# Chapter 2
# End to End Machine Learning

DuLi

2019 年 3 月 27 日

# 目录

**1 Outline** **3**

**2 Working with Real Data** **3**

**3 Look at the Big Picture** **4**

3.1 Frame the Problem . . . . . 4

3.2 Select a Performance Measure . . . . . 5

**4 Get the data** **6**

4.1 Create the workspace . . . . . 6

4.2 Take a Quick Look at the Data Structure . . . . . 6

4.3 Create a Testset . . . . . 9

**5 Discover and Visualize the Data ti Gain Insights** **12**

5.1 Visualizing Geographical Data . . . . . 12

5.2 Looking for Correlations . . . . . 14

# 1 Outline

Here are main steps you will go through:

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data fpr machine learning algorithms.
5. Select a model and train it.
6. Fine-tune your model.
7. Present your solution.
8. Launch,monitor,and maintain your system.

# 2 Working with Real Data

Here are a afew places you can look to get data:

- Popular open data open repositories:

- UC Irvine Machine Learning Repositories.

- Kaggle datasets.

- Amazon's AWS datasets.

- Meta Portals(they list open data repositories)

- dataportals.org

- opendatamonitor.eu

- quandl.com

- Other pages listing many popular open data repositories

- Wikipedia's list of Machine Learning datasets.
- Quora.com question.
- Datasets subreddit.

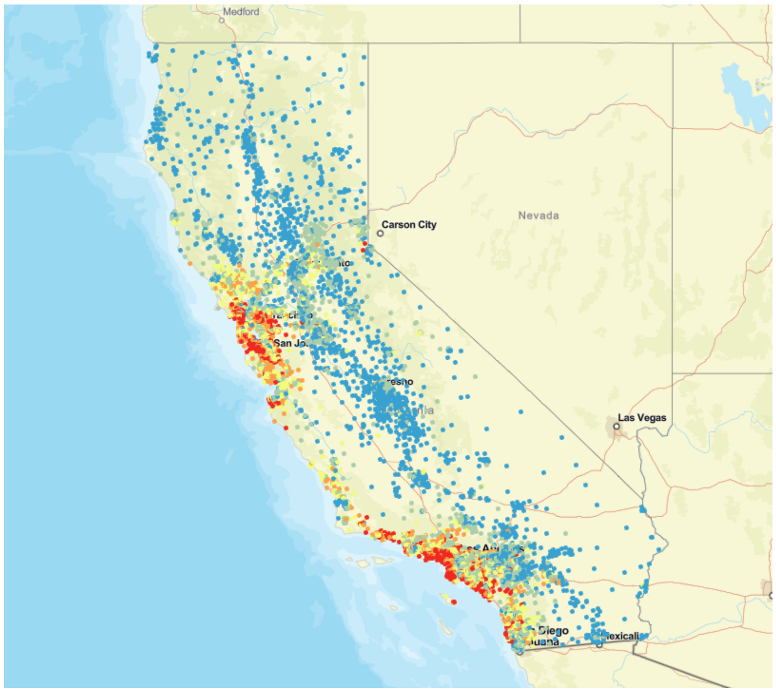


图 1: California Housing Prices Databases

# 3 Look at the Big Picture

## 3.1 Frame the Problem

The first question to ask you is what exactly is the bussiness objective;building a model is probably not the end goal.How do you expect use and benefit from this model?This is improtant because it will determine how you frame the problem,what algorithms you will select,what performance measure you will use to evaluate your model,and how much effort you should spend tweaking it. Your model output (a predicting of a district's median housing price) will be fed to another Machine Learning system along with many other signals.This downstream system will determine whether it is worth investing in a given area or not.

