

# Discover and Visualize the Data to Gain Insights

So far you have only taken a quick glance at the data to get a general understanding of the kind of data you are manipulating. Now the goal is to go into a little more depth.

First, make sure you have put the test set aside and you are only exploring the training set. Also, if the training set is very large, you may want to sample an exploration set, to make manipulations easy and fast. In our case, the set is quite small, so you can just work directly on the full set. Let's create a copy so that you can play with it without harming the training set:

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

## Visualizing Geographical Data

Since there is geographical information (latitude and longitude), it is a good idea to create a scatterplot of all districts to visualize the data (Figure 2-11):

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

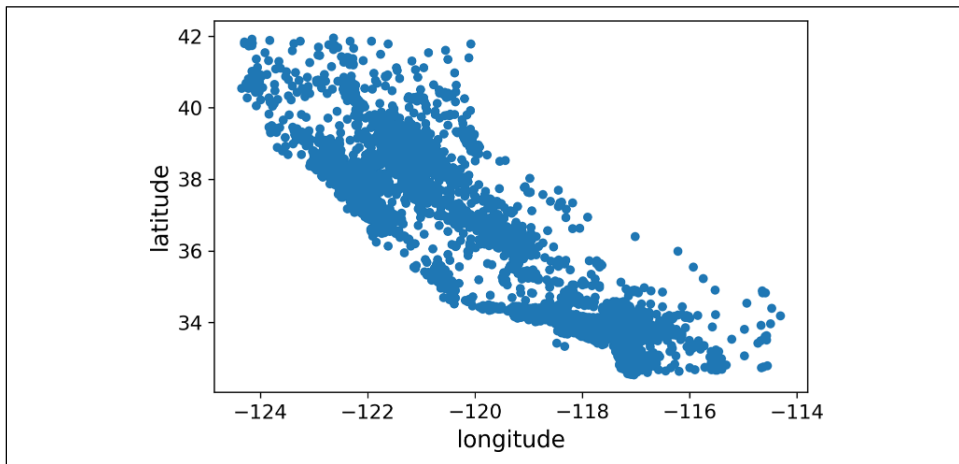


Figure 2-11. A geographical scatterplot of the data

This looks like California all right, but other than that it is hard to see any particular pattern. Setting the alpha option to 0.1 makes it much easier to visualize the places where there is a high density of data points (Figure 2-12):

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

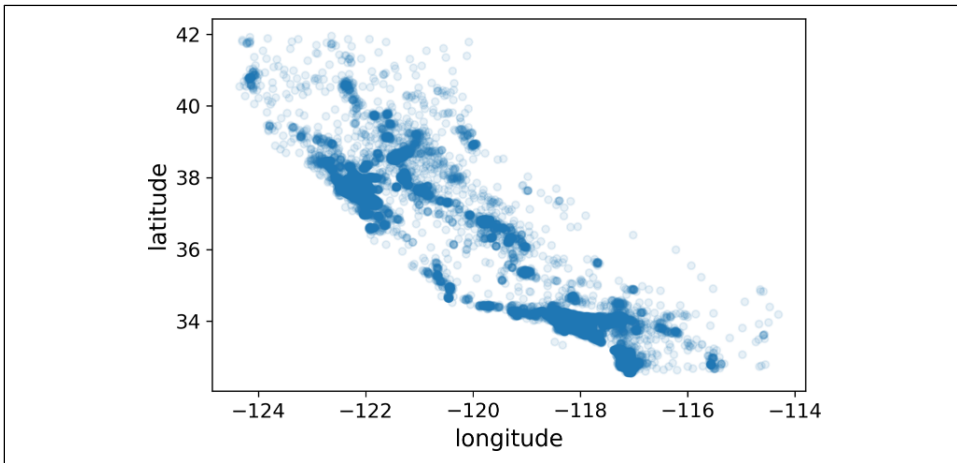


Figure 2-12. A better visualization that highlights high-density areas

Now that's much better: you can clearly see the high-density areas, namely the Bay Area and around Los Angeles and San Diego, plus a long line of fairly high density in the Central Valley, in particular around Sacramento and Fresno.

Our brains are very good at spotting patterns in pictures, but you may need to play around with visualization parameters to make the patterns stand out.

Now let's look at the housing prices (Figure 2-13). 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):<sup>16</sup>

```
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
              s=housing["population"]/100, label="population", figsize=(10,7),
              c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,
            )
plt.legend()
```

<sup>16</sup> If you are reading this in grayscale, grab a red pen and scribble over most of the coastline from the Bay Area down to San Diego (as you might expect). You can add a patch of yellow around Sacramento as well.

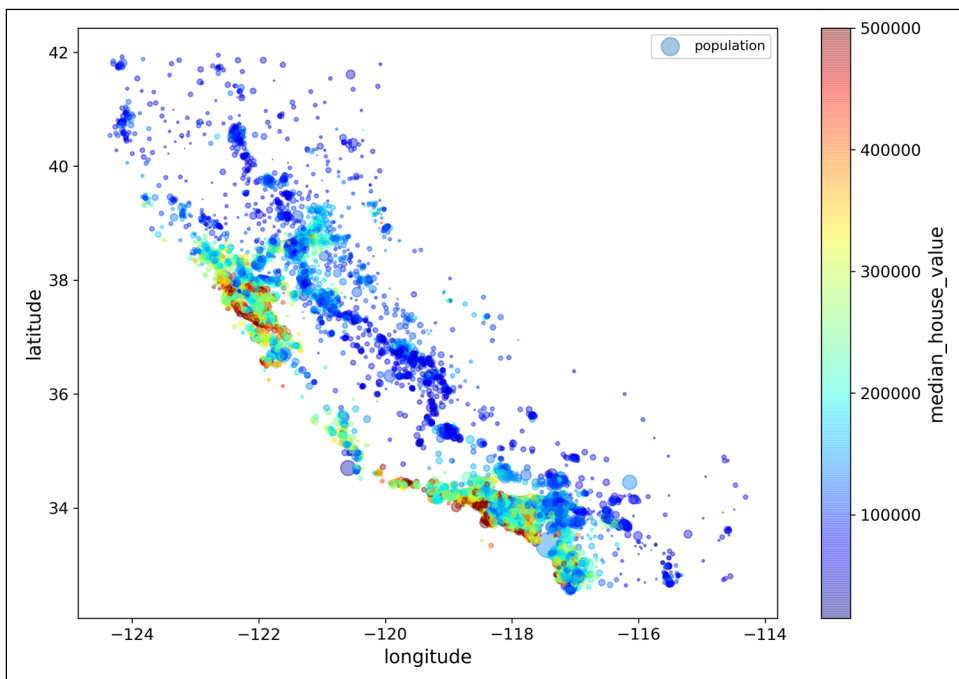


Figure 2-13. California housing prices: red is expensive, blue is cheap, larger circles indicate areas with a larger population

This image tells you that the housing prices are very much related to the location (e.g., close to the ocean) and to the population density, as you probably knew already. A clustering algorithm should be useful for detecting the main cluster and for adding new features that measure the proximity to the cluster centers. The ocean proximity attribute may be useful as well, although in Northern California the housing prices in coastal districts are not too high, so it is not a simple rule.

## Looking for Correlations

Since the dataset is not too large, you can easily compute the *standard correlation coefficient* (also called *Pearson's r*) between every pair of attributes using the `corr()` method:

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

Now let's look at how much each attribute correlates with the median house value:

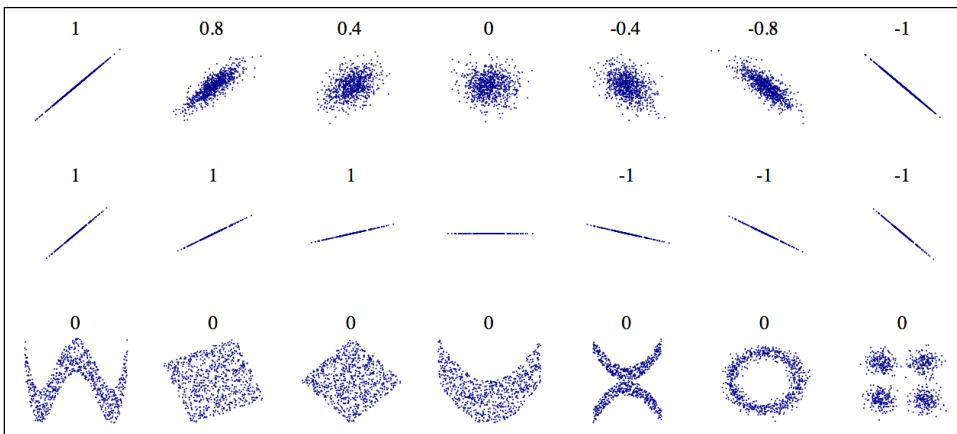
```
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value    1.000000
median_income         0.687170
total_rooms           0.135231
housing_median_age    0.114220
```

```

households          0.064702
total_bedrooms      0.047865
population          -0.026699
longitude           -0.047279
latitude            -0.142826
Name: median_house_value, dtype: float64

```

The correlation coefficient ranges from  $-1$  to  $1$ . When it is close to  $1$ , it means that there is a strong positive correlation; for example, the median house value tends to go up when the median income goes up. When the coefficient is close to  $-1$ , it means that there is a strong negative correlation; you can see a small negative correlation between the latitude and the median house value (i.e., prices have a slight tendency to go down when you go north). Finally, coefficients close to  $0$  mean that there is no linear correlation. **Figure 2-14** shows various plots along with the correlation coefficient between their horizontal and vertical axes.



*Figure 2-14. Standard correlation coefficient of various datasets (source: Wikipedia; public domain image)*



The correlation coefficient only measures linear correlations (“if  $x$  goes up, then  $y$  generally goes up/down”). It may completely miss out on nonlinear relationships (e.g., “if  $x$  is close to  $0$ , then  $y$  generally goes up”). Note how all the plots of the bottom row have a correlation coefficient equal to  $0$ , despite the fact that their axes are clearly not independent: these are examples of nonlinear relationships. Also, the second row shows examples where the correlation coefficient is equal to  $1$  or  $-1$ ; notice that this has nothing to do with the slope. For example, your height in inches has a correlation coefficient of  $1$  with your height in feet or in nanometers.

Another way to check for correlation between attributes is to use the pandas `scatter_matrix()` function, which plots every numerical attribute against every

other numerical attribute. Since there are now 11 numerical attributes, you would get  $11^2 = 121$  plots, which would not fit on a page—so let's just focus on a few promising attributes that seem most correlated with the median housing value (Figure 2-15):

```
from pandas.plotting import scatter_matrix

attributes = ["median_house_value", "median_income", "total_rooms",
             "housing_median_age"]
scatter_matrix(housing[attributes], figsize=(12, 8))
```

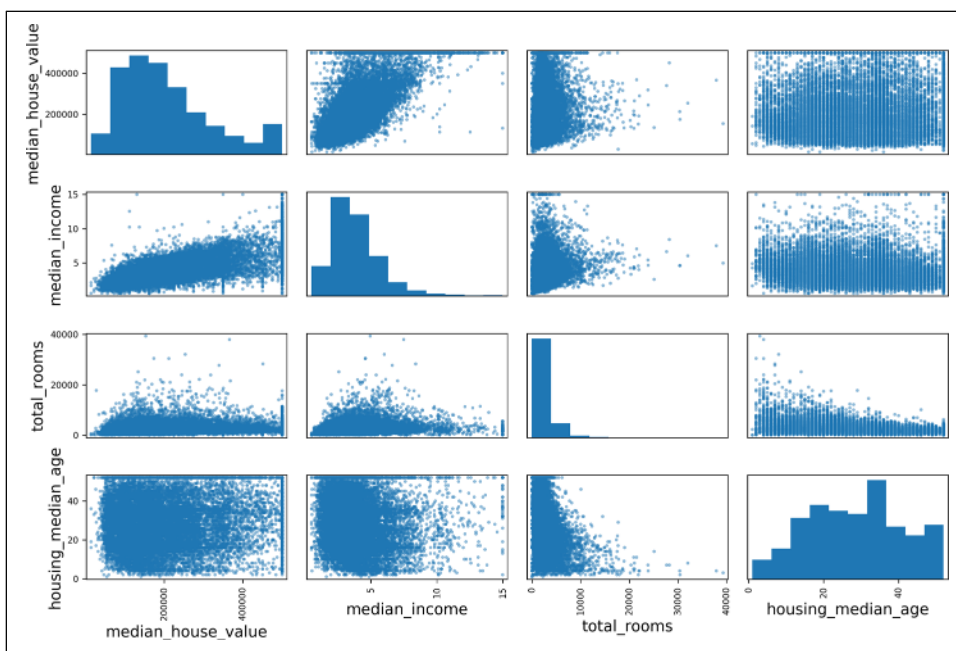


Figure 2-15. This scatter matrix plots every numerical attribute against every other numerical attribute, plus a histogram of each numerical attribute

The main diagonal (top left to bottom right) would be full of straight lines if pandas plotted each variable against itself, which would not be very useful. So instead pandas displays a histogram of each attribute (other options are available; see the pandas documentation for more details).

The most promising attribute to predict the median house value is the median income, so let's zoom in on their correlation scatterplot (Figure 2-16):

```
housing.plot(kind="scatter", x="median_income", y="median_house_value",
             alpha=0.1)
```

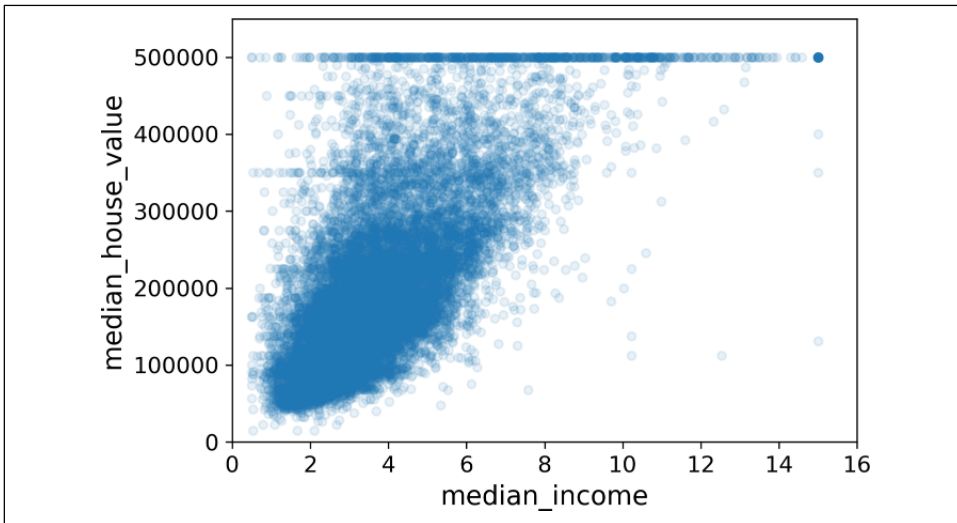


Figure 2-16. Median income versus median house value

This plot reveals a few things. First, the correlation is indeed very strong; you can clearly see the upward trend, and the points are not too dispersed. Second, the price cap that we noticed earlier is clearly visible as a horizontal line at \$500,000. But this plot reveals other less obvious straight lines: a horizontal line around \$450,000, another around \$350,000, perhaps one around \$280,000, and a few more below that. You may want to try removing the corresponding districts to prevent your algorithms from learning to reproduce these data quirks.

## Experimenting with Attribute Combinations

Hopefully the previous sections gave you an idea of a few ways you can explore the data and gain insights. You identified a few data quirks that you may want to clean up before feeding the data to a Machine Learning algorithm, and you found interesting correlations between attributes, in particular with the target attribute. You also noticed that some attributes have a tail-heavy distribution, so you may want to transform them (e.g., by computing their logarithm). Of course, your mileage will vary considerably with each project, but the general ideas are similar.

One last thing you may want to do before preparing the data for Machine Learning algorithms is to try out various attribute combinations. For example, the total number of rooms in a district is not very useful if you don't know how many households there are. What you really want is the number of rooms per household. Similarly, the total number of bedrooms by itself is not very useful: you probably want to compare it to the number of rooms. And the population per household also seems like an interesting attribute combination to look at. Let's create these new attributes: