CHƯƠNG 2:
DỰ ÁN MÁY HỌC

Khoa Khoa học và Kỹ thuật thông tin
Bộ môn Khoa học dữ liệu

NỘI DUNG

1. Xác định bối cảnh và định nghĩa bài toán.
2. Thu thập dữ liệu.
3. Khám phá dữ liệu.
4. Chuẩn bị dữ liệu.
5. Huấn luyện mô hình.
6. Tinh chỉnh mô hình.
7. Triển khai mô hình.

1. XÁC ĐỊNH BỐI CẢNH VÀ ĐỊNH NGHĨA BÀI TOÁN

MỤC TIÊU

- Phát biểu được bài toán / tác vụ máy học sẽ thực hiện.
- Công việc thực hiện:
  + Xác định đầu vào.
  + Xác định đầu ra.
  + Xác định loại bài toán máy học.
  + Xác định độ đo đánh giá.

Xác định bài toán

VD: Xây dựng mô hình dự đoán giá nhà tại bang California dựa trên dữ liệu điều tra dân số của bang này.

- Dữ liệu này có các số liệu (thuộc tính): *dân số, thu nhập trung bình, giá nhà trung bình,...* cho mỗi *khu vực* tại bang California.
- Bài toán:
  + Đầu vào: 1 khu vực bất kỳ và các dữ kiện khác đi kèm.
  + Đầu ra: Giá nhà trung bình.

Bối cảnh sử dụng

- Hệ thống này sẽ được dùng như thế nào?
- Đầu ra sẽ thế nào?
- Phương pháp thực hiện là gì ?
- Hệ thống tin cậy tới mức nào ?

Bối cảnh sử dụng

- Hệ thống này sẽ được dùng như thế nào?
  - →Trả lời: dùng để đưa ra dự đoán giá nhà trung bình dựa trên các dữ kiện đầu vào ➔ **hỗ trợ** cho các nhà đầu tư.
- Đầu ra sẽ thế nào?
  - →Trả lời: Đầu ra là một giá trị thể hiện giá nhà trung bình.
- Phương pháp thực hiện là gì ?
  - →Trả lời: **Phương pháp hồi quy (regression).**

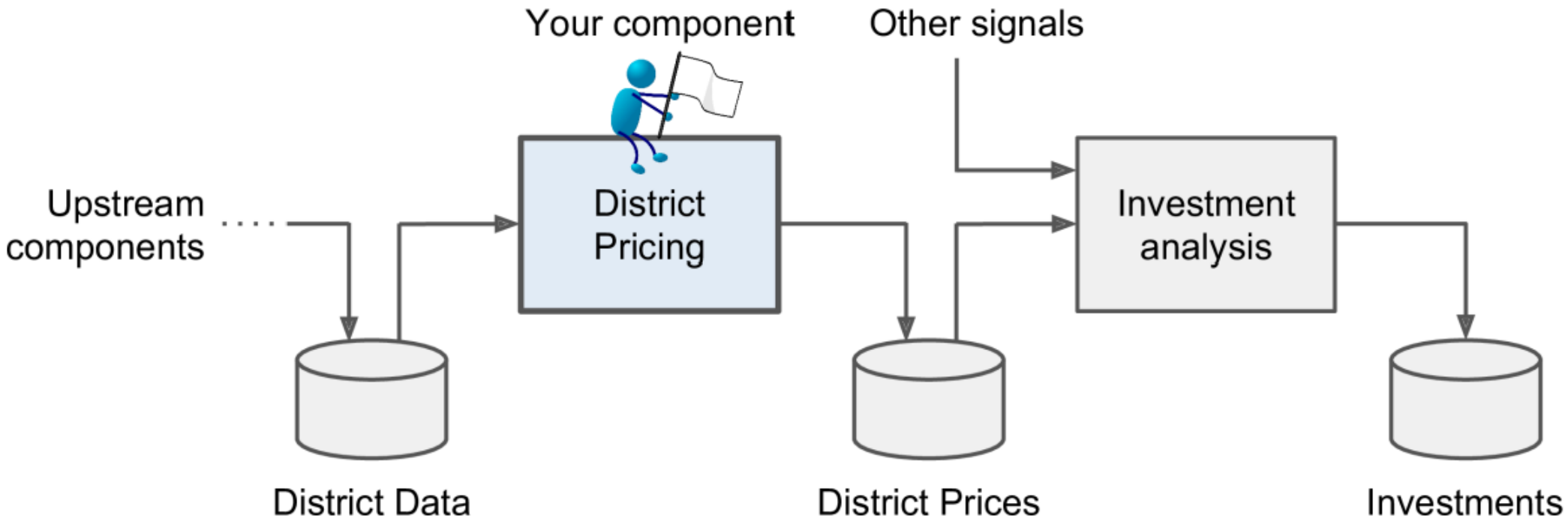
Bối cảnh sử dụng

– Hệ thống tin cậy tới mức nào?

→Trả lời: Đo lường độ tin cậy của hệ thống dựa trên dữ liệu kiểm thử bằng một độ đo cụ thể là RMSE (Root-Mean-Square Error).

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

Pipeline for ML process



2. THU THẬP DỮ LIỆU

MỤC TIÊU

- Thu thập dữ liệu phục vụ cho việc huấn luyện và kiểm tra mô hình máy học.
- Yêu cầu:
  + Nguồn dữ liệu có tin cậy hay không?
  + Dữ liệu phù hợp với bài toán:
    - Miền (domain) của dữ liệu.
    - Biến target của dữ liệu (đối với bài toán học có giám sát).

Ví dụ

Bài toán: Xây dựng mô hình dự đoán giá nhà tại bang California dựa trên dữ liệu điều tra dân số của bang này.

- Bộ dữ liệu: **California Housing**
- Nguồn tải: https://github.com/ageron/handson-ml/tree/master/datasets/housing
- Định dạng: CSV.
- Publication: Pace, R. K., & Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, *33*(3), 291-297.

Ý nghĩa từng thuộc tính

1. longitude: kinh độ.
2. latitude: vĩ độ.
3. housingMedianAge: tuổi của ngôi nhà.
4. totalRooms: Tổng số phòng.
5. totalBedrooms: Số phòng ngủ.
6. population: tổng dân số cư trú.
7. households: tổng số hộ gia đình.
8. medianIncome: Thu nhập trung bình của từng hộ gia đình.
9. medianHouseValue: Giá nhà.
10. ocean_proximity: Vị trí nhà (trong đất liền, giáp biển, giáp vịnh, trên đảo).