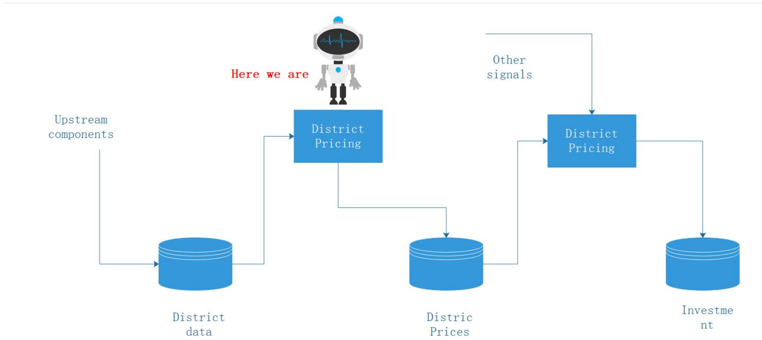


图 2: A machine learning pipeline for real estate investment

The next step is framing the problem:is it supervised,unsupervised,or Reinforcement learning?Is it a classification task,a regression task,or something else?Should we use batch learning or online learning technique? Clearly it is supervised,we need historical data to train the model.Moreover it is a typical regression task,more specifically,this is a multivariate regression since the system will use mutiple features to make a prediction.In first chapter,we predicted life satisfiction based on just one feature,the GDP per captia.Finally,there is no continuous flow of data coming in the system,there is no particular need to adjust to changing data rapidly,and the data is small enough to fit in memory,so plain batch learning should do just fine.

## 3.2 Select a Performance Measure

The next step is to select a performance measure.A typical performance measure for regression problems is the Root Mean Square Error(均方根误差),it measures the standard deviation of the errors the system makes in its prediction.

$$RMSE(X, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(X^i) - y^i)^2} \tag{1}$$

Even though the RMSE is generally the perferred performance measure for regression tasks,in some contexts you may prefer to use another function.For example,suppose that there are many outlier districts.In that case,we may consider using the Mean Absolute Error(平均绝对误差):

$$MAE(X,h)=\frac{1}{m}\sum_{i=1}^{m}|h(X^i)-y^i| \tag{2}$$

Both the RMSE and MAE are ways to measure the *distance* between two vectors.Various distance measures or norms are possible:

- Euclidean norm(欧几里得距离)。
- Manhattan norm(曼哈顿距离),it measures the distance between two points in a city if you can only travel along orthogonal city blocks.也就是指城市中两点之间沿着街区边缘走路的距离。
- More generally,the $l_k$ norm of a vector $v$ containing n elements is defined as:

$$\| v \| = (| v_o |^k + | v_1 |^k + \cdots + | v_n |^k)^{\frac{1}{k}} \tag{3}$$

# 4 Get the data

It's time to get your hands dirty.

## 4.1 Create the workspace

First,you need to have Python enviornment installed.We recommand you installed anaconda on your computer.https://www.anaconda.com/

## 4.2 Take a Quick Look at the Data Structure

Let's take a look a the Data Structure.I download the database from Kaggle.https://www.kaggle.com/camnugent/california-housing-prices.Let's take a glance of the data structure. Each row represent one district.There are 10 attributes(Figure 3):longitude,lattidute,housing_median_age,total_rooms,total_bedrooms,population,households,median_income,median_house_value,ocean_proximity.

| | longitude | latitude | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | median_house_value | ocean_proximity |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -122.23 | 37.88 | 41.0 | 880.0 | 129.0 | 322.0 | 126.0 | 8.3252 | 452600.0 | NEAR BAY |
| 1 | -122.22 | 37.86 | 21.0 | 7099.0 | 1106.0 | 2401.0 | 1138.0 | 8.3014 | 358500.0 | NEAR BAY |
| 2 | -122.24 | 37.85 | 52.0 | 1467.0 | 190.0 | 496.0 | 177.0 | 7.2574 | 352100.0 | NEAR BAY |
| 3 | -122.25 | 37.85 | 52.0 | 1274.0 | 235.0 | 558.0 | 219.0 | 5.6431 | 341300.0 | NEAR BAY |
| 4 | -122.25 | 37.85 | 52.0 | 1627.0 | 280.0 | 565.0 | 259.0 | 3.8462 | 342200.0 | NEAR BAY |
| 5 | -122.25 | 37.85 | 52.0 | 919.0 | 213.0 | 413.0 | 193.0 | 4.0368 | 269700.0 | NEAR BAY |
| 6 | -122.25 | 37.84 | 52.0 | 2535.0 | 489.0 | 1094.0 | 514.0 | 3.6591 | 299200.0 | NEAR BAY |
| 7 | -122.25 | 37.84 | 52.0 | 3104.0 | 687.0 | 1157.0 | 647.0 | 3.1200 | 241400.0 | NEAR BAY |
| 8 | -122.26 | 37.84 | 42.0 | 2555.0 | 665.0 | 1206.0 | 595.0 | 2.0804 | 226700.0 | NEAR BAY |
| 9 | -122.25 | 37.84 | 52.0 | 3549.0 | 707.0 | 1551.0 | 714.0 | 3.6912 | 261100.0 | NEAR BAY |
| 10 | -122.26 | 37.85 | 52.0 | 2202.0 | 434.0 | 910.0 | 402.0 | 3.2031 | 281500.0 | NEAR BAY |
| 11 | -122.26 | 37.85 | 52.0 | 3503.0 | 752.0 | 1504.0 | 734.0 | 3.2705 | 241800.0 | NEAR BAY |
| 12 | -122.26 | 37.85 | 52.0 | 2491.0 | 474.0 | 1098.0 | 468.0 | 3.0750 | 213500.0 | NEAR BAY |
| 13 | -122.26 | 37.84 | 52.0 | 696.0 | 191.0 | 345.0 | 174.0 | 2.6736 | 191300.0 | NEAR BAY |

