790     307.043
X6            -17.4144     51.376     -0.339      0.735    -118.110      83.281
==============================================================================
"""
```

Now, we check the intercept model.

[12]:
```
incpt_model = sm.GLM.from_formula('Y ~ 1', data=train,
family=sm.families.Gaussian()).fit()
print(f"Intercept Model, AIC: {np.round(incpt_model.aic,2)}")
```

```
Intercept Model, AIC: 3004.52
```

So, based on the AIC criterion, the spline model has a better fit on here than either the (linear) GLM and the intercept model.

Now, we start to check and evaluate the spline model we got.

[13]:
```
# get the degree of freedom of each variable
fixed_l
```

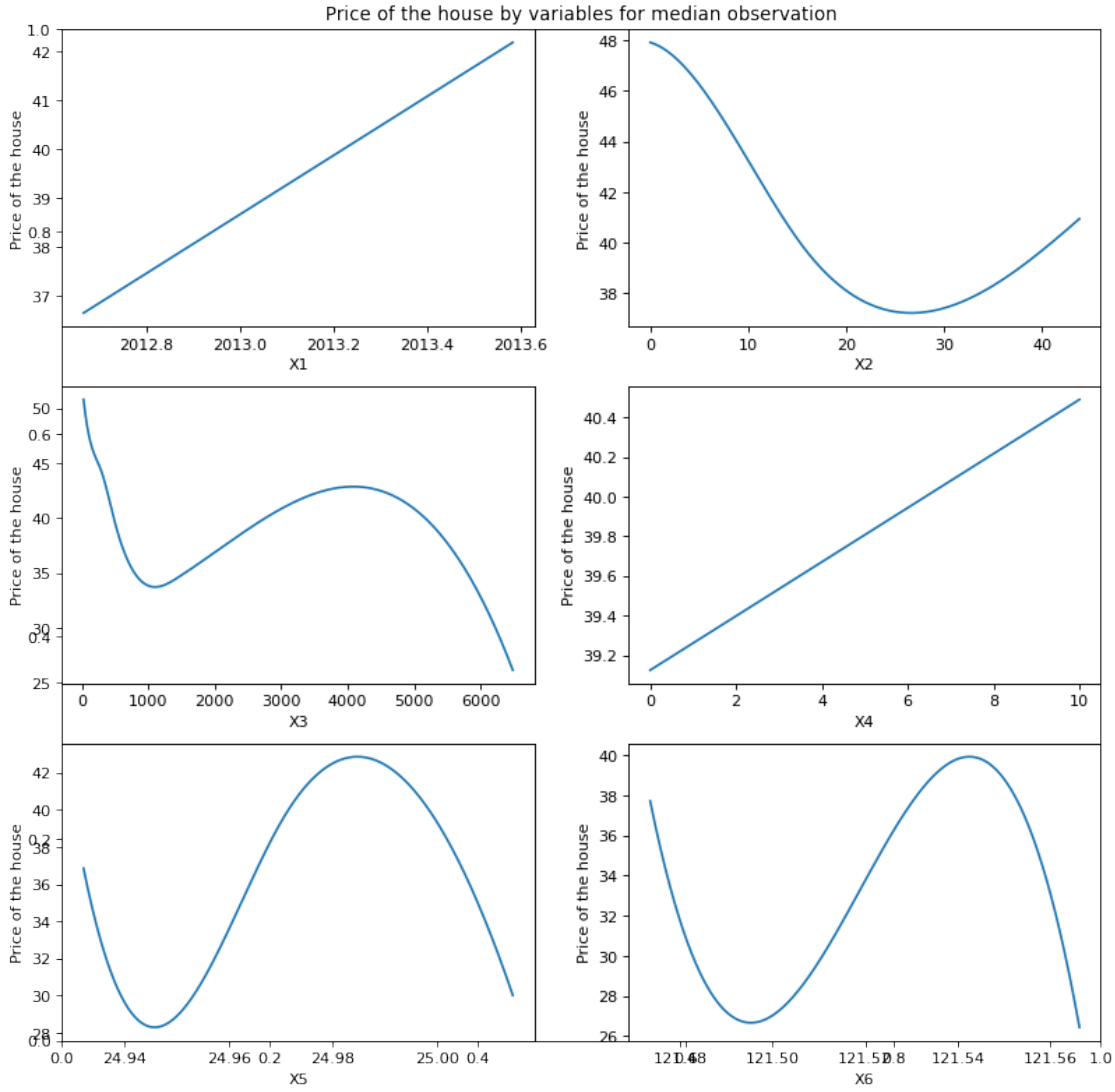
[13]:
```
[('X3', 7), ('X5', 5), ('X2', 5), ('X1', 1), ('X6', 4), ('X4', 1)]
```

[14]:
```
fmla = 'Y ~ X1 + bs(X2, 4) + bs(X3, 6) + X4 + bs(X5, 4) + bs(X6, 3)'
fsw_model = sm.GLM.from_formula(fmla, data=train, family=sm.families.Gaussian())
fsw_results = fsw_model.fit()
vrbls = ['X1', 'X2', 'X3', 'X4', 'X5', 'X6']
```

[15]:
```
def get_median_mode(dat, num):
    d_mat = {}
    for var in vrbls:
        if np.issubdtype(dat[var].dtype, np.number):
            d_mat[var] = dat[var].median()
        else:
```

