Discover & Visualize Data

Tuesday, December 17, 2019 10:18 AM

Note Written by: Zulfadli Zainal

Let's dive deep into the data!

Visualizing Geographical Data

Since we are working on only training data, lets simplify the database by creating a copy of training database.

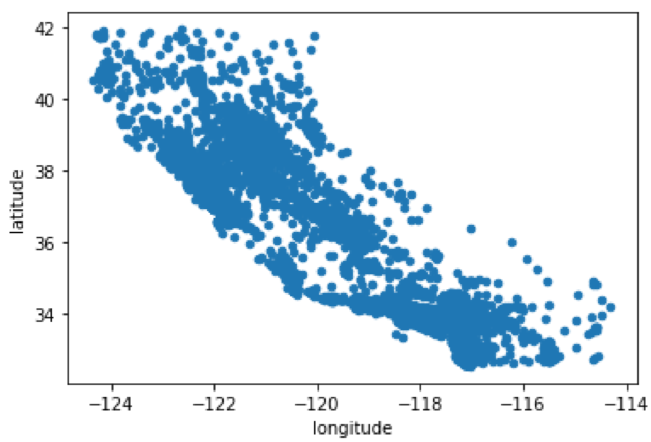
Form Book:
```
housing = strat_train_set.copy()
```

From my code:
```
housing = train_set_05.copy()
```

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

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

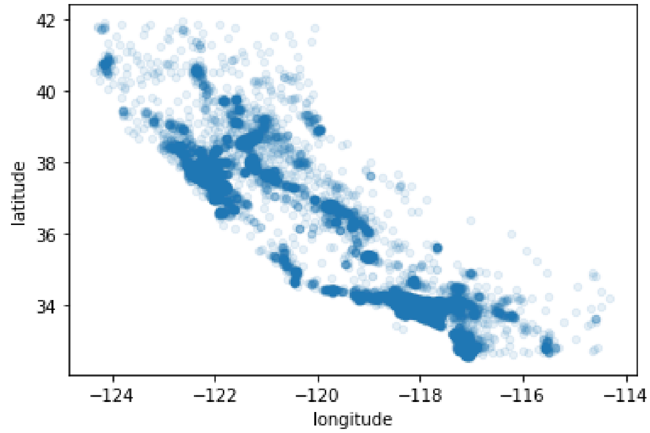
```
housing.plot(kind="scatter", x="longitude", y="latitude")
```



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.

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

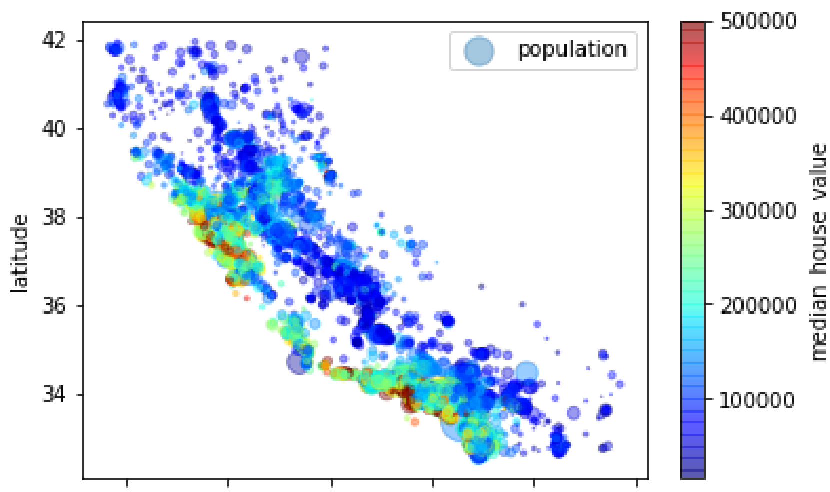
```
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
```



