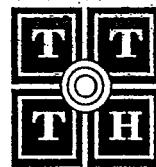


**TRUNG TÂM TIN HỌC**

ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP. HCM

227 Nguyễn Văn Cừ - Quận 5 – Tp. Hồ Chí Minh

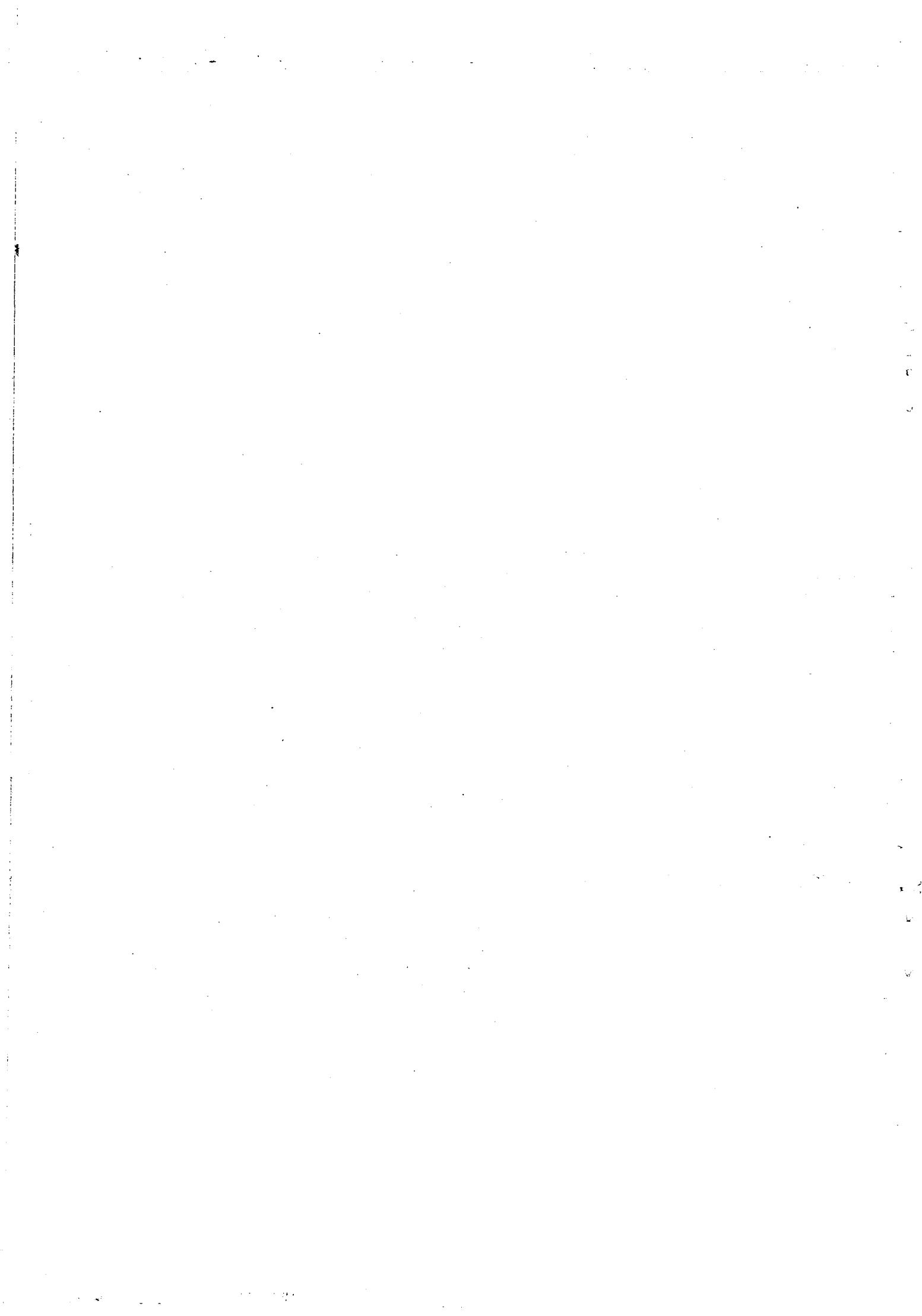
Tel.: 028 38351056 – Fax : 38324466 – Email: ttth@hcmus.edu.vn

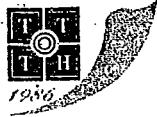


*Tài liệu học tập:*

**DATA PRE-PROCESSING  
AND ANALYSIS**

Tp. HCM – Tháng 07/2019





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

# DATA PRE-PROCESSING AND ANALYSIS

## Bài 1:Tổng quan Data Pre-processing

Phòng LT & Mạng

[https://csc.edu.vn/lap-trinh-va-cSDL/Data-Pre-processing-and-Analysis\\_196](https://csc.edu.vn/lap-trinh-va-cSDL/Data-Pre-processing-and-Analysis_196)

2019



## Nội dung

1. Giới thiệu
2. Quy trình Pre-processing





## Giới thiệu

- **Dữ liệu có thể rất đáng sợ đối với những ai làm việc với nó.** Nếu chúng ta có dataset thô trong tay và nếu chúng ta là người làm việc với dữ liệu thì nên bắt đầu nghĩ đến những thứ khác nhau có thể làm với tập dữ liệu thô này. Vì vậy, trong quá trình học tập để trở thành một người làm việc với dữ liệu, cần phải nắm các kỹ thuật tiền xử lý dữ liệu khác nhau bởi vì chúng rất cần thiết cho công việc.



## Giới thiệu

- **Tiền xử lý dữ liệu (Data Preprocessing) là gì?**

- Là một kỹ thuật khai thác dữ liệu (data mining technique) bao gồm chuyển đổi dữ liệu thô thành định dạng dễ hiểu, có thể sử dụng được. Dữ liệu trong thế giới thực thường không đầy đủ, không nhất quán và/hoặc thiếu một số hành vi/xu hướng nhất định và có khả năng chứa nhiều lỗi.
- Là một phương pháp đã được chứng minh để giải quyết các vấn đề về dữ liệu
- Là một bước quan trọng trong quy trình khai thác dữ liệu



## Giới thiệu

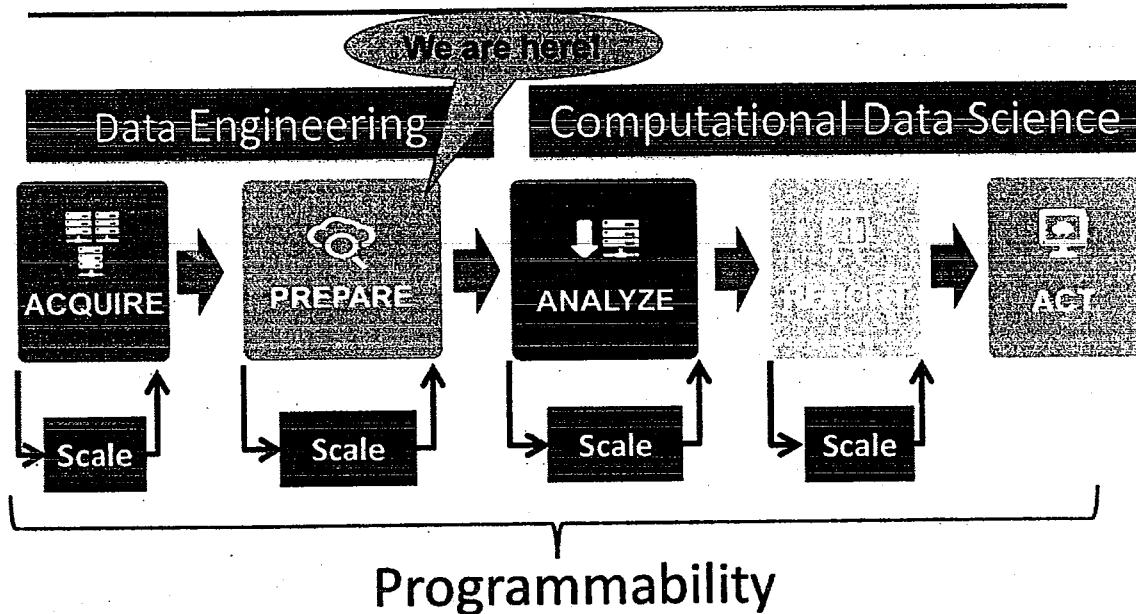
### ☐ Tại sao chúng ta phải tiền xử lý dữ liệu?

- Dữ liệu trong thế giới thực thường không đầy đủ: thiếu các giá trị thuộc tính, hoặc chỉ chứa dữ liệu tổng hợp.
- Các phương pháp thu thập dữ liệu thường được kiểm soát lỏng lẻo, dẫn đến các giá trị ngoài phạm vi (out-of-range value), ví dụ: Tuổi -30; kết hợp dữ liệu không phù hợp, ví dụ: Giới tính: Nam, Mang thai: Có; thiếu giá trị...
- Nếu có nhiều thông tin không liên quan và dư thừa hoặc dữ liệu nhiễu (noisy) và không đáng tin cậy, không nhất quán (Inconsistent) thì việc khám phá kiến thức trong giai đoạn huấn luyện sẽ khó khăn hơn.

## Giới thiệu

- Sản phẩm của tiền xử lý dữ liệu là training dataset.
- Phân tích dữ liệu không được sàng lọc cẩn thận như vậy có thể tạo ra kết quả sai lệch. Do đó, tiền xử lý dữ liệu cần phải làm trước khi tiến hành phân tích.
- Thông thường, tiền xử lý dữ liệu là giai đoạn quan trọng nhất của dự án máy học, nó làm sạch và phân tích dữ liệu, chuẩn bị dữ liệu cho mô hình.

## Giới thiệu



Data Pre-processing and Analysis

7

## Nội dung

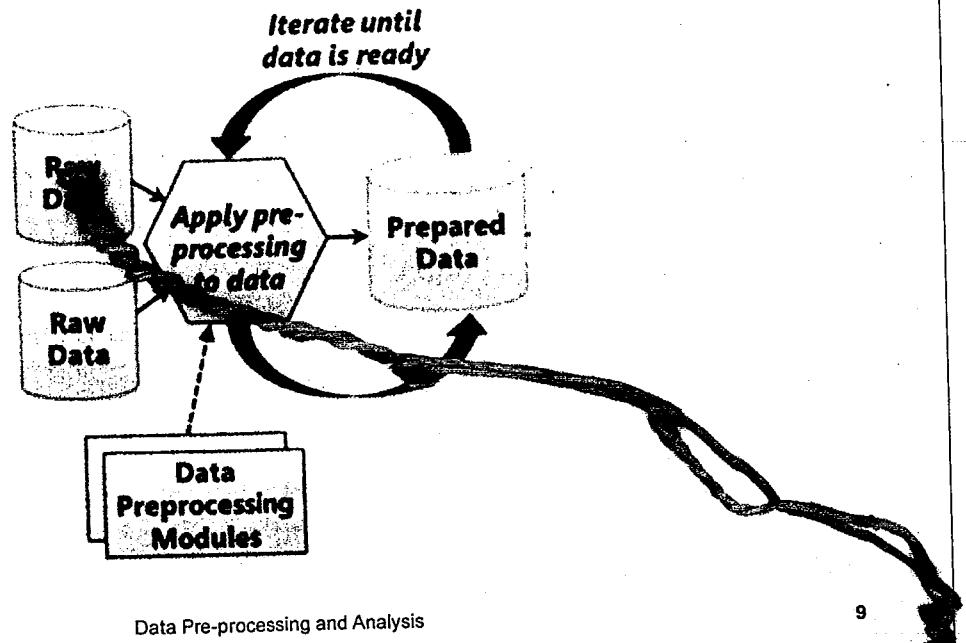
1. Giới thiệu
2. Quy trình Pre-processing

Data Pre-processing and Analysis

8

## Quy trình Pre-processing

### □ Quy trình Pre-processing



## Quy trình Pre-processing

### □ Gồm các bước:

- Import thư viện
- Đọc dữ liệu, lựa chọn thuộc tính
- Kiểm tra dữ liệu thiếu (missing value)
- Kiểm tra dữ liệu phân loại (categorical data)
- Chuẩn hóa dữ liệu (Data standardizing)
- Chuyển đổi dữ liệu (PCA transformation)
- Chia dữ liệu (Data splitting)

## Quy trình Pre-processing

### □ Import thư viện

- Nhập các thư viện cần thiết bằng từ khóa import. Ví dụ:
  - NumPy là package hỗ trợ cho việc tính toán với Python. (tính toán)
  - Pandas là package hỗ trợ việc thao tác và phân tích dữ liệu.
  - Matplotlib là package vẽ biểu đồ 2D của Python
  - Seaborn là package trực quan hóa dữ liệu Python dựa trên matplotlib, cung cấp giao diện cấp cao để vẽ biểu đồ thống kê hấp dẫn hơn.
  - ...



## Quy trình Pre-processing

### □ Đọc dữ liệu

- Dữ liệu có thể đến từ nhiều nguồn và định dạng khác nhau như CSDL, tập tin (.csv, .xls, .txt, .json, .xml)...
- Tuy nhiên, người ta thường sử dụng .csv vì đây là loại tập tin nhẹ, truy cập và sử dụng nhanh chóng.
- Sau khi đọc dữ liệu, cần xem tổng quan về dữ liệu thông qua shape, info, head, tail, describe
- Lựa chọn các thuộc tính cần thiết trong dữ liệu đã đọc vào.



## Quy trình Pre-processing

### □ Kiểm tra dữ liệu thiếu (missing value)

- Khái niệm về missing value rất quan trọng cho việc hiểu để quản lý dữ liệu thành công. Nếu các giá trị còn thiếu không được nhà nghiên cứu xử lý đúng cách, thì cuối cùng họ có thể rút ra một suy luận không chính xác về dữ liệu.
- Missing data trong training data set có thể làm giảm sức mạnh/ sự phù hợp của một mô hình hoặc có thể dẫn đến một mô hình sai lệch vì chúng ta chưa phân tích chính xác hành vi và mối quan hệ với các biến khác. Nó có thể dẫn đến dự đoán hoặc phân loại sai.

Data Pre-processing and Analysis

13

## Quy trình Pre-processing

### □ Kiểm tra dữ liệu phân loại (categorical data)

- Biến phân loại chúng ta có trong tập dữ liệu có thể là biến thuộc tính hoặc biến kết quả target variable.
- Với các biến thuộc tính phân loại chưa có định dạng số thì cần phải chuyển đổi dữ liệu phân loại sang định dạng số. Một số cách chuyển đổi như: Label Encoder, Binary Encoder, One hot code Encoder...

Data Pre-processing and Analysis

14

# Quy trình Pre-processing



#### Chuẩn hóa dữ liệu (Data standardizing)

- Tập dữ liệu đầu vào thường chỉ chứa các thuộc tính số và do đó có thể cần phải chia tỷ lệ (Feature scaling) cho các thuộc tính trong dữ liệu trước khi thực hiện công việc tiếp theo như PCA.
  - Chia tỷ lệ là phương pháp giới hạn phạm vi của các thuộc tính để chúng có thể được so sánh dựa trên các căn cứ chung. Một số cách để chia tỷ lệ là Standard Scaler, Min-max Scaler...

cân jìn chia hì le ui: 15  
VP: hinh → lbn + Data Pre-processing and Analysis  
Tfn → nhô → cc nhô → lbn → xuat boan có khung thua ve doi hoi  
lbn → xin ý nghia → do dó cân scale  
lưu ý: đảm bảo tính công bằng

## Quy trình Pre-processing

#### Chuyển đổi dữ liệu (PCA transformation)

- PCA chủ yếu sử dụng để giảm kích thước của không gian tính năng trong khi vẫn giữ lại càng nhiều thông tin càng tốt.



## Quy trình Pre-processing

### ❑ Chia dữ liệu (Data splitting)

- Trong hầu hết các mô hình Machine Learning, chúng ta thường chia dataset thành hai bộ riêng biệt: training set và test set (còn gọi là cross-validation, xác thực chéo). Việc phân chia này giúp chúng ta có dữ liệu riêng biệt cho hai công việc là huấn luyện mô hình (training set) và kiểm tra mô hình (test set).
- Tỷ lệ phân chia dataset (training set: test set) có thể là 70:30 hoặc 80:20. Tuy nhiên, việc phân chia này có thể thay đổi tùy theo hình dạng và kích thước của dataset.