Đọc dữ liệu

- Sử dụng thư viện pandas.

  pandas là thư viện dùng để thao tác trên các dataframe. pandas cung cấp rất nhiều hàm hỗ trợ cho việc đọc/xuất dữ liệu, trích xuất cột/dòng, thống kê, ... trên dữ liệu.

- Load thư viện pandas:

```
import pandas as pd
```

- Đọc dữ liệu:

```
data = pd.read_csv('đường dẫn tới file csv')
```

3. KHÁM PHÁ DỮ LIỆU

Mục tiêu

- Hiểu và có cái nhìn tổng quan về bộ dữ liệu.
- Các thông tin cần phải nắm rõ:
  + Kích thước dữ liệu: số thuộc tính, số dòng (số mẫu) dữ liệu.
  + Thuộc tính nào là thuộc tính dự đoán?
  + Đặc điểm của giá trị của các thuộc tính:
    - Loại giá trị: *numerical, categorical, text, character, ...*
    - Đặc điểm của giá trị: *phân bố, miền giá trị, độ dài, ...*

Tổng quan dữ liệu

- Bộ dữ liệu: **California Housing.**
- Số lượng thuộc tính: 10.
- Số lượng dữ liệu (số mẫu dữ liệu): 20,640.

Xem thông tin 10 dòng dữ liệu đầu tiên

– Sử dụng lệnh head():

```
data.head(10)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	299200.0
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200	241400.0
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	261100.0

Xem thông tin các thuộc tính

– Sử dụng lệnh: info()

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Thông số cho các thuộc tính số

- Có 2 dạng giá trị đối với dữ liệu:
  + Giá trị categorical (tạm dịch: giá trị phân loại)
  + Giá trị số (numerical).
- Các thông số cần quan tâm đối với thuộc tính số:
  + Giá trị trung bình (mean).
  + Giá trị lớn nhất (max).
  + Giá trị nhỏ nhất (min).
  + Giá trị độ lệch chuẩn (std - standard deviation).
  + Tứ phân vị (quantile).

Xem thông tin đối với các thuộc tính số

– Sử dụng hàm describe()

```
data.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100

Trực quan hoá các thông số

- Thư viện sử dụng: matplotlib

  matplotlib là thư viện chuyên dụng để vẽ đồ thị trên ngôn ngữ Python. Thư viện này hỗ trợ nhiều loại đồ thị khác nhau như: barplot (biểu đồ cột), lineplot (biểu đồ đường),…

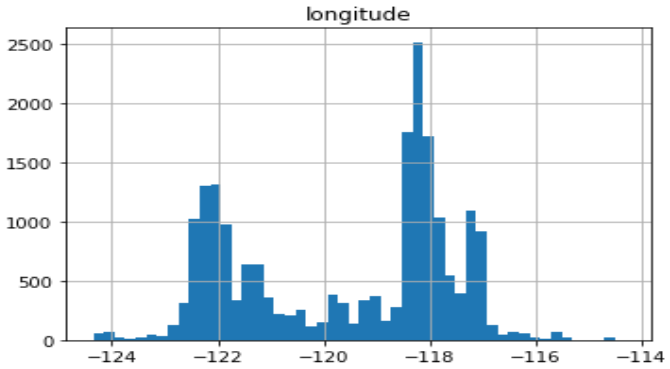
- Load thư viện:

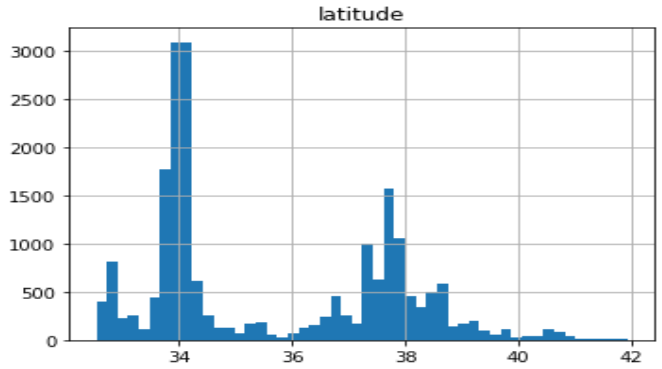
```
import matplotlib.pyplot as plt
```

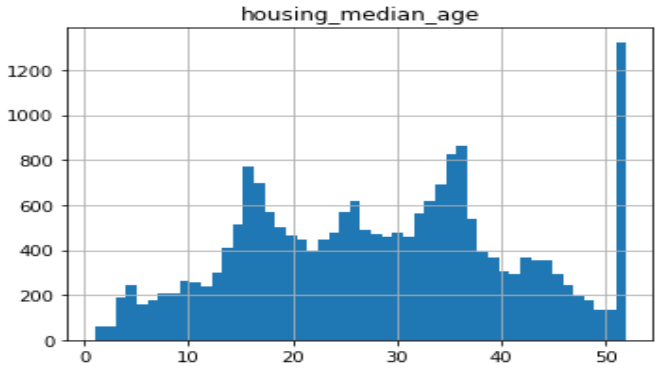
- Trực quan hoá dữ liệu dạng số:

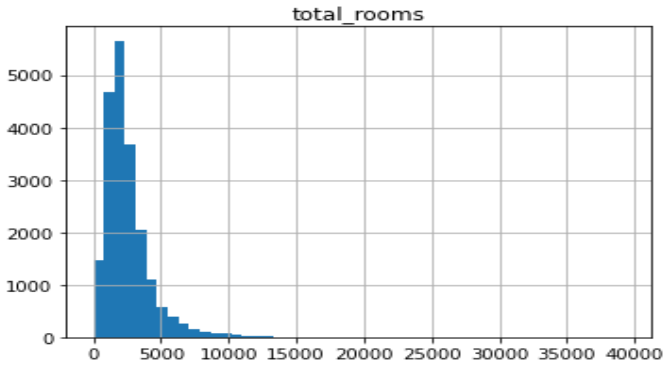
```
1. data.hist(bins=50, figsize=(20,15))
2. plt.show()
```

