End-to-End machine learning project

Main steps for machine learning project

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for 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.

Working with real world data

- Popular open data repositories:
  - UC Irvine Machine Learning Repository- *http://archive.ics.uci.edu/ml/index.php*
  - Kaggle datasets - *https://www.kaggle.com/datasets*
  - Amazon’s AWS datasets - *https://registry.opendata.aws/*
- Meta portals (they list open data repositories):
  - *http://dataportals.org/*
  - *http://opendatamonitor.eu/*
  - *http://quandl.com/*
- Other pages listing many popular open data repositories:
  - Wikipedia’s list of Machine Learning datasets - *https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research*
  - Quora.com question - *https://www.quora.com/Where-can-I-find-large-datasets-open-to-the-public*
  - Datasets subreddit - *https://www.reddit.com/r/datasets/*
- General economy, financials, demographics, and NYC opendata
  - *http://www.census.gov/data.html*
  - *https://fred.stlouisfed.org/*
  - *https://opendata.cityofnewyork.us/*

1. Look at the big picture

California housing prices

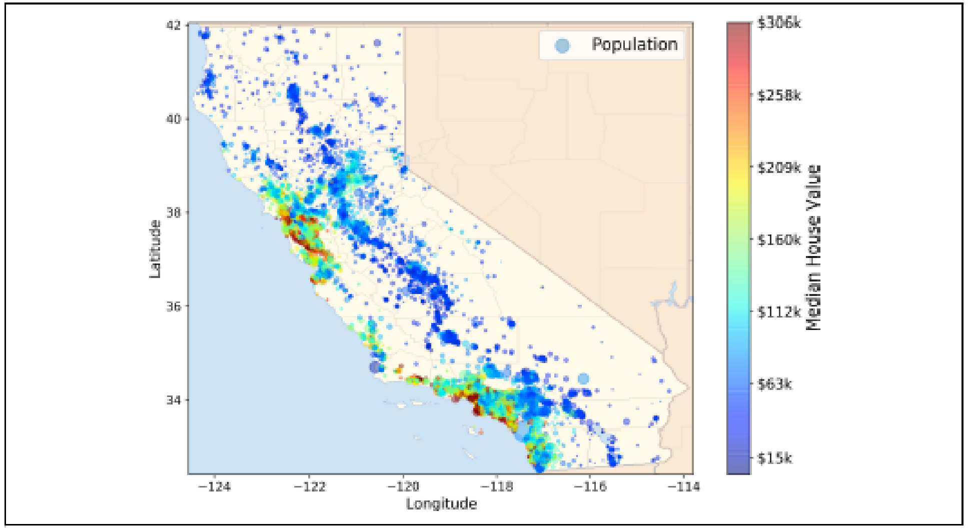


Figure 2-1. California housing prices

- 1990 Census Data: population, median income, median housing price, etc for each block group

Frame the problem

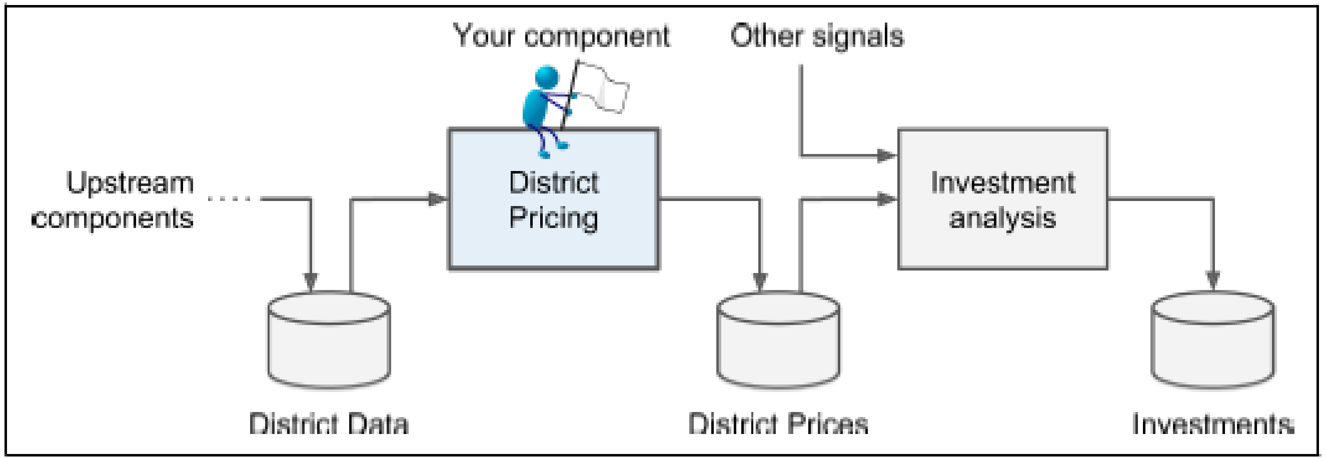


Figure 2-2. A Machine Learning pipeline for real estate investments

○ Importance of your model output to the investment decision

Frame the problem

- What is the business objective?
- What is the status quo for the business on how the problem is being solved?
- Start designing your system
- Frame the problem: is it supervised, unsupervised, or reinforcement learning?
- Is it regression?
- Is it multiple regression?

Select a performance measure

- Root mean square error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

- Mean Absolute Error (MAE)

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

Check the assumptions

- It is a good practice to list and verify the assumptions that were made so far

2. Get the data

Get the data

- Full Jupyter notebook is available at: https://github.com/ageron/handson-ml2
- You can set up your environment follow through page 45 – 48 of the textbook
- Anaconda Navigator: PyCharm, Jupyter, RStudio, Spyder etc https://docs.anaconda.com/anaconda/install/

Download the data

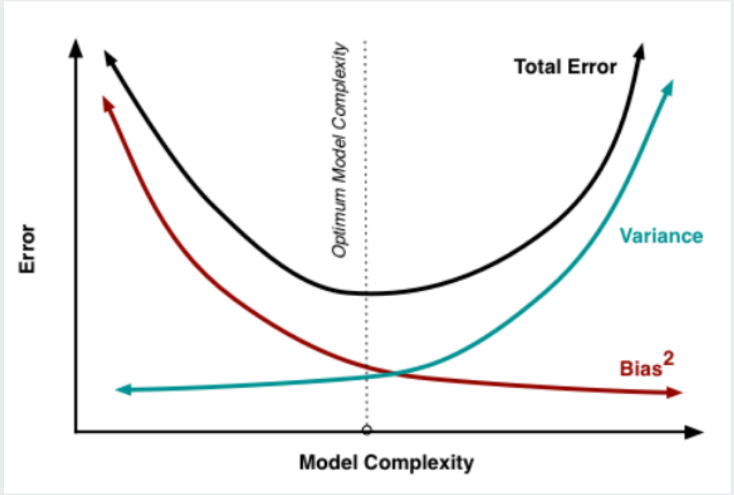
- Use the function fetch_housing_data
- Download the entire GIT repository and put it in your working directory (preferred approach) *https://github.com/ageron/handson-ml2*

Take a quick look at the data structure

- df.head() (head(df) in R), top five rows in the dataframe
- df.info() (struct(df) in R), get quick description of the data
- Pay attention to the missing data on some of the dimensions
- df[“col”].value_counts() (summary(df[“col”]) in R) to see the categorical variable distribution
- df.describe() (summary(df) in R) to see summary statistics for numerical variables
- df.hist() (hist(df) in R) to see the histogram of the data

Create a Test Set

- Training data vs. Testing data



- Understanding the Bias-Variance Tradeoff (fortmann-roe.com)

Create a Test Set

- Random sampling (running the risk that the test data is not a fixed set)
- Solutions:
  - Save the test set so you can reuse it
  - Set random number generator seed
  - Add identifier to the data (preferred way)
- In scikit-learn, you can use train_test_split, can also leverage random_state in the function.

Stratified sampling

○ The population is divided into homogeneous subgroups called *strata*,
and the right number of instances is sampled from each stratum to guarantee that the test set is representative of the overall population.

○ Suppose you chatted with experts who told you that 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.

Stratified sampling

```
housing["income_cat"].hist()
```

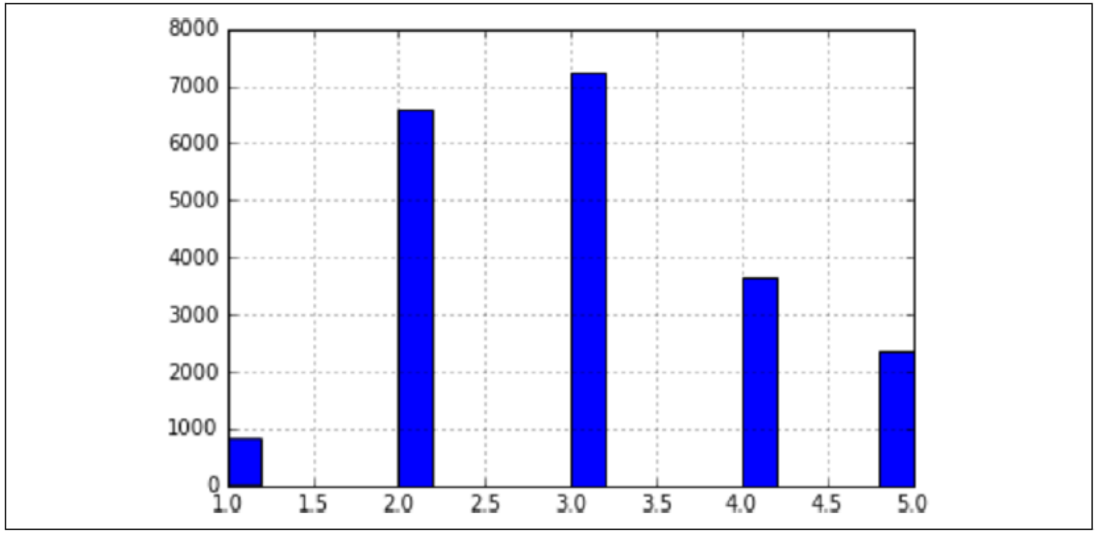


Figure 2-9. Histogram of income categories

3. Discover and visualize the data to gain insights

Visualizing geographical data

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

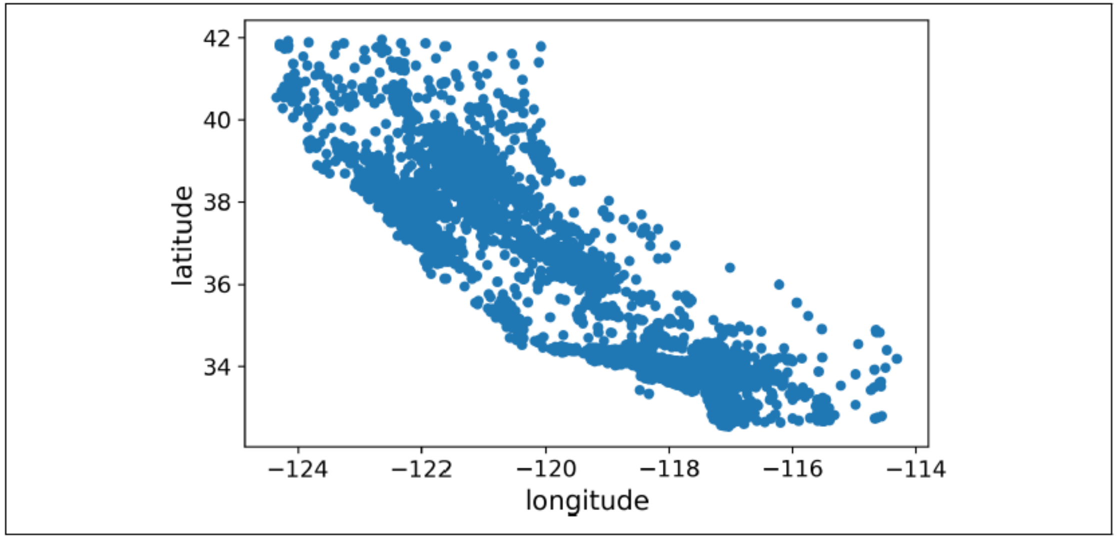


Figure 2-11. A geographical scatterplot of the data

Even better visualization

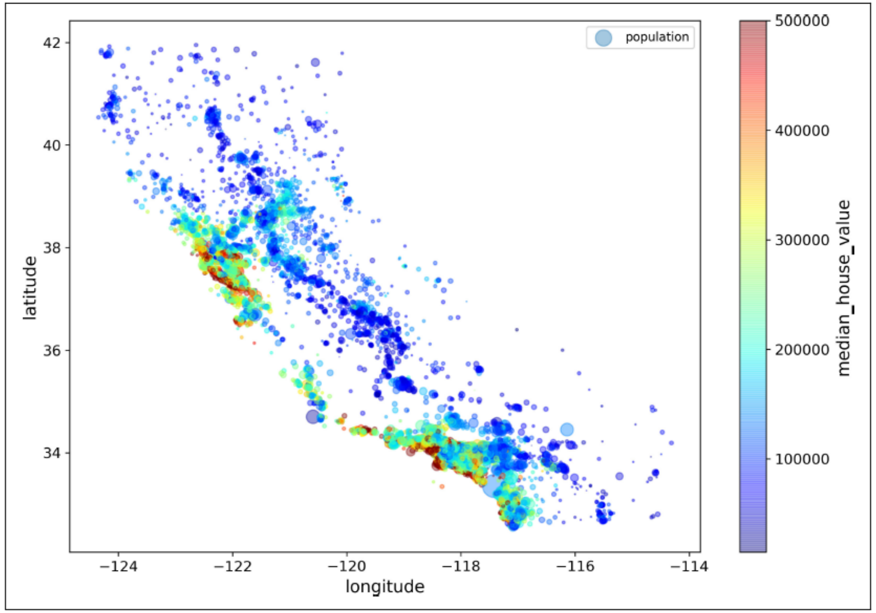
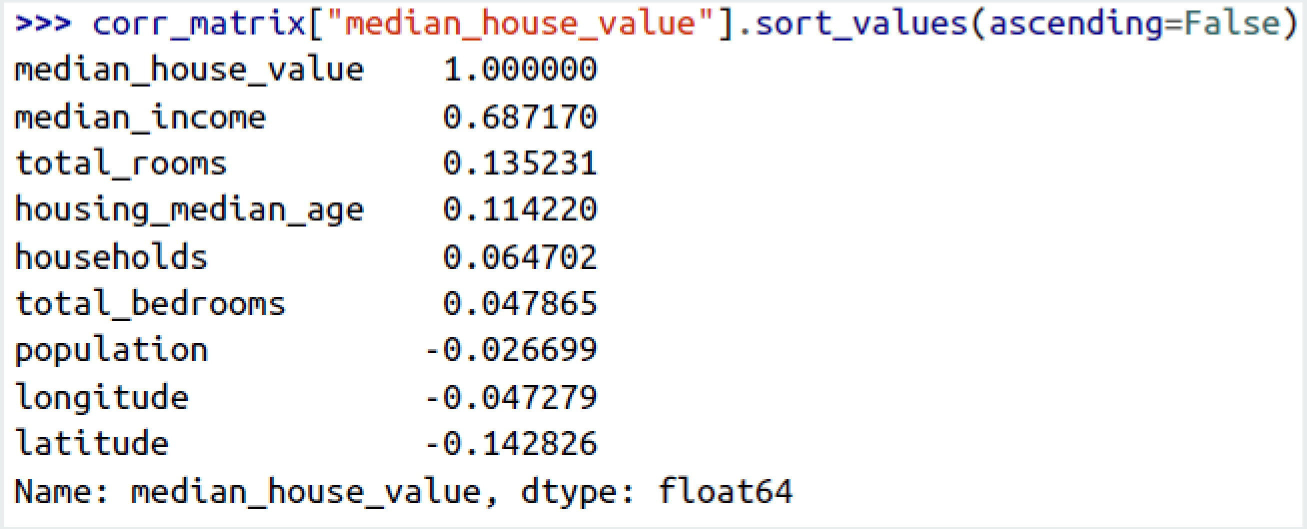


Figure 2-13. California housing prices

Correlation coefficients


```
>>> 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
```

Correlation coefficients

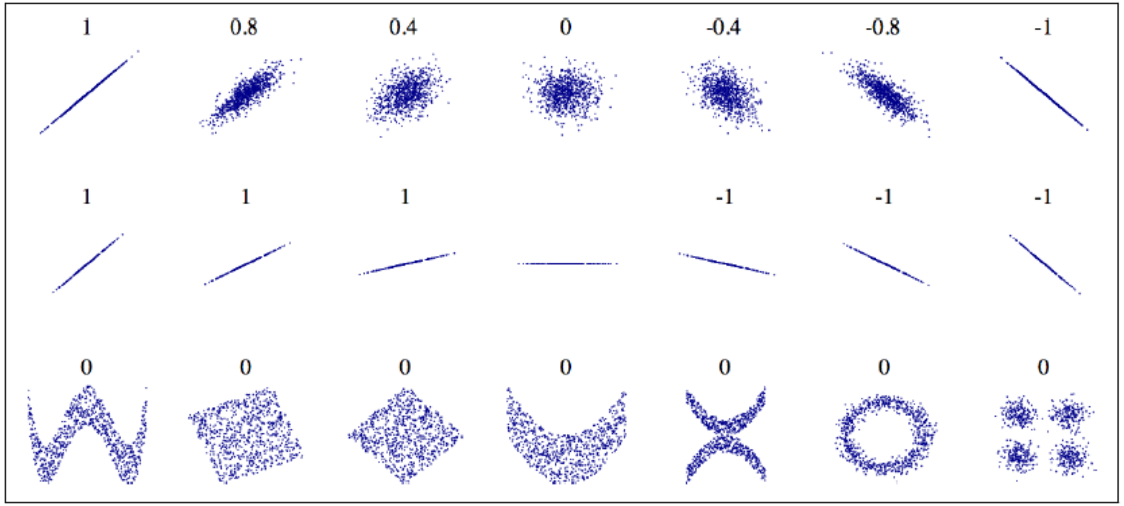


Figure 2-14. Standard correlation coefficient of various datasets (source: Wikipedia; public domain image)

Scatter matrix

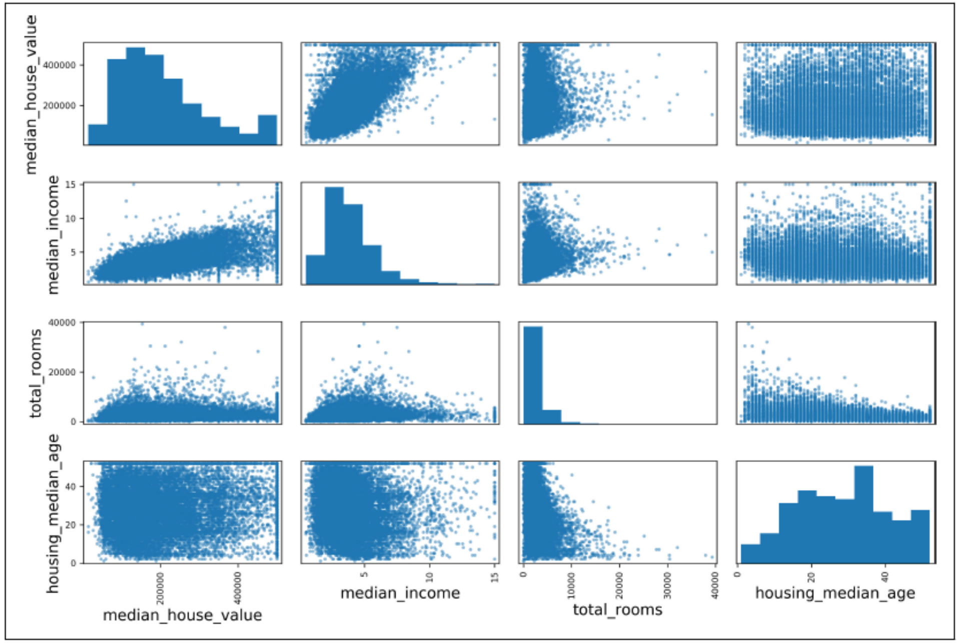
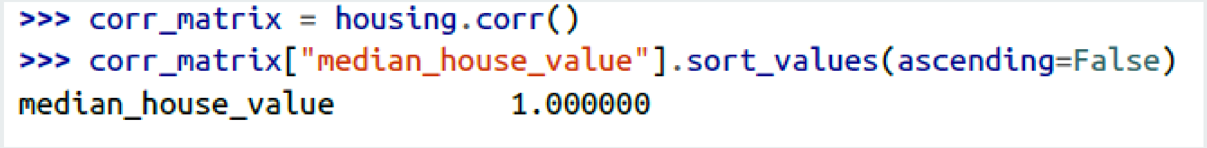
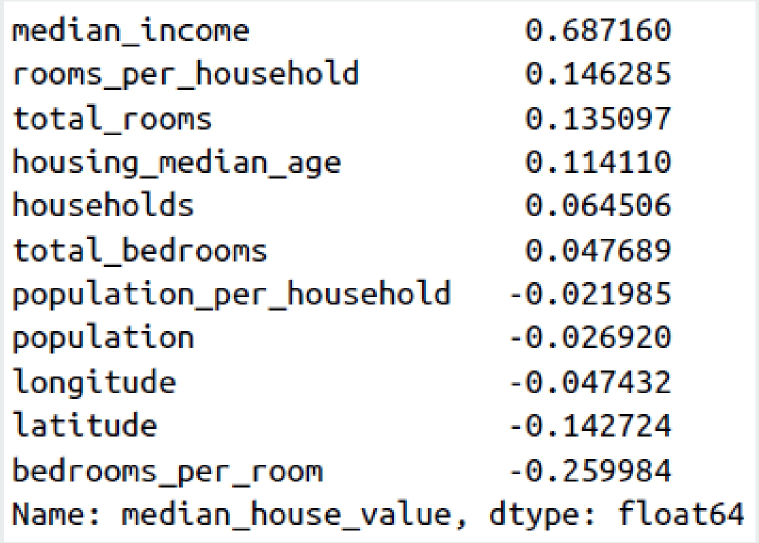


Figure 2-15. Scatter matrix

Derived variables/predictors



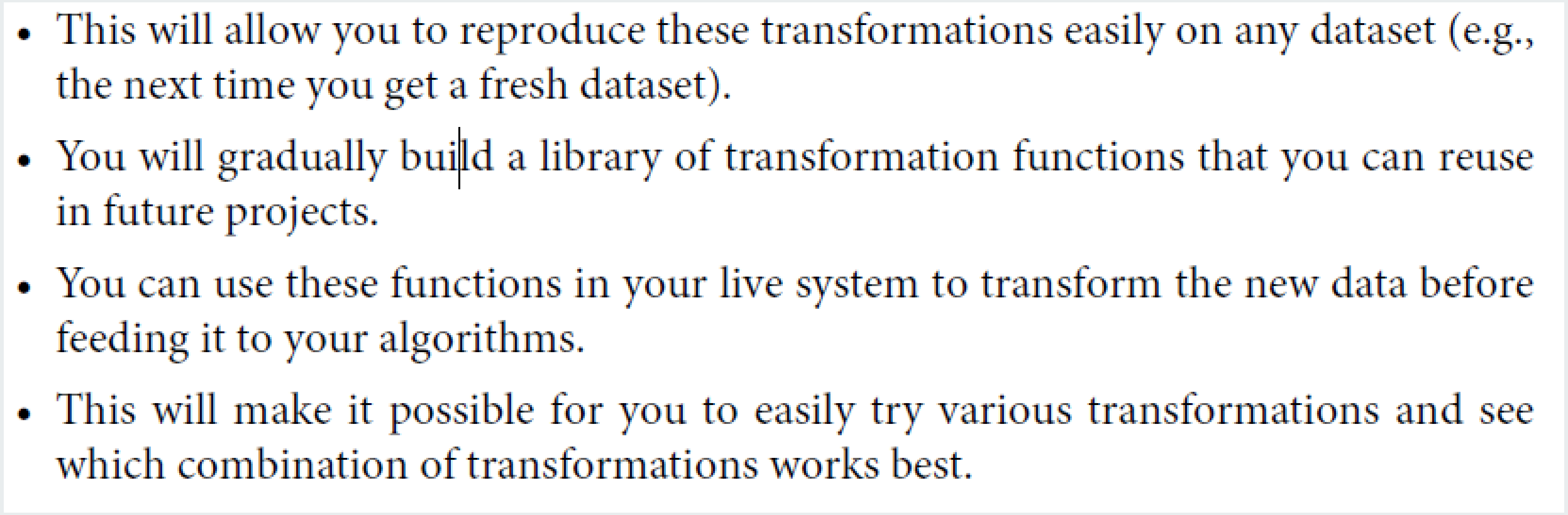
```
>>> corr_matrix = housing.corr()
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value          1.000000
```



```
median_income                0.687160
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
```

4. Prepare the data for machine learning algorithms

Automate data preparation



- This will allow you to reproduce these transformations easily on any dataset (e.g., the next time you get a fresh dataset).
- You will gradually build a library of transformation functions that you can reuse in future projects.
- You can use these functions in your live system to transform the new data before feeding it to your algorithms.
- This will make it possible for you to easily try various transformations and see which combination of transformations works best.

Data cleaning

- **Missing values**
  - Drop the missing values
  - Drop the variable
  - Median/mean to fillna
  - Infer from other variables

Text and categorical attributes/variables

- **Create dummy variables (ocean_proximity)**
  - “<1H OCEAN”
  - “INLAND”
  - 'NEAR BAY‘
  - ……

Custom transformation

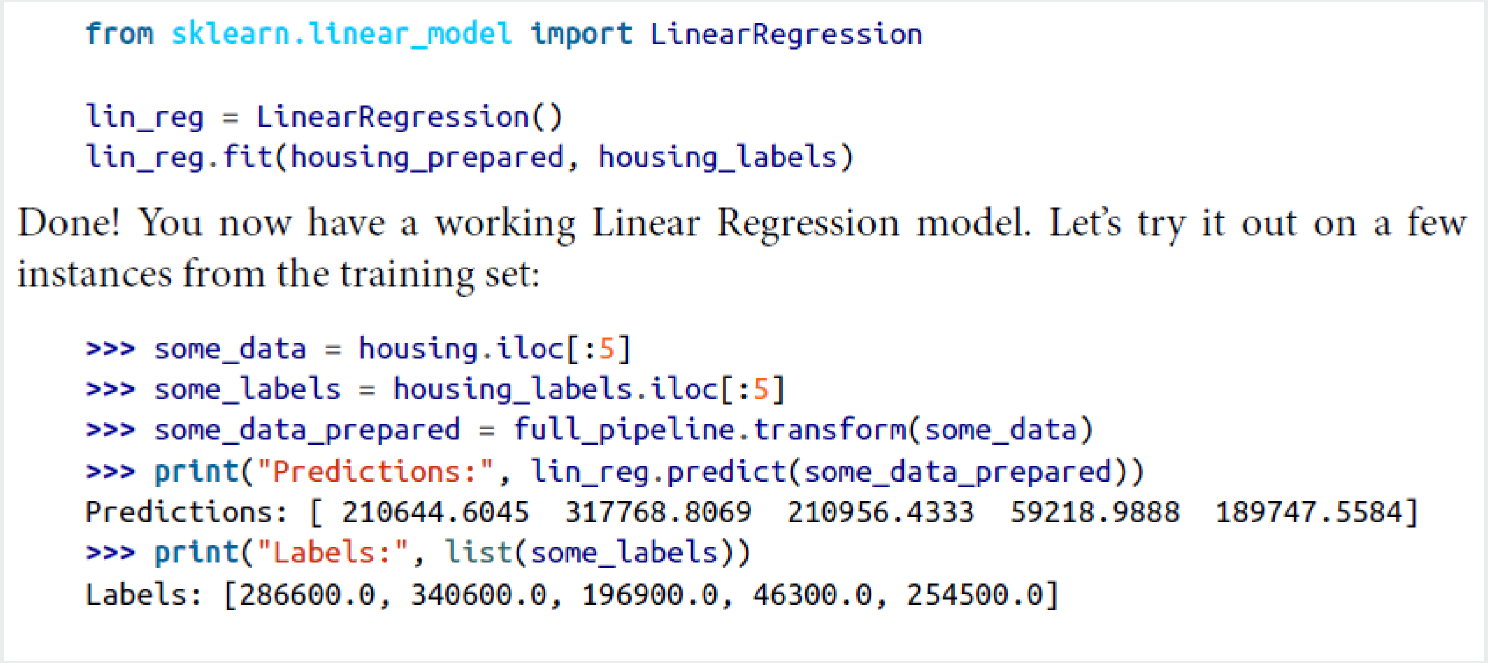
- Automate the process of defining new variables

Feature scaling

- Min-max scaling (normalization): rescale so that value ranging from 0 to 1
- Standardization: first it subtracts the mean value (so standardized values always have a zero mean), and then it divides by the standard deviation so that the resulting distribution has unit variance.

5. Select and train a model

Training and evaluating on the training set



```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)
```

Done! You now have a working Linear Regression model. Let's try it out on a few instances from the training set:

```
>>> some_data = housing.iloc[:5]
>>> some_labels = housing_labels.iloc[:5]
>>> some_data_prepared = full_pipeline.transform(some_data)
>>> print("Predictions:", lin_reg.predict(some_data_prepared))
Predictions: [ 210644.6045  317768.8069  210956.4333  59218.9888  189747.5584]
>>> print("Labels:", list(some_labels))
Labels: [286600.0, 340600.0, 196900.0, 46300.0, 254500.0]
```

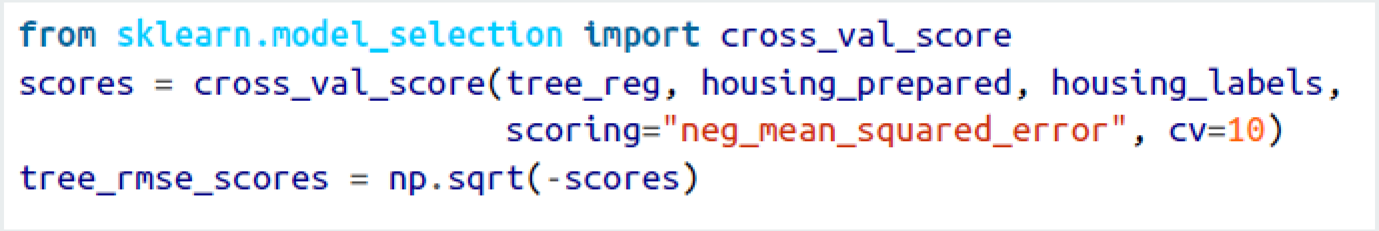
Training and evaluating on the training set

- Linear regression model?
- Decision tree model?
- Random forest model?

Better evaluation using cross-validation

- Sikit-Learn’s k-fold cross validation

The following code randomly splits the training set into 10 distinct subsets called *folds*, then it trains and evaluates the Decision Tree model 10 times, picking a different fold for evaluation every time and training on the other 9 folds.

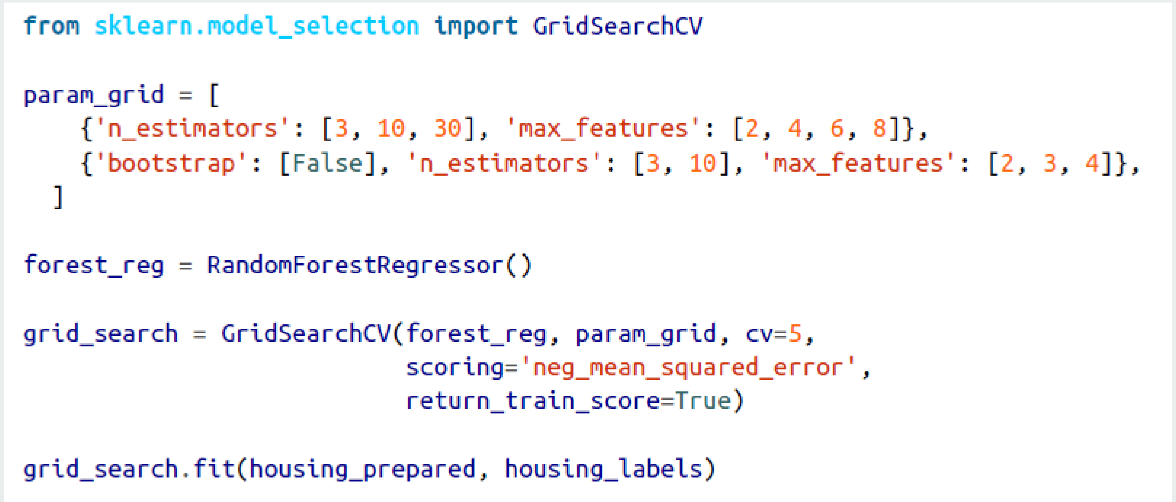


```python
from sklearn.model_selection import cross_val_score
scores = cross_val_score(tree_reg, housing_prepared, housing_labels,
                         scoring="neg_mean_squared_error", cv=10)
tree_rmse_scores = np.sqrt(-scores)
```

6. Fine-tune your model

Fine-tune your model

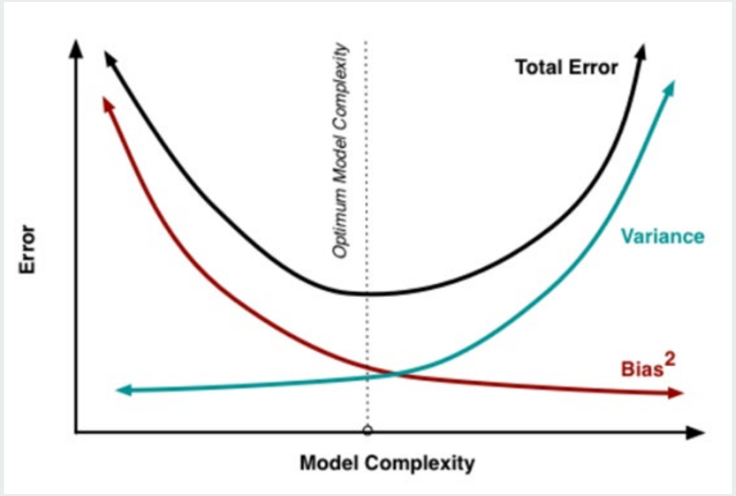
- Hyperparameter (Scikit-Learn’s GridSearchCV): RandomForestRegressor as an example.



```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(housing_prepared, housing_labels)
```

Evaluate your system on the test set

- Evaluate the final model on the test set



7. Present your results

7. Present your results

- Present your results to technical audience
- Present your results to non-technical audience
- Present your results to senior management
- Obtain approval for putting the model into production

8. Launch, monitor, and maintain your system

Launch, monitor, and maintain your system

- Productionize your model
- Monitor model performance
- Maintain your system
- Improve your model and system as needed

Recap: Main steps for machine learning project

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for 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.