# EDA - Understanding Data

Module 1 | Chapter 1 | Notebook 1

---

In this notebook we will explore the data that we will use in this chapter for linear regressions. In Data Science jargon, this process is called **Exploratory Data Analysis** or **EDA** for short. Like all regressions, linear regression is used to predict continuous values. This could be revenue, user numbers or even machine temperatures. In our case it's house prices. At the end of this notebook you will have a good understanding of the data for this chapter.

---

## Importing and understanding data

**Scenario:** A Taiwanese investor comes to you to find out how much his properties in Taiwan are actually worth. He may want to resell them and needs an estimate of the price he can charge for his ten properties. Using a data set from 2012/2013, you will train a prediction model that will help the investor. Your training set contains the data from 314 houses.

To look at the training data, first import `pandas` with its conventional alias `pd`.

In [1]:
```
import pandas as pd
```

The data is stored in an excel file called *Taiwan_real_estate_training_data.xlsx*. Import it using `pandas.read_excel()`, store it in a `DataFrame` named `df` and define the `'No'` column as the column containing the row names ( `index_col` parameter). Then use `my_df.head()` to get an impression of the data.

In [4]:
```
df = pd.read_excel('Taiwan_real_estate_training_data.xlsx', index_col='No')
df.head()
```

Out[4]:

| No | X1 house age | X2 distance to the nearest MRT station | X3 number of convenience stores | X4 number of parking spaces | X5 air pollution | X6 light pollution | X7 noise pollution | X8 neighborhood quality | cri sc |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 32.0 | 84.87882 | 10 | 89 | 29.754370 | 197.289414 | 0.852160 | 0.348743 | 0.593 |
| **2** | 19.5 | 306.59470 | 9 | 99 | 111.751859 | 179.272296 | 11.394151 | 0.279919 | 0.679 |
| **3** | 13.3 | 561.98450 | 5 | 79 | 394.335266 | 310.258310 | 13.691476 | 0.518158 | 0.585 |
| **4** | 13.3 | 561.98450 | 5 | 82 | 411.776028 | 273.979285 | 7.798633 | 0.431828 | 0.785 |
| **5** | 5.0 | 390.56840 | 5 | 72 | 98.966440 | 223.585153 | 12.628267 | 0.371121 | 0.752 |

The data dictionary for this data is as follows:

| Column number | Column name | Type | Description |
|---|---|---|---|
| 0 | `'house_age'` | continuous ( `float` ) | age of the house in years |
| 1 | `'metro_distance'` | continuous ( `float` ) | distance in meters to the next metro station |
| 2 | `'number_convenience_stores'` | continuous ( `int` ) | Number of convenience stores nearby |
| 3 | `'number_parking_spaces'` | continuous ( `int` ) | Number of parking spaces nearby |
| 4 | `'air_pollution'` | continuous ( `float` ) | Air pollution value near the house |
| 5 | `'light_pollution'` | continuous ( `float` ) | Light pollution value near the house |
| 6 | `'light_pollution'` | continuous ( `float` ) | Light pollution value near the house |
| 7 | `'neighborhood_quality'` | continuous ( `float` ) | average quality of life in the neighborhood |
| 8 | `'crime_score'` | continuous ( `float` ) | crime score according to police |
| 9 | `'energy_consumption'` | continuous ( `float` ) | The property's energy consumption |
| 10 | `'longitude'` | continuous ( `float` ) | The property's longitude |
| 11 | 'price_per_ping' | continuous (`float') | House price in Taiwan dollars per ping, one ping is 3.3 $m^2$ |

The column names in the data dictionary are different o those in the Excel file. They are summarized in `col_names`.

In [6]:
```
col_names = ['house_age',
             'metro_distance',
             'number_convenience_stores',
             'number_parking_spaces',
             'air_pollution',
             'light_pollution',
             'noise_pollution',
             'neighborhood_quality',
             'crime_score',
             'energy_consumption',
             'longitude',
             'price_per_ping']
```

Change the column names in `df` accordingly. They should be the same as the names in the data dictionary.