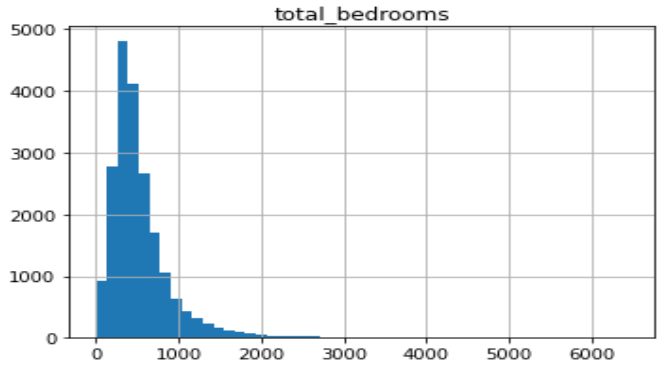
Kết quả trực quan hoá dữ liệu

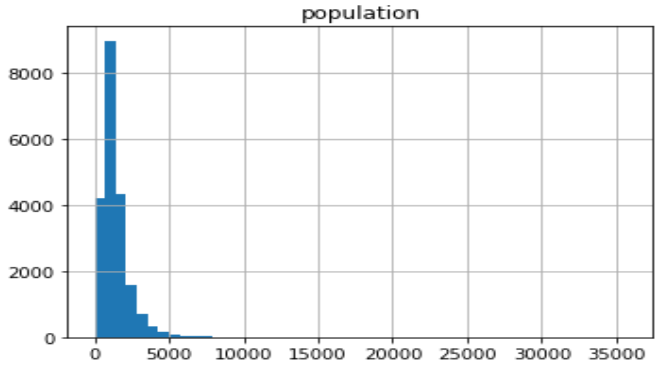


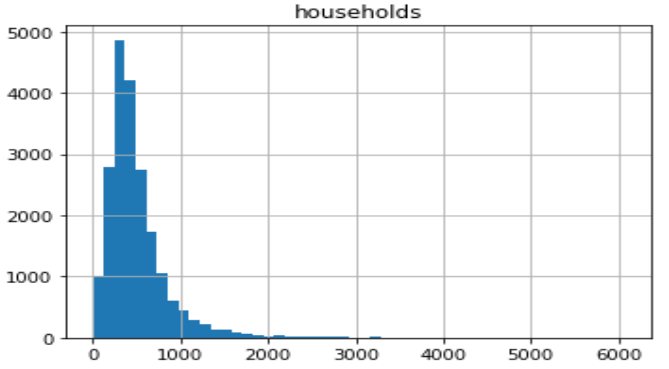


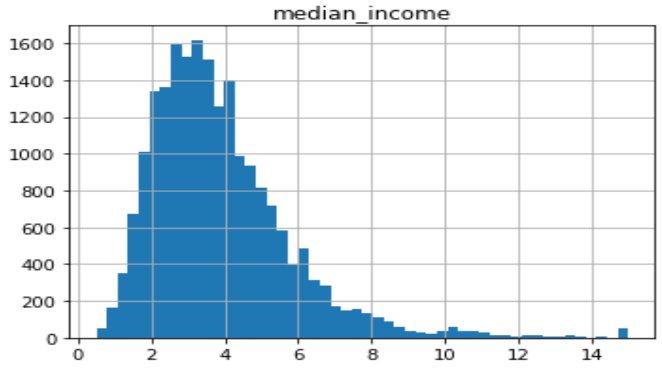


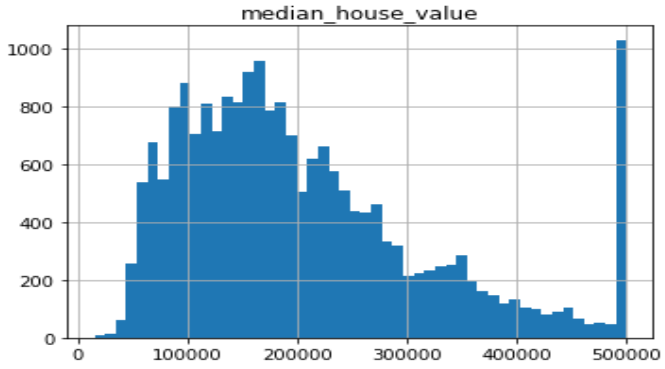












Một số nhận xét

- Phân bố dữ liệu: các thuộc tính có dạng phân bố chuẩn (dạng hình chuông) hay không?
- Các thuộc tính nào bị lệch? Lệch về phía nào?
- Khoảng giá trị (scale) của các thuộc tính như thế nào? Có phải các thuộc tính đều có cùng khoảng giá trị?
- Có dữ liệu dạng N/A hay không?

Lưu ý: Sinh viên dựa vào thông tin về các thuộc tính số và đồ thị trực quan để trả lời các câu hỏi trên.

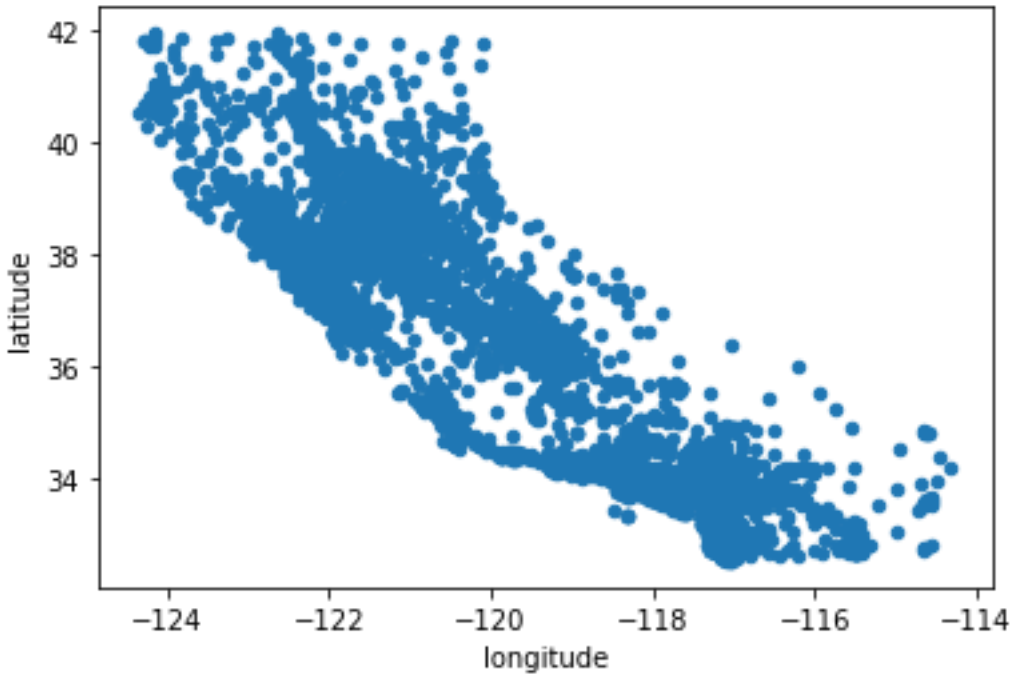
Trực quan hoá dữ liệu địa lý

- Hai thuộc tính:
  + longitude (kinh độ).
  + latitude (vĩ độ).
- Thư viện sử dụng: matplotlib.
- Loại biểu đồ: scatter plot (tạm dịch: biểu đồ phân tán/biểu đồ tán xạ).
- Thực hiện trực quan hoá dữ liệu:

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

Kết quả

Trực quan hoá từ dữ liệu



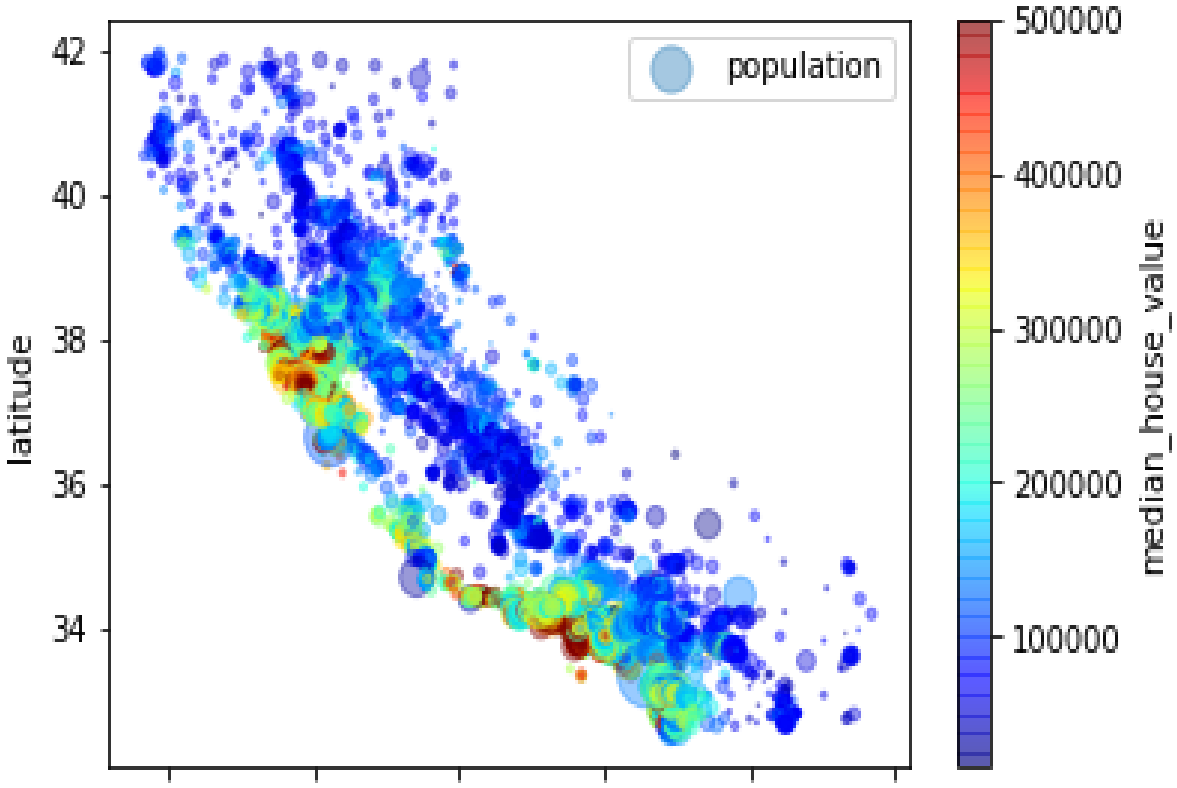
Dữ liệu thực tế của bang California

Trực quan hoá dữ liệu giá nhà theo khu vực

- Câu hỏi đặt ra: giá nhà có phụ thuộc vào khu vực không (VD: gần trung tâm, gần biển).
- Trực quan hoá dữ liệu: sử dụng biểu đồ màu (color map).
- Thuộc tính sử dụng:
  + longitude, latitude: Trực quan hoá vị trí bằng scatter plot.
  + median_house_value: Trực quan hoá giá nhà theo thang màu.
- Thực hiện:

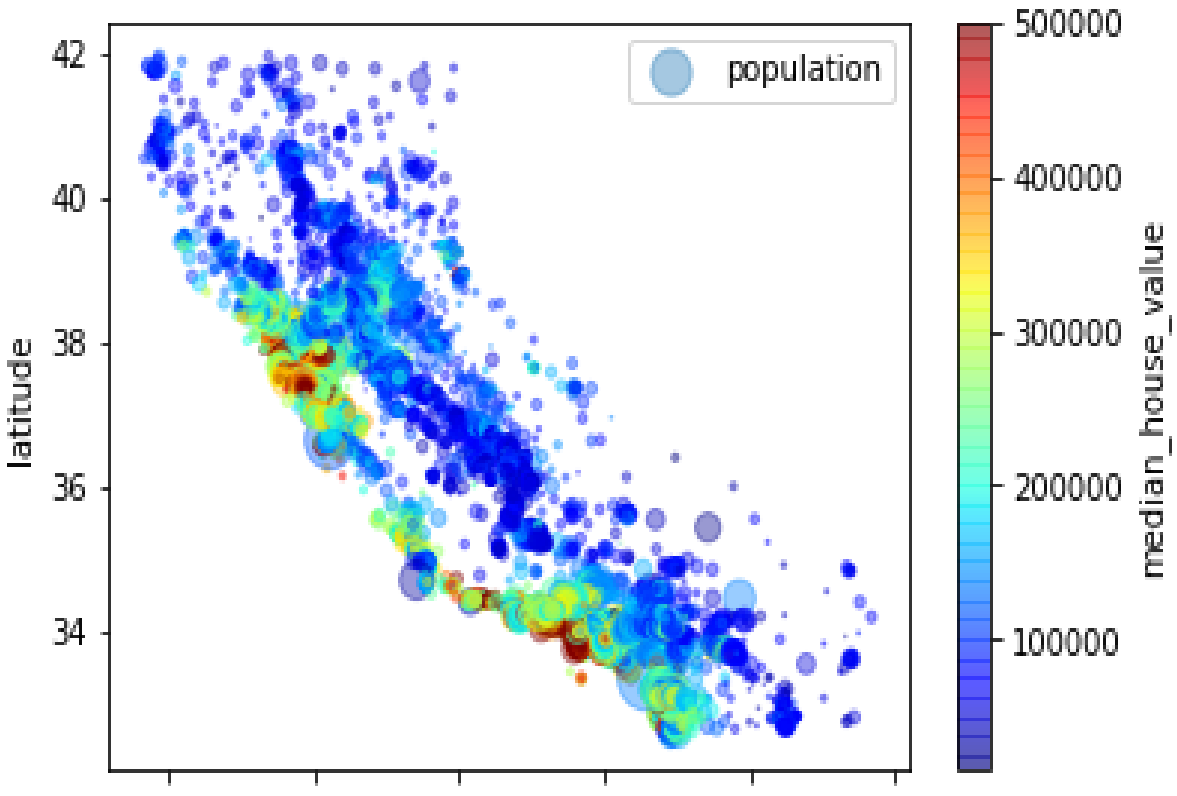
```
1. data.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
             s=data["population"]/100, label="population",
             c="median_house_value", cmap=plt.get_cmap("jet"),
             colorbar=True,
   )
2. plt.legend()
```

Kết quả



1. Giá nhà có phụ thuộc vào vị trí không?
2. Vị trí nào có giá nhà đắt nhất?
3. Giữa giá nhà và dân số có mối quan hệ nào không?

Kết quả



1. Giá nhà có phụ thuộc vào vị trí không?
   + Vị trí nhà có ảnh hưởng tới giá nhà.
2. Vị trí nào giá nhà đắt nhất ?
   + Gần biển và trên đảo.
3. Giá nhà và dân số có mối quan hệ nào không?
   + Khu vực đông dân thì giá nhà cao.

4. CHUẨN BỊ DỮ LIỆU

Mục tiêu

- Chuẩn bị dữ liệu để huấn luyện và kiểm thử cho các mô hình máy học.
- Các yêu cầu về làm sạch dữ liệu:
  + Xử lý dữ liệu thiếu.
  + Mã hoá dữ liệu.
  + Chuẩn hoá dữ liệu.

Xử lý thuộc tính bị thiếu giá trị

- Các thuộc tính bị thiếu giá trị (missing value) sẽ làm cho hiệu năng của các mô hình máy học bị giảm xuống.
- Cách xử lý đối với các thuộc tính bị thiếu giá trị:
  1. Bỏ đi các dòng (mẫu) dữ liệu bị thiếu.
  2. Bỏ đi cả thuộc tính.
  3. Điền giá trị thiếu:
     - Điền theo giá trị tương tự.
     - Điền theo giá trị trung bình hoặc trung vị.
     - Điền bằng tay.

Liệt kê các thuộc tính bị thiếu giá trị

– Thực hiện câu lệnh sau:

```
data.isna().sum()
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms      207
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

Xử lý bằng cách điền giá trị theo trung vị (median)

- Thư viện sử dụng: sklearn.

  sklearn là thư viện máy học được xây dựng dùng cho ngôn ngữ python. Thư viện này hỗ trợ rất nhiều các hàm và module dùng để xử lý dữ liệu, huấn luyện mô hình và kiểm tra mô hình.

- Điền dữ liệu thiếu theo trung vị: sử dụng module SimpleImputer.

Các bước thực hiện

- Bước 1: Khai báo lớp SimpleImputer, sử dụng phương pháp median (trung vị).

```
1. from sklearn.impute import SimpleImputer
2. imputer = SimpleImputer(strategy='median')
```

- Bước 2: Loại bỏ đi thuộc tính ocean_proximity ra khỏi dataframe do đây không phải thuộc tính số (**trung vị chỉ tính toán được trên giá trị số**).

```
housing_num = data.drop("ocean_proximity", axis=1)
```

- Bước 3: Huấn luyện lớp SimpleImputer.

```
imputer.fit(housing_num)
```

- Bước 4: Áp dụng vào bộ dữ liệu.

```
X = imputer.transform(housing_num)
housing_tr = pd.DataFrame(X, columns=housing_num.columns)
```

Kết quả thực hiện

Trước khi xử lý

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms      207
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

Sau khi xử lý

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
dtype: int64
```

Mã hoá thuộc tính có giá trị categorical

- Mục tiêu: chuyển các thuộc tính có giá trị là text hoặc string thành giá trị số.
- VD: X = [NEAR_BAY, INLAND, ISLAND]
  => X' = [0, 1, 2]
- Thư viện hỗ trợ: LabelEncoder
- Thực hiện:

```
1. from sklearn.preprocessing import LabelEncoder
2. encoder = LabelEncoder()
3. ocean_proximity = data["ocean_proximity"]
4. housing_cat_encoded = encoder.fit_transform(ocean_proximity)
```

CHUẨN HOÁ DỮ LIỆU

- Chuẩn hoá dữ liệu nhằm mục tiêu biến đổi giá trị dữ liệu về các miền giá trị nhỏ hơn nhằm tăng khả năng xử lý của các thuật toán phân lớp.
- Các phương pháp chuẩn hoá thường dùng:
  + Chuẩn hoá *Min-max*.
  + Chuẩn hoá *z-score*.

CHUẨN HOÁ DỮ LIỆU

– Chuẩn hoá Min-max cho thuộc tính A:

$$v'_i = \frac{v_i - minA}{\max(A) - \min(A)}(new\_\max(A) - new\_\min(A)) + new\_\min(A)$$

– Trong đó:

+ v'_i: giá trị mới của dữ liệu.

+ v_i: giá trị hiện tại của dữ liệu.

+ new_min(A), new_max(A): giá trị miền dữ liệu mới.

+ min(A), max(A): giá trị của miền dữ liệu hiện tại.

CHUẨN HOÁ DỮ LIỆU

– VD: Cho A = [34, 30, 50, 42]. Chuẩn hoá thuộc tính A về miền dữ liệu [1, 10].

max(A) = 50.

min(A) = 30.

new_max(A) = 10.

new_min(A) = 1.

$$v'_1 = \frac{34-30}{50-30}(10-1)+1 = 2.8.$$

$$v'_2 = \frac{30-30}{50-30}(10-1)+1 = 1.$$

$$v'_3 = \frac{50-30}{50-30}(10-1)+1 = 10.$$

$$v'_2 = \frac{42-30}{50-30}(10-1)+1 = 6.4.$$

=> A’ = [2.8, 1, 10, 6.4]

CHUẨN HOÁ DỮ LIỆU

– Chuẩn hoá z-score cho thuộc tính A.

$$v'_i = \frac{v_i - \bar{A}}{\sigma_A}$$

– Trong đó:

+ v'_i: giá trị mới của dữ liệu.

+ v_i: giá trị hiện tại của dữ liệu.

+ $\bar{A}$: giá trị trung bình thuộc tính A.

+ σ_A: độ lệch chuẩn (dùng để đo độ biến động giá trị của 1 biến).

$$\sigma_A = \sqrt{\frac{\sum_{i=1}^{N}(v_i - \bar{A})^2}{N}}$$

CHUẨN HOÁ DỮ LIỆU

– VD: Cho A = [34, 30, 50, 42]. Chuẩn hoá thuộc tính A về miền dữ liệu [1, 10].

$$\bar{A} = \frac{34+30+50+42}{4} = 39$$

$$\sigma_A = \sqrt{\frac{(34-39)^2+(30-39)^2+(50-39)^2+(42-39)^2}{4}} = 7.68$$

$$v'_1 = \frac{34-39}{7.68} = -0.65.$$

$$v'_2 = \frac{30-39}{7.68} = -1.17.$$

$$v'_3 = \frac{50-39}{7.68} = 1.43.$$

$$v'_4 = \frac{42-39}{7.68} = 0.39.$$

=> A' = [-0.65, -1.17, 1.43, 0.39]

Chuẩn hoá dữ liệu

- Chuẩn hoá MinMax: sử dụng module MinMaxScaler()
- Chuẩn hoá z-score: sử dụng module StandardScaler()

Sử dụng pipline

```
1. from sklearn.pipeline import Pipeline
2. from sklearn.preprocessing import StandardScaler
3. housing_num = data.drop("ocean_proximity", axis=1)
   num_pipeline = Pipeline([
                ('imputer', SimpleImputer(strategy='median')),
                ('std_scaler', StandardScaler()),
            ])
4. housing_num_tr = num_pipeline.fit_transform(housing_num)
```

Kết hợp các thao tác lại với nhau

- Kết hợp `housing_num_tr` và `housing_cat_encoded` lại với nhau.

```
data_pre_process = np.append(
    housing_num_tr, axis = 1,
    housing_cat_encoded.reshape(-1, 1)
)
```

- Xem số chiều của `data_pre_process`:

```
data_pre_process.shape
```

5. HUẤN LUYỆN MÔ HÌNH

MỤC TIÊU

- Huấn luyện mô hình dự đoán cho bài toán.
- Các công việc cần làm:
  + Xác định loại bài toán: hồi quy, phân lớp,…
  + Chọn mô hình phù hợp cho từng loại bài toán.
  + Phân chia dữ liệu huấn luyện.
  + Kiểm tra hiệu năng của mô hình.

Bài toán dự đoán giá nhà

- Loại bài toán áp dụng: Hồi quy (regression).
- Mô hình áp dụng: Hồi quy tuyến tính (Linear regression).
  + Input: Các thông số về giá nhà.
  + Output: Giá nhà dự kiến.
- Chia dữ liệu thành 2 biến như sau:
  + Biến X: các thuộc tính trong bộ dữ liệu.
  + Biến y: thuộc tính đích (target) – thuộc tính median_house_value.

Chia dữ liệu thành X và y

– Chia dữ liệu:

```
y = data_pre_process[:, -1]

X = np.delete(data_pre_process, -1, axis=1)
```

Phân chia dữ liệu

- Phân chia dữ liệu:
  - + Dữ liệu huấn luyện (training): dùng để huấn luyện cho mô hình.
  - + Dữ liệu kiểm thử (test): dùng để kiểm tra khả năng dự đoán của mô hình.
- Thông thường, dữ liệu huấn luyện và dữ liệu kiểm thử sẽ có tỉ lệ là 80 : 20 (Train : Test = 80 : 20).
- Phân chia dữ liệu: sử dụng hàm train_test_split() trong sklearn.

```
1. from sklearn.model_selection import train_test_split
2. X_train, X_test, y_train, y_test = train_test_split(X,
   y, test_size=0.2, random_state=42)
```

Huấn luyện mô hình

- Huấn luyện mô hình trên tập dữ liệu huấn luyện (training).
- Thư viện sử dụng: *LinearRegression*.

```
1. from sklearn.linear_model import LinearRegression
2. lin_reg = LinearRegression()
3. lin_reg.fit(X_train, y_train)
```

Kiểm tra mô hình

- Kiểm tra mô hình trên tập dữ liệu kiểm thử (test).
- Mục tiêu của kiểm tra: đánh giá khả năng dự đoán chính xác của mô hình bằng cách so sánh kết quả dự đoán bởi mô hình (nhãn dự đoán) với kết quả thật (truth label) ➔ độ đo đánh giá.
- Đối với bài toán hồi quy: Sử dụng độ đo **Root Mean Square Error** (RMSE).

```
1.from sklearn.metrics import mean_squared_error
2.y_predictions = lin_reg.predict(X_test)
3.mse = mean_squared_error(y_test, y_predictions)
4.rmse = np.sqrt(mse)
```

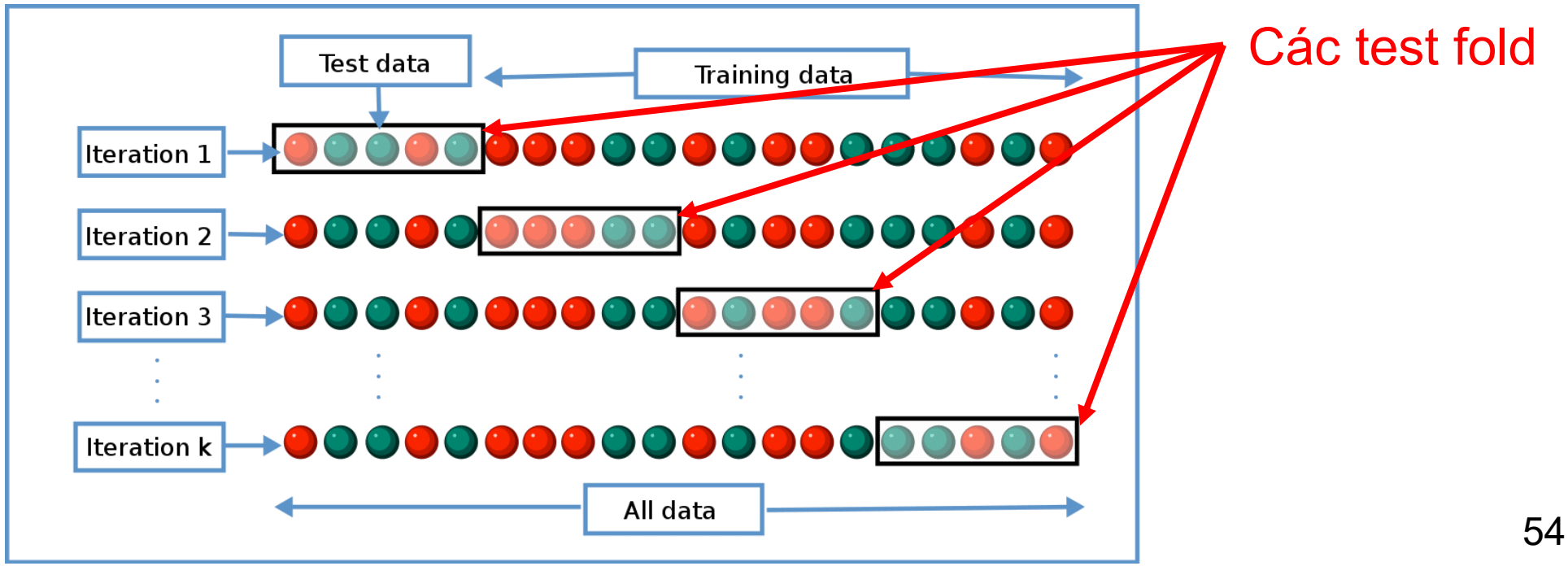
Lưu mô hình đã huấn luyện

- Sau khi huấn luyện, cần lưu lại mô hình đã huấn luyện để sử dụng cho lần sau, hoặc triển khai hệ thống.
- Nhớ lưu lại mô hình sau mỗi lần huấn luyện, nhất là các mô hình lớn, phức tạp và tốn nhiều thời gian huấn luyện.
- Thực hiện lưu mô hình:

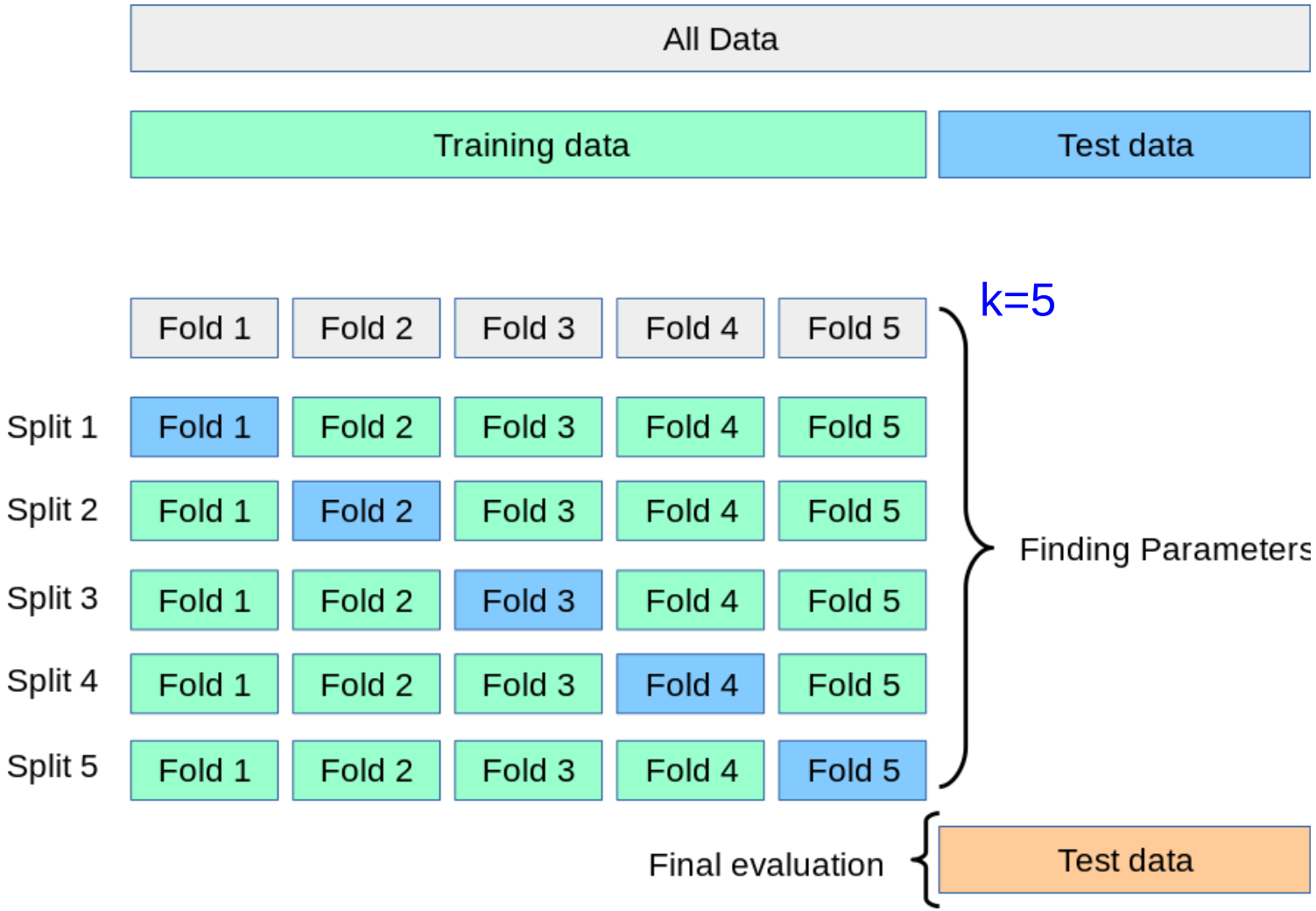
```
1. from sklearn.externals import joblib
2. joblib.dump(lin_reg, "linear_regression.pkl")
```

CROSS VALIDATION

- Cross validation (kiểm tra chéo) là kỹ thuật chia tập dữ liệu ban đầu ra làm các phần huấn luyện và kiểm thử khác nhau gọi là các fold, sau đó huấn luyện và kiểm tra mô hình trên từng fold.
- Mục tiêu: giảm hiện tượng overfiting.



CROSS VALIDATION



CROSS VALIDATION

– Cần phải xác định sẽ chia bao nhiêu fold (xác định giá trị k).

– Sử dụng cross validation:

```
1. from sklearn.model_selection import cross_val_score
2. scores = cross_val_score(lin_reg, X, y,
      scoring="neg_mean_squared_error", cv=10)
3. rmse_scores = np.sqrt(-scores)
```

6. TINH CHỈNH MÔ HÌNH

TINH CHỈNH MÔ HÌNH

- Tinh chỉnh mô hình (fine tuning) là phương pháp điều chỉnh giá trị của các siêu tham số (hyper parameter) của mô hình sao cho mô hình tối ưu nhất.
- Các phương pháp tinh chỉnh tham số cho mô hình:
  + Grid Search.
  + Randomized search.
  + Ensemble method.
- Lưu ý: cần phân biệt rõ khái niệm tham số (parameter) và siêu tham số (hyper parameter) của mô hình.

Ví dụ

- Mô hình RandomForestRegressor có các siêu tham số như sau:
  - + n_estimators: The number of trees in the forest.
  - + max_features: The number of features to consider when looking for the best split.
- Làm sao để chọn ra 2 siêu tham số sao cho mô hình đạt kết quả cao nhất trên bộ dữ liệu ?

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Grid search

- Grid search sẽ liệt kê tất cả các giá trị có khả năng của siêu tham số, sau đó tiến hành tìm ra giá trị siêu tham số tối ưu nhất.
- Các thành phần cần thiết của Grid search.
  + Mô hình cần tinh chỉnh (estimator).
  + Tập hợp các siêu tham số (parameter space).
  + Phương pháp dò tìm (sampling candidate).
  + Cross validation scheme → Cơ chế cross validation.
  + Score function ➔ độ đo để tìm ra mô hình tối ưu.
- Khi số lượng siêu tham số tăng lên thì Grid Search không khả thi do không gian giá trị quá lớn.

Ví dụ

– Xây dựng tập các giá trị khả dĩ của các siêu tham số:

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

– n_estimators: 3 x 2 = 6

– max_features: 4 x 3 = 12

➔ Số lượng tham số khả dĩ: 18.

Thực hiện Grid Search

- Sử dụng cross validation với k = 5 trên bộ dữ liệu California House Price

```
1. from sklearn.model_selection import GridSearchCV
2. param_grid = [
        {'n_estimators': [3, 10, 30],'max_features': [2, 4, 6, 8]},
        {'bootstrap': [False], 'n_estimators': [3, 10],
   max_features': [2, 3, 4]},
3. ]
4. model = RandomForestRegressor()
5. grid_search = GridSearchCV(model, param_grid, cv=5,
   scoring='neg_mean_squared_error')
6. grid_search.fit(X, y)
```

Kết quả

- Kết quả thực hiện GridSearch: bộ tham số tối ưu nhất

```
grid_search.best_params_
```

- Kết quả:

```
{'max_features': 2, 'n_estimators': 30}
```

- Xem danh sách các bộ tham số khả dĩ:

```
grid_search.cv_results_
```

Random search

Random search cũng tìm ra các tập siêu tham số tối ưu khả dĩ cho mô hình. Tuy nhiên, Random search chỉ chọn ra ngẫu nhiên một số giá trị hữu hạn.

➔ Tiết kiệm thời gian thực hiện hơn so với Grid Search. Tuy nhiên không đảm bảo giá trị tìm được là tối ưu trên toàn cục.

7. TRIỂN KHAI MÔ HÌNH

Triển khai mô hình

- Trình bày hệ thống máy học đã huấn luyện và triển khai. Cần làm rõ các ý sau:
  + Những kết quả đạt được sau quá trình xây dựng hệ thống.
  + Cái gì đã thử nghiệm, thấy hiệu quả và không hiệu quả.
  + Những giả định đã sử dụng trong hệ thống.
  + Những hạn chế của hệ thống.
- Trong nhiều trường hợp, hệ thống máy học làm không tốt bằng các chuyên gia con người. Tuy nhiên, hệ thống máy học chỉ đóng vai trò hỗ trợ quyết định. Quyền quyết định vẫn thuộc về chuyên gia.

Triển khai mô hình

— Đọc thêm: Triển khai dự án máy học trong thực tế

+ Sách: *Mark Treveil et al. (2020), Introducing MLOps, O'Reilly Media, Inc. ISBN: 9781492083290*

+ Khoá học: *Machine Learning Engineering for Production (MLOps) Specialization by Andrew Ng.*

(Link: https://www.coursera.org/specializations/machine-learning-engineering-for-production-mlops#courses)

Tổng kết

- Có 7 bước nhằm xây dựng và bảo trì một dự án máy học:
  1. Xác định bối cảnh và định nghĩa bài toán.
  2. Thu thập dữ liệu.
  3. Khám phá dữ liệu.
  4. Chuẩn bị dữ liệu.
  5. Huấn luyện mô hình.
  6. Tinh chỉnh mô hình.
  7. Triển khai mô hình.

Tổng kết

- Đối với bất kỳ dự án hay đề tài nào về máy học, xác định được bài toán hay tác vụ đang làm là quan trọng nhất.
- Dữ liệu đóng vai trò quan trọng đối với một dự án máy học. Quá trình thu thập, tiền xử lý và khám phá dữ liệu chiếm phần lớn thời gian trong quá trình thực hiện.

Tài liệu tham khảo

Chương 2 của sách: *Hands-on Machine Learning with ScikitLearn, Keras & TensorFlow.*