图 3: Data Structure

The info method is useful to get a quick description of the data,in particular the total number of rows,and each attribute's type and number of non-null values.There are 20640 instances in the dataset,which means that it is fairly small by Machine Learning standards,but it's perfect to get started.Notice that the total_bedrooms attribute has only 20433 non-null values,meaning that 207 districts are missing this feature.We will need to take care of this later.

```
In [5]: raw_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

图 4: Data Info

All attribute are numercial,except the ocean_proximity field.We can find out what categories exist and how many districts belong to each category by using the value_counts() method:

```
In [6]: raw_data['ocean_proximity'].value_counts()

Out[6]: <1H OCEAN     9136
        INLAND        6551
        NEAR OCEAN    2658
        NEAR BAY      2290
        ISLAND           5
        Name: ocean_proximity, dtype: int64
```

图 5: Data Counts

Let's look at the other fields.The describe() method shows a summary of the numercial attributes.(Figure 6).The count,mean,min and max rows are self-explanatory.

```
In [7]: raw_data.describe()
```

Out[7]:

| | longitude | latitude | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | median_house_value |
|---|---|---|---|---|---|---|---|---|---|
| count | 20640.000000 | 20640.000000 | 20640.000000 | 20640.000000 | 20433.000000 | 20640.000000 | 20640.000000 | 20640.000000 | 20640.000000 |
| mean | -119.569704 | 35.631861 | 28.639486 | 2635.763081 | 537.870553 | 1425.476744 | 499.539680 | 3.870671 | 206855.816909 |
| std | 2.003532 | 2.135952 | 12.585558 | 2181.615252 | 421.385070 | 1132.462122 | 382.329753 | 1.899822 | 115395.615874 |
| min | -124.350000 | 32.540000 | 1.000000 | 2.000000 | 1.000000 | 3.000000 | 1.000000 | 0.499900 | 14999.000000 |
| 25% | -121.800000 | 33.930000 | 18.000000 | 1447.750000 | 296.000000 | 787.000000 | 280.000000 | 2.563400 | 119600.000000 |
| 50% | -118.490000 | 34.260000 | 29.000000 | 2127.000000 | 435.000000 | 1166.000000 | 409.000000 | 3.534800 | 179700.000000 |
| 75% | -118.010000 | 37.710000 | 37.000000 | 3148.000000 | 647.000000 | 1725.000000 | 605.000000 | 4.743250 | 264725.000000 |
| max | -114.310000 | 41.950000 | 52.000000 | 39320.000000 | 6445.000000 | 35682.000000 | 6082.000000 | 15.000100 | 500001.000000 |

图 6: Data Describe

Another quick way to get a feel of the type of data you are dealing with is to plot a histogram for each numercial attribute.

```
In [10]: import matplotlib.pyplot as plt
         raw_data.hist(bins = 50,figsize= (20,15))
         plt.show()
```

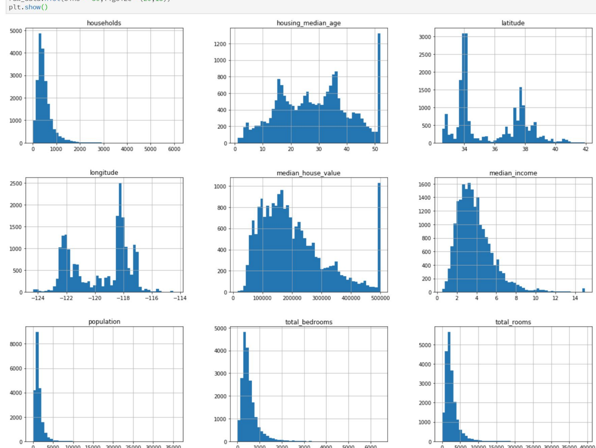


图 7: A histogram for each numercial attribute

### 4.3 Create a Testset

Create a test set is theoretically quite simple:just pick some instances randomly,typically 20% of the dataset,and set them aside.

```
import numpy as np

def split_train_test(data,test_ratio):
  shuffled_indices = np.random.permutation(len(data))
  test_set_size = int(len(data)*test_ratio)
  test_indices = shuffled_indices[:test_size]
  train_indices = shuffled_indices[test_size:]
  return data.iloc[train_indices],data.iloc[test_indices]
```

We can then using the function like this:

```
train_set ,test_set = split_train_test(housing,0.2)
```

This works,but it not perfect:if you run the program again,it will generate a different test set.One solution is to save the test set on the first run and then load it in subsequent runs.Another option is to set the random number generate's seed(eg. np.random.seed(52))before calling np.permutation().But both these solution will break next time you fetch an updated dataset.A common solution is to use each instance's identifier to decide whether or not it should go in the test set.For example,you could compute a hash of each instance's identifier.

```
import hashlib

def test_set_check(identifier ,test_ratio ,hash):
    return hash(np.int64(identifier).digest()[-1]<256*test_ratio)

def split_train_test_by_id(data,test_ratio ,id_column,hash=hashlib.md5):
    ids = data[id_column]
    in_test_set = ids.apply(lambda id_:test_set_check(id_,test_ratio ,hash))
    return data.iloc[~in_test_set],data.iloc[in_test_set]
```

Unfortunately,the hosuing dataset does not have an identifier column.The simplest solution is to use the row index as the ID:

```

housing_with_id = housing.reset_index()
train_set , test_set = split_train_test_by_id(hosuing_with_id ,0.2 ,"index")
```

We can utilize the train_testsplit function to reah the same effect of the split function above.

```
from sklearn.model_selection import train_test_split

train_set , test_set = train_test_split(housing , test_size = 0.25 ,random_state = 0)
```

This generally fine if your dataset is large enough,bu if it is not,you run the risk of introducing a significant sampling bias.For example,when a survey company decides to call 1000 people to ask them a few questions ,they don't just pick 1000 people randomly in a phone booth.They try to ensure that these 1000 people are representative of the whole population,eg. the US population is composed of 51.3% female and 48.7% male,so a survey in the US should try to maintain this ratio in the sample:513 female and 487 male.This is called *stratified sampling.*

Since the median income is a very important attribute to predict median housing prices. You may want to ensure that the test set is representative of the various categories of incomes in the whole dataset

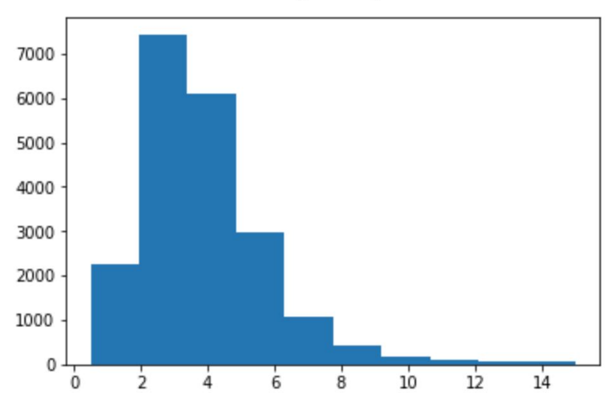


图 8: Histogram of income categories

Let's look at the histogram of income categories,most median income values are clustered around 2-5(tens of thousands of dallors),but some median incomes go far beyond 6.It is important to have a sufficient number of instances in your dataset for each stratum,or else the estimate of the stratum'simportance may br biased,so we should not have too many strata(dividing the median income by 1.5 and rounding up using cel and then merge all the categories greater than 5 into category 5).

