Real Estate Valuation

Data Science Project

Real estate valuation is the process of determining the value of real estate property.

Some factors affecting the value of houses are location, home size and useable space, age and condition of the house, and availability of social amenities.

By examining real estate valuation, we understand the reasons for people's choice when purchasing a real estate property

Details on the dataset

Title: Real estate valuation data set

The dataset used is of market historical dataset of real estate valuation collected from Sindian District, New Taipei City, Taiwan.

The price per unit is based on a unit measure of 3.3 metres squared. There are 414 instances with 7 attributes.

Sources

- Original Owner and Donor:
  - Name: Prof. I-Cheng Yeh
  - Institutions: Department of Civil Engineering, Tamkang University, Taiwan.
- UCI Irvine Machine Learning Repository
- https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set

Description of columns

- House no: House number
- Transaction date: Date when the purchase was done
- House age(years): The age of the house in years
- MRT station(metres): Distance from MRT station in metres
- Convenience stores: Number of convenience stores around
- Latitude: Geographical coordinate
- Longitude: Geographical coordinate
- Unit price: House price of unit area(3.3 metres squared)

Problem Statement.

To develop a model that can be used to predict real estate value.

1. Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sb
import plotly.express as px
from sklearn import metrics
from sklearn import tree
from sklearn.linear_model import Ridge, LinearRegression, ElasticNet
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor, AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.ensemble import BaggingRegressor, AdaBoostRegressor


import missingno as msno
%matplotlib inline
plt.style.use('ggplot')
import warnings
warnings.filterwarnings('ignore')
print('Import Libraries-Done')
```

```
Import Libraries-Done
```

2. Load the dataset

In [4]:
```
valuation = pd.read_excel('Real estate valuation data set.xlsx')
valuation.head()
```

Out[4]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.916667	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.916667	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583333	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500000	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833333	5.0	390.56840	5	24.97937	121.54245	43.1

3. Exploratory Data Analysis

In [5]:
```
#renaming the columns
valuation=valuation.rename(columns={'No':'House no','X1 transaction date':'Transaction date','X2 house age':'House age(years)','
```

In [6]:
```
valuation.head()
```

Out[6]:

	House no	Transaction date	House age(years)	MRT station(metres)	Convenience stores	Latitude	Longitude	Unit price
0	1	2012.916667	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.916667	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583333	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500000	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833333	5.0	390.56840	5	24.97937	121.54245	43.1

In [7]:
```
valuation.tail()
```

Out[7]:

	House no	Transaction date	House age(years)	MRT station(metres)	Convenience stores	Latitude	Longitude	Unit price
409	410	2013.000000	13.7	4082.01500	0	24.94155	121.50381	15.4
410	411	2012.666667	5.6	90.45606	9	24.97433	121.54310	50.0
411	412	2013.250000	18.8	390.96960	7	24.97923	121.53986	40.6
412	413	2013.000000	8.1	104.81010	5	24.96674	121.54067	52.5
413	414	2013.500000	6.5	90.45606	9	24.97433	121.54310	63.9

In [10]:
```
valuation.dtypes
```

Out[10]:
```
House no                 int64
Transaction date       float64
House age(years)       float64
MRT station(metres)    float64
Convenience stores       int64
Latitude               float64
Longitude              float64
Unit price             float64
dtype: object
```

In [12]:
```
valuation.shape
```

Out[12]:
```
(414, 8)
```

In [13]:
```
valuation.describe().T
```

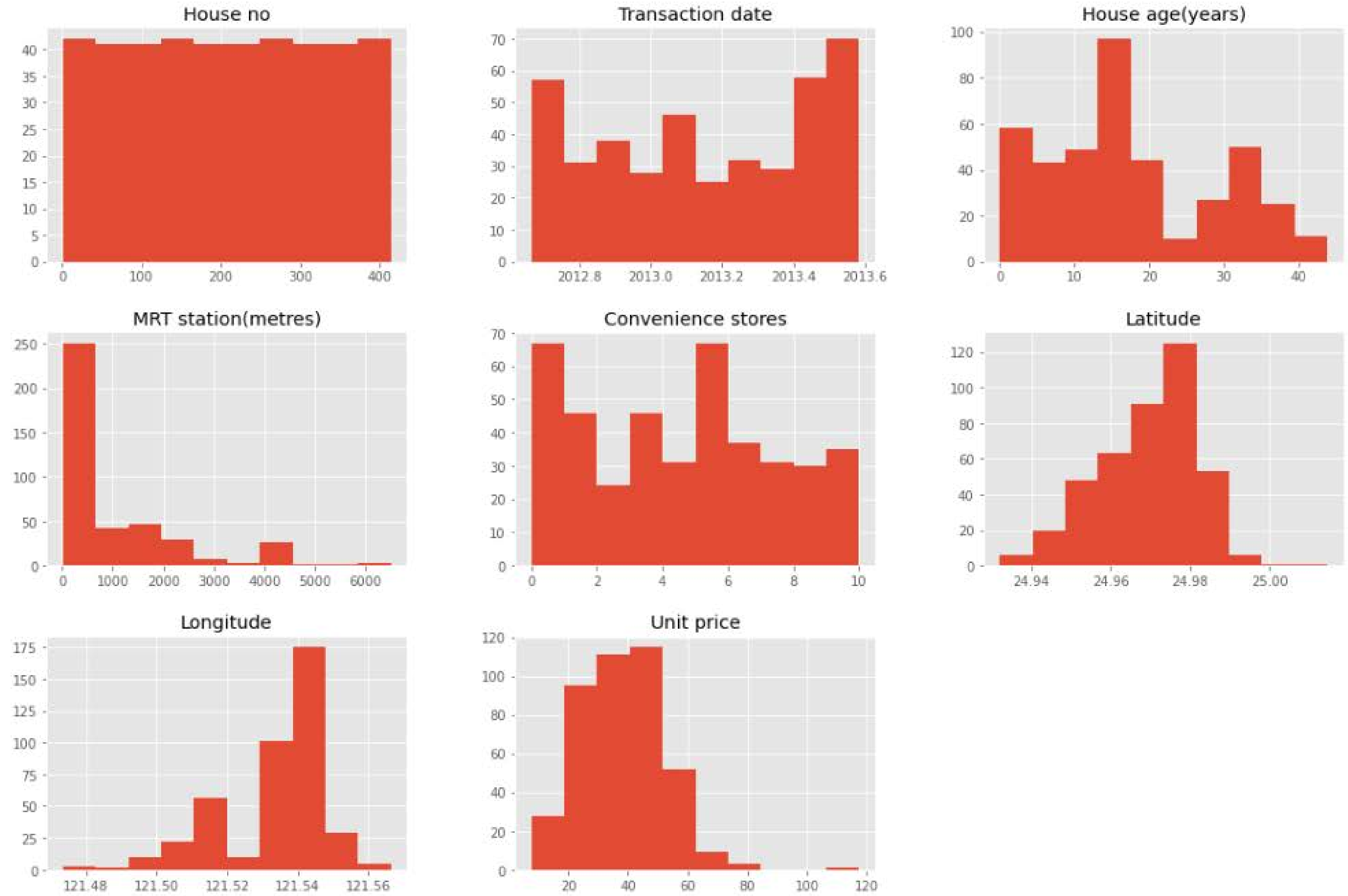
Out[13]:

	count	mean	std	min	25%	50%	75%	max
House no	414.0	207.500000	119.655756	1.000000	104.250000	207.500000	310.750000	414.000000
Transaction date	414.0	2013.148953	0.281995	2012.666667	2012.916667	2013.166667	2013.416667	2013.583333
House age(years)	414.0	17.712560	11.392485	0.000000	9.025000	16.100000	28.150000	43.800000
MRT station(metres)	414.0	1083.885689	1262.109595	23.382840	289.324800	492.231300	1454.279000	6488.021000
Convenience stores	414.0	4.094203	2.945562	0.000000	1.000000	4.000000	6.000000	10.000000
Latitude	414.0	24.969030	0.012410	24.932070	24.963000	24.971100	24.977455	25.014590
Longitude	414.0	121.533361	0.015347	121.473530	121.528085	121.538630	121.543305	121.566270
Unit price	414.0	37.980193	13.606488	7.600000	27.700000	38.450000	46.600000	117.500000

In [14]:
```
#check for missing values
valuation.isnull().any()
```

Out[14]:
```
House no               False
Transaction date       False
House age(years)       False
MRT station(metres)    False
Convenience stores     False
Latitude               False
Longitude              False
Unit price             False
dtype: bool
```

In [18]:
```
valuation.hist(figsize=(18,12))
plt.show()
```



In [16]:
```
fig=px.scatter_mapbox(valuation,lat='Latitude',lon='Longitude',hover_name='Unit price',color='Unit price',zoom=15,mapbox_style='