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()
```

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)



What is the conclusion from here?

This image tells you that the housing prices are very much related to the location. (Close to Ocean & Population Density)

It will probably be useful to use a clustering algorithm to detect the main clusters.

Looking for Correlation

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:

For each features, we try to correlate with each and every features.

Basic Method: Table Calculation

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

corr_matrix = housing.corr()

Index	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	income_cat
longitude	1	-0.924478	-0.105848	0.048871	0.0765985	0.10803	0.0630696	-0.0195833	-0.0474324	-0.0162433
latitude	-0.924478	1	0.0057661	-0.0391837	-0.0724189	-0.115222	-0.077647	-0.0752054	-0.142724	-0.0788483
housing_medi…	-0.105848	0.0057661	1	-0.364509	-0.325047	-0.29871	-0.306428	-0.11136	0.11411	-0.139385
total_rooms	0.048871	-0.0391837	-0.364509	1	0.929379	0.855109	0.918392	0.200087	0.135097	0.22169
total_bedroo…	0.0765985	-0.0724189	-0.325047	0.929379	1	0.87632	0.98017	-0.00973978	0.0476886	0.012502
population	0.10803	-0.115222	-0.29871	0.855109	0.87632	1	0.904637	0.00238001	-0.02692	0.0228479
households	0.0630696	-0.077647	-0.306428	0.918392	0.98017	0.904637	1	0.0107813	0.0645063	0.0349951
median_income	-0.0195833	-0.0752054	-0.11136	0.200087	-0.00973978	0.00238001	0.0107813	1	0.68716	0.902156
median_house…	-0.0474324	-0.142724	0.11411	0.135097	0.0476886	-0.02692	0.0645063	0.68716	1	0.642274
income_cat	-0.0162433	-0.0788483	-0.139385	0.22169	0.012502	0.0228479	0.0349951	0.902156	0.642274	1

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

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

```
In [41]: corr_matrix["median_house_value"].sort_values(ascending=False)
Out[41]:
median_house_value    1.000000
median_income         0.687160
income_cat            0.642274
total_rooms           0.135097
housing_median_age    0.114110
households            0.064506
total_bedrooms        0.047689
population           -0.026920
longitude            -0.047432
latitude             -0.142724
Name: median_house_value, dtype: float64
```

What Correlation means??

Correlation Coefficient is between -1 to 1.

When close to 1 -> There is strong +ve correlation.

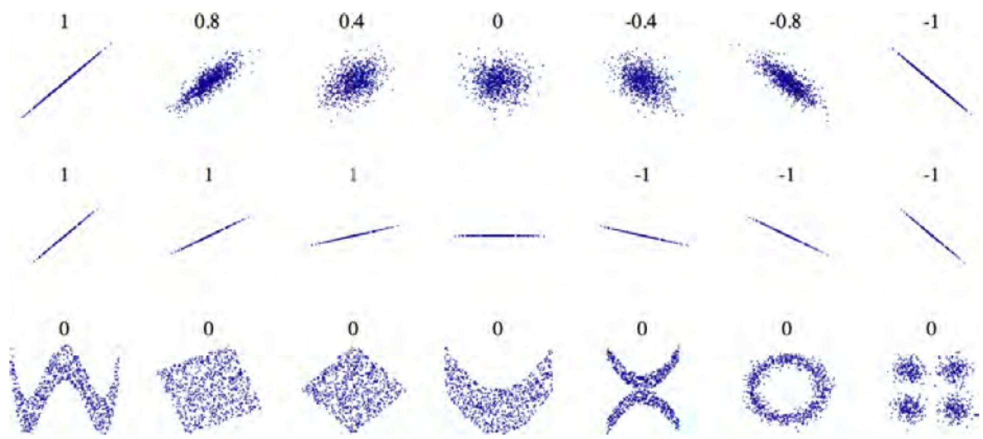
Eg: the median house value tends to go up when the median income goes up.

When close to -1 -> There is strong -ve correlation.

Eg: 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)

When close to 0 -> There is no linear correlation.

Sample correlation charts:



The correlation coefficient only measures linear correlations
-> if x goes up, then y generally goes up/down

<u>Other Method: Pandas Scatter Matrix</u>

Pandas Scatter Matrix: Plots every numerical attributes

Since in our data we have 11 columns, Pandas will plot 11^2 plot = 121 plots.

```
from pandas.tools.plotting import scatter_matrix
```

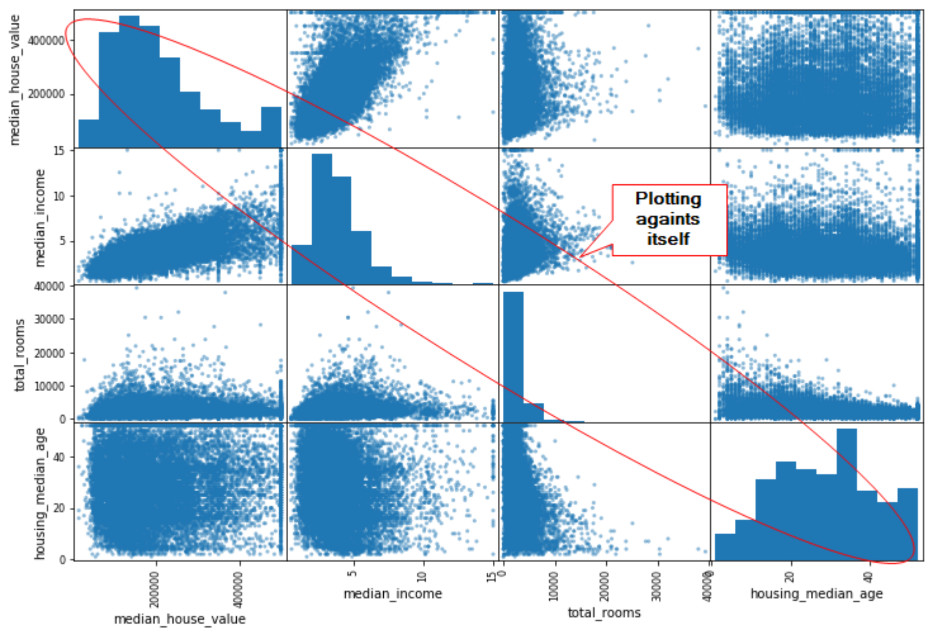
New Pandas - Remove '.tools'

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

from pandas.plotting import scatter_matrix

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

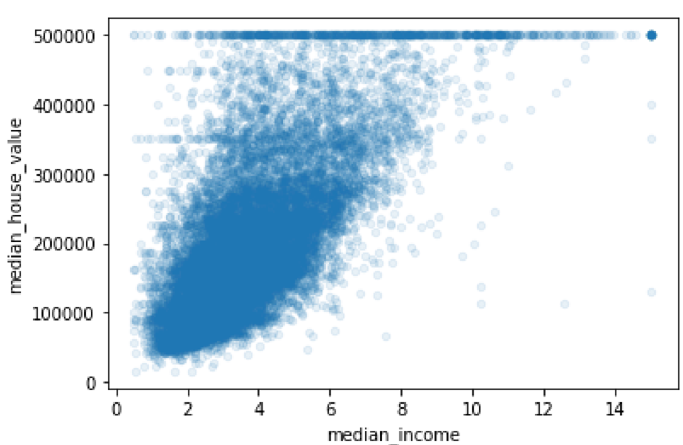
#Scatter selected attributes with each other
scatter_matrix(housing[attributes], figsize=(12, 8))

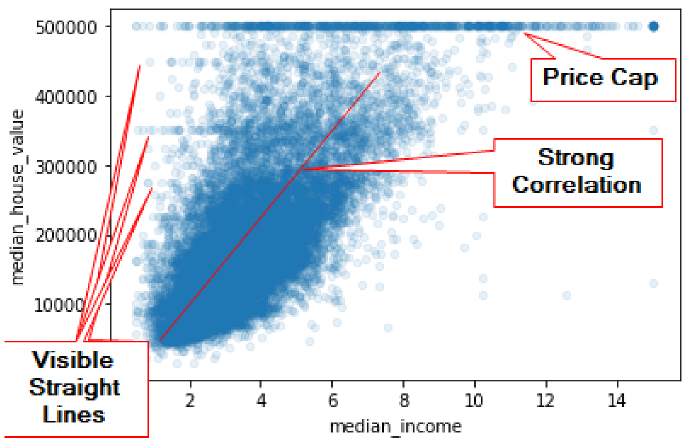


Looks like the most interesting insight that we going to get is median house value vs median income.
Lets zoom in to get more detail result.

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

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





Findings from this scatter:

1. Correlation seems really strong - Upward Trend
2. Price cap visible at 500K
3. Shows some visible straight lines at 450K, 350K, and 280K - Maybe need to remove corresponding districts to prevent your algorithms from learning to reproduce these data quirks.

Experimenting with Attributes Combinations

Try to experiment with your data. Not just visualize.

For example: Total number of rooms data might not be useful if you don't know how many household they are.

Lets create a new attributes - That would help to understand the data more!

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
housing["population_per_household"]=housing["population"]/housing["households"]
```

housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
housing["population_per_household"] = housing["population"]/housing["households"]

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

#Finding correlation with new created Attributes
corr_matrix = housing.corr()
corr_matrix["median_house_value"].sort_values(ascending = False)

```
In [23]: corr_matrix = housing.corr()
    ...: corr_matrix["median_house_value"].sort_values(ascending = False)
Out[23]:
median_house_value          1.000000
median_income               0.687160
income_cat                  0.642274
rooms_per_household         0.146285
total_rooms                 0.135097
housing_median_age          0.114110
households                  0.064506
total_bedrooms              0.047689
population_per_household   -0.021985
population                 -0.026920
longitude                  -0.047432
latitude                   -0.142724
bedrooms_per_room          -0.259984
Name: median_house_value, dtype: float64
```

Better Correlation!

Seems like we can gain more insights by creating more valuable attributes.