

Now you 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)
```

We spent quite a bit of time on test set generation for a good reason: this is an often neglected but critical part of a Machine Learning project. Moreover, many of these ideas will be useful later when we discuss cross-validation. Now it's time to move on to the next stage: exploring the data.

## 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 a little bit more in 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 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")
```

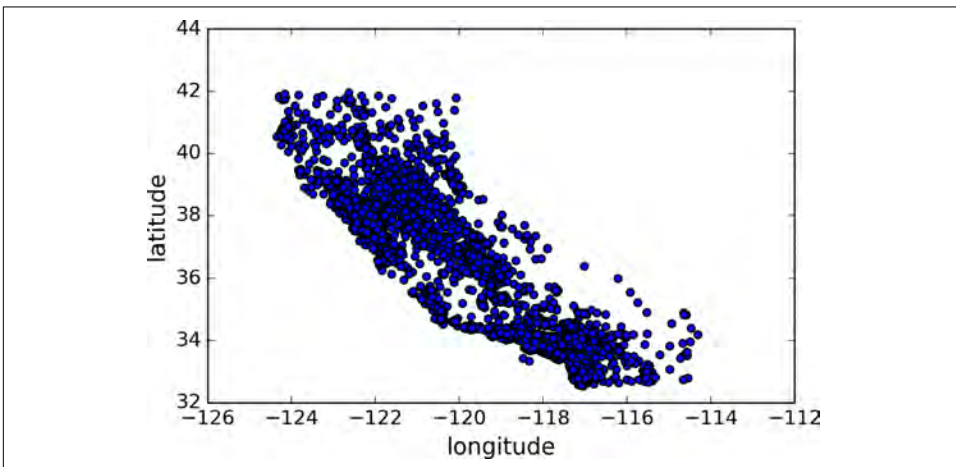


Figure 2-11. A geographical scatterplot of the data

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)
```

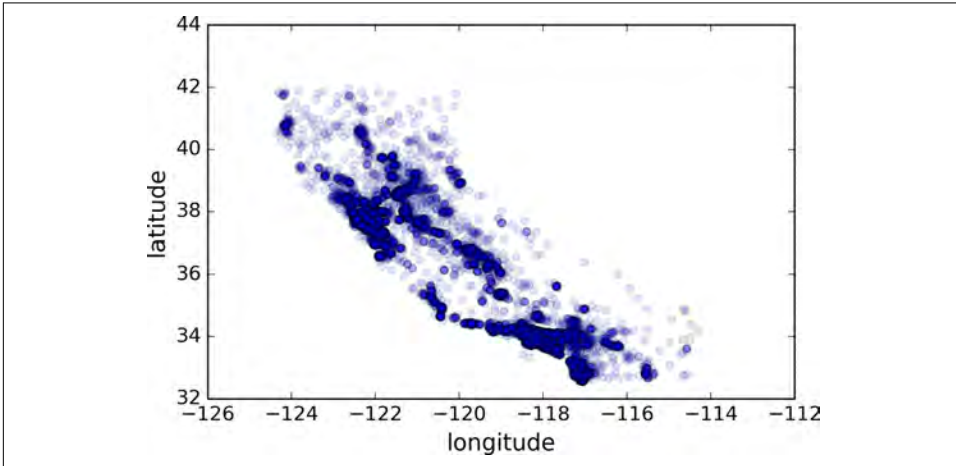


Figure 2-12. A better visualization highlighting high-density areas

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.

More generally, our brains are very good at spotting patterns on 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>15</sup>

```
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()
```

<sup>15</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.

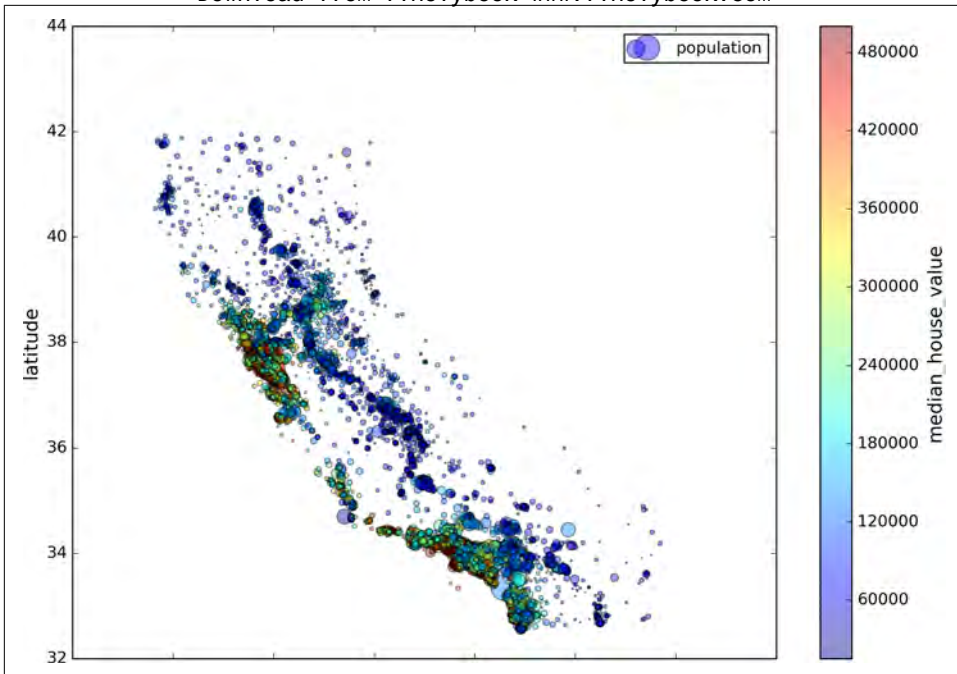


Figure 2-13. California housing prices

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. It will probably be useful to use a clustering algorithm to detect the main clusters, and add 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
```