```
raw_data["income_cat"] = np.ceil(raw_data["median_income"]/1.5)
raw_data["income_cat"].where(raw_data["income_cat"]>5,5.0,inplace = False)
```

Now you are ready to do stratified sampling based on the income category.For this we can use Scikit-learn's StratifiedShuffleSplit class:

```
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits = 1,test_size = 0.2, random_state = 17)

for train_index, test_index in split.split(raw_data, raw_data["income_cat"]):
    strat_train_set = raw_data.loc[train_index]
    strat_test_set = raw_data.loc[test_index]
```

By this way,the category proportions in the test_set which generated with stratified sampling almost identical to those in the full dataset.

We spent quite a bit of time on the test set generation for a good reason:this is an often neglected but critical part of a Machine Learning project. At last,we should remove the income_cat attribute so the data is back to its original state:

```
for set in(strat_train_set,strat_test_set):
  set.drop(["income_cat"],axis = 1,inplace = True)
```

这个部分看似平淡无奇，其实还蛮有用的，之前在选数据集的时候，从来没有考虑过这个问题。回头想想，选择有代表性的数据集对于一个监督学习的系统来说，还是非常重要的。使得在训练阶段也能提高精度，这个道理我想是不言自明的。

# 5 Discover and Visualize the Data ti Gain Insights

So far we have only taken a quick glance at the data to get a general understanding of the kind of data we are manipulating.In our case,the set is quite small so you can just work directly on the full set.Let's create a copy so you can play with it without harming the training set.

```
housing = strat_train_set.copy()
```

## 5.1 Visualizing Geographical Data

Since there is geographical information(latitude and longitude),let's plot it!

```
housing.plot(kind="scatter",x="longitude",y="latitude")
```

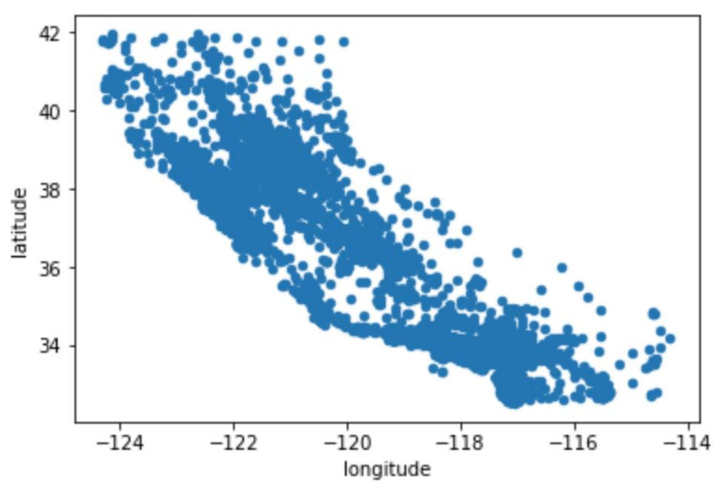


图 9: A geographical scatterplot of the data

Setting the alpha option to 0.1 makes it much easier to visualize the places where there is a high density of data points.

```
housing.plot(kind="scatter",x="longitude",y="latitude",alpha = 0.1)
```

alpha:float (0.0 transparent through 1.0 opaque),这里的alpha指的是透明度,所以密度越大的地方，因为重叠的原因，颜色就会越深。

It's better now,let's look at the housing prices.The radius of each circle represents the district's population(option s),and the color represents the price(option c).We will use a predefined color map(option cmap)called jet,which ranges from blue(low values) to red(high prices):

```
housing.plot(kind="scatter",x="longitude",y="latitude",alpha=0.4,s=housing["population"]/100,
    label="population",c="median_house_value",cmap=plt.get_cmap("jet"),colorbar=True)
plt.legend()
```

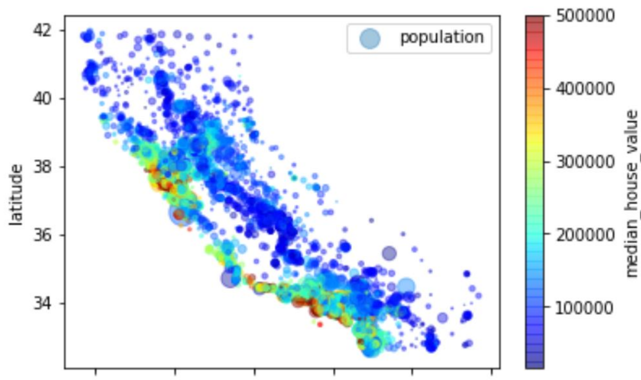


图 10: California housing prices

观察最后得到的图我们大概能看得出一些结论，这些结论也与我们的常识相符。首先，房价和位置的关系很大，临海的房价普遍要高一些，还有某些位置的房价普遍要高一些，这些地方应该是加州沿海的一些城市，旧金山，洛杉矶等等。当然这也不是全部，北部沿海的房价也会低一些。

## 5.2 Looking for Correlations

Since the data set is not too large ,we can easily compute the *Standard correlation coefficient*(also called Pearson's r 皮尔森相关系数) between every pair of attributes using the corr() method:

```
corr_matrix = housing.corr()
```

The matrix is big,so let's look at a specific attribute(eg.median house value).

```
corr_matrix["median_house_value"].sort_values(ascending=False)
```

The correlation coefficient ranges from 1 to -1.接近1代表正相关，接近-1代表负相关，我们可以看出平均房价和收入是正相关关系，而和经度是负相关的，也就是内陆房价低，沿海高。Finally,coefficients close to zero mean there is no linear correlation.

```
median_house_value    1.000000
median_income         0.689774
total_rooms           0.137847
housing_median_age    0.098007
households            0.069283
total_bedrooms        0.053267
population           -0.023161
longitude            -0.048948
latitude             -0.140303
Name: median_house_value, dtype: float64
```

下面这张图来自维基百科，从左到右依次表示皮尔森相关系数在不同值时的情况。

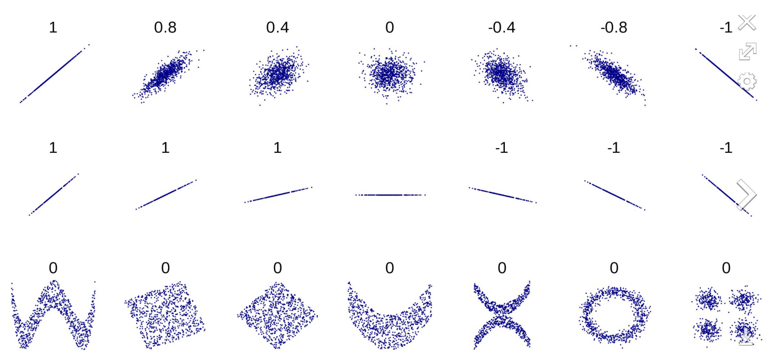


图 11: Scatter Matrix

The correlation coefficient only measures linear correlations.

Another way to check for correlations between attributes is to use Pandas' scatter_matrix function.Since there are now 11 numerical attributes,you would get $11^2$ plots,so let's just focus on a few promising attributes that seem most correlated with the median housing value.

```
from pandas.plotting import scatter_matrix

attributes = ["median_house_value","median_income","total_rooms","housing_median_age"]

scatter_matrix(housing[attributes],figsize=(12,8))
```

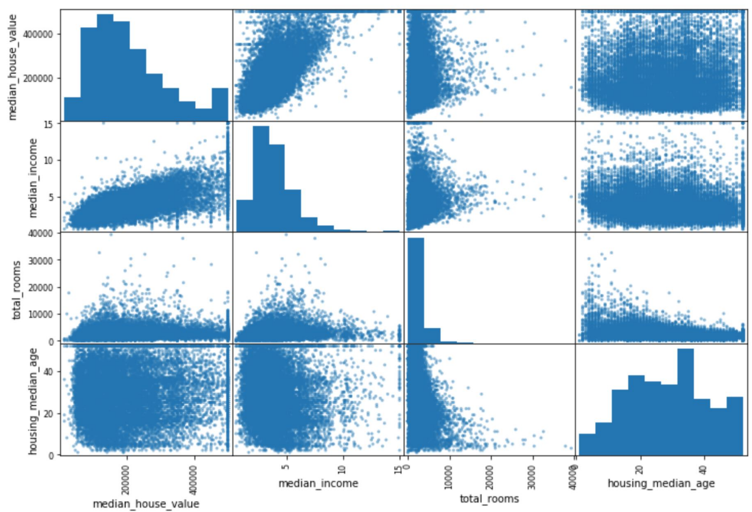


图 12: Scatter Matrix