THỰC HÀNH BUỔI 2

(Data Collection and Preprocessing)

Phần 1. Xử lý các giá trị ngoại lai và các phương pháp

1.1 Ngoại lai trong machine learning

Có một bộ dữ liệu chuẩn chứa chiều cao và cân nặng của từng người và một dữ liệu chứa một số dữ liệu ngoại lai.

```
import pandas as pd
df_example = pd.DataFrame(
    data={
        "height": [147, 150, 153, 158, 163, 165, 168, 170, 173, 175, 178, 180, 183],
        "weight": [49, 50, 51, 54, 58, 59, 60, 62, 63, 64, 66, 67, 68],
        "height_2": [110, 150, 153, 158, 163, 165, 168, 170, 173, 175, 178, 180, 183],
        "weight_2": [49, 90, 51, 54, 58, 59, 60, 62, 63, 64, 66, 67, 68],

    }
)
df_example
```

	height	weight	height_2	weight_2
0	147	49	110	49
1	150	50	150	90
2	153	51	153	51
3	158	54	158	54
4	163	58	163	58
5	165	59	165	59
6	168	60	168	60
7	170	62	170	62
8	173	63	173	63
9	175	64	175	64
10	178	66	178	66
11	180	67	180	67
12	183	68	183	68

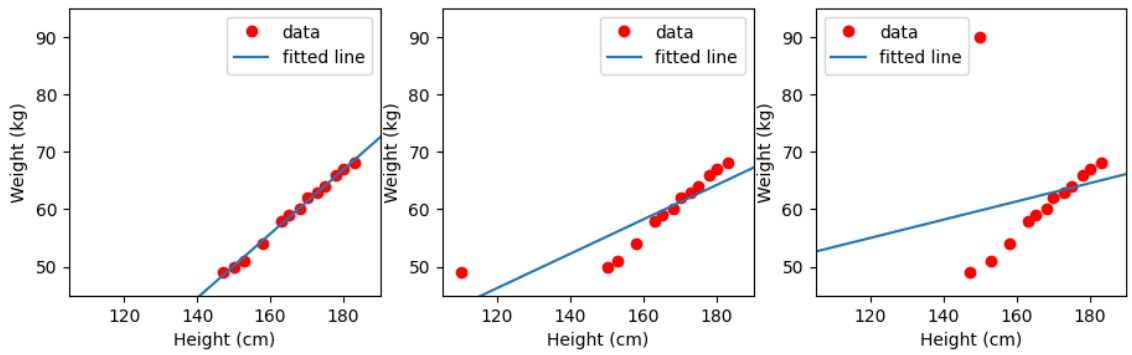
Giả sử ta cần dùng bộ dữ liệu này để xây dựng một mô hình dự đoán cân nặng theo chiều cao. Ta có thể thấy rằng cân nặng *thường* tỉ lệ thuận với chiều cao nên mô hình hồi quy tuyến tính sẽ phù hợp cho công việc này. Hình vẽ dưới đây thể hiện kết quả mà mô hình hồi quy tuyến tính học được trong ba trường hợp:

- TH1 (trái): dữ liệu trong cột **height** làm đầu vào, trong cột **weight** làm nhãn.
- TH2 (giữa): dữ liệu trong cột **height_2** làm đầu vào, trong cột **weight** làm nhãn.
- TH3 (phải): dữ liệu trong cột **height** làm đầu vào, trong cột **weight_2** làm nhãn.

```python
from matplotlib import pyplot as plt
from sklearn.linear_model import LinearRegression

def fit_linear_regression_and_visualize(df: pd.DataFrame, input_col: str, label_col: str):
    # fit the model by Linear Regression
    lin_reg = LinearRegression(fit_intercept=True)
    lin_reg.fit(df[[input_col]], df[label_col])
    w1 = lin_reg.coef_
    w0 = lin_reg.intercept_
    # visualize
    plt.plot(df[input_col], df[label_col], "ro", label="data")
    plt.axis([105, 190, 45, 75])
    plt.xlabel("Height (cm)")
    plt.ylabel("Weight (kg)")
    plt.ylim(45, 95)
    plt.plot([105, 190], [w1 * 105 + w0, w1 * 190 + w0], label="fitted line")
    plt.legend()

plt.figure(figsize=(17, 6))
plt.subplot(1, 3, 1)
fit_linear_regression_and_visualize(df_example, input_col="height", label_col="weight")
```

```python
plt.subplot(1, 3, 2)
fit_linear_regression_and_visualize(df_example, input_col="height_2", label_col="weight")

plt.subplot(1, 3, 3)
fit_linear_regression_and_visualize(df_example, input_col="height", label_col="weight_2")
```