## Chapter 1: TỔNG QUAN TIỀN XỬ LÝ DỮ LIỆU

### Ex1: Điểm thi THPT Quốc Gia 2016

- Cho tập tin Diemthi\_thpt\_quocgia\_2016.xlsx chứa bộ dữ liệu điểm thi THPT Quốc Gia năm 2016 của gần 35.000 thí sinh.
- Đọc dữ liệu. Xem thông tin dữ liệu:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34826 entries, 0 to 34825
Data columns (total 6 columns):
SOBAODANH      34826 non-null object
HO_TEN          34826 non-null object
NGAY_SINH       34826 non-null object
TEN_CUMTHI      34826 non-null object
GIOI_TINH        34826 non-null object
DIEM_THI         34826 non-null object
dtypes: object(6)
memory usage: 1.6+ MB
```

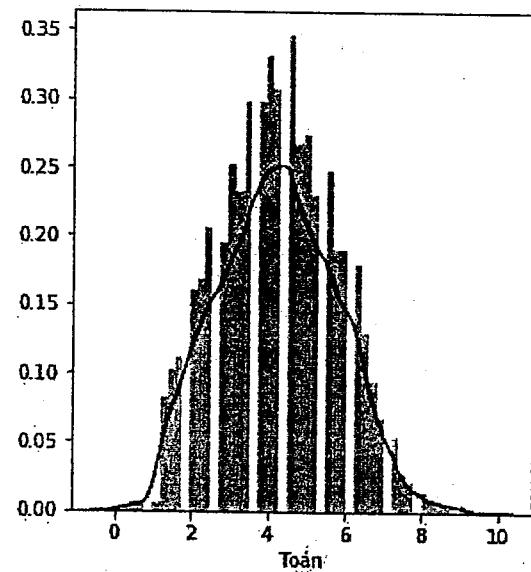
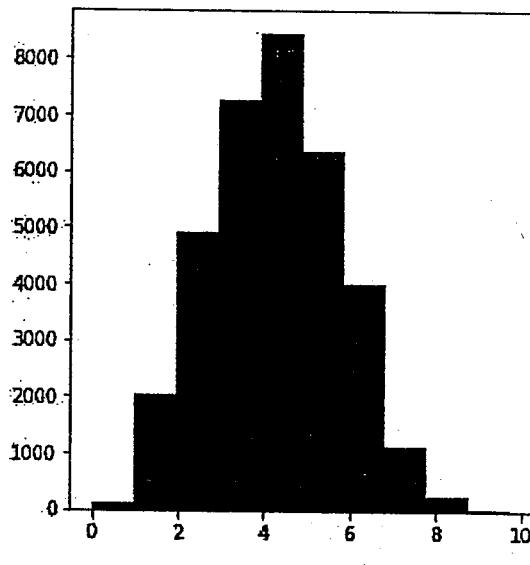
- Biết danh sách các môn thi là: "Toán", "Ngữ văn", "Địa lí", "Tiếng Anh", "Sinh học", "Vật lí", "Hóa học", "Lịch sử". Một thí sinh chỉ thi các môn bắt buộc chung còn các môn tự chọn có thể khác nhau.
- Với dữ liệu hiện tại, cột DIEM\_THI là chuỗi chứa điểm thi của tất cả các môn mà một thí sinh thi:

|   | SOBAODANH | HO_TEN        | NGAY_SINH  | TEN_CUMTHI         | GIOI_TINH | DIEM_THI                                |
|---|-----------|---------------|------------|--------------------|-----------|---|
| 0 | 018000001 | DƯƠNG VIỆT AN | 12/03/1998 | Sở GD&ĐT Bắc Giang | Nam       | Toán: 2.00 Ngữ văn: 5.50 Địa lí: 3....  |
| 1 | 018000002 | ĐỖ VĂN AN     | 09/12/1998 | Sở GD&ĐT Bắc Giang | Nam       | Toán: 5.50 Ngữ văn: 5.25 Địa lí: 5.5... |
| 2 | 018000003 | ĐỖ XUÂN AN    | 12/08/1997 | Sở GD&ĐT Bắc Giang | Nam       | Toán: 4.50 Ngữ văn: 5.50 Địa lí: 3.7... |
| 3 | 018000004 | ĐẶNG PHÚC AN  | 19/03/1998 | Sở GD&ĐT Bắc Giang | Nữ        | Toán: 3.00 Ngữ văn: 6.00 Địa lí: 5.5... |
| 4 | 018000005 | ĐẶNG VĂN AN   | 25/10/1998 | Sở GD&ĐT Bắc Giang | Nam       | Toán: 2.25 Ngữ văn: 4.75 Địa lí: 5.2... |

- Và như vậy thì chúng ta sẽ không phân tích được điểm thi của thí sinh. Do đó, việc đầu tiên là phải tiền xử lý dữ liệu. Từ dữ liệu trong cột DIEM\_THI, hãy tạo ra các cột tương ứng với danh sách các môn thi nói trên và đưa điểm của thí sinh từ chuỗi vào các cột, môn nào thí sinh không thi thì sẽ để NaN, như sau:

| SOBAODANH | HO_TEN                   | NGAY_SINH  | TEN_CUMTHI         | GIOI_TINH | DIEM_THI                              | Toán | Ngữ<br>văn | Địa<br>lí | Tiếng<br>Anh | Sinh<br>học | Vật<br>lí | Hóa<br>học | Lịch<br>sử |
|-----------|--------------------------|------------|--------------------|-----------|---------------------------------------|------|------------|-----------|--------------|-------------|-----------|------------|------------|
| 0         | 018000001. DƯƠNG VIỆT AN | 12/03/1998 | Sở GD&ĐT Bắc Giang | Nam       | Toán: 2.00 Ngữ văn: 5.50 Địa lý: 3.50 | 2.00 | 5.50       | 5.00      | NaN          | NaN         | NaN       | NaN        | 3.0        |
| 1         | 018000002 ĐỖ VĂN AN      | 09/12/1998 | Sở GD&ĐT Bắc Giang | Nam       | Toán: 5.50 Ngữ văn: 5.25 Địa lí: 5.50 | 5.50 | 5.25       | 5.50      | 3.68         | NaN         | NaN       | NaN        | NaN        |
| 2         | 018000003 ĐỖ XUÂN AN     | 12/08/1997 | Sở GD&ĐT Bắc Giang | Nam       | Toán: 4.50 Ngữ văn: 5.50 Địa lí: 3.75 | 4.50 | 5.50       | 3.75      | 2.25         | NaN         | NaN       | NaN        | NaN        |
| 3         | 018000004 ĐẶNG PHÚC AN   | 19/03/1998 | Sở GD&ĐT Bắc Giang | Nữ        | Toán: 3.00 Ngữ văn: 6.00 Địa lí: 5.50 | 3.00 | 6.00       | 5.50      | 1.50         | NaN         | NaN       | NaN        | NaN        |
| 4         | 018000005 ĐẶNG VĂN AN    | 25/10/1998 | Sở GD&ĐT Bắc Giang | Nam       | Toán: 2.25 Ngữ văn: 4.75 Địa lí: 5.25 | 2.25 | 4.75       | 5.25      | 2.00         | NaN         | NaN       | NaN        | NaN        |

- Hãy vẽ biểu đồ phân phối tần suất điểm thi, mỗi điểm thi là một biểu đồ, nhận xét trên từng biểu đồ: các thống kê mô tả, phân phối chuẩn hay nghiêng? Đường cong cao hơn hay thấp hơn phân phối chuẩn...
- Ví dụ: Môn “Toán”



- Nhận xét: ...



## DATA PRE-PROCESSING AND ANALYSIS

### Bài 2: Data Pre-processing – Data Exploration

Phòng LT & Mạng

<http://kientruc.usth.edu.vn/~datatranh/Data%20Pre-processing%20and%20Analysis.pdf>

2019

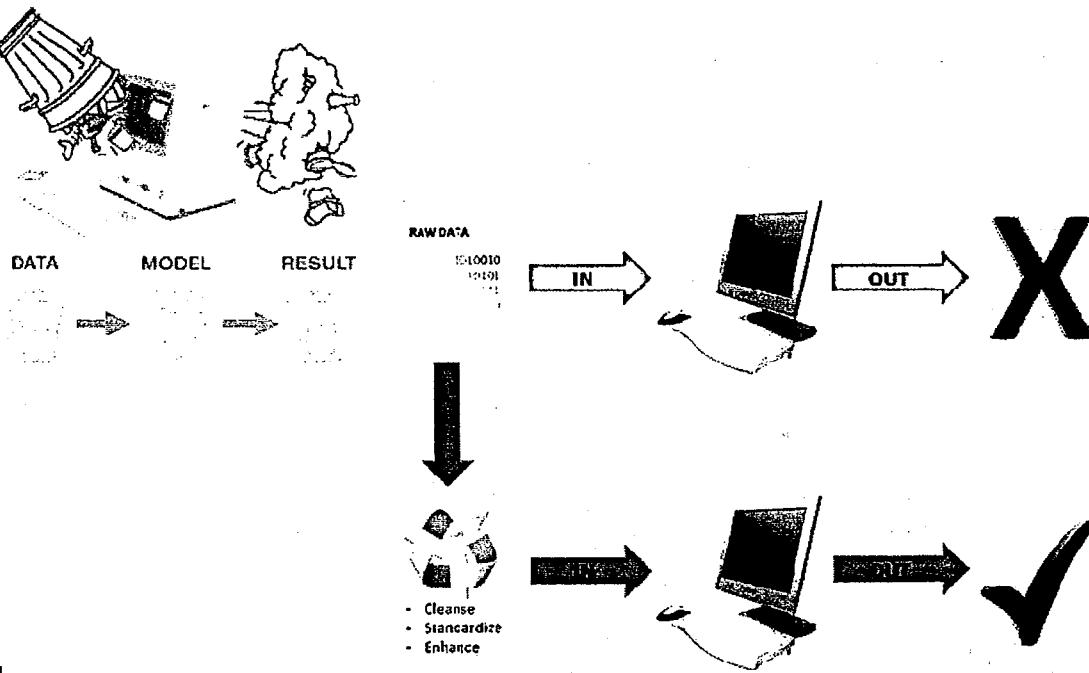


### Giới thiệu

- Khám phá dữ liệu là một bước thực hiện trong quá trình làm sạch dữ liệu (Data Cleaning)
- Không có lối tắt để khám phá dữ liệu. Khi làm việc với Machine Learning, chúng ta sẽ nhận ra rằng chúng ta luôn phải vật lộn để cải thiện độ chính xác của mô hình. Trong những tình huống như vậy, các kỹ thuật khám phá dữ liệu sẽ rất hữu ích.
- Chất lượng input sẽ quyết định chất lượng output. Thường thì việc khám phá dữ liệu, làm sạch và chuẩn hóa dữ liệu chiếm phần lớn thời gian của dự án (70% ~ 80%)



## Giới thiệu



Data Pre-processing and Analysis

3

## Giới thiệu

### ❑ Các công việc cần làm trước tiên:

- Xác định các thuộc tính/ biến (Variable Identification)
- Phân tích đơn biến (Univariate Analysis) (ján trú hàng biến' (single variable))
- Phân tích hai biến (Bi-variate Analysis) (có thể có 2 biến' liên hue (bivariate) 1 biến ján loanh / 1 biến (Anova).)
- Xử lý dữ liệu thiếu (Missing values)
- Xử lý dữ liệu ngoại lai (Outlier values)

## Nội dung

1. Xác định các thuộc tính
2. Phân tích đơn biến
3. Phân tích hai biến
4. Cách xử lý dữ liệu thiếu
5. Phát hiện và xử lý ngoại lệ



## Xác định các thuộc tính

### Để xác định thuộc tính:

- Đầu tiên, xác định các biến đầu vào (Predictor/ Input) và biến đầu ra (Target/ Output).
- Tiếp theo, xác định loại dữ liệu (data type: string, numeric...) loại (category: categorical, continuous) của các biến.



## Xác định các thuộc tính

### • Ví dụ

- Students: dự đoán sinh viên chơi cricket hay không dựa trên dữ liệu:

| Student_ID | Gender | Prev_Exam_Marks | Height(cm) | Weight Category(kgs) | Play Cricket |
|------------|--------|-----------------|------------|----------------------|--------------|
| S001       | M      | 65              | 178        | 61                   | 1            |
| S002       | F      | 75              | 174        | 56                   | 0            |
| S003       | M      | 45              | 163        | 62                   | 1            |
| S004       | M      | 57              | 175        | 70                   | 0            |
| S005       | F      | 59              | 162        | 67                   | 0            |

Data Pre-processing and Analysis

7

## Xác định các thuộc tính

- Các thuộc tính được xác định trong các danh mục khác nhau:

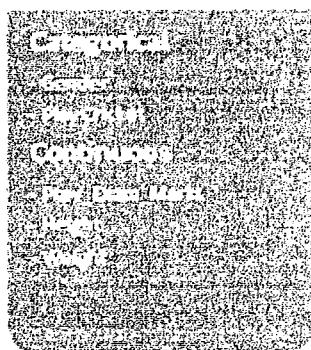
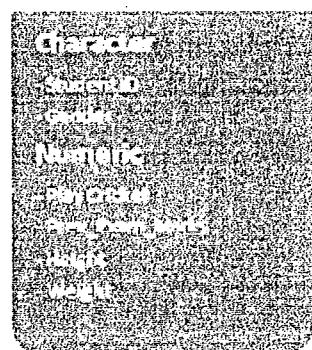
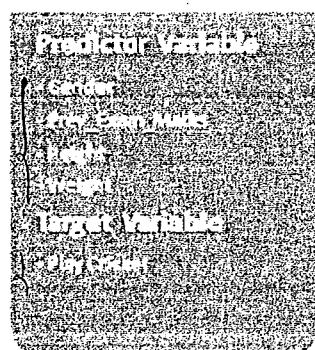
Type of Variable

lôgik/không

Data Type

lôgik/dữ liệu

Variable Category



8

# Xác định các thuộc tính

## • Ví dụ

- Titanic: dự đoán sự sống còn của hành khách trên tàu Titanic

| Passengerid | Survived | Pclass | Name   | Sex    | Age  | SibSp | Parch | Ticket           | Fare    | Cabin | Embarked | Quá trình làm việc |           |           |
|-------------|----------|--------|--|--------|------|-------|-------|------------------|---------|-------|----------|--------------------|-----------|-----------|
|             |          |        |  |        |      |       |       |                  |         |       |          | gộp thành          | gộp thành | gộp thành |
| 1           | 0        | 3      | Braund, Mr. Owen Harris                            | male   | 22.0 | 1     | 0     | A/5 21171        | 7.2500  | Nan   | S        |                    |           |           |
| 2           | 1        | 1      | Cumings, Mrs. John Bradley (Florence Briggs Th...) | female | 38.0 | 1     | 0     | PC 17599         | 71.2833 | C85   | C        |                    |           |           |
| 3           | 1        | 3      | Heikkinen, Miss. Laina                             | female | 26.0 | 0     | 0     | STON/O2. 3101282 | 7.9250  | Nan   | S        |                    |           |           |
| 4           | 1        | 1      | Futrelle, Mrs. Jacques Heath (Lily May Peel)       | female | 35.0 | 1     | 0     | 113803           | 53.1000 | C123  | S        |                    |           |           |
| 5           | 0        | 3      | Allen, Mr. William Henry                           | male   | 35.0 | 0     | 0     | 373450           | 8.0500  | Nan   | S        |                    |           |           |

- Hãy tìm các thuộc tính của dataset và đưa vào các danh mục phù hợp: Type of variable,

Data Type, Variable category

| Type of Variable   | Data Type | Variable Category                |
|--------------------|-----------|----------------------------------|
| Predictor Variable | character | - Category                       |
| + Pclass           | + Sex     | + Survival                       |
| + Cabin            | + Cabin   | Data Pre-processing and Analysis |
| + Age              | numeric   | + Pclass                         |
| + SibSp            | + Pclass  | + Sex                            |
| + Cabin            | + Age     | + Cabin                          |
| - target Variable  | + Cabin   | + SibSp                          |
| + survived         | + Age     | continuous                       |
|                    | + Cabin   | + Fare                           |

## Nội dung

- Xác định các thuộc tính
- Phân tích đơn biến
- Phân tích hai biến
- Cách xử lý dữ liệu thiếu
- Phân chia và xử lý ngoại lệ



## Phân tích đơn biến

- Ở giai đoạn này, chúng ta khám phá từng biến một.
- Phương pháp để thực hiện phân tích đơn biến sẽ phụ thuộc vào loại biến là phân loại (categorical) hay liên tục (continuous).

## Phân tích đơn biến

### Biến liên tục □ Continuous Variables

- Trong trường hợp các biến liên tục, chúng ta cần tìm hiểu xu hướng trung tâm và sự lây lan của biến.  
→ min, max, mean, median, range, IQR, ~~jống số~~, ~~đo lát~~, ~~đo diện~~, ~~vẽ histogram~~ or ~~boxplot~~
- Chúng được đo bằng các phương pháp trực quan số liệu thống kê khác nhau như sau:

| Central Tendency | Measure of Dispersion | Visualization Methods |
|------------------|-----------------------|-----------------------|
| Mean             | Range                 | Histogram             |
| Median           | Quartile              | Box Plot              |
| Mode             | IQR                   |                       |
| Min              | Variance              |                       |
| Max              | Standard Deviation    |                       |
|                  | Skewness and Kurtosis |                       |

Phân tích cho biến

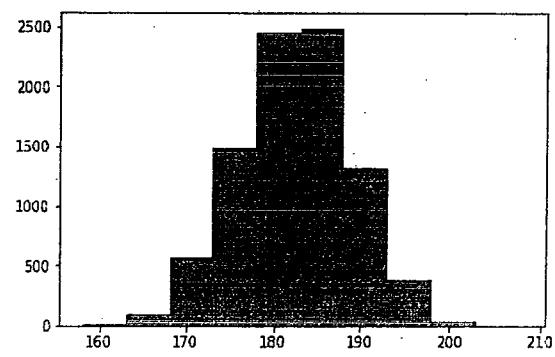
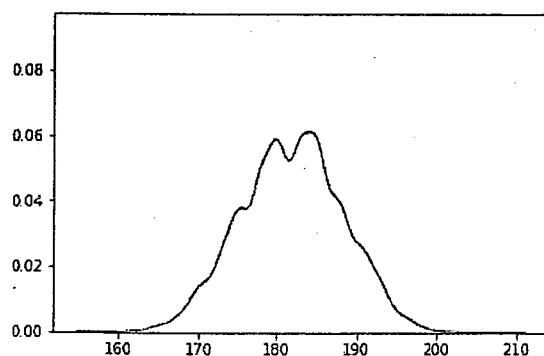
## Phân tích đơn biến

- Chú ý: Phân tích đơn biến cũng được sử dụng để làm nổi bật các giá trị bị thiếu và ngoại lệ.



## Phân tích đơn biến

- Ví dụ: Phân tích biến heights (liên tục) trong bộ dữ liệu baseball (như gợi ý của bảng liệt kê phía trên)



## Phân tích đơn biến

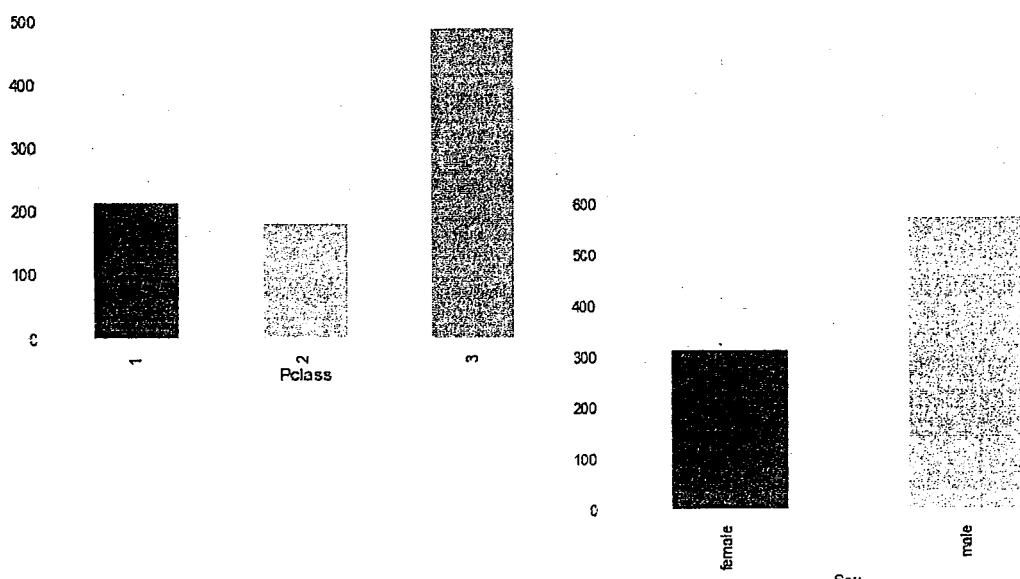
### Categorical Variables

- Đối với các biến phân loại, chúng ta sử dụng bảng tần số để hiểu phân phối của từng loại, cũng có thể đọc theo tỷ lệ phần trăm của các giá trị theo từng danh mục.
- Biến có thể được đếm bằng Count theo từng Category.
- Bar chart được dùng để trực quan hóa dữ liệu.



## Phân tích đơn biến

- Ví dụ: phân tích biến phân loại Pclass, Sex,...



# Nội dung

1. Xác định các thuộc tính
2. Phân tích đơn biến
3. Phân tích hai biến
4. Cách xử lý dữ liệu thiếu
5. Phát hiện và xử lý ngoại lệ



## Phân tích hai biến

- Phân tích hai biến tìm ra mối quan hệ giữa hai biến: tìm kiếm sự liên kết (association) và không liên kết (disassociation) giữa các biến ở mức ý nghĩa được xác định trước.**
- Chúng ta có thể thực hiện phân tích hai biến cho bất kỳ sự kết hợp nào của các biến phân loại và liên tục. Sự kết hợp có thể là: Phân loại & Phân loại, Phân loại & Liên tục và Liên tục & Liên tục.**
- Các phương pháp khác nhau được sử dụng để giải quyết các kết hợp này trong quá trình phân tích.**



## Phân tích hai biến

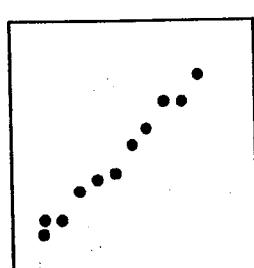
### □ Continuous & Continuous

- Khi thực hiện phân tích hai biến liên tục, chúng ta nên xem xét biểu đồ phân tán. Đó là một cách phù hợp để tìm ra mối quan hệ giữa hai biến.
- Mẫu biểu đồ phân tán biểu thị mối quan hệ giữa các biến. Mỗi quan hệ có thể là tuyến tính hoặc phi tuyến tính.

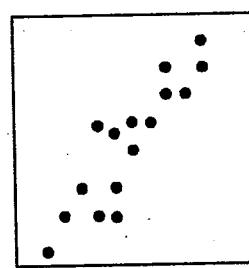


## Phân tích hai biến

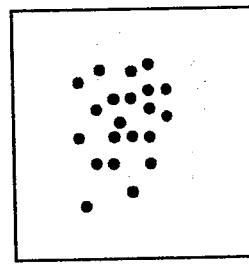
Scatter plot ,



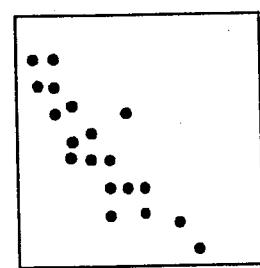
Strong positive correlation



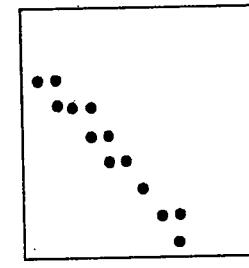
Moderate positive correlation



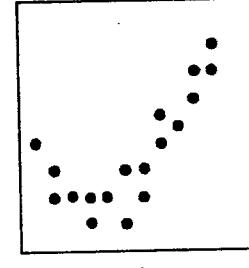
No correlation



Moderate negative correlation



Strong negative correlation



Curvilinear relationship



## Phân tích hai biến

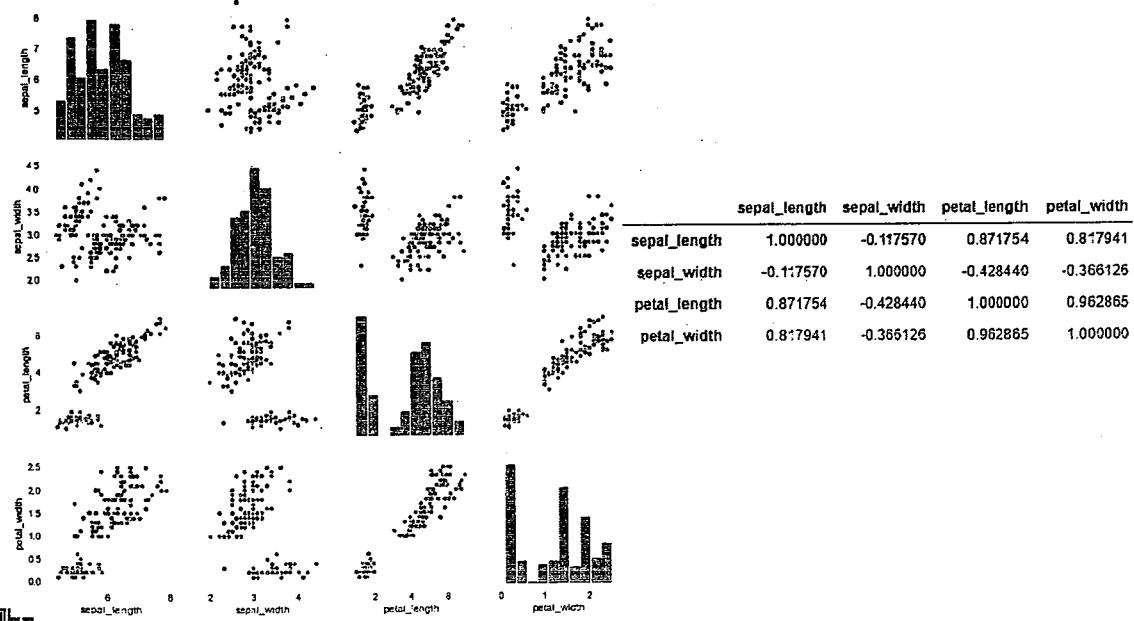
- Biểu đồ phân tán cho thấy mối quan hệ giữa hai biến nhưng không chỉ ra sức mạnh của mối quan hệ giữa chúng. Để tìm ra sức mạnh của mối quan hệ, cần dùng Correlation function. Tương quan khác nhau giữa -1 và +1.
  - 1: tương quan tuyến tính âm hoàn hảo
  - +1: tương quan tuyến tính dương hoàn hảo
  - 0: không tương quan

↑ Pearson



## Phân tích hai biến

- Ví dụ: Iris



## Phân tích hai biến

### □ Categorical & Categorical:

- Để tìm mối quan hệ giữa hai biến phân loại, chúng ta có thể sử dụng các phương pháp sau:

- Two-way table 
- Stacked Column Chart 
- Chi-Square Test



## Phân tích hai biến

### • Two-way table

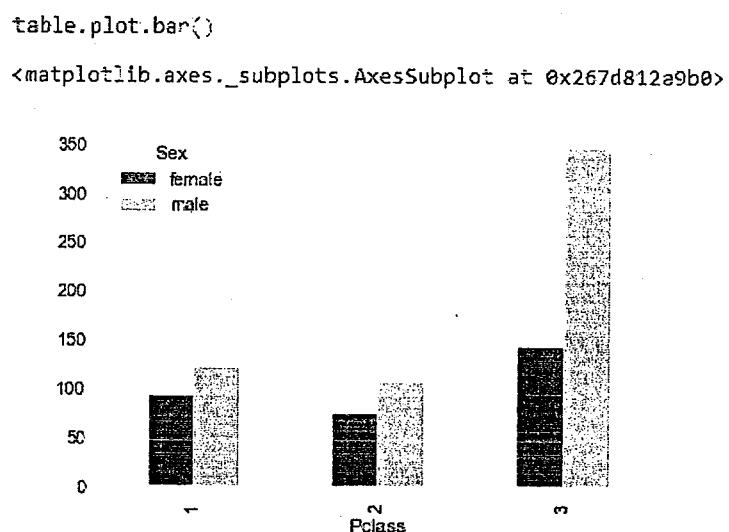
- Bắt đầu phân tích mối quan hệ bằng cách tạo bảng hai chiều Count.
- Các dòng đại diện cho category của một biến và các cột đại diện cho các loại của biến khác. Hiển thị Count của các mẫu có sẵn trong mỗi kết hợp của các loại dòng và cột.



## Phân tích hai biến

- Ví dụ: Bảng 2 chiều count của Pclass và Sex