```
            d_mat[var] = dat[var].mode()
    d_mat = pd.DataFrame([d_mat], columns = d_mat.keys())
    return(sm.add_constant(pd.concat([d_mat]*num)))

def plot_spline(var, plt, fsw_results, dat):
    if np.issubdtype(dat[var].dtype, np.number):
        num = 500
        d_mat = get_median_mode(dat, num)
        d_mat[var] = np.linspace(dat[var].min(), dat[var].max(), num)
        y = fsw_results.predict(exog = d_mat)
        plt.plot(d_mat[var], y)
    else:
        d_mat = get_median_mode(dat, dat[var].nunique())
        d_mat[var] = dat[var].unique()
        y = fsw_results.predict(exog = d_mat)
        plt.bar(d_mat[var], y)
    plt.set(xlabel=var, ylabel='Price of the house')
```

[16]:
```
fig=plt.figure(figsize=(12, 12), dpi=80)
plt.title(f'Price of the house by variables for median observation')
for itr, var in enumerate(vrbls):
    ax=fig.add_subplot(3, 2, itr+1)
    plot_spline(var, ax, fsw_results, train)
```



Here we plotting splines as prices, each plot holds all other variable median values constant while varying one variable. In this way we can have an overall view of how each variable influence the price of the house.

Here each variable stands for:

- "X1 transaction date":"X1"
- "X2 house age":"X2",
- "X3 distance to the nearest MRT station":"X3"
- "X4 number of convenience stores":"X4"
- "X5 latitude":"X5"
- "X6 longitude":"X6"

So, here, we can see that:

- Both longitude and latitude have a shape of convex, this means in the middle of the area

include, the price is higher. This result consist with the result from the map plot.

- Both transection date and number of conveniece stores are positively correlated with the price of the house.
- For house age, the U-shape can be intepreted as the linear shape. So the Price of the house will decreas with the age of the house.
- It seems that the price of the house are negatively correlated with the distance to the MRT station in general.

Next, we test the prediction power of the model.

[17]:
```
real_test = list(test["Y"])
test_input = test.drop("Y", axis=1)
```

[18]:
```
prediction_list = []
count = 0
for i in range(len(test)):
    input = dict(test_input.iloc[i])
    pred = fsw_results.predict(input).values[0]
    prediction_list.append(pred)
```

First, we check about the MSE of our spline model.

[19]:
```
from sklearn.metrics import mean_squared_error
import math
print(mean_squared_error(real_test, prediction_list))
```

```
47.01749877908336
```

Check the MSE of the GLM model.

[20]:
```
pred_list = []
count = 0
for i in range(len(test)):
    input = dict(test_input.iloc[i])
    pred = model_base.predict(input).values[0]
    pred_list.append(pred)
```

[21]:
```
print(mean_squared_error(real_test, pred_list))
```

```
69.49220810125891
```

We can use the plot to check the prediction result of the model.

Based on the MSEs of our two models, $47.01 < 69.49$, we have again confirmed that our spline model is better than the GLM model.

[22]:
```
fig=plt.figure(figsize=(5, 5), dpi=80)
plt.scatter(real_test, prediction_list)
plt.plot([0,75], [0,75], color = 'black', linewidth = 2)
plt.title("Q-Q plot of the predicted price v.s. real price")
plt.show()
```

Based on the QQ plot, we can have that the prediction v.s. the real value in our test data set basicly lies on the 45 degree line of the QQ plot, so our model did a great job in the predction task.

Here is the prediction example:

[23]:
```
# new data input
housing_info = {'X1': 2013.3333333,
                'X2': 19.2,
                'X3': 383.7129,
                'X4': 8.0,
                'X5': 24.972,
                'X6': 121.54477}
expexted_cost = fsw_results.predict(input).values[0]
print(f"The expected cost of the new give housing information is␣
 ↪{expexted_cost}.")
```

```
The expected cost of the new give housing information is 33.86411998479687.
```

[ ]: