Machine Learning

| Discover and Visualize the Data

Prof. Dr. Mostafa Elhosseini

https://youtube.com/drmelhosseini

Agenda

- Discover and Visualize the Data to Gain Insights
-

Remind

```
strat_train_set, strat_test_set = train_test_split(
    housing, test_size=0.2, stratify=housing["income_cat"], random_state=42)
```

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

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

Remind

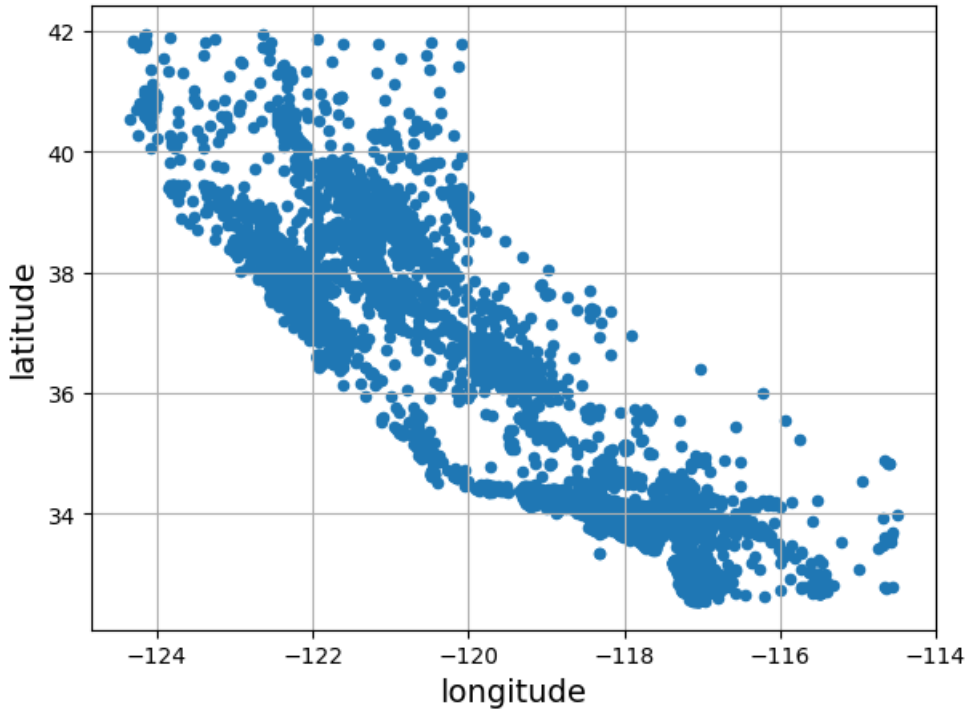
- Ensure the **test set is set aside** to prevent data snooping during the exploration of the training set.
- For large training sets, consider sampling a smaller subset for easier and faster exploration.
- If the training set is small, it's feasible to work directly on the entire dataset.
- Create a copy of the full training set before applying any transformations, allowing you to revert to the original state if needed.

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

Geographical Data

- Create a scatterplot of all the districts to visualize the data.

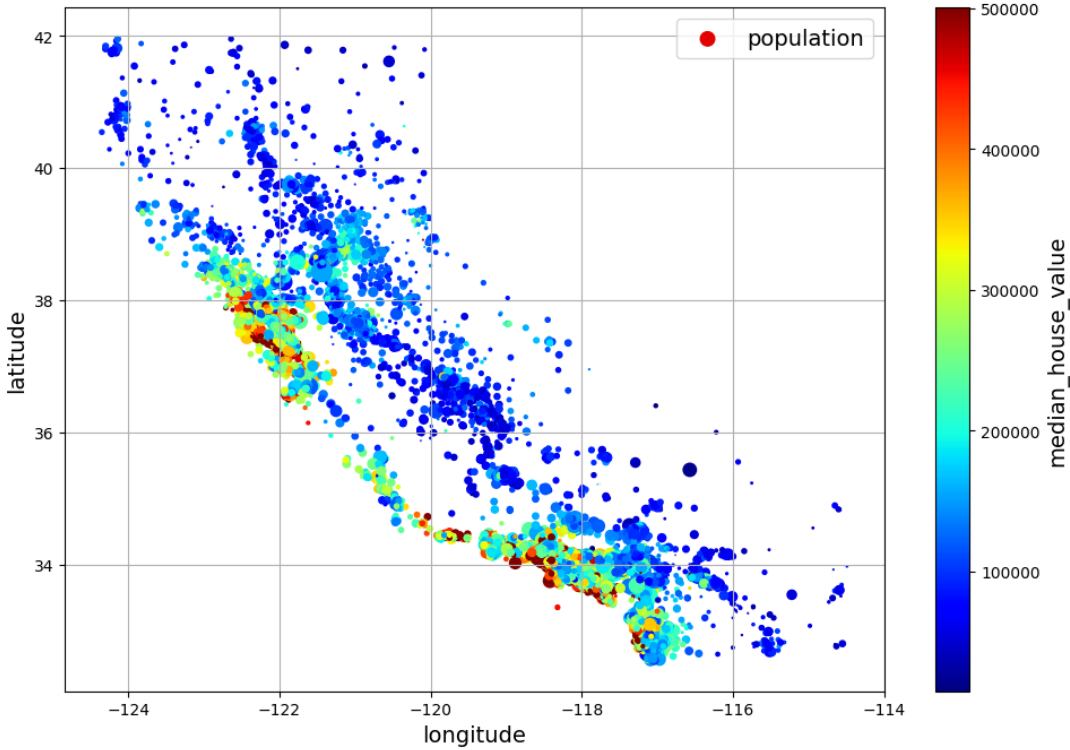
```
housing.plot(kind="scatter", x="longitude", y="latitude", grid=True)
save_fig("bad_visualization_plot")  # extra code
plt.show()
```



Geographical Data

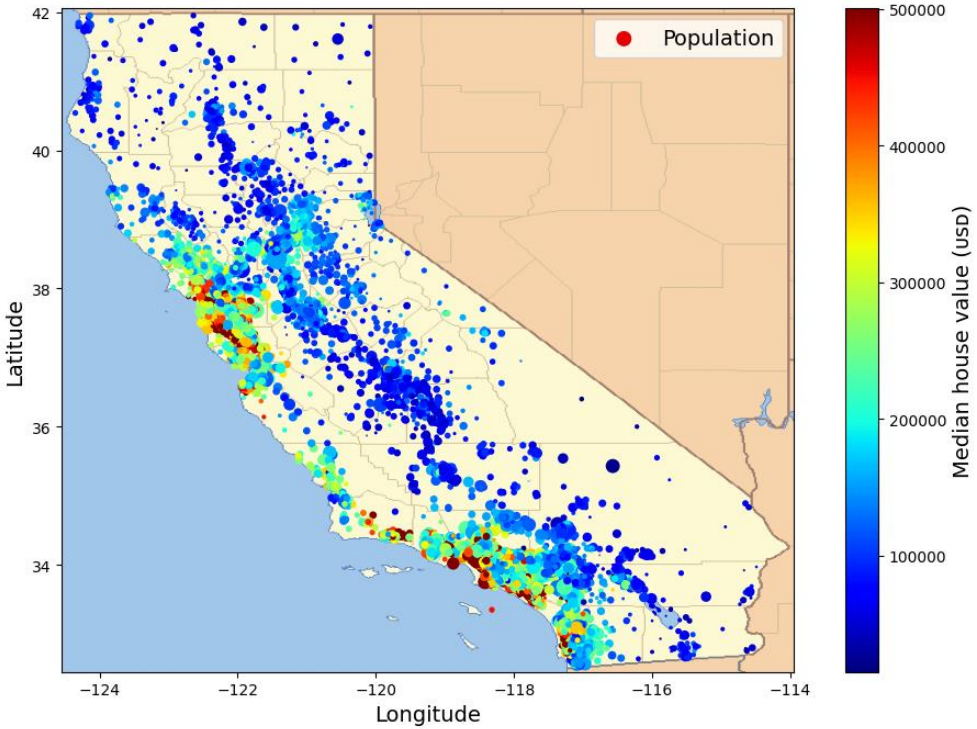
- Create a scatterplot of all the districts to visualize the data.

```
housing.plot(kind="scatter", x="longitude", y="latitude", grid=True,
             s=housing["population"] / 100, label="population",
             c="median_house_value", cmap="jet", colorbar=True,
             legend=True, sharex=False, figsize=(10, 7))
save_fig("housing_prices_scatterplot")  # extra code
plt.show()
```



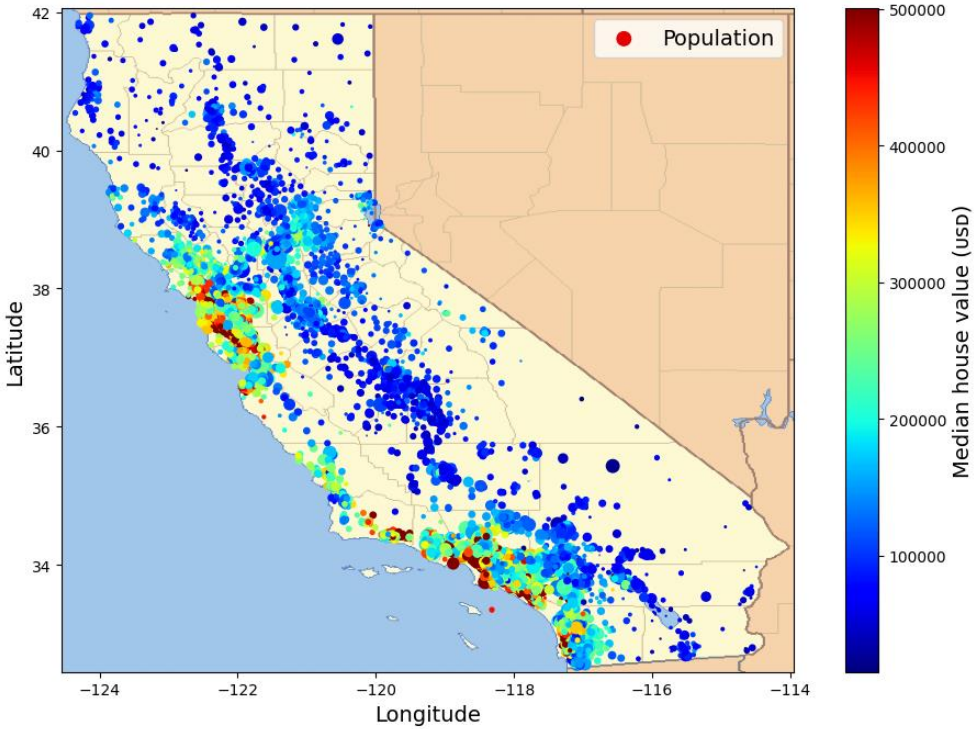
Geographical Data

- Create a scatterplot of all the districts to visualize the data.



Geographical Data

- The **housing prices** are very much **related to the location** (e.g., close to the ocean) and to the population density.



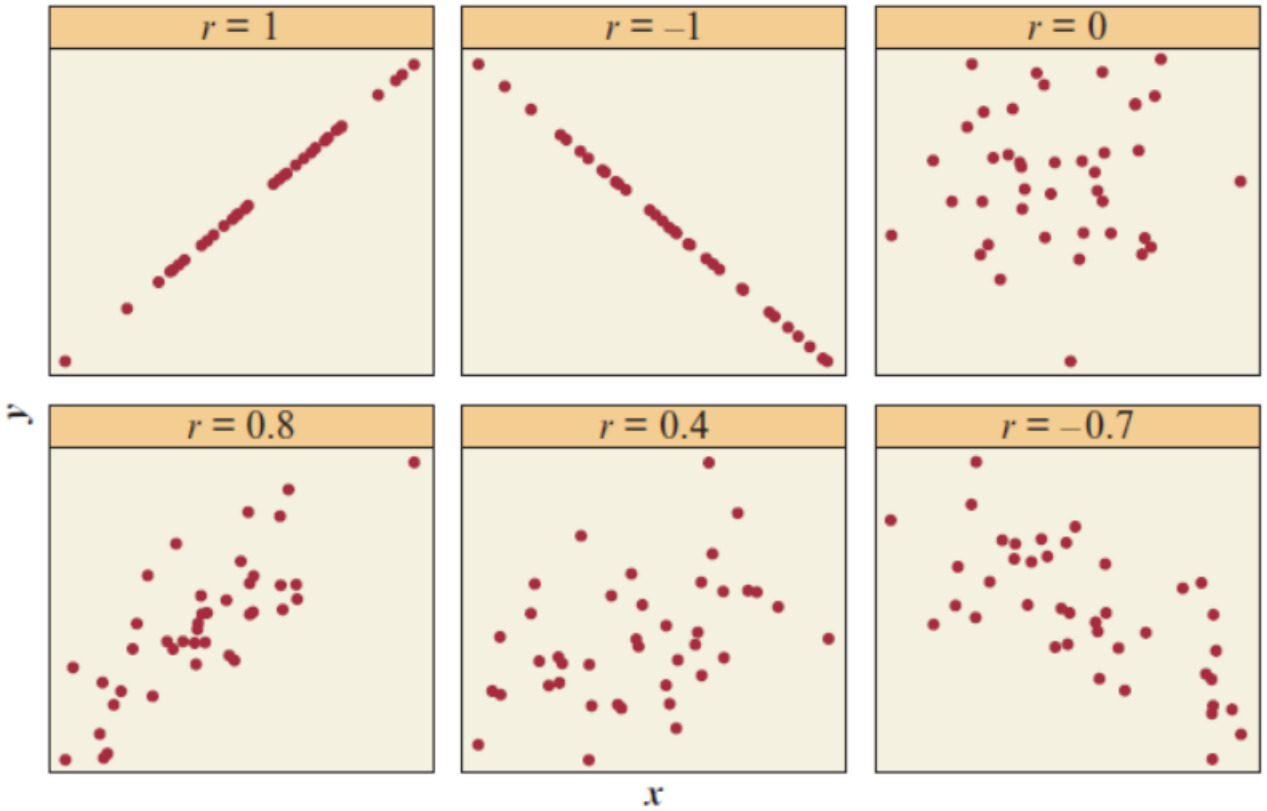
Look for Correlations

- In practice, data points in a scatterplot sometimes fall close to a straight-line trend
- When the data points follow a roughly straight-line trend, the variables are said to have an approximately linear relationship.
- In some cases, the data points fall close to a straight line, but more often there is quite a bit of variability of the points around the straight-line trend.
- A summary measure called the correlation describes the strength of the linear association

Look for Correlations

- The correlation summarizes the **strength and direction of the linear** (straight-line) association between two quantitative variables. Denoted by r, it takes values between -1 and +1
- A positive value for r indicates a positive association, and a negative value for r indicates a negative association
- An r value close to +1 or -1 indicates a strong linear association.
- An r value close to 0 indicates a weak association.
- A positive correlation indicates a positive association, and a negative correlation indicates a negative association
- The value of the correlation does not depend on the variables' units
- The correlation does not depend on which variable is treated as the response and which as the explanatory variable
- **Pearson correlation** refers to the British statistician, Karl Pearson. In 1896 he provided the formula used to compute the correlation value from sample data

Look for Correlations



Look for Correlations

- Compute the standard correlation coefficient (also called Pearson's r) between every pair of attributes using the **corr()** method

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

- 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         0.688380
total_rooms           0.137455
housing_median_age    0.102175
households            0.071426
total_bedrooms        0.054635
population           -0.020153
longitude            -0.050859
latitude             -0.139584
Name: median_house_value, dtype: float64
```

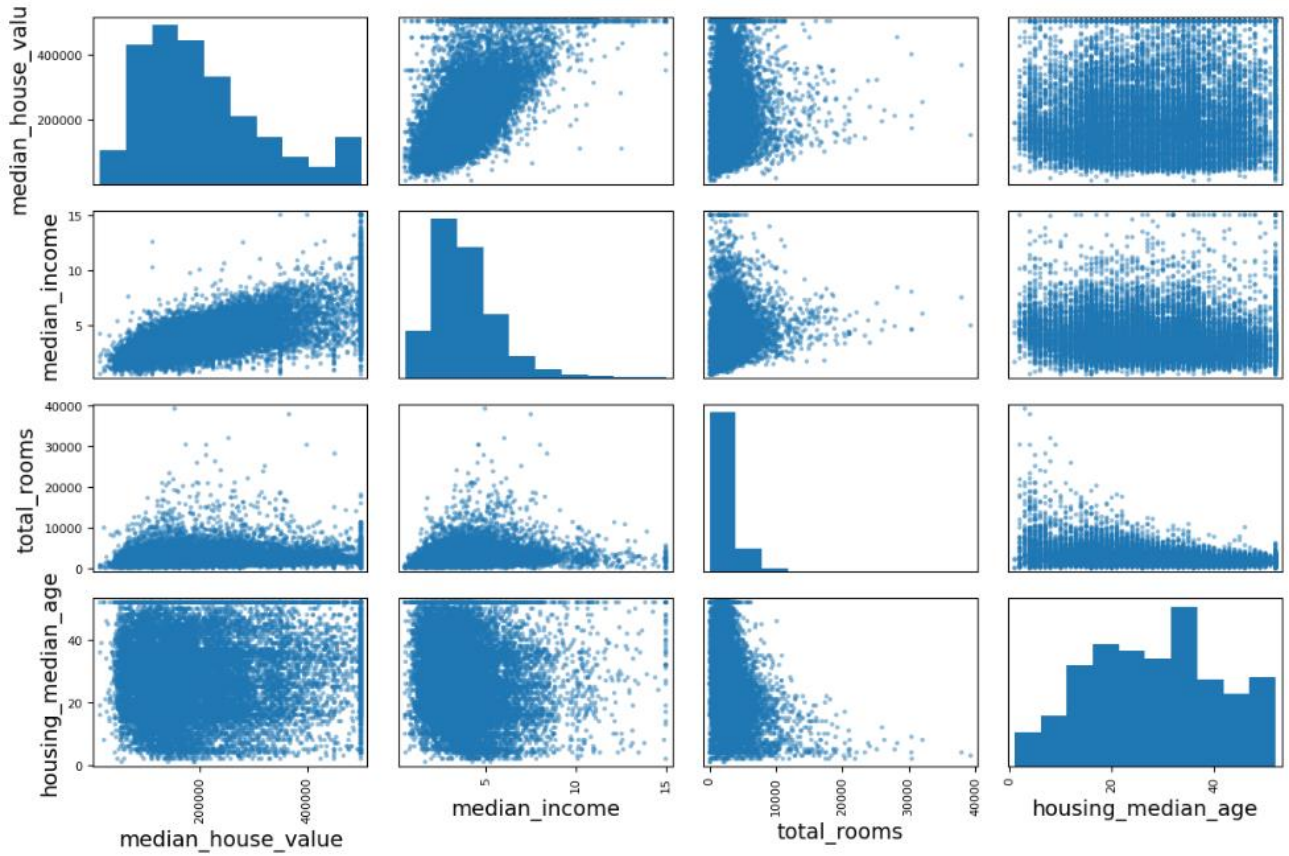
Look for Correlations

```
median_house_value    1.000000
median_income         0.688380
total_rooms           0.137455
housing_median_age    0.102175
households            0.071426
total_bedrooms        0.054635
population           -0.020153
longitude            -0.050859
latitude             -0.139584
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).
- Coefficients close to 0 mean that there is no linear correlation.

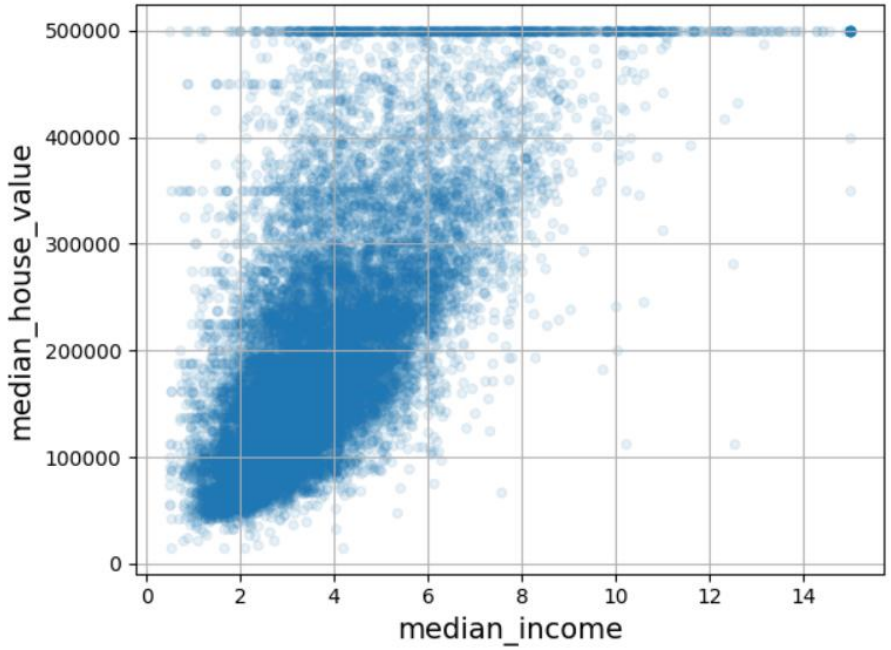
Look for Correlations

- Pandas scatter_matrix() function



Look for Correlations

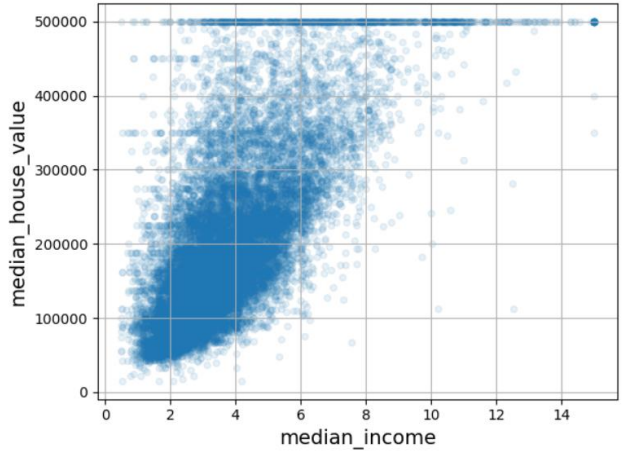
- Clear upward trend, indicating a strong correlation between variables, with relatively low dispersion of points.
- A visible horizontal line at $500,000 indicates a price cap in the data.
- Other less noticeable horizontal lines at various price points (e.g., $450,000, $350,000, $280,000) suggest more data quirks.
- You may want to try remove districts corresponding to these artificial caps to prevent machine learning algorithms from learning these anomalies.



Look for Correlations

- Looking at the **correlation scatterplots**, it seems like the **most promising** attribute to predict the median house value is the **median income**, so you zoom in on their scatterplot

```
housing.plot(kind="scatter", x="median_income", y="median_house_value",
             alpha=0.1, grid=True)
save_fig("income_vs_house_value_scatterplot")  # extra code
plt.show()
```



- The correlation coefficient only measures **linear correlations** ("as x goes up, y generally goes up/down").

Skewness

- **Skewness** is a **measure of the asymmetry** of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive, negative, or undefined. In a perfectly symmetrical distribution, the skewness is zero.

$$\text{Skewness} = \frac{N}{(N-1)(N-2)} \times \sum_{i=1}^{N} \left(\frac{X_i - \bar{X}}{s} \right)^3$$

- **Positive Skewness**: Indicates that the tail on the right side of the distribution is longer or fatter than the left side. In a positively skewed distribution, the mean and median will be greater than the mode.
- in Python, use **scipy.stats.skew()**

Skewness

- You also noticed that some attributes have a **skewed-right** distribution, so you may want to transform them (e.g., by computing their **logarithm** or **square root**)
- The **logarithmic transformation** is particularly effective for data that is positively skewed. It **compresses the long tail** and **stretches out the data towards the left**, making the distribution more **symmetric**. This transformation is useful when the data spans several orders of magnitude.

Experiment with Attribute Combinations

- 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

Experiment with Attribute Combinations

```
housing["rooms_per_house"] = housing["total_rooms"] / housing["households"]
housing["bedrooms_ratio"] = housing["total_bedrooms"] / housing["total_rooms"]
housing["people_per_house"] = housing["population"] / housing["households"]
```

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

```
median_house_value    1.000000
median_income         0.688380
rooms_per_house       0.143663
total_rooms           0.137455
housing_median_age    0.102175
households            0.071426
total_bedrooms        0.054635
population           -0.020153
people_per_house     -0.038224
longitude            -0.050859
latitude             -0.139584
bedrooms_ratio       -0.256397
Name: median_house_value, dtype: float64
```

Experiment with Attribute Combinations

- The new **bedrooms_ratio** attribute is much more correlated with the median house value than the total number of rooms or bedrooms.
- Apparently houses with a lower bedroom/room ratio tend to be more expensive.
- The **number of rooms per household** is also more informative than the total number of rooms in a district—obviously the larger the houses, the more expensive they are.

Remind – for the Upcoming Lecture

```
strat_train_set.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
13096	-122.42	37.80	52.0	3321.0	1115.0	1576.0	1034.0	2.0987	458300.0	NEAR BAY
14973	-118.38	34.14	40.0	1965.0	354.0	666.0	357.0	6.0876	483800.0	<1H OCEAN
3785	-121.98	38.36	33.0	1083.0	217.0	562.0	203.0	2.4330	101700.0	INLAND
14689	-117.11	33.75	17.0	4174.0	851.0	1845.0	780.0	2.2618	96100.0	INLAND
20507	-118.15	33.77	36.0	4366.0	1211.0	1912.0	1172.0	3.5292	361800.0	NEAR OCEAN

Remind – for the Upcoming Lecture

- Revert to a clean training set (by copying **strat_train_set** once again).
- You should also **separate the predictors and the lab**els, since you don't necessarily want to apply the same transformations to the predictors and the target values (note that **drop()** creates a copy of the data and does not affect strat_train_set)

Remind – for the Upcoming Lecture

```
housing = strat_train_set.drop("median_house_value", axis=1)
housing_labels = strat_train_set["median_house_value"].copy()
```

```
housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity
13096	-122.42	37.80	52.0	3321.0	1115.0	1576.0	1034.0	2.0987	NEAR BAY
14973	-118.38	34.14	40.0	1965.0	354.0	666.0	357.0	6.0876	<1H OCEAN
3785	-121.98	38.36	33.0	1083.0	217.0	562.0	203.0	2.4330	INLAND
14689	-117.11	33.75	17.0	4174.0	851.0	1845.0	780.0	2.2618	INLAND
20507	-118.15	33.77	36.0	4366.0	1211.0	1912.0	1172.0	3.5292	NEAR OCEAN

Remind – for the Upcoming Lecture

```
housing = strat_train_set.drop("median_house_value", axis=1)
housing_labels = strat_train_set["median_house_value"].copy()
```

```
housing_labels.head()
```

```
13096    458300.0
14973    483800.0
3785     101700.0
14689     96100.0
20507    361800.0
Name: median_house_value, dtype: float64
```