Output:



Các điểm màu đỏ thể hiện các điểm dữ liệu với trục hoành là cân nặng và trục tung là chiều cao. Đường thẳng màu xanh là đường thẳng mà mô hình hồi quy tuyến tính học được. Ta có thể thấy rằng đường màu xanh trong hình bên trái khá khớp dữ liệu, trong khi hai đường thẳng ở hai trường hợp còn lại bị lệch đi khá nhiều dù chỉ có một điểm dữ liệu ngoại lệ trong mỗi trường hợp.

Như vậy, với dữ liệu rất đơn giản này, dữ liệu ngoại lệ dù ở đầu vào mô hình hay nhãn đều mang lại kết quả không tốt.

1.2 Xác định và xử lý các điểm ngoại lệ

Có 2 loại ngoại lai chính:

- Các giá trị không nằm trong miền xác định của dữ liệu. Ví dụ, tuổi, điểm số hay khoảng cách không thể là số âm.
- Giá trị có khả năng xảy ra nhưng xác suất rất thấp. Ví dụ 120 tuổi, thu nhập 1 triệu đô la/tháng. Những giá trị này có khả năng xảy ra nhưng thực sự hiếm có.

Cách xử lý:

- Với dữ liệu thuộc nhóm thứ nhất ta có thể thay bằng nan và coi như giá trị đó bị thiếu và tiến hàng xử lý nó trong bước xử lý missing data.
- Với dữ liệu ở nhóm hai, thường thì sẽ sử dụng phương pháp chặn trên hoặc chặn dưới (clipping hay capping). Tức là một giá trị quá lớn hoặc quá nhỏ, ta đưa nó về giá trị lớn nhất hoặc nhỏ nhất, được coi là điểm không thuộc outlier. Ví dụ với một giá trị của tuổi là 120, ta có thể đưa nó về 70 và giả sử như điểm dữ liệu này có những đặc tính chung của "người cao tuổi". Một điểm đáng lưu ý là việc chọn giá trị lớn nhất/nhỏ nhất cũng tùy thuộc vào dữ liệu. Nếu dữ liệu chỉ toàn bao gồm người cao tuổi tử 65 trở lên thì rõ ràng chặn trên bởi 70 là không hợp lý vì 70 vẫn là quá trẻ trong bộ dữ liệu này.

Phương pháp IQR

– Cách phổ biến nhất là sử dụng Box plot. Box plot vừa giúp xác định xem dữ liệu có điểm ngoại lệ không, vừa giúp tìm ra ngưỡng lớn nhất và nhỏ nhất để làm điểm cắt.

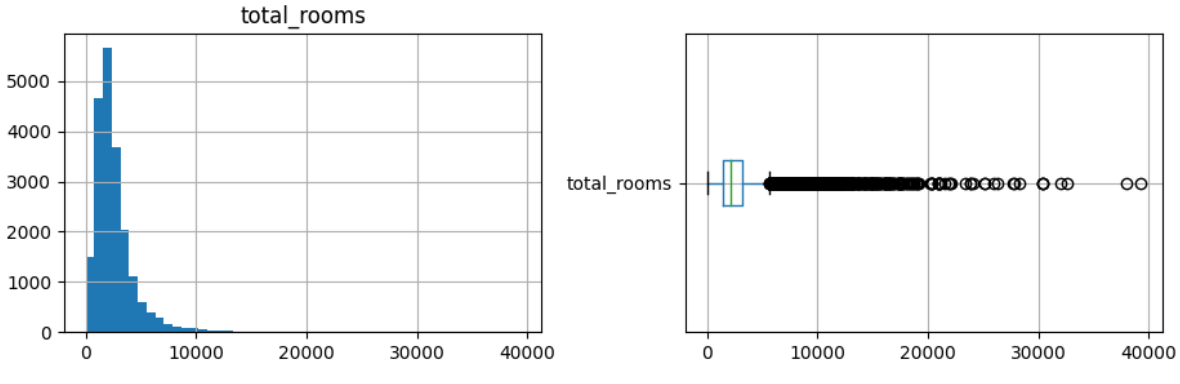
– Để minh họa cho cách sử dụng box plot, ta sẽ sử dụng bộ dữ liệu California Housing.

```
import pandas as pd

housing_path = "https://media.githubusercontent.com/media/tiepvupsu/tabml_data/master/california_housing/"
df_housing = pd.read_csv(housing_path + "housing.csv")
df_housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

– Dưới đây là histogram và box plot của cột total_rooms

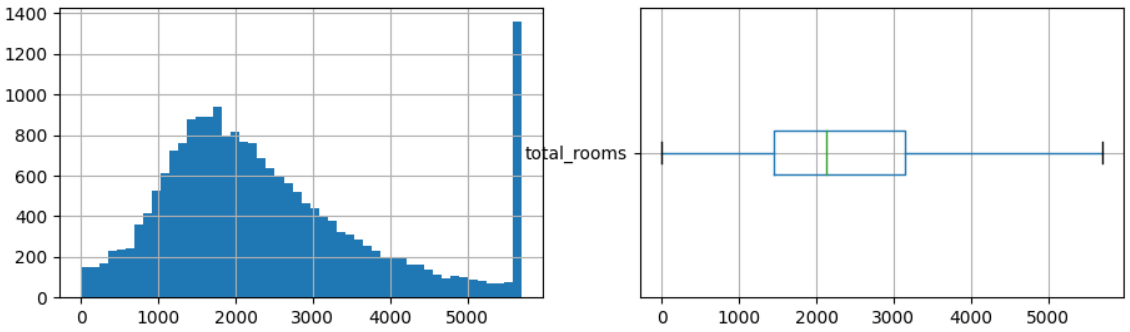
```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(11, 3))
df_housing[["total_rooms"]].hist(bins=50, ax=axes[0]);
df_housing[["total_rooms"]].boxplot(ax=axes[1], vert=False);
```



Ta thấy trên hình có khá nhiều giá trị lẻ tẻ hướng về bên phải. Do vậy ta cần đưa chúng về giá trị cực tiểu hoặc cựa đại của Boxplot. Ở đây mình sử dụng sklearn để làm điều này.

```
from typing import Tuple
from sklearn.base import BaseEstimator, TransformerMixin

def find_boxplot_boundaries(col: pd.Series, whisker_coeff: float = 1.5) -> Tuple[float, float]:
    Q1 = col.quantile(0.25)
    Q3 = col.quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - whisker_coeff * IQR
    upper = Q3 + whisker_coeff * IQR
    return lower, upper

class BoxplotOutlierClipper(BaseEstimator, TransformerMixin):
    def __init__(self, whisker_coeff: float = 1.5):
        self.whisker = whisker_coeff
        self.lower = None
        self.upper = None

    def fit(self, X: pd.Series):
        self.lower, self.upper = find_boxplot_boundaries(X, self.whisker)
        return self

    def transform(self, X):
        return X.clip(self.lower, self.upper)
```

```
clipped_total_rooms = BoxplotOutlierClipper().fit_transform(df_housing["total_rooms"])

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(11, 3))
clipped_total_rooms.hist(bins=50, ax=axes[0])
clipped_total_rooms.to_frame().boxplot(ax=axes[1], vert=False);
```



Sau khi cắt dữ liệu theo cực đại và cự tiểu của boxplot ta thấy rằng dữ liệu đã đỡ lệch đi. Và ta không còn thấy các điểm ngoại lai nằm ngoài cực đại và cực tiểu.

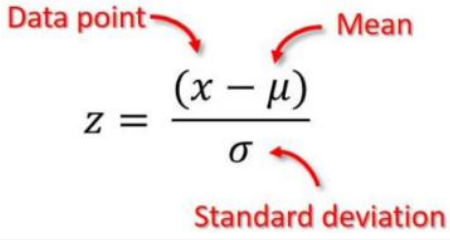
Phương pháp Z-score

Nếu dữ liệu tuân theo phân phối chuẩn, bạn có thể áp dụng quy tắc 3σ cho phân phối chuẩn.

Trong phân phối chuẩn, giả sử μ là kỳ vọng và σ là độ lệch chuẩn. Quy tắc 3σ cho phân phối chuẩn nói rằng:

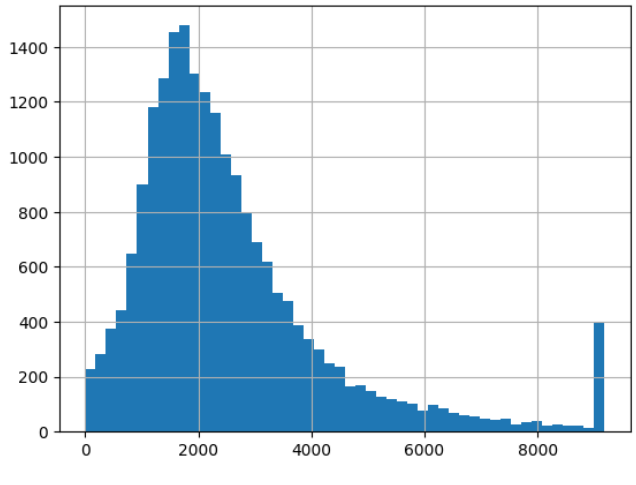
- 68% các điểm dữ liệu nằm trong khoảng $\mu \pm \sigma$
- 95% các điểm dữ liệu nằm trong khoảng $\mu \pm 2\sigma$
- 99.7% các điểm dữ liệu nằm trong khoảng $\mu \pm 3\sigma$

Với một điểm dữ liệu , z-score của nó được tính bởi:




```
class ZscoreOutlierClipper(BaseEstimator, TransformerMixin):
    def __init__(self, z_threshold: float = 3):
        self.z_threshold = z_threshold
        self.lower = None
        self.upper = None

    def fit(self, X: pd.Series):
        mean = X.mean()
        std = X.std()
        self.lower = mean - self.z_threshold * std
        self.upper = mean + self.z_threshold * std
        return self

    def transform(self, X):
        return X.clip(self.lower, self.upper)
```

```
clipped_total_rooms2 = ZscoreOutlierClipper().fit_transform(df_housing["total_rooms"])
clipped_total_rooms2.hist(bins=50);
```



Nhận xét:

- So với box plot, z-score trong trường hợp này trả về khoảng giá trị rộng hơn, các giá trị lớn hơn 9000 mới được coi là ngoại lệ trong khi con số chặn trên của boxplot chỉ khoảng 6000.
- Khi z-score xử lý ngoại lệ có thể làm dịch chuyển kỳ vọng và độ lệch chuẩn do vậy giá trị chặn trên và chặn dưới cũng sẽ có thay đổi. Nghĩa là lần xác định tiếp theo sẽ khác so với lần hiện tại.
- Còn với IQR dù giá trị ngoại lai có như thế nào cũng sẽ không làm ảnh hưởng tới chặn trên và dưới, vì nó chỉ cần thay giá trị ngoại lai vào giá trị chặn trên hoặc dưới.

1.3 Cách xác định và xử lý các điểm ngoại lai là categorical

Khác với dữ liệu là số, dữ liệu ngoại lệ trong các trường categorical khó xác định hơn. Một phần là khó sẽ biểu đồ histogram và muốn xác định dữ liệu ngoại lệ cần có kiến thức chuyên môn cao.

- Do sai khác trong cách nhập dữ liệu
- Hoặc có thể là nhập sai lỗi chính tả, để xử lý chúng ta có thể vẽ histogram thể hiện tần suất của từng giá trị trong toàn bộ dữ liệu. Thông thường những lỗi chính tả thì nó sẽ ở mức thấp.
- Đối với một số dữ liệu có liên quan tới dữ liệu nhãn mà chúng có nhiều dữ liệu khác nhau khó one-hot hay enlable thì ta có thể nhóm chúng thay những mục mới có điểm chung.

1.4 Sinh viên thực hành lại với bộ dữ liệu ở phần 1.1

Phần 2. Data Pre-processing.

Yêu cầu: Sinh viên thực hiện tiền xử lý dữ liệu trên bộ dữ liệu **Data.csv**

1) Kiểm tra các giá trị bị thiếu (Checking for Missing Values)
2) Xử lý các giá trị bị thiếu (Handling the Missing Values)
3) Chuyển đổi và mã hóa dữ liệu (Chuyển đổi các biến Categorical thành dạng số (nếu cần thiết) bằng mã hóa one-hot encoding hoặc mã hóa số học)
4) Thực hiện chia tập dữ liệu (Train, test)
5) Xử lý ngoại lệ (Outlier Handling)
6) Chuẩn hóa dữ liệu (z-score, Min-Max Scaling).