In [7]:
```
df.columns = col_names
```

Now that we have correctly imported the data, we can start checking it. First we should always check whether the data is clean. That means answering the following questions:

- Do all the columns have the right name?
- Do all the columns have the right data types?
- Are there any missing values?

You can answer all these questions with just one command: `my_df.info()` Execute the command and assess whether the data is clean.

In [13]:
```
df.info()
df.isna().sum()
```

Out[13]:
```
house_age                    0
metro_distance               0
number_convenience_stores    0
number_parking_spaces        0
air_pollution                0
light_pollution              0
noise_pollution              0
neighborhood_quality         0
crime_score                  0
energy_consumption           0
longitude                    0
price_per_ping               0
dtype: int64
```

Next we should check whether they have meaningful values. Display the 8 value summary of each column using `my_df.describe()`. Do you notice any unusual values?

In [15]:
```
df.describe()
```

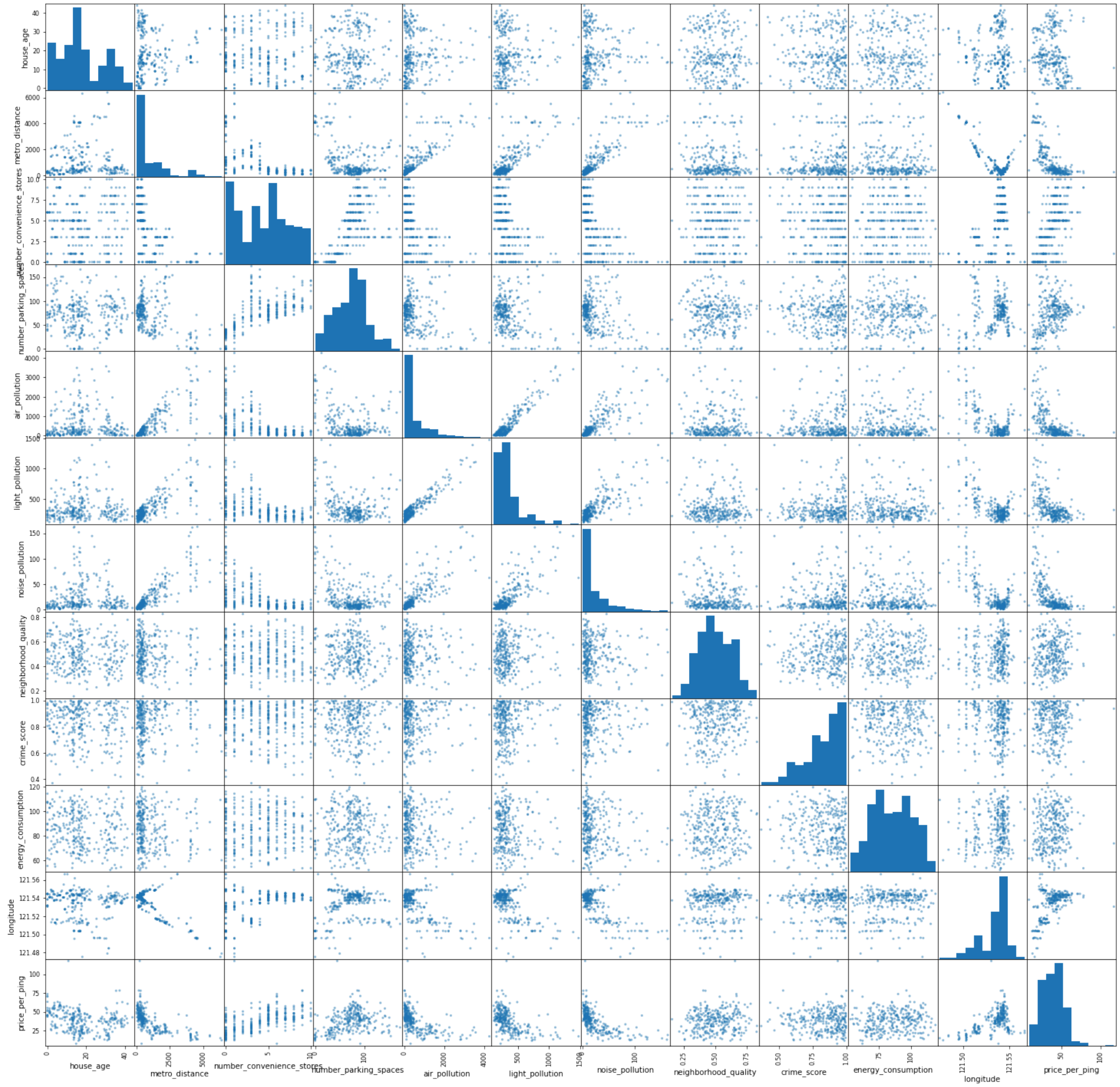
Out[15]:

| | house_age | metro_distance | number_convenience_stores | number_parking_spaces | air_pollution |
|---|---|---|---|---|---|
| **count** | 314.000000 | 314.000000 | 314.000000 | 314.000000 | 314.000000 |
| **mean** | 17.964968 | 1080.554101 | 4.073248 | 72.544586 | 552.604234 |
| **std** | 11.302234 | 1265.305662 | 2.917018 | 33.956459 | 724.980887 |
| **min** | 0.000000 | 23.382840 | 0.000000 | 0.000000 | 0.129046 |
| **25%** | 10.025000 | 292.997800 | 1.000000 | 47.250000 | 105.270986 |
| **50%** | 16.250000 | 492.231300 | 4.000000 | 74.000000 | 273.298306 |
| **75%** | 28.350000 | 1412.735250 | 6.000000 | 91.000000 | 650.921839 |
| **max** | 43.800000 | 6396.283000 | 10.000000 | 173.000000 | 4257.231953 |

The data seems plausible. The `'count'` row shows `314` values. Since there are also 314 row in `df`, this means that there are no missing values. Otherwise, the `'min'` and `'max'` values of each column also look realistic.

Now let's have a visual look at the distribution of the individual columns to identify outliers and things like that. Use the `pandas.plotting.scatter_matrix()` function, or the `seaborn.pairplot()` function. Pass them both the `DataFrame` `df` and they will draw histograms of each column and scatter plots of all column combinations.

In [24]:
```
pd.plotting.scatter_matrix(df,  figsize=(25,25));
```




The price of real estate uses the local area measurement, *ping*. This is not used in the West. Convert the prices per ping into prices per square meter. Store the converted values in a new column `'price_per_m2'` which you can add to `df`. The conversion formula is as follows:

\begin{equation*} House price\, \ per\, \ square meter = \frac{house price\, \ per\, \ ping}{3.3} \end{equation*}

In [25]:
```
df.loc[:,'price_per_m2'] = df.loc[:,'price_per_ping']/3.3
df.head()
```

Out[25]:

| No | house_age | metro_distance | number_convenience_stores | number_parking_spaces | air_pollution | ligl |
|---|---|---|---|---|---|---|
| 1 | 32.0 | 84.87882 | 10 | 89 | 29.754370 | |
| 2 | 19.5 | 306.59470 | 9 | 99 | 111.751859 | |
| 3 | 13.3 | 561.98450 | 5 | 79 | 394.335266 | |
| 4 | 13.3 | 561.98450 | 5 | 82 | 411.776028 | |
| 5 | 5.0 | 390.56840 | 5 | 72 | 98.966440 | |

**Congratulations:** You have cleaned the real estate investor's data. Unusually, you don't need to clean this data. This means we can turn to a simple linear regression next. You will get a first impression of the Python module `sklearn`.

**Remember:**

- Before you start using *machine learning*, it's important to explore and clean the data.
- Regressions are used to predict continuous data.

---

Do you have any questions about this exercise? Look in the forum to see if they have already been discussed.

---

Found a mistake? Contact Support at support@stackfuel.com.

---