```
table = pd.crosstab(data['Pclass'], data['Sex'])  
table  
  
Sex female male  
Pclass  
---  
1    94   122  
2    76   108  
3   144   347
```



hai biến PClass và sex có quan hệ gì với nhau 100%?  
→ dùng Chi-Square.

25

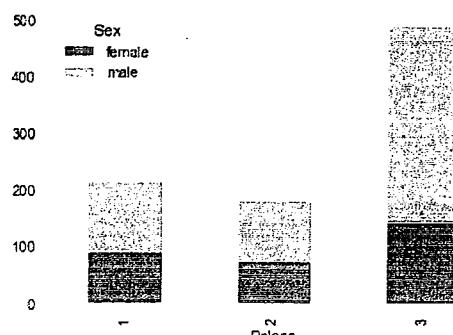
## Phân tích hai biến

### • Stacked Column Chart

- Biểu đồ cột xếp chồng là biểu đồ trực quan hóa dữ liệu của bảng hai chiều

- Ví dụ: Stacked Column Chart của Pclass và Sex

```
table.plot(kind='bar', stacked=True)  
<matplotlib.axes._subplots.AxesSubplot at 0x272ea055cc0>
```



26

## Phân tích hai biến

### • Chi-Square Test

- Thử nghiệm này được sử dụng để rút ra ý nghĩa thống kê của mối quan hệ giữa các biến.
- Kiểm định chi bình phương được sử dụng khi chúng ta muốn xem liệu có mối quan hệ giữa hai biến phân loại (categorical variables) trong một tổng thể hay không. Nó thường được sử dụng với dữ liệu phân loại như trình độ học vấn, màu sắc hoặc giới tính.
- Ngoài ra, nó kiểm tra xem bằng chứng trong mẫu có đủ mạnh để khái quát mối quan hệ cho một quần thể lớn hơn không.

Data Pre-processing and Analysis

27

## Phân tích hai biến

- Ví dụ, chúng ta quan tâm liệu có mối quan hệ giữa giới tính (gender) và loại hình doanh nghiệp (section) mà người lao động tham gia làm việc hay không?
- Khi đó, giả thuyết của Chi-squared test được phát biểu như sau:
  - $H_0$ : Biến gender và biến section là hai biến độc lập
  - $H_1$ : Biến gender và biến section không phải là hai biến độc lập

Data Pre-processing and Analysis

28

## Phân tích hai biến

- Chúng ta có thể giải thích test statistic trong bối cảnh chi-squared distribution với số lượng degrees of freedom (dof) như sau:

Cáu 1  
critical value

- Nếu Statistic  $\geq$  Critical Value: bác bỏ giả thuyết null ( $H_0$ )  $\Rightarrow$  hai biến không độc lập.
- Nếu Statistic  $<$  Critical Value: không bác bỏ null hypothesis ( $H_0$ )  $\Rightarrow$  hai biến độc lập.
- degrees of freedom của chi-squared distribution được tính dựa trên kích thước của contingency table: degrees of freedom:  $(\text{rows} - 1) * (\text{cols} - 1)$



## Phân tích hai biến

- chuẩn phán định  
dùng  $\alpha = 0.05$
- Về mặt p-value và mức ý nghĩa được chọn (chosen significance level - alpha), thử nghiệm có thể được hiểu như sau:

Cáu 2  
về p-value

- Nếu  $p\text{-value} \leq \alpha$ : bác bỏ null hypothesis ( $H_0$ )  $\Rightarrow$  hai biến không độc lập.
  - Nếu  $p\text{-value} > \alpha$ : không bác bỏ null hypothesis ( $H_0$ )  $\Rightarrow$  hai biến độc lập.
- Để thử nghiệm có hiệu quả, ít nhất năm mẫu được yêu cầu trong mỗi ô của contingency table.

dưới 5 mẫu / ô  $\rightarrow$  Sử dụng chi-square test thay vì chia tách -



## Phân tích hai biến

\* Ví dụ: Có sự liên quan giữa Pclass và Sex hay không?

```
# chi-squared test with similar proportions
from scipy.stats import chi2_contingency
from scipy.stats import chi2

# contingency table: H0: Pclass and Sex independent
table = pd.crosstab(data['Pclass'], data['Sex'])
table
```

|        | Sex | female | male |
|--------|-----|--------|------|
| Pclass |     |        |      |
| 1      | 94  | 122    |      |
| 2      | 76  | 108    |      |
| 3      | 144 | 347    |      |

## Phân tích hai biến

```
stat, p, dof, expected = chi2_contingency(table)
print('dof=%d' % dof)
print(expected)

dof=2
[[ 76.12121212 139.87878788]
 [ 64.84399551 119.15600449]
 [173.03479237 317.96520763]]

# interpret test-statistic
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%f, critical=%f, stat=%f' % (prob, critical, stat))
probability=0.950, critical=5.991, stat=16.971

# interpret p-value
alpha = 1.0 - prob
print('significance=%f, p=%f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
significance=0.050, p=0.000
Dependent (reject H0)

if abs(stat) >= critical:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
Dependent (reject H0)
```

## Phân tích hai biến

### □ Categorical & Continuous

- Trong tìm hiểu mối quan hệ giữa các biến phân loại và biến liên tục, chúng ta có thể vẽ boxplot cho từng level của các categorical variable. Nếu level có số lượng nhỏ, nó sẽ không hiển thị ý nghĩa thống kê. Để xem xét ý nghĩa thống kê, chúng ta có thể thực hiện Z-test, T-test hoặc ANOVA.



## Phân tích hai biến

- **Z-Test/ T-Test:** Thử nghiệm đánh giá xem giá trị trung bình của hai nhóm có khác nhau về mặt thống kê hay không. Nếu xác suất của Z nhỏ thì chênh lệch của hai trung bình là đáng kể hơn. Thử nghiệm T rất giống với thử nghiệm Z nhưng nó được sử dụng khi số lượng quan sát cho cả hai loại nhỏ hơn 30.
- **ANOVA:** đánh giá trung bình của nhiều nhóm (2+) có khác nhau về mặt thống kê hay không.



## Phân tích hai biến

• Ví dụ: Pclass và Sex có ảnh hưởng đến Fare hay không?

• ANOVA: PClass, Sex có ảnh hưởng đến Fare hay không?  
d\_melt = data[['Pclass', 'Sex', 'Fare']]  
d\_melt.head()

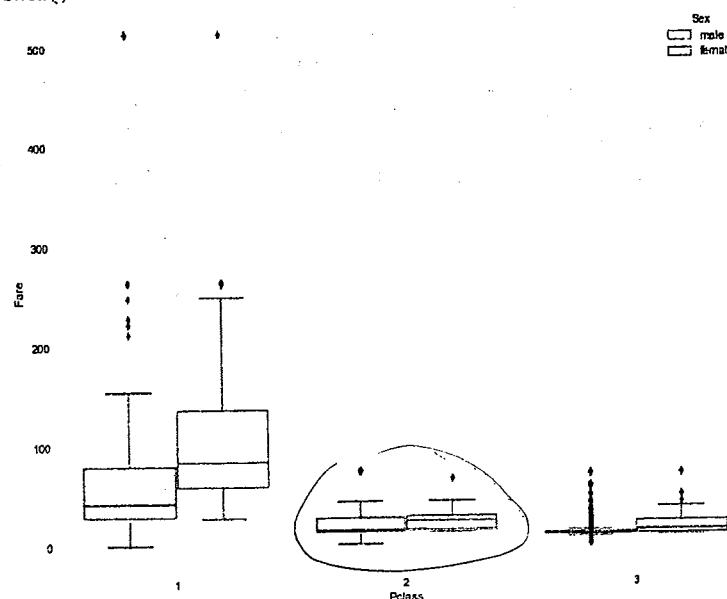
|   | Pclass | Sex    | Fare    |
|---|--------|--------|---------|
| 0 | 3      | male   | 7.2500  |
| 1 | 1      | female | 71.2833 |
| 2 | 3      | female | 7.9250  |
| 3 | 1      | female | 53.1000 |
| 4 | 3      | male   | 8.0500  |

Data Pre-processing and Analysis

35

## Phân tích hai biến

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,10))
sns.boxplot(x="Pclass", y="Fare", hue="Sex", data=d_melt, palette="Set3")
plt.show()
```



Data Pre-processing and Analysis

36

## Phân tích hai biến

Pclass có ảnh hưởng về ko?  
Pclass và age có ảnh hưởng về ko?  
import statsmodels.api as sm  
from statsmodels.formula.api import ols  
model = ols('Fare ~ C(Pclass) + C(Sex) + C(Pclass):C(Sex)', data=d\_melt).fit()  
anova\_table = sm.stats.anova\_lm(model, typ=2)

|                  | sum_sq       | df    | F          | PR(>F)             |
|------------------|--------------|-------|------------|--------------------|
| C(Pclass)        | 7.347122e+05 | 2.0   | 242.620968 | 9.763645e-85 <0.05 |
| C(Sex)           | 3.174857e+04 | 1.0   | 20.968394  | 5.337586e-06 <0.05 |
| C(Pclass):C(Sex) | 5.002816e+04 | 2.0   | 16.520591  | 9.034206e-08 <0.05 |
| Residual         | 1.339992e+06 | 885.0 | NaN        | NaN                |

Giải thích: P-value thu được từ phân tích ANOVA cho Pclass, Sex và phối hợp có ý nghĩa thống kê ( $P < 0.05$ ). Kết luận: Pclass ảnh hưởng đáng kể đến Fare, Sex ảnh hưởng đến Fare và sự phối hợp của cả Pclass và Sex ảnh hưởng đáng kể đến Fare

Bây giờ, chúng ta biết rằng sự khác biệt về Pclass, Sex có ý nghĩa thống kê, nhưng ANOVA không cho biết Pclass và Sex khác nhau đáng kể với nhau. Để biết các cặp Pclass và Sex khác nhau đáng

Tukey HSD, thực hiện post-hoc comparison sử dụng Tukey HSD.

Data Pre-processing and Analysis

37

## Phân tích hai biến

```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
for name, grouped_df in d_melt.groupby('Pclass'):
    print('Pclass {}'.format(name), pairwise_tukeyhsd(grouped_df['Fare'], grouped_df['Sex'], alpha=0.05))

Pclass 1 Multiple Comparison of Means - Tukey HSD,FWER=0.05
=====
group1 group2 meandiff lower upper reject
-----
female male -38.8997 -59.4959 -18.3035 True

Pclass 2 Multiple Comparison of Means - Tukey HSD,FWER=0.05
=====
group1 group2 meandiff lower upper reject
-----
female male -2.2283 -6.1896 1.7329 False ✎

Pclass 3 Multiple Comparison of Means - Tukey HSD,FWER=0.05
=====
group1 group2 meandiff lower upper reject
-----
female male -3.4572 -5.7329 -1.1815 True
```

Các kết quả trên từ Tukey HSD cho thấy rằng ngoại trừ Pclass = 2, group1 = female, group 2 = male, tất cả các so sánh cặp khác đều bác bỏ null hypothesis và chỉ ra sự khác biệt đáng kể về mặt thống kê.

Data Pre-processing and Analysis

38

## Nội dung

1. Xác định các thuộc tính
2. Phân tích đơn biến
3. Phân tích hai biến
4. Cách xử lý dữ liệu thiếu
5. Phát hiện và xử lý ngoại lệ



## Cách xử lý dữ liệu thiếu

### Tại sao phải xử lý giá trị thiếu?

- Thiếu dữ liệu trong tập dữ liệu huấn luyện (training data set) có thể làm giảm sức mạnh/ sự phù hợp của một mô hình hoặc có thể dẫn đến một mô hình sai lệch vì chúng ta chưa phân tích hành vi và mối quan hệ với các biến khác một cách chính xác. Nó có thể dẫn đến dự đoán hoặc phân loại sai.



## Cách xử lý dữ liệu thiếu

### • Ví dụ

| Name        | Weight | Gender | Play Cricket/ Not |
|-------------|--------|--------|-------------------|
| Mr. Amit    | 58     | M      | Y                 |
| Mr. Anil    | 61     | M      | Y                 |
| Miss Swati  | 58     | F      | N                 |
| Miss Richa  | 55     | ✓      | Y                 |
| Mr. Steve   | 55     | M      | N                 |
| Miss Reena  | 64     | F      | Y                 |
| Miss Rashmi | 57     | ✗      | Y                 |
| Mr. Kunal   | 57     | M      | N                 |

| Name        | Weight | Gender | Play Cricket/ Not |
|-------------|--------|--------|-------------------|
| Mr. Amit    | 58     | M      | Y                 |
| Mr. Anil    | 61     | M      | Y                 |
| Miss Swati  | 58     | F      | N                 |
| Miss Richa  | 55     | F      | Y                 |
| Mr. Steve   | 55     | M      | N                 |
| Miss Reena  | 64     | F      | Y                 |
| Miss Rashmi | 57     | F      | Y                 |
| Mr. Kunal   | 57     | M      | N                 |

| Gender    | #Students | #Play Cricket | %Play Cricket |
|-----------|-----------|---------------|---------------|
| F         | 2         | 1             | 50%           |
| M         | 4         | 2             | 50%           |
| Missing ✗ | 2         | 2             | 100%          |

| Gender | #Students | #Play Cricket | %Play Cricket |
|--------|-----------|---------------|---------------|
| F      | 4         | 3             | 75%           |
| M      | 4         | 2             | 50%           |

Chú ý: các giá trị bị thiếu trong hình trên: Ở cột trái, không xử lý các giá trị bị thiếu: Suy luận từ bộ dữ liệu này là cơ hội chơi cricket của nam cao hơn nữ. Mặt khác, ở cột phải, đã xử lý giá trị bị thiếu (dựa trên giới tính), chúng ta có thể thấy rằng phụ nữ có cơ hội chơi cricket cao hơn so với nam giới.



## Cách xử lý dữ liệu thiếu

### □ Tại sao dữ liệu lại bị thiếu?

- Lý do xuất hiện các giá trị bị thiếu có thể xảy ra ở hai giai đoạn:
  - Khai thác dữ liệu (Data Extraction)
  - Thu thập dữ liệu (Data collection): Những lỗi xảy ra tại thời điểm thu thập dữ liệu và khó sửa chữa. Chúng có thể là được phân thành bốn loại: thiếu hoàn toàn ngẫu nhiên (missing completely at random), thiếu ngẫu nhiên (missing at random), thiếu phụ thuộc vào input không quan sát (missing that depends on unobserved predictors), thiếu phụ thuộc vào chính giá trị còn thiếu.



## Cách xử lý dữ liệu thiếu

### • Data Extraction

- Có thể có vấn đề với quá trình trích xuất. Trong những trường hợp như vậy, chúng ta nên kiểm tra kỹ lại dữ liệu chính xác với người quản trị dữ liệu.
- Lỗi ở giai đoạn trích xuất dữ liệu thường dễ tìm và cũng có thể được sửa một cách dễ dàng.



## Cách xử lý dữ liệu thiếu

### • Data collection

~~Thiếu hoàn toàn ngẫu nhiên~~ → Xử lý bằng cách "loá" với ~~kết quả~~ nghĩa là xác suất của biến bị thiếu là như nhau cho tất cả các mẫu.

- Ví dụ: trong quá trình thu thập dữ liệu, mỗi người quyết định rằng họ sẽ tuyên bố thu nhập của mình sau khi tung đồng xu công bằng nếu ngửa thì nói mà sấp thì không. Ở đây, mẫu có khả năng thiếu giá trị như nhau.

~~Thiếu ngẫu nhiên~~ → Missing at random: Đây là trường hợp khi biến bị thiếu ở tỷ lệ ngẫu nhiên và tỷ lệ thiếu thay đổi đối với các giá trị/cấp độ khác nhau của các biến đầu vào khác nhau. → Dùng mean, median (biến số lượng), mode (av biến phân loại)

- Ví dụ: trong quá trình thu thập dữ liệu về tuổi thấy rằng nữ có số lượng giá trị thiếu cao hơn so với nam.



→ Các tên rõ bên dưới → Nếu biến "tuổi" là số lượng bối, biến khác là ~~mean~~ mean tổng thể → khay thế  
2) Prediction model → tên dữ kiện  
3) CNN → đưa vào dữ liệu "hỗ trợ".  
Nếu "chiều cao", "cân nặng" là thuộc y/n thì →  
tính mean cho từng nhóm và điền all thay

## Cách xử lý dữ liệu thiếu

### ▪ Missing that depends on unobserved predictors:

Đây là trường hợp khi các giá trị bị thiếu không ngẫu nhiên và có liên quan đến biến đầu vào không quan sát được.

- Ví dụ: Trong một nghiên cứu y khoa, nếu một chẩn đoán cụ thể gây ra sự khó chịu, thì khả năng bị bỏ qua cao hơn. Giá trị bị thiếu này không phải là ngẫu nhiên.

### ▪ Missing that depends on the missing value itself:

Đây là trường hợp khi xác suất thiếu giá trị tương quan trực tiếp với chính giá trị bị thiếu.

- Ví dụ: Những người có thu nhập cao hơn hoặc thấp hơn có khả năng cung cấp thông tin non-response với thu nhập của họ.



## Cách xử lý dữ liệu thiếu

### □ Các cách xử lý dữ liệu thiếu

● Xóa

● Dùng Mean/ Mode/ Median

● Mô hình dự đoán (Prediction Model)

● Dùng KNN



## Cách xử lý dữ liệu thiếu

### • Deletion

- Chúng ta xóa các mẫu trong đó có bất kỳ biến nào bị thiếu.
- Sự đơn giản là một trong những lợi thế của phương pháp này. Nhưng nó lại làm giảm sức mạnh của mô hình vì làm giảm kích thước mẫu.
- Các phương thức xóa được sử dụng khi bản chất của dữ liệu bị thiếu là “Missing completely at random”, ngược lại các giá trị thiếu không ngẫu nhiên khác có thể làm sai lệch output của mô hình.

Data Pre-processing and Analysis

47

## Cách xử lý dữ liệu thiếu

### Ví dụ:

|   | Gender | Manpower | Sales |  |
|---|--------|----------|-------|--|
| 0 | M      | 25.0     | 343.0 | # Create a new DataFrame dropping all incomplete rows<br>no_missing_values_rows = data.dropna(how='any') |
| 1 | F      | NaN      | 280.0 | # Print the shape of the new DataFrame<br>print(no_missing_values_rows.shape)                            |
| 2 | M      | 33.0     | 332.0 | (4, 3)   |
| 3 | M      | NaN      | 272.0 |  |
| 4 | F      | 25.0     | NaN   |  |
| 5 | M      | 29.0     | 326.0 | no_missing_values_rows   |
| 6 | NaN    | 26.0     | 259.0 |  |
| 7 | M      | 32.0     | 297.0 |  |

|   | Gender | Manpower | Sales |
|---|--------|----------|-------|
| 0 | M      | 25.0     | 343.0 |
| 2 | M      | 33.0     | 332.0 |
| 5 | M      | 29.0     | 326.0 |
| 7 | M      | 32.0     | 297.0 |

Data Pre-processing and Analysis

48

## Cách xử lý dữ liệu thiếu

### • Mean/ Mode/ Median Imputation:

- Là một phương pháp điền vào chỗ các giá trị còn thiếu với các giá trị ước tính. Mục tiêu là sử dụng các mối quan hệ đã biết có thể được xác định trong các giá trị hợp lệ của tập dữ liệu để hỗ trợ ước tính các giá trị còn thiếu.
- Là một trong những phương pháp được sử dụng thường xuyên nhất.
- Bao gồm việc thay thế dữ liệu bị thiếu cho một thuộc tính nhất định bằng mean hoặc median (thuộc tính định lượng - quantitative attribute) hoặc mode (thuộc tính định tính - qualitative attribute) của tất cả các giá trị đã biết của biến đó.



Thay =

Mean: nếu  $\{x_1, x_2, \dots, x_n\}$  là outlier  $\rightarrow$   $\bar{x} = \frac{\sum x_i}{n}$

Median: nếu  $\{x_1, x_2, \dots, x_n\}$  là  $\rightarrow$   $\tilde{x} = \text{giá trị trung位}$

Mode: có thể đếm  $\rightarrow$   $\text{dạng chuẩn}$ .

49

## Cách xử lý dữ liệu thiếu

### • Có 2 loại:

- Tổng quát hóa (Generalized Imputation): Trong trường hợp này, chúng ta tính giá trị trung bình hoặc trung vị cho tất cả các giá trị không thiếu của biến sau đó thay thế giá trị thiếu bằng giá trị trung bình hoặc trung vị.

Ví dụ: Giống như trong bảng trên, biến "Manpower" bị thiếu, vì vậy chúng ta lấy trung bình của tất cả các giá trị không bị thiếu của "Manpower" (28.33) và sau đó thay thế giá trị bị thiếu bằng nó.

- Trường hợp tương tự (Similar case Imputation): Trong trường hợp này, chúng ta tính trung bình cho giới tính "Male" (29.75) và "Female" (25) của các giá trị không bị thiếu sau đó thay thế giá trị bị thiếu dựa trên giới tính.

Đối với "Male", chúng ta sẽ thay thế các giá trị  $x_{\text{new}}$  với giá trị "Manpower" còn thiếu bằng 29.75 và đối với "Female" thay thế bằng 25.



## Cách xử lý dữ liệu thiếu

### Ví dụ

```
index_M = data.Gender[data.Gender == 'M']
index_M
0    M
2    M
3    M
5    M
7    M
Name: Gender, dtype: object

mean_M = data['Manpower'][index_M.index].mean()
mean_M
29.75

index_F = data.Gender[data.Gender == 'F']
index_F
1    F
4    F
Name: Gender, dtype: object

mean_F = data['Manpower'][index_F.index].mean()
mean_F
25.0
```

Data Pre-processing and Analysis

51

## Cách xử lý dữ liệu thiếu

```
index_M_M = data.Manpower[(data.Gender == 'M') & (data.Manpower.isna())]
index_M_M
3    NaN
Name: Manpower, dtype: float64
```

```
index_F_M = data.Manpower[(data.Gender == 'F') & (data.Manpower.isna())]
index_F_M
```

```
1    NaN
Name: Manpower, dtype: float64
```

```
data['Manpower'][index_M_M.index] = mean_M
data['Manpower'][index_F_M.index] = mean_F
data
```

|   | Gender | Manpower | Sales |
|---|--------|----------|-------|
| 0 | M      | 25.00    | 343.0 |
| 1 | F      | 25.00    | 280.0 |
| 2 | M      | 33.00    | 332.0 |
| 3 | M      | 29.75    | 272.0 |
| 4 | F      | 25.00    | Nan   |
| 5 | M      | 29.00    | 326.0 |
| 6 | NaN    | 26.00    | 259.0 |
| 7 | M      | 32.00    | 297.0 |

Data Pre-processing and Analysis

52

## Cách xử lý dữ liệu thiếu

### • Prediction Model

- Mô hình dự đoán là một trong những phương pháp tinh vi để xử lý dữ liệu bị thiếu. Ở đây, chúng ta tạo ra một mô hình dự đoán để ước tính các giá trị sẽ thay thế dữ liệu bị thiếu.
- Trong trường hợp này, chúng ta chia bộ dữ liệu thành hai bộ: Một bộ không có giá trị thiếu cho biến và bộ khác có giá trị bị thiếu.  
Tập dữ liệu đầu tiên trở thành tập dữ liệu huấn luyện của mô hình trong khi tập dữ liệu thứ hai có giá trị thiếu là tập dữ liệu thử nghiệm và biến có giá trị thiếu được coi là biến đích.
- Tiếp theo, chúng ta tạo một mô hình để dự đoán biến đích dựa trên các thuộc tính khác của tập dữ liệu huấn luyện và điền các giá trị thiếu của tập dữ liệu kiểm tra. Chúng ta có thể sử dụng regression, ANOVA, logistic regression và các kỹ thuật mô hình hóa khác nhau để thực hiện việc này.



## Cách xử lý dữ liệu thiếu

### ◦ Có 2 nhược điểm cho phương pháp này:

- Các giá trị ước tính của mô hình thường được xử lý tốt hơn các giá trị thực
- Nếu không có mối quan hệ giữa các thuộc tính trong tập dữ liệu và thuộc tính có các giá trị bị thiếu thì mô hình sẽ không chính xác khi ước tính các giá trị bị thiếu.



## Cách xử lý dữ liệu thiếu

- Ví dụ: Sử dụng interpolate() để nội suy tuyến tính cho các giá trị Sales đang thiếu

data.interpolate()

|   | Gender | Manpower | Sales |
|---|--------|----------|-------|
| 0 | M      | 25.00    | 343.0 |
| 1 | F      | 25.00    | 280.0 |
| 2 | M      | 33.00    | 332.0 |
| 3 | M      | 29.75    | 272.0 |
| 4 | F      | 25.00    | 299.0 |
| 5 | M      | 29.00    | 326.0 |
| 6 | NaN    | 26.00    | 259.0 |
| 7 | M      | 32.00    | 297.0 |

*sales ↑ manpower  
 $y = f(x)$  → nếu dữ liệu thiếu ít (5%, 10%)  
mới áp dụng đc.*



## Cách xử lý dữ liệu thiếu

### • KNN Imputation

- Trong phương pháp này, các giá trị bị thiếu của một thuộc tính được xác định bằng cách sử dụng số lượng thuộc tính đã cho giống với thuộc tính có giá trị bị thiếu. Sự giống nhau của hai thuộc tính được xác định bằng cách sử dụng hàm khoảng cách.
- Cách này cũng có những lợi thế và bất lợi nhất định.





# Cách xử lý dữ liệu thiếu

## Ưu điểm

- KNN có thể dự đoán cả hai thuộc tính định tính và định lượng
- Không cần tạo mô hình dự đoán cho từng thuộc tính có dữ liệu bị thiếu
- Các thuộc tính có nhiều giá trị bị thiếu có thể được xử lý dễ dàng
- Cấu trúc tương quan của dữ liệu được xem xét

## Khuyết điểm

- KNN rất tốn thời gian trong việc phân tích dữ liệu lớn. Nó tìm kiếm thông qua tất cả dữ liệu để tìm các trường hợp tương tự nhất.
- Lựa chọn giá trị k là rất quan trọng. Giá trị k cao sẽ bao gồm các thuộc tính khác biệt đáng kể so với những gì chúng ta cần trong khi giá trị k thấp hơn có thể thiếu các thuộc tính quan trọng.

Data Pre-processing and Analysis

Sê học ở môn LDS6

57



# Nội dung

1. Xác định các thuộc tính
2. Phân tích đơn biến
3. Phân tích hai biến
4. Cách xử lý dữ liệu thiếu
5. Phát hiện và xử lý ngoại lệ



Data Pre-processing and Analysis

58

## Phát hiện và xử lý ngoại lệ.

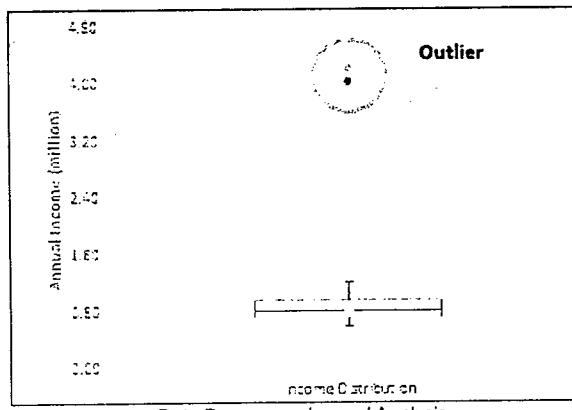
### ☐ Ngoại lệ (Outlier) là gì?

- Outlier là một thuật ngữ thường được sử dụng bởi các nhà phân tích và nhà khoa học dữ liệu vì nó cần được chú ý tới, nó có thể dẫn đến ước tính sai lầm lớn.
- Outlier là một mẫu xuất hiện ở xa và tách khỏi tổng thể.



## Phát hiện và xử lý ngoại lệ

- Ví dụ: Lập hồ sơ khách hàng và thấy rằng thu nhập trung bình hàng năm của khách hàng là 0.8 triệu USD. Nhưng, có hai khách hàng có thu nhập hàng năm là 4 triệu USD và 4.2 triệu USD. Hai khách hàng này có thu nhập hàng năm cao hơn nhiều so với phần còn lại của quần thể. Hai mẫu này sẽ được coi là Outliers.



## Phát hiện và xử lý ngoại lệ

### □ Phân loại ngoại lệ

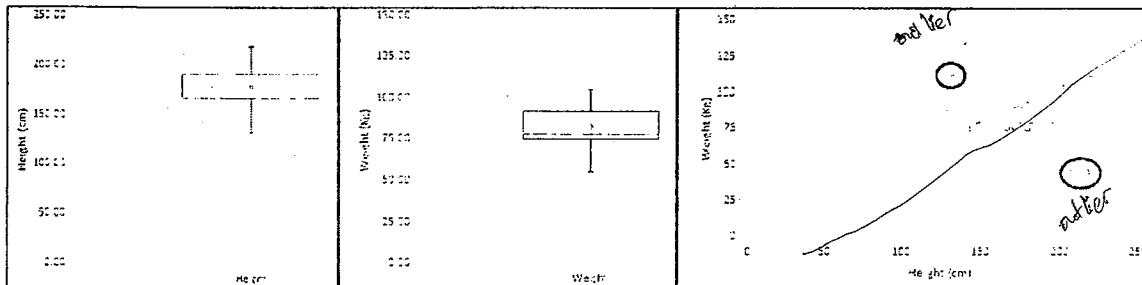
- Ngoại lệ có thể có hai loại: đơn biến (Univariate) và đa biến (Multivariate).

- Ngoại lệ đơn biến: Các ngoại lệ này có thể được tìm thấy khi chúng ta xem xét phân phối của một biến duy nhất. Ví dụ trước là ngoại lệ đơn biến.
  - Ngoại lệ đa biến là các ngoại lệ trong một không gian n chiều. Để tìm thấy chúng, chúng ta phải xem xét các bản phân phối theo nhiều chiều.
    - Vẽ scatter plot trên  $(x, y)$  → để tìm mối quan hệ có tuyến tính hay  $\text{tín}$ ? (khi xem xét trên bốn chiều).



## Phát hiện và xử lý ngoại lệ

- Ví dụ: Tìm hiểu mối quan hệ giữa chiều cao và cân nặng. Có phân phối đơn biến và hai biến cho Chiều cao, Cân nặng. Quan sát boxplot, không có bất kỳ ngoại lệ nào.
- Nếu xét đến pp (trên và dưới  $1.5 * \text{IQR}$ ). Quan sát scatter plot, có hai giá trị dưới và một giá trị trên mức trung bình trong một phân khúc cụ thể về cân nặng và chiều cao.
- Nếu xét pp → dùng 2-score  $[-2.5, 2.5], [-2.5, 2.5], [-3; 3]$  → Tùy vào dữ liệu.



⇒ Nếu nhìn bảng có biết pp nào thì dùng IQR.



## Phát hiện và xử lý ngoại lệ

### ☒ Nguyên nhân gây ra ngoại lệ?

- Bất cứ khi nào chúng ta bắt gặp các ngoại lệ, cách lý tưởng để giải quyết chúng là tìm ra lý do có những ngoại lệ này. Phương pháp để đối phó với ngoại lệ sẽ phụ thuộc vào lý do xuất hiện của chúng. Nguyên nhân của các ngoại lệ có thể được phân loại thành hai loại chính:
  - Nhân tạo (Artificial (Error) / Non-natural)
  - Tự nhiên (Natural)



## Phát hiện và xử lý ngoại lệ.

### • Lỗi nhập dữ liệu (Data Entry Error)

- Lỗi do người gây ra trong quá trình thu thập, ghi hoặc nhập dữ liệu có thể gây ra các ngoại lệ trong dữ liệu.
  - Ví dụ: Thu nhập hàng năm của khách hàng là 100.000 USD. Vô tình, toán tử nhập dữ liệu đặt sẵn một số 0 làm cho thu nhập trở thành 1.000.000 USD, cao gấp 10 lần. Đây sẽ là giá trị ngoại lệ khi so sánh với phần còn lại của quần thể.

### • Lỗi đo lường (Measurement Error)

- Đây là nguồn phỗ biến nhất của các ngoại lệ. Điều này được gây ra khi cụ đo được sử dụng bị lỗi.
  - Ví dụ: Có 10 máy cân. Trong đó có 9 máy cân chính xác, 1 bị lỗi: trọng lượng được đo bởi những người trên máy bị lỗi sẽ cao hơn/thấp hơn so với những người còn lại trong nhóm. Các trọng lượng đo được trên máy bị lỗi có thể dẫn đến các ngoại lệ.



## Phát hiện và xử lý ngoại lệ

### • Lỗi thử nghiệm (Experimental Error)

- Một nguyên nhân khác của ngoại lệ là lỗi thử nghiệm.
  - Ví dụ: Trong một cuộc chạy nước rút 100m của 7 vận động viên, một người chạy đã mất tập trung vào lệnh xuất phát khiến anh ta bắt đầu muộn. Điều này khiến thời gian chạy của anh ta nhiều hơn các vận động viên khác. Tổng thời gian chạy của anh ta có thể là một ngoại lệ.

### • Ngoại lệ có chủ ý (Intentional Outlier)

- Điều này thường được tìm thấy trong các tự báo cáo liên quan đến dữ liệu nhạy cảm.
  - Ví dụ: Thanh thiếu niên báo cáo lượng rượu mà họ tiêu thụ. Chỉ một phần trong số họ báo cáo giá trị thực tế. Ở đây các giá trị thực tế có thể trông giống như ngoại lệ vì phần còn lại của thanh thiếu niên báo cáo dưới mức tiêu thụ.



## Phát hiện và xử lý ngoại lệ

### • Lỗi xử lý dữ liệu (Data Processing Error)

- Bất cứ khi nào chúng ta thực hiện khai thác dữ liệu, chúng ta thường trích xuất dữ liệu từ nhiều nguồn. Một số lỗi thao tác hoặc trích xuất có thể dẫn đến các ngoại lệ trong bộ dữ liệu.

### • Lỗi lấy mẫu (Sampling error)

- Ví dụ: Chúng ta phải đo chiều cao của vận động viên. Do nhầm lẫn, chúng ta đo luôn một vài cầu thủ bóng rổ. Việc này có khả năng gây ra các ngoại lệ trong bộ dữ liệu.



## Phát hiện và xử lý ngoại lệ.

### • Ngoại lệ tự nhiên (Natural Outlier)

▪ Khi một ngoại lệ không do người gây ra, đó là một ngoại lệ tự nhiên.

- Ví dụ: Trong nhiệm vụ với một trong những công ty bảo hiểm nổi tiếng, nhà phân tích nhận thấy rằng hiệu suất của 50 cỗ vấn tài chính hàng đầu cao hơn nhiều so với phần còn lại của quần thể. Đáng ngạc nhiên, đó không phải là do bất kỳ lỗi nào. Do đó, bất cứ khi nào họ thực hiện hoạt động khai thác dữ liệu với các cỗ vấn, họ thường xử lý riêng phần khúc.



## Phát hiện và xử lý ngoại lệ

### □ Tác động của ngoại lệ lên tập dữ liệu?

- Các ngoại lệ có thể thay đổi mạnh mẽ kết quả phân tích dữ liệu và mô hình thống kê. Có rất nhiều tác động bất lợi của các ngoại lệ trong bộ dữ liệu:
  - Ngoại lệ làm tăng phương sai lỗi và giảm sức mạnh của kiểm định thống kê
  - Nếu các ngoại lệ được phân phối không ngẫu nhiên, chúng có thể làm giảm tính quy tắc
  - Chúng có thể thiên vị (bias) hoặc ảnh hưởng đến các ước tính có thể được quan tâm thực sự
  - Chúng cũng có thể tác động đến giả định cơ bản của regression, ANOVA và các giả định mô hình thống kê khác.



## Phát hiện và xử lý ngoại lệ

Ví dụ: Kiểm tra xem điều gì xảy ra với tập dữ liệu có và không có ngoại lệ trong tập dữ liệu.

| Without Outlier                 | With Outlier                         |
|---------------------------------|--------------------------------------|
| 4, 4, 5, 5, 5, 5, 6, 6, 6, 7, 7 | 4, 4, 5, 5, 5, 5, 6, 6, 6, 7, 7, 300 |
| Mean = 5.45                     | Mean = 30.00                         |
| Median = 5.00                   | Median = 5.50                        |
| Mode = 5.00                     | Mode = 5.00                          |
| Standard Deviation = 1.04       | Standard Deviation = 85.03           |

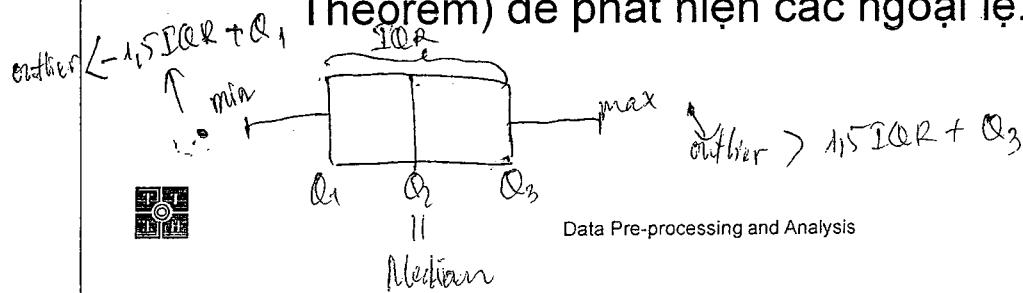
Dữ liệu được thiết lập với các ngoại lệ có độ lệch chuẩn và trung bình khác nhau đáng kể. Khi không có ngoại lệ, trung bình là 5.45. Nhưng với ngoại lệ, trung bình tăng vọt lên 30. Điều này sẽ thay đổi hoàn toàn ước tính.



## Phát hiện và xử lý ngoại lệ

### □ Cách phát hiện ngoại lệ

- Phương pháp được sử dụng phổ biến nhất để phát hiện các ngoại lệ là trực quan hóa: dùng boxplot, histogram, scatter plot.
- Một số nhà phân tích dùng các quy tắc ngón tay cái khác nhau (Ví dụ: Three Sigma Theorem) để phát hiện các ngoại lệ.



$$\text{IQR} = Q_3 - Q_1$$

## Phát hiện và xử lý ngoại lệ

### • Một số quy tắc phát hiện ngoại lệ:

- Bất kỳ giá trị nào vượt quá phạm vi  $Q1 - 1.5 \times IQR$  đến  $Q3 + 1.5 \times IQR$
- Sử dụng capping method: Bất kỳ giá trị nào nằm ngoài phạm vi của phân vị thứ 5 và phân vị thứ 95 có thể được coi là ngoại lệ
- Điểm dữ liệu, độ lệch chuẩn ba hoặc lớn hơn so với giá trị trung bình được coi là ngoại lệ
- Các ngoại lệ Bivariate và multivariate thường được đo bằng cách sử dụng chỉ số ảnh hưởng hoặc khoảng cách.
- Phát hiện ngoại lệ chỉ là một trường hợp đặc biệt của việc kiểm tra dữ liệu cho các điểm dữ liệu có ảnh hưởng và nó cũng phụ thuộc vào sự hiểu biết của doanh nghiệp.

Data Pre-processing and Analysis

71

## Phát hiện và xử lý ngoại lệ

### □ Cách loại bỏ ngoại lệ

- Hầu hết các cách để đối phó với các ngoại lệ tương tự như các cách đối phó với thiếu giá trị như xóa các mẫu, bień đổi chúng, binning, coi chúng như một nhóm riêng biệt, đưa ra các giá trị và các phương pháp thống kê khác...

Data Pre-processing and Analysis

72

## Phát hiện và xử lý ngoại lệ

### • Xóa mẫu (Deleting observation)

- Chúng ta xóa các giá trị ngoại lệ nếu đó là do lỗi nhập dữ liệu, lỗi xử lý dữ liệu hoặc các quản sát

khi xóa nó ảnh hưởng đến ngoại lệ rất nhỏ về số lượng. Chúng ta cũng có thể sử dụng cắt tỉa ở cả hai đầu để loại bỏ các ngoại lệ.



## Phát hiện và xử lý ngoại lệ

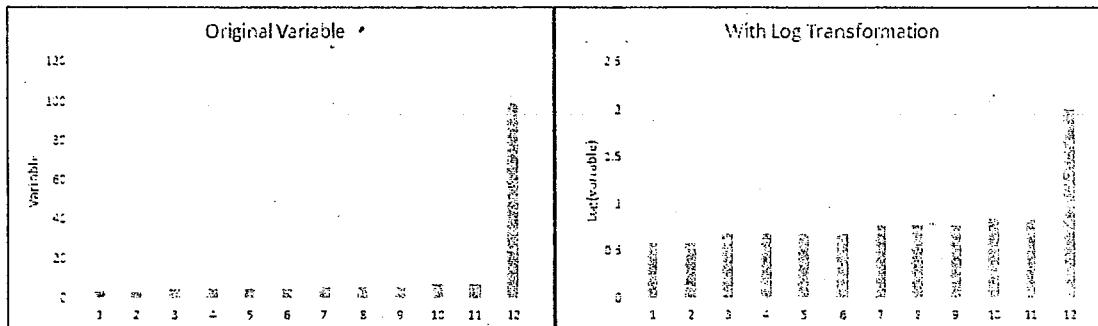
### • Biến đổi giá trị (Transforming and binning value)

- Biến đổi các biến cũng có thể loại bỏ các ngoại lệ. Áp dụng log tự nhiên của một giá trị làm giảm sự thay đổi gây ra bởi các giá trị cực nhỏ/lớn. Binning cũng là một hình thức biến đổi. Thuật toán Cây quyết định (Decision Tree) cho phép xử lý tốt các ngoại lệ do việc tạo biến. Chúng ta cũng có thể sử dụng quá trình gán trọng số cho các mẫu khác nhau.



## Phát hiện và xử lý ngoại lệ

### Ví dụ



## Phát hiện và xử lý ngoại lệ

### Điền giá trị (Imputing)

Giống như việc loại bỏ các giá trị bị thiếu, chúng ta cũng có thể điền giá trị cho các ngoại lệ bằng mean, median, mode. Trước khi đưa ra các giá trị, chúng ta nên phân tích xem đó là ngoại lệ tự nhiên hay nhân tạo. Nếu nó là nhân tạo, chúng ta có thể điền với các giá trị. Chúng ta cũng có thể sử dụng mô hình thống kê để dự đoán các giá trị mẫu ngoại lệ và sau đó chúng ta có thay nó với các giá trị dự đoán.

## Phát hiện và xử lý ngoại lệ

### • Tách riêng (Treat separately)

\* Nếu có số lượng ngoại lệ đáng kể, chúng ta nên xử lý chúng riêng biệt trong mô hình thống kê.

Một trong những cách tiếp cận là tách thành hai nhóm khác nhau và xây dựng mô hình riêng cho cả hai nhóm sau đó kết hợp đầu ra.

(Ví: trước năm 2018, dữ liệu 10 vạn số từ 2018, doanh nghiệp thêm Kỹ thuật dữ liệu → Tách 2 nhóm) và sử dụng 1 thuật toán → có thể gồm 80 or 10?  
Hoặc Ví: Lượng mua sắm cao gấp 5 lần so với tập cũ lại → Tách ra riêng 2 nhóm.



## Phát hiện và xử lý ngoại lệ

### • Thực hiện các phân tích khác nhau

\* Phân tích dữ liệu trong cả hai trường hợp: có và không có ngoại lệ, và ghi lại chú thích về kết quả đã thay đổi như thế nào trong cả hai trường hợp

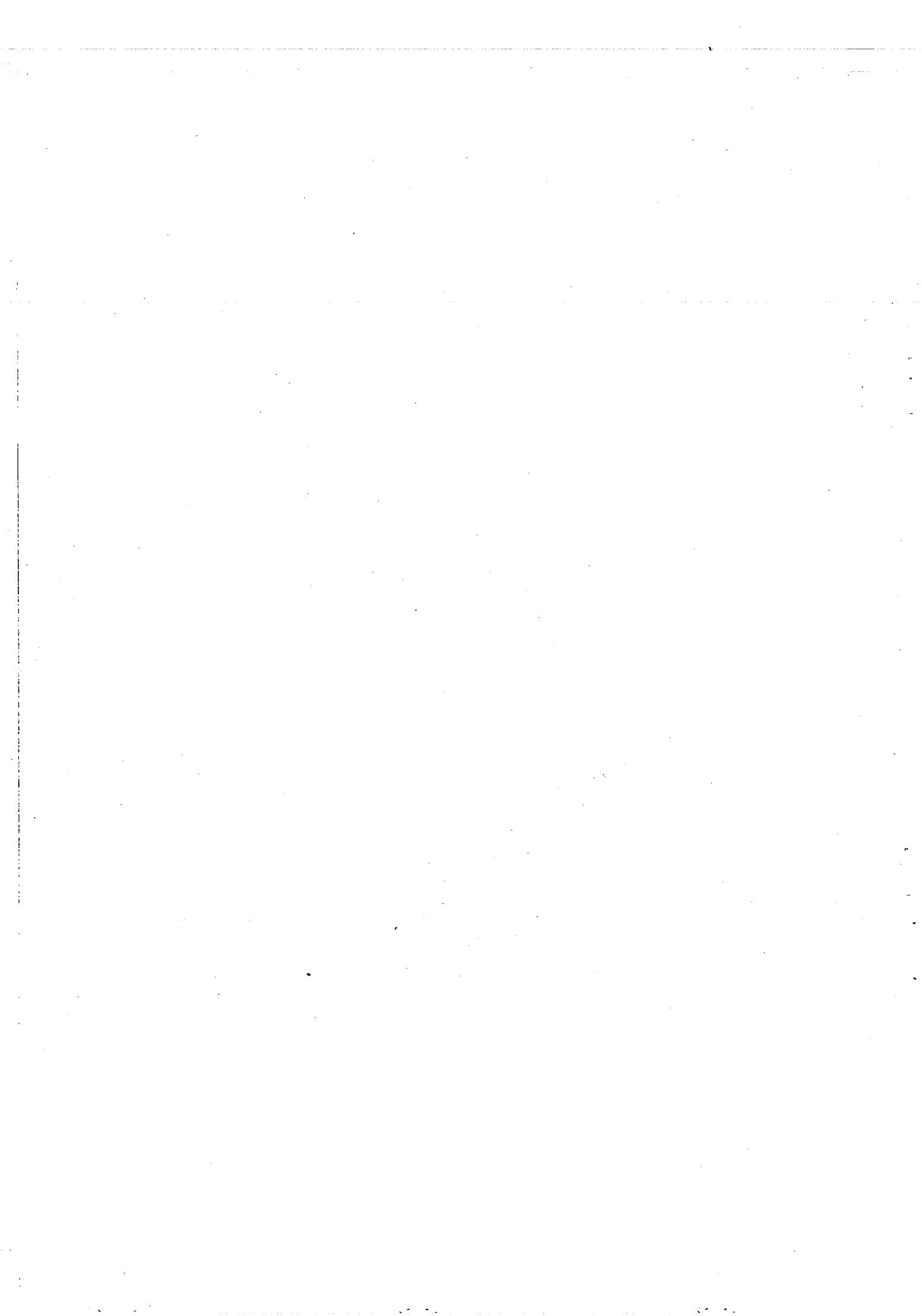




## Phát hiện và xử lý ngoại lệ

- Tuy nhiên, outliers có thể là những mẫu hợp pháp và đôi khi lại là những điều thú vị nhất.  
=> Rất quan trọng để điều tra bản chất của ngoại lệ trước khi quyết định.







## Chapter 2 - Ex1: Housing prices

Cho dữ liệu housing-prices-dataset/train.csv

**Yêu cầu: Thực hiện các công việc sau**

1. Xác định các thuộc tính
2. Phân tích đơn biến
  - 2.1 Để dự đoán giá nhà, giả sử cần các thông tin sau: 'LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd' => phân tích các biến này
3. Phân tích hai biến
4. Xử lý dữ liệu thiếu
5. Phát hiện và xử lý ngoại lệ

biến này.

In [1]: # Link: <https://www.kaggle.com/alphaepsilon/housing-prices-dataset>

In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
```

In [3]:

```
df = pd.read_csv("housing-prices-dataset/train.csv")
df.shape
```

Out[3]: (1460, 81)

In [4]: # df.info()

In [5]: # df.isnull().sum() # dùng để đếm các giá trị null từ các cột

### 1. Xác định các thuộc tính

1. Input: <> SalePrice
2. Output: SalePrice
3. Type of variable:
  - 3.1 Predictor: khác SalePrice
  - 3.2 Target: SalePrice
4. Data Type:
  - 4.1 Charactor/String
  - 4.2 Numeric
5. Variable Category:

bên numeric

In [6]: numbers = [f for f in df.columns if df.dtypes[f] != 'object'] # Quantitative:



```
In [7]: list_nums = ', '.join(numbers)
list_nums
```

```
Out[7]: 'Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, YearBuilt, Yea  
rRemodAdd, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrS  
F, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, BsmtHalfBath, FullBath, Hal  
fBath, BedroomAbvGr, KitchenAbvGr, TotRmsAbvGrd, Fireplaces, GarageYrBlt, Garag  
eCars, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPor  
ch, PoolArea, MiscVal, MoSold, YrSold, SalePrice'
```

*Tim's last bln' object*

```
In [8]: objects = [f for f in df.columns if df.dtypes[f] == 'object'] # Qualitative
```

```
In [9]: list_obj = ', '.join(objects)
list_obj
```

```
Out[9]: 'MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConfig, LandSlop  
e, Neighborhood, Condition1, Condition2, BldgType, HouseStyle, RoofStyle, RoofM  
atl, Exterior1st, Exterior2nd, MasVnrType, ExterQual, ExterCond, Foundation, Bs  
mtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, Heating, HeatingQC,  
CentralAir, Electrical, KitchenQual, Functional, FireplaceQu, GarageType, Garag  
eFinish, GarageQual, GarageCond, PavedDrive, PoolQC, Fence, MiscFeature, SaleTy  
pe, SaleCondition'
```

## 5. Variable

### 5.1 Categorical:

### 5.2 Continuous

# bản phân loại, Kết luận

7/24/2019

Ex1\_house\_pricing-print



```
In [10]: # Categorical:     group him theo bùn, không có dữ liệu biến phân loại
          i = 1
          for obj in objects:
              print(i, "/", obj, "\t", len(df[obj].unique()), ":" , df[obj].unique())
              i = i+1
```

bảng ta unique (các bùn, không có dữ liệu).

|                   |   |   |
|-------------------|---|---|
| 1 / MSZoning      | 5 : ['RL' 'RM' 'C (all)' 'FV' 'RH']   | → 5 loại → phân loại (gồm 5 loại?)                |
| 2 / Street        | 2 : ['Pave' 'Grvl']   | → 2 loại (pave và cát) → phân loại                |
| 3 / Alley         | 3 : [nan 'Grvl' 'Pave']   | → có dữ liệu nan" → giá xem xét null → ua xem xét |
| 4 / LotShape      | 4 : ['Reg' 'IR1' 'IR2' 'IR3']   |   |
| 5 / LandContour   | 4 : ['Lvl' 'Bnk' 'Low' 'HLS']   |   |
| 6 / Utilities     | 2 : ['AllPub' 'NoSeWa']   |   |
| 7 / LotConfig     | 5 : ['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']   |   |
| 8 / LandSlope     | 3 : ['Gtl' 'Mod' 'Sev']   |   |
| 9 / Neighborhood  | 25 : ['CollgCr' 'Veenker' 'Crawfor' 'NoRidge' 'Mitche<br>l' 'Somerst' 'NWAmes'<br>'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR'<br>'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill'<br>'Blmgtn' 'BrDale' 'SWISU' 'Blueste'] |   |
| 10 / Condition1   | 9 : ['Norm' 'Feedr' 'PosN' 'Artery' 'RRAe' 'RRNn' 'RR<br>An' 'PosA' 'RRNe']   |   |
| 11 / Condition2   | 8 : ['Norm' 'Artery' 'RRNn' 'Feedr' 'PosN' 'PosA' 'RR<br>An' 'RRAe']  |   |
| 12 / BldgType     | 5 : ['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']   |   |
| 13 / HouseStyle   | 8 : ['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'SL<br>vl' '2.5Unf' '2.5Fin']   |   |
| 14 / RoofStyle    | 6 : ['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']   |   |
| 15 / RoofMatl     | 8 : ['CompShg' 'WdShngl' 'Metal' 'WdShake' 'Membran' 'Tar&Gr<br>v' 'Roll'<br>'ClyTile']   |   |
| 16 / Exterior1st  | 15 : ['VinylSd' 'MetalSd' 'Wd Sdng' 'HdBoard' 'BrkFac<br>e' 'WdShing' 'CemntBd'<br>'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc'<br>'CBlock']  |   |
| 17 / Exterior2nd  | 16 : ['VinylSd' 'MetalSd' 'Wd Shng' 'HdBoard' 'Plywoo<br>d' 'Wd Sdng' 'CmentBd'<br>'BrkFace' 'Stucco' 'AsbShng' 'Brk Cmn' 'ImStucc' 'AsphShn' 'Stone'<br>'Other' 'CBlock']  |   |
| 18 / MasVnrType   | 5 : ['BrkFace' 'None' 'Stone' 'BrkCmn' 'nan']   |   |
| 19 / ExterQual    | 4 : ['Gd' 'TA' 'Ex' 'Fa']   |   |
| 20 / ExterCond    | 5 : ['TA' 'Gd' 'Fa' 'Po' 'Ex']  |   |
| 21 / Foundation   | 6 : ['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Stone']   |   |
| 22 / BsmtQual     | 5 : ['Gd' 'TA' 'Ex' 'nan' 'Fa']   |   |
| 23 / BsmtCond     | 5 : ['TA' 'Gd' 'nan' 'Fa' 'Po']   |   |
| 24 / BsmtExposure | 5 : ['No' 'Gd' 'Mn' 'Av' 'nan']   |   |
| 25 / BsmtFinType1 | 7 : ['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' 'nan' 'LWQ']   |   |
| 26 / BsmtFinType2 | 7 : ['Unf' 'BLQ' 'nan' 'ALQ' 'Rec' 'LwQ' 'GLQ']   |   |
| 27 / Heating      | 6 : ['GasA' 'GasW' 'Grav' 'Wall' 'OthW' 'Floor']  |   |
| 28 / HeatingQC    | 5 : ['Ex' 'Gd' 'TA' 'Fa' 'Po']  |   |
| 29 / CentralAir   | 2 : ['Y' 'N']   |   |
| 30 / Electrical   | 6 : ['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix' 'nan']   |   |
| 31 / KitchenQual  | 4 : ['Gd' 'TA' 'Ex' 'Fa']   |   |
| 32 / Functional   | 7 : ['Typ' 'Min1' 'Maj1' 'Min2' 'Mod' 'Maj2' 'Sev']   |   |
| 33 / FireplaceQu  | 6 : [nan 'TA' 'Gd' 'Fa' 'Ex' 'Po']  |   |
| 34 / GarageType   | 7 : ['Attchd' 'Detchd' 'BuiltIn' 'CarPort' 'nan' 'Basme<br>nt' '2Types']  |   |
| 35 / GarageFinish | 4 : ['RFn' 'Unf' 'Fin' 'nan']   |   |

7/24/2019

Ex1\_house\_pricing-print



```
36 / GarageQual      6 : ['TA' 'Fa' 'Gd' 'nan 'Ex' 'Po']
37 / GarageCond      6 : ['TA' 'Fa' 'nan 'Gd' 'Po' 'Ex']
38 / PavedDrive      3 : ['Y' 'N' 'P']
39 / PoolQC          4 : [nan 'Ex' 'Fa' 'Gd']
40 / Fence            5 : [nan 'MnPrv' 'GdWo' 'GdPrv' 'MnWw']
41 / MiscFeature     5 : [nan 'Shed' 'Gar2' 'Othr' 'TenC']
42 / SaleType         9 : ['WD' 'New' 'COD' 'ConLD' 'ConLI' 'CWD' 'ConLw' 'Con' 'Ot
h']
43 / SaleCondition   6 : ['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca'
'Family']
```

Kiểm tra những giá trị & biến số.

7/24/2019

Ex1\_house\_pricing-print



In [11]: # Categorical & Continuous

```
i = 1
for obj in numbers:
    print(i, "/", obj, "\t", len(df[obj].unique()), ":", df[obj].unique() if len(df[obj].unique()) < 150 else ")")
```

```
1 / Id 1460 :
2 / MSSubClass 15 : [ 60 20 70 50 190 45 90 120 30 85 80 160 75 180 Khi số lượng >= 40] → dữ liệu
3 / LotFrontage 111 : [ 65. 80. 68. 60. 84. 85. 75. nan 51. Nếu 150 <= n
50. 70. 91. 72. 66. 101. 57. 44. 110. 98. 47. 108. 112. 74. 115. 61. 48. 33. 52. 100. 24. 89. 63. 76. 81. 95. 69. 21. 32. 78. 121. 122. 40. 105. 73. 77. 64. 94. 34. 90. 55. 88. 82. 71. 120. 107. 92. 134. 62. 86. 141. 97. 54. 41. 79. 174. 99. 67. 83. 43. 103. 93. 30. 129. 140. 35. 37. 118. 87. 116. 150. 111. 49. 96. 59. 36. 56. 102. 58. 38. 109. 130. 53. 137. 45. 106. 104. 42. 39. 144. 114. 128. 149. 313. 168. 182. 138. 160. 152. 124. 153. 46. ]
4 / LotArea 1073 : Một số (1500 số) từ 1083 đến 1083, bao gồm cả số
5 / OverallQual 10 : [ 7 6 8 5 9 4 10 3 1 2]
6 / OverallCond 9 : [ 5 8 6 7 4 2 3 9 1]
7 / YearBuilt 112 : [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 1965 2005 1962 2006
1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
1979 1885 1919 1990 1969 1935 1988 1971 1952 1936 1923 1924 1984 1926
1940 1941 1987 1986 2008 1908 1892 1916 1932 1918 1912 1947 1925 1900
1980 1989 1992 1949 1880 1928 1978 1922 1996 2010 1946 1913 1937 1942
1938 1974 1893 1914 1906 1890 1898 1904 1882 1875 1911 1917 1872 1905]
8 / YearRemodAdd 61 : [2003 1976 2002 1970 2000 1995 2005 1973 1950 19
65 2006 1962 2007 1960
2001 1967 2004 2008 1997 1959 1990 1955 1983 1980 1966 1963 1987 1964
1972 1996 1998 1989 1953 1956 1968 1981 1992 2009 1982 1961 1993 1999
1985 1979 1977 1969 1958 1991 1971 1952 1975 2010 1984 1986 1994 1988
1954 1957 1951 1978 1974]
9 / MasVnrArea 328 :
10 / BsmtFinSF1 637 :
11 / BsmtFinSF2 144 : [ 0 32 668 486 93 491 506 712 362
41 169 869 150 670
28 1080 181 768 215 374 208 441 184 279 306 180 580 690
692 228 125 1063 620 175 820 1474 264 479 147 232 380 544
294 258 121 391 531 344 539 713 210 311 1120 165 532 96
495 174 1127 139 202 645 123 551 219 606 612 480 182 132
336 468 287 35 499 723 119 40 117 239 80 472 64 1057
127 630 128 377 764 345 1085 435 823 500 290 324 634 411
841 1061 466 396 354 149 193 273 465 400 682 557 230 106
791 240 547 469 177 108 600 492 211 168 1031 438 375 144
81 906 608 276 661 68 173 972 105 420 546 334 352 872
110 627 163 1029]
12 / BsmtUnfSF 780 :
13 / TotalBsmtSF 721 :
14 / 1stFlrSF 753 :
15 / 2ndFlrSF 417 :
16 / LowQualFinSF 24 : [ 0 360 513 234 528 572 144 392 371 390 420 473
156 515 80 53 232 481
120 514 397 479 205 384]
```



17 / GrLivArea 861 :  
 18 / BsmtFullBath 4 : [1 0 2 3]  
 19 / BsmtHalfBath 3 : [0 1 2]  
 20 / FullBath 4 : [2 1 3 0]  
 21 / HalfBath 3 : [1 0 2]  
 22 / BedroomAbvGr 8 : [3 4 1 2 0 5 6 8]  
 23 / KitchenAbvGr 4 : [1 2 3 0]  
 24 / TotRmsAbvGrd 12 : [ 8 6 7 9 5 11 4 10 12 3 2 14]  
 25 / Fireplaces 4 : [0 1 2 3]  
 26 / GarageYrBlt 98 : [2003. 1976. 2001. 1998. 2000. 1993. 2004. 1973.  
 1931. 1939. 1965. 2005.  
 1962. 2006. 1960. 1991. 1970. 1967. 1958. 1930. 2002. 1968. 2007. 2008.  
 1957. 1920. 1966. 1959. 1995. 1954. 1953. nan 1983. 1977. 1997. 1985.  
 1963. 1981. 1964. 1999. 1935. 1990. 1945. 1987. 1989. 1915. 1956. 1948.  
 1974. 2009. 1950. 1961. 1921. 1900. 1979. 1951. 1969. 1936. 1975. 1971.  
 1923. 1984. 1926. 1955. 1986. 1988. 1916. 1932. 1972. 1918. 1980. 1924.  
 1996. 1940. 1949. 1994. 1910. 1978. 1982. 1992. 1925. 1941. 2010. 1927.  
 1947. 1937. 1942. 1938. 1952. 1928. 1922. 1934. 1906. 1914. 1946. 1908.  
 1929. 1933.]  
 27 / GarageCars 5 : [2 3 1 0 4]  
 28 / GarageArea 441 :  
 29 / WoodDeckSF 274 :  
 30 / OpenPorchSF 202 :  
 31 / EnclosedPorch 120 : [ 0 272 228 205 176 87 172 102 37 144 64 11  
 4 202 128 156 44 77 192  
 140 180 183 39 184 40 552 30 126 96 60 150 120 112 252 52 224 234  
 244 268 137 24 108 294 177 218 242 91 160 130 169 105 34 248 236 32  
 80 115 291 116 158 210 36 200 84 148 136 240 54 100 189 293 164 216  
 239 67 90 56 129 98 143 70 386 154 185 134 196 264 275 230 254 68  
 194 318 48 94 138 226 174 19 170 220 214 280 190 330 208 145 259 81  
 42 123 152 286 168 20 301 198 221 212 50 99]  
 32 / 3SsnPorch 20 : [ 0 320 407 130 180 168 140 508 238 245 196 144 182 162  
 23 216 96 153  
 290 304]  
 33 / ScreenPorch 76 : [ 0 176 198 291 252 99 184 168 130 142 192 410  
 224 266 170 154 153 144  
 128 259 160 271 234 374 185 182 90 396 140 276 180 161 145 200 122 95  
 120 60 126 189 260 147 385 287 156 100 216 210 197 204 225 152 175 312  
 222 265 322 190 233 63 53 143 273 288 263 80 163 116 480 178 440 155  
 220 119 165 40]  
 34 / PoolArea 8 : [ 0 512 648 576 555 480 519 738]  
 35 / MiscVal 21 : [ 0 700 350 500 400 480 450 15500 1200  
 800 2000 600  
 3500 1300 54 620 560 1400 8300 1150 2500]  
 36 / MoSold 12 : [ 2 5 9 12 10 8 11 4 1 7 3 6]  
 37 / YrSold 5 : [2008 2007 2006 2009 2010]  
 38 / SalePrice 663 :

In [12]: # Quan sát 2 kết quả trên để kết Luận

## 2. Phân tích đơn biến

In [13]: features = ['LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'BedroomAb



```
In [14]: i = 1
for obj in features:
    print(i, "/", obj, "\t", len(df[obj].unique()), ":", df[obj].unique() if len(df[obj].unique()) > 10 else df[obj].unique().head(10))
    i = i+1

1 / LotArea      1073 :
2 / YearBuilt    112 : [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 1965
2005 1962 2006
1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
1979 1885 1919 1990 1969 1935 1988 1971 1952 1936 1923 1924 1984 1926
1940 1941 1987 1986 2008 1908 1892 1916 1932 1918 1912 1947 1925 1900
1980 1989 1992 1949 1880 1928 1978 1922 1996 2010 1946 1913 1937 1942
1938 1974 1893 1914 1906 1890 1898 1904 1882 1875 1911 1917 1872 1905]
3 / 1stFlrSF     753 :
4 / 2ndFlrSF     417 :
5 / FullBath     4 : [2 1 3 0]
6 / BedroomAbvGr 8 : [3 4 1 2 0 5 6 8]
7 / TotRmsAbvGrd 12 : [ 8 6 7 9 5 11 4 10 12 3 2 14]
```

## Continuous variable

- LotArea
- 1stFlrSF
- 2ndFlrSF

```
In [15]: # LotArea
# Central Tendency
df['LotArea'].describe(include='all')
```

```
Out[15]: count      1460.000000
mean      10516.828082
std       9981.264932 độ lệch chuẩn lsn.
min      1300.000000
25%      7553.500000
50%      9478.500000
75%      11601.500000
max      215245.000000 đt ln' nkt »> mean.
Name: LotArea, dtype: float64
```

```
In [16]: LotArea_median = df.LotArea.median()
LotArea_median
```

Out[16]: 9478.5

```
In [17]: LotArea_mode = df.LotArea.mode()
LotArea_mode
```

Out[17]: 0 7200
dtype: int64

7/24/2019

Ex1\_house\_pricing-print



In [18]: # Measure of Dispersion  
LotArea\_range = df.LotArea.ptp()  
LotArea\_range

Out[18]: 213945

In [19]: Q1 = np.percentile(df.LotArea, 25)  
Q1

Out[19]: 7553.5

In [20]: Q3 = np.percentile(df.LotArea, 75)  
Q3

Out[20]: 11601.5

In [21]: LotArea\_iqr = scipy.stats.iqr(df.LotArea)  
LotArea\_iqr

Out[21]: 4048.0

In [22]: LotArea\_var = df.LotArea.var()  
LotArea\_var

Out[22]: 99625649.6503417

In [23]: LotArea\_std = df.LotArea.std()  
LotArea\_std

Out[23]: 9981.264932379147

In [24]: LotArea\_skew = df.LotArea.skew()  
LotArea\_skew

Out[24]: 12.207687851233496

In [25]: # LotArea\_skew > 0 => phân phối lệch phải  
scipy.stats.skew(df.LotArea)

Out[25]: 12.195142125084478

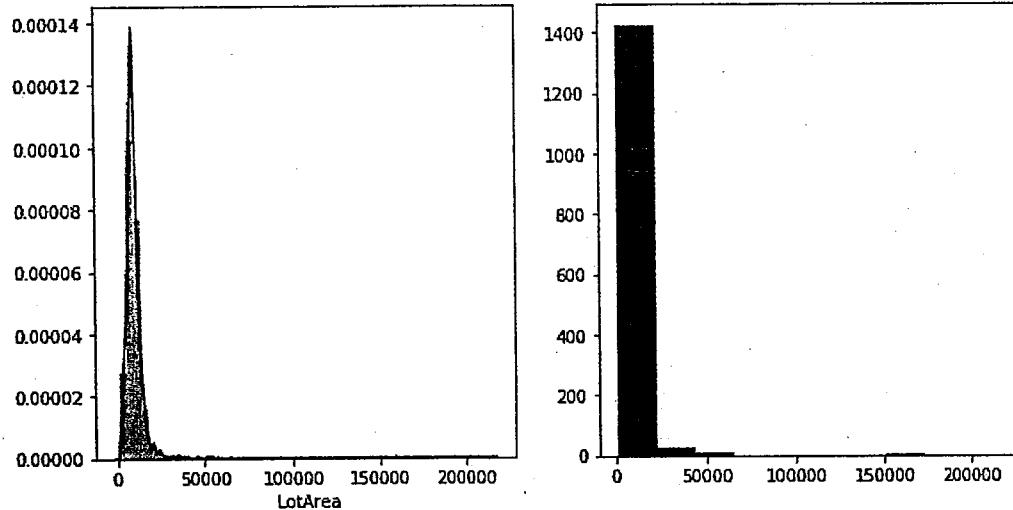
In [26]: LotArea\_kur = df.LotArea.kurtosis()  
LotArea\_kur

Out[26]: 203.24327101886033

In [27]: # LotArea\_kur > 0: phân phối nhọn hơn phân phối chuẩn  
scipy.stats.kurtosis(df.LotArea)

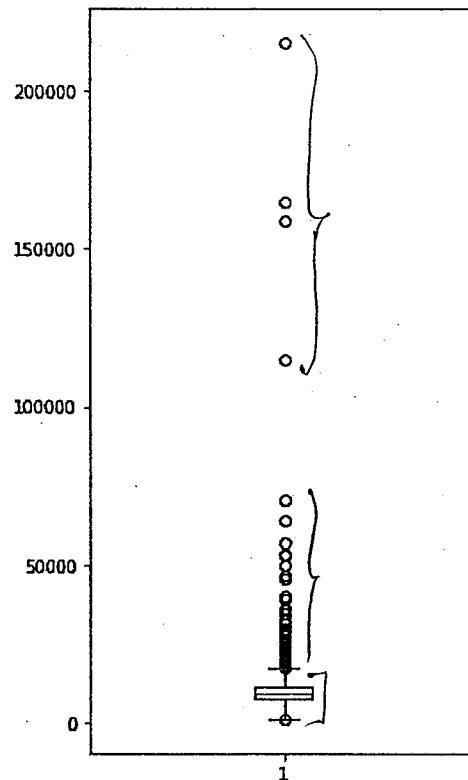
Out[27]: 202.5437927513529

```
In [28]: # Visualization  
# Histogram  
plt.figure(figsize=(10, 5))  
plt.subplot(1, 2, 1)  
sns.distplot(df.LotArea)  
plt.subplot(1, 2, 2)  
plt.hist(df.LotArea)  
plt.show()
```





```
In [29]: # Boxplot
plt.figure(figsize=(4,8))
plt.boxplot(df.LotArea)
plt.show()
```



```
In [30]: # Number of upper outliers
n_O_upper = df[df.LotArea > (Q3 + 1.5*LotArea_iqr)].shape[0]
n_O_upper
```

Out[30]: 67

```
In [31]: # Number of lower outliers
n_O_lower = df[df.LotArea < (Q1 - 1.5*LotArea_iqr)].shape[0]
n_O_lower
```

Out[31]: 2

```
In [32]: # Percentage of outliers
outliers_per = (n_O_lower + n_O_upper)/df.shape[0]
outliers_per
```

Out[32]: 0.04726027397260274 → 5% outlier

```
In [33]: # xem xét loại bỏ outliers ???
```

Note: Hai thuộc tính còn lại ((1stFlrSF, 2ndFlrSF)) làm tương tự như trên



## Categorical Variables

- FullBath
- BedroomAbvGr
- TotRmsAbvGrd

In [34]: `# FullBath  
FullBath_count = (df.groupby("FullBath").count())["Id"]` → bùn jān lõai  
FullBath\_count

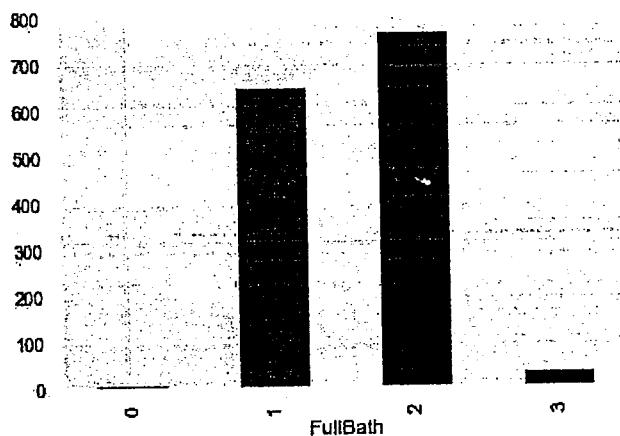
Out[34]: FullBath  

|   |     |
|---|-----|
| 0 | 9   |
| 1 | 650 |
| 2 | 768 |
| 3 | 33  |

Name: Id, dtype: int64

In [35]: `sns.set()  
FullBath_count.plot.bar()` → dùng bar chart .

Out[35]: <matplotlib.axes.\_subplots.AxesSubplot at 0x190699037b8>



In [36]: `# Note: Hai thuộc tính còn Lại ((BedroomAbvGr, TotRmsAbvGrd)) Làm tương tự như trê`

7/24/2019

Ex1\_house\_pricing-print

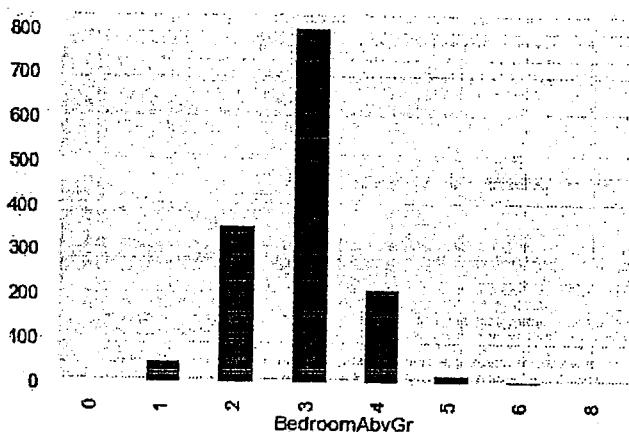


```
In [37]: # BedroomAbvGr  
BedroomAbvGr_count = (df.groupby("BedroomAbvGr").count())["Id"]  
BedroomAbvGr_count
```

```
Out[37]: BedroomAbvGr      85' fony hym'  
0      6  
1     50  
2    358  
3   804  
4   213  
5    21  
6     7  
8     1  
Name: Id, dtype: int64
```

```
In [38]: sns.set()  
BedroomAbvGr_count.plot.bar()
```

```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x190699781d0>
```



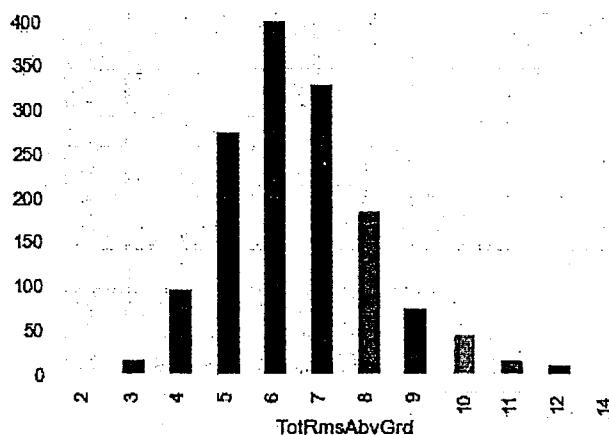
```
In [39]: # TotRmsAbvGrd  
TotRmsAbvGrd_count = (df.groupby("TotRmsAbvGrd").count())["Id"]  
TotRmsAbvGrd_count
```

```
Out[39]: TotRmsAbvGrd      15' fony  
2      1  
3     17  
4    97  
5   275  
6   402  
7   329  
8   187  
9    75  
10   47  
11   18  
12   11  
14   1 (Nhà ngắn kẽ có fony 13)  
Name: Id, dtype: int64
```



```
In [40]: sns.set()  
TotRmsAbvGrd_count.plot.bar()
```

```
Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x190699d8ac8>
```



### 3. Phân tích hai biến

- Continuous & Continuous ✓
- Categorical & Categorical ✓
- Categorical & Continuous ✓

7/24/2019

Ex1\_house\_pricing-print

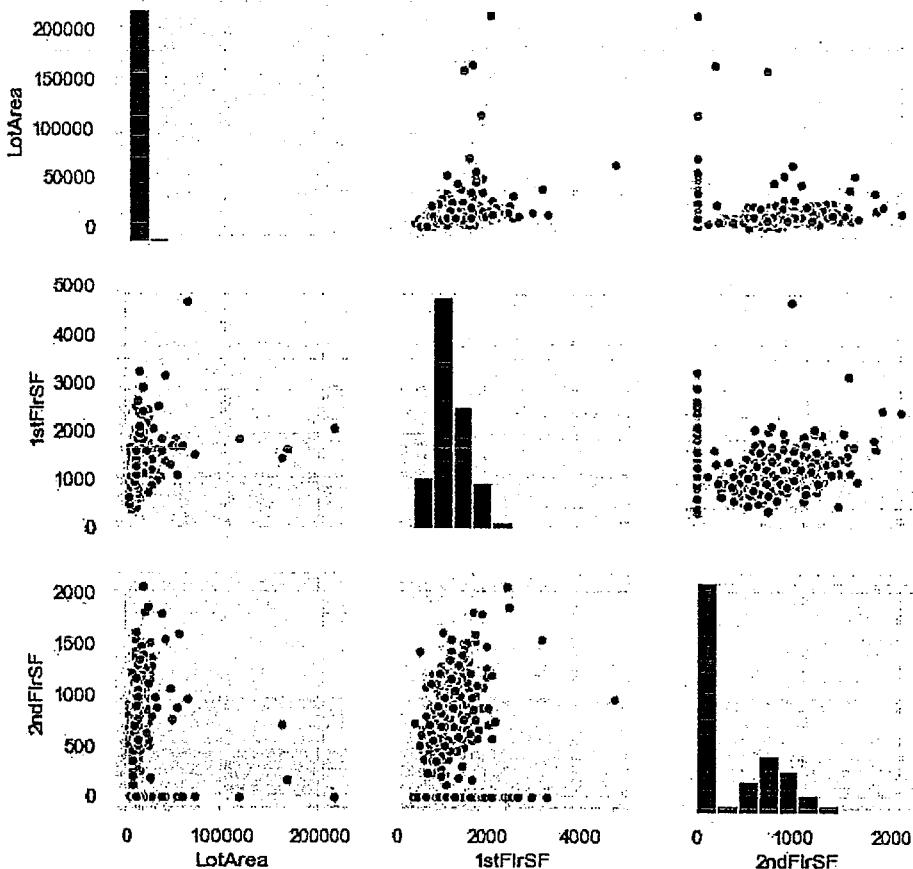


In [41]: # Continuous & Continuous \*

```
sns.pairplot(df[['LotArea', '1stFlrSF', '2ndFlrSF']])
```

Out[41]: <seaborn.axisgrid.PairGrid at 0x19069a306a0>

3 biến liên tục  
→ Xem xét  
quan hệ  
các cặp biến  
l天赋



In [42]: # Không quan hệ tuyến tính  
df[['LotArea', '1stFlrSF', '2ndFlrSF']].corr()

Out[42]:

|          | LotArea  | 1stFlrSF  | 2ndFlrSF  |
|----------|----------|-----------|-----------|
| LotArea  | 1.000000 | 0.299475  | 0.050986  |
| 1stFlrSF | 0.299475 | 1.000000  | -0.202646 |
| 2ndFlrSF | 0.050986 | -0.202646 | 1.000000  |

→ 3 biến liên tục ko liên quan với  
nhau → jùi sd 3 biến để pt  
(ko bù đù)

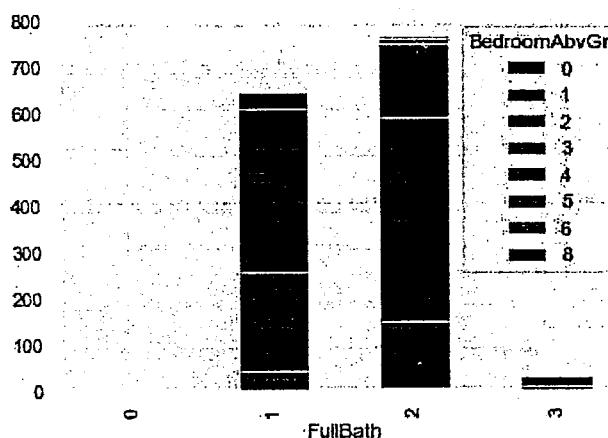


```
In [43]: # Categorical & Categorical
# 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd'
# contingency.table: Ho: 'FullBath' and 'BedroomAbvGr' independent
table_FB = pd.crosstab(df['FullBath'], df['BedroomAbvGr'])
table_FB
```

Out[43]:

| BedroomAbvGr | 0 | 1 | 2  | 3   | 4   | 5   | 6  | 8 |
|--------------|---|---|----|-----|-----|-----|----|---|
| FullBath     | 0 | 5 | 2  | 2   | 0   | 0   | 0  | 0 |
| 0            | 1 | 1 | 42 | 213 | 352 | 37  | 5  | 0 |
| 1            | 2 | 0 | 6  | 143 | 443 | 156 | 12 | 7 |
| 2            | 3 | 0 | 0  | 0   | 9   | 20  | 4  | 0 |
| 3            | 0 | 0 | 0  | 0   | 0   | 0   | 0  | 0 |

In [44]: table\_FB.plot(kind='bar', stacked=True) → Dùng stacked (bên trên logo)



In [45]: # chi-squared test with similar proportions
from scipy.stats import chi2\_contingency
from scipy.stats import chi2

In [46]: # Chi-Square Test
stat, p, dof, expected = chi2\_contingency(table\_FB)
print('dof=%d' % dof)
print('p=', p)

dof=21  
p= 9.022959522651409e-177 → {0.05 (hàc bô thô)}

In [47]: # interpret test-statistic
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f' % (prob, critical, stat))

probability=0.950, critical=32.671, stat=898.930



```
In [48]: # interpret p-value
alpha = 1.0 - prob
print('significance=% .3f, p=% .3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

significance=0.050, p=0.000  
Dependent (reject H0)

→ Kết luận rằng có liên quan mật (giả thiết).

Note: Các cặp còn lại làm tương tự như trên

```
In [49]: # Categorical & Continuous
# 'FullBath' có bị ảnh hưởng bởi 'LotArea'? => ANOVA ONEWAY
table_FB = pd.crosstab(df['FullBath'], df['BedroomAbvGr'])
table_FB
```

Out[49]:

| BedroomAbvGr | 0 | 1  | 2   | 3   | 4   | 5  | 6 | 8 |
|--------------|---|----|-----|-----|-----|----|---|---|
| FullBath     |   |    |     |     |     |    |   |   |
| 0            | 5 | 2  | 2   | 0   | 0   | 0  | 0 | 0 |
| 1            | 1 | 42 | 213 | 352 | 37  | 5  | 0 | 0 |
| 2            | 0 | 6  | 143 | 443 | 156 | 12 | 7 | 1 |
| 3            | 0 | 0  | 0   | 9   | 20  | 4  | 0 | 0 |

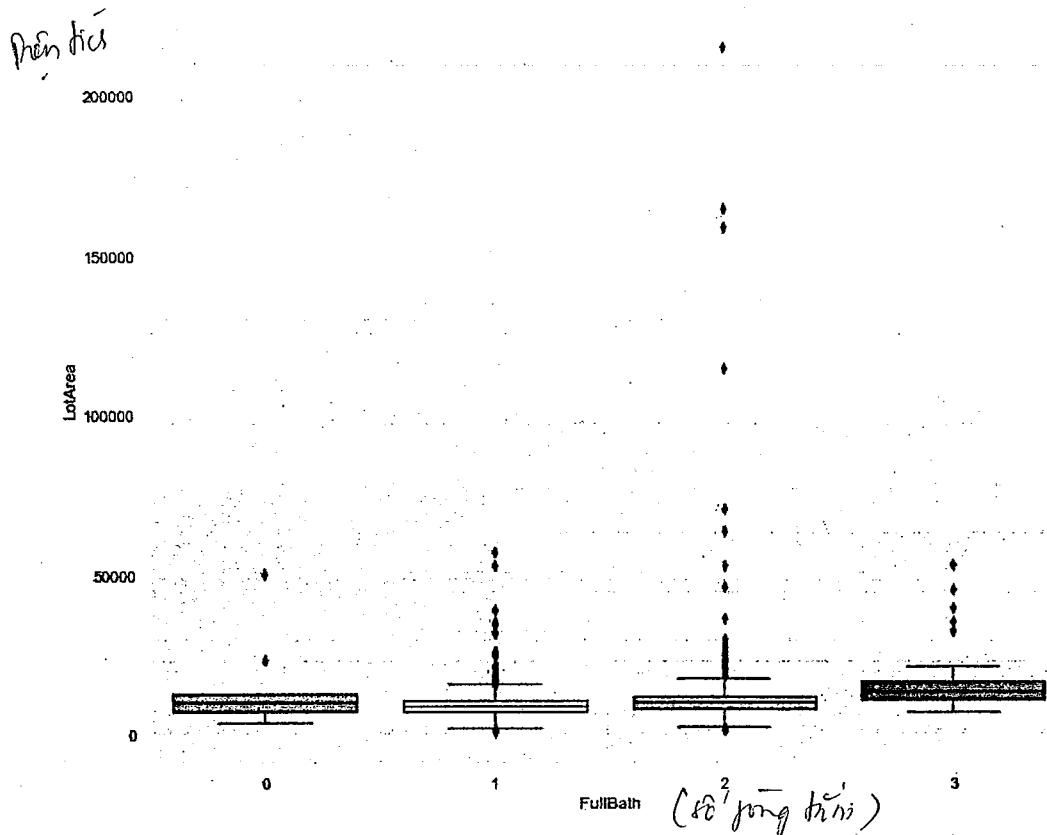
```
In [50]: df_sub = df[['FullBath', 'LotArea']]
df_sub.head()
```

Out[50]:

|   | FullBath | LotArea |
|---|----------|---------|
| 0 | 2        | 8450    |
| 1 | 2        | 9600    |
| 2 | 2        | 11250   |
| 3 | 1        | 9550    |
| 4 | 2        | 14260   |



```
In [91]: import matplotlib.pyplot as plt
plt.figure(figsize=(12,10))
sns.boxplot(x="FullBath", y="LotArea", data=df_sub, palette="Set3")
plt.show()
```



```
In [52]: import statsmodels.api as sm
from statsmodels.formula.api import ols
```

```
In [53]: model = ols('LotArea ~ C(FullBath)', data=df_sub).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

Xem lại lý thuyết

Out[53]:

|             | sum_sq       | df     | F         | PR(>F)       |
|-------------|--------------|--------|-----------|--------------|
| C(FullBath) | 3.332090e+09 | 3.0    | 11.386809 | 2.207543e-07 |
| Residual    | 1.420217e+11 | 1456.0 | NaN       | NaN          |

$p < 0.05 \rightarrow$  có liên quan nhau.

```
In [54]: # Giải thích: P-value thu được từ phân tích ANOVA cho LotArea và FullBath phối hợp
# Kết Luận: LotArea ảnh hưởng đáng kể đến FullBath
```

7/24/2019

xem lín lý thuyết?

Ex1\_house\_pricing-print



```
In [55]: from statsmodels.stats.multicomp import pairwise_tukeyhsd  
# perform multiple pairwise comparison (Tukey HSD)  
m_comp = pairwise_tukeyhsd(endog=df_sub['LotArea'], groups=df_sub['FullBath'], alpha=0.05)
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

| group1 | group2 | meandiff   | lower       | upper      | reject |
|--------|--------|------------|-------------|------------|--------|
| 0      | 1      | -5695.6583 | -14221.4451 | 2830.1286  | False  |
|        | 2      | -3587.0109 | -12103.848  | 4929.8263  | False  |
|        | 3      | 2477.0404  | -7075.4392  | 12029.52   | False  |
| 1      | 2      | 2108.6474  | 754.7975    | 3462.4974  | True   |
|        | 3      | 8172.6987  | 3639.9003   | 12705.4971 | True   |
| 2      | 3      | 6064.0513  | 1548.1091   | 10579.9935 | True   |

chấp H0

Bíc bao H0

- Các kết quả trên từ Tukey HSD cho thấy 0-1, 0-2, 0-3: chấp nhận H0, các so sánh cặp khác về số phòng bắc bắc H0 và chỉ ra sự khác biệt đáng kể về mặt thống kê.

## 4. Xử lý dữ liệu thiếu

```
In [56]: # Kiểm tra dữ liệu thiếu  
df_now = df[['LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd']]  
df_now.isnull().sum()
```

```
Out[56]: LotArea      0  
YearBuilt     0  
1stFlrSF      0  
2ndFlrSF      0  
FullBath      0  
BedroomAbvGr   0  
TotRmsAbvGrd  0  
dtype: int64
```

```
In [57]: # Không có dữ liệu thiếu
```

## 5. Phát hiện và xử lý ngoại lệ