fig.show()
```

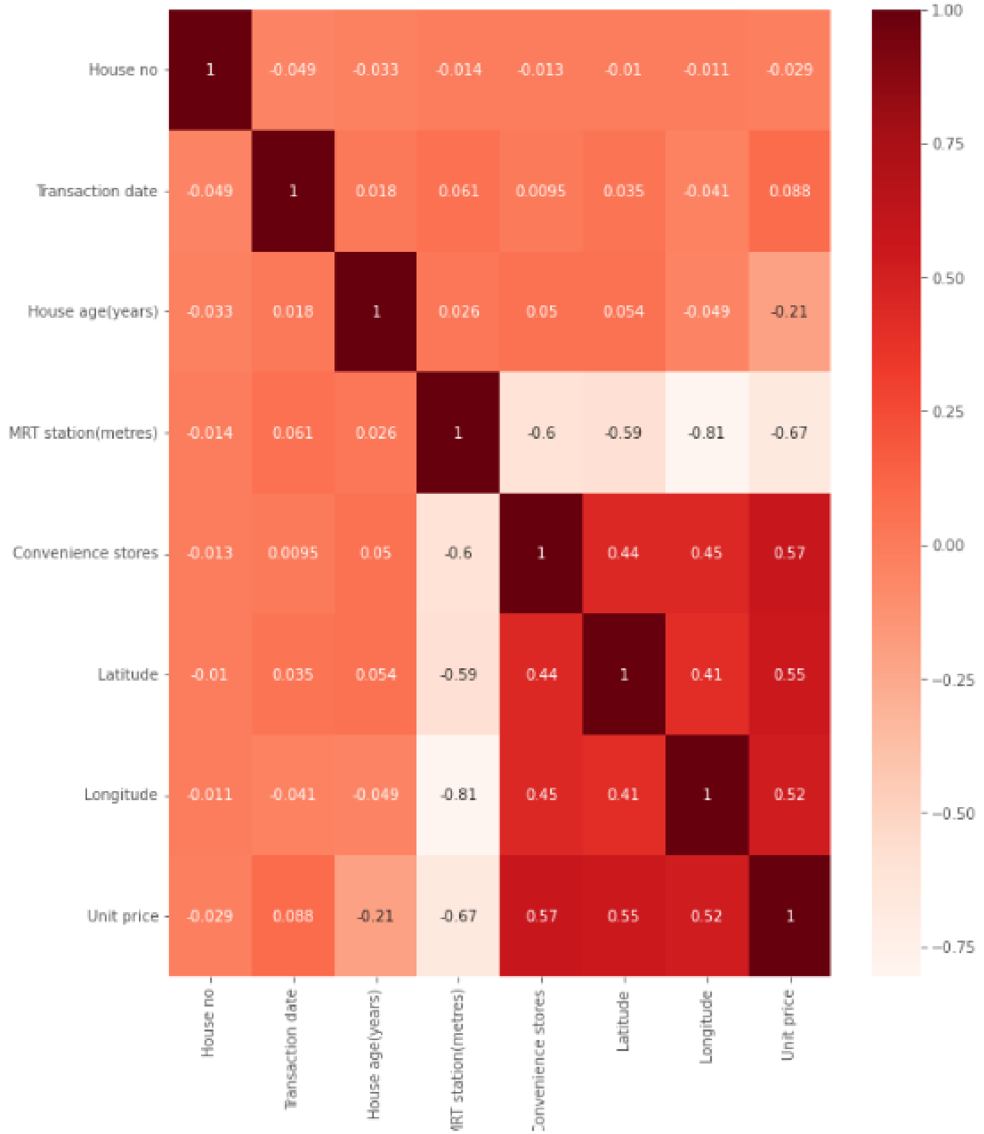
Observation

- The house ages range from 0 to over 40 years. Majority of the houses range from 10 years to 20 years.
- Most houses are within 1000 metres(1 kilometre) to the MRT station.
- The houses in this region are not evenly distributed with most distribution on the eastern side of the town.
- The western side of the town has houses of lower price per unit area as compared to the eastern side.

4. Feature Engineering

In [17]:
```
#use Pearson correlation

plt.figure(figsize=(10,12))
cor = valuation.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



```
In [18]: #correlation between the variables
         cor_target=abs(cor['Unit price'])
         cor_target
```

```
Out[18]: House no                 0.028587
         Transaction date         0.087529
         House age(years)         0.210567
         MRT station(metres)      0.673613
         Convenience stores       0.571005
         Latitude                 0.546307
         Longitude                0.523287
         Unit price               1.000000
         Name: Unit price, dtype: float64
```

```
In [19]: #selecting highly correlated features
         relevant_feature =cor_target[cor_target>0.5]
         relevant_feature
```

```
Out[19]: MRT station(metres)      0.673613
         Convenience stores       0.571005
         Latitude                 0.546307
         Longitude                0.523287
         Unit price               1.000000
         Name: Unit price, dtype: float64
```

The variables that are highly correlated to our target column (Unit price) are:

- MRT station(metres)
- Convenience stores
- Latitude
- Longitude

These variables will have an effect on the prediction of Real Estate Valuation

5. Modelling

Train-Test-Split Data

In [20]:
```
x = valuation.drop(columns='Unit price')
y = valuation['Unit price']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

In [21]:
```
x_train.head()
```

Out[21]:

	House no	Transaction date	House age(years)	MRT station(metres)	Convenience stores	Latitude	Longitude
302	303	2013.500000	16.5	2288.0110	3	24.95885	121.51359
20	21	2013.416667	4.5	2275.8770	3	24.96314	121.51151
303	304	2013.500000	38.3	439.7105	0	24.97161	121.53423
142	143	2013.416667	19.8	640.6071	5	24.97017	121.54647
14	15	2013.500000	13.2	1164.8380	4	24.99156	121.53406

In [22]:
```
y_test.head()
```

Out[22]:
```
356    45.3
170    14.4
224    46.0
331    15.6
306    50.2
Name: Unit price, dtype: float64
```

Creation and model selection

In [23]:
```
scaler = StandardScaler()

X_train = scaler.fit_transform(x_train)
X_test = scaler.transform(x_test)
```

In [24]:
```
regular_reg = ElasticNet()
lin_reg = LinearRegression()
ridge = Ridge()
rf_reg= RandomForestRegressor(random_state=0)
dt_reg = DecisionTreeRegressor(random_state = 0)
bag_reg = BaggingRegressor(random_state = 0)
boost_reg = AdaBoostRegressor(random_state = 0)

models = {'ElasticNet': regular_reg,'LinearRegression':lin_reg,'Ridge':ridge, 'RandomForestRegressor':rf_reg, 'DecisionTreeRegres
          'AdaBoostRegressor': boost_reg}
```

In [25]:
```
def cross_valid(models, X, y, process = 'Training'):
    print(f'Process: {process}')
    for model_name, model in models.items():
        scores = cross_val_score(model, X, y, cv = 5)
        print(f'Model: {model_name}')
        print(f'Cross validation mean score: {round(np.mean(scores), 4)}')
        print(f'Cross validation deviation: {round(np.std(scores), 4)}')
        print('\n')
```

```
In [26]: cross_valid(models, X_train, y_train, process = 'Training')
```

```
Process: Training
Model: ElasticNet
Cross validation mean score: 0.5162
Cross validation deviation: 0.1058


Model: LinearRegression
Cross validation mean score: 0.5487
Cross validation deviation: 0.1187


Model: Ridge
Cross validation mean score: 0.549
Cross validation deviation: 0.1188


Model: RandomForestRegressor
Cross validation mean score: 0.6877
Cross validation deviation: 0.1146


Model: DecisionTreeRegressor
Cross validation mean score: 0.441
Cross validation deviation: 0.1574


Model: BaggingRegressor
Cross validation mean score: 0.6735
Cross validation deviation: 0.1049


Model: AdaBoostRegressor
Cross validation mean score: 0.6438
Cross validation deviation: 0.1102
```

```
In [27]: cross_valid(models, X_test, y_test, process = 'Testing')
```

```
Process: Testing
Model: ElasticNet
Cross validation mean score: 0.5654
Cross validation deviation: 0.1576


Model: LinearRegression
Cross validation mean score: 0.6039
Cross validation deviation: 0.1647


Model: Ridge
Cross validation mean score: 0.6045
Cross validation deviation: 0.1651


Model: RandomForestRegressor
Cross validation mean score: 0.6663
Cross validation deviation: 0.082


Model: DecisionTreeRegressor
Cross validation mean score: 0.4231
Cross validation deviation: 0.2053


Model: BaggingRegressor
Cross validation mean score: 0.6415
Cross validation deviation: 0.1238


Model: AdaBoostRegressor
Cross validation mean score: 0.5767
Cross validation deviation: 0.1554
```

In [28]:
```
rf_reg.fit(x_train, y_train)
```
Out[28]:
```
RandomForestRegressor(random_state=0)
```

In [29]:
```
def model_evaluation(model, X, y):
    y_predict = np.exp(model.predict(X))
    y = np.exp(y)
    print(f'Mean Squared Error: {mean_squared_error(y, y_predict)}')
    print(f'Mean Absolute Error: {mean_absolute_error(y, y_predict)}')
    print(f'R2 Score: {r2_score(y, y_predict)}')
```

In [30]:
```
model_evaluation(rf_reg, x_train, y_train)
```
```
Mean Squared Error: 3.462384169764902e+99
Mean Absolute Error: 3.234250320436159e+48
R2 Score: -0.0030303029962064354
```

In [31]:
```
model_evaluation(rf_reg, x_test, y_test)
```
```
Mean Squared Error: 1.2343485258950025e+66
Mean Absolute Error: 1.2198107404772023e+32
R2 Score: -0.012195206299479011
```

Random Forest Regressor is the most suitable model for prediction

In [32]:
```
#save the model

import pickle
pickle.dump(rf_reg, open('Random_Forest_Regression.pkl', 'wb'))
```

Conclusion

The value of a house is affected by Latitude and Longitude(location), how far it is from the MRT station and the number of convenience stores around.

House age(years) and Transaction date have an effect but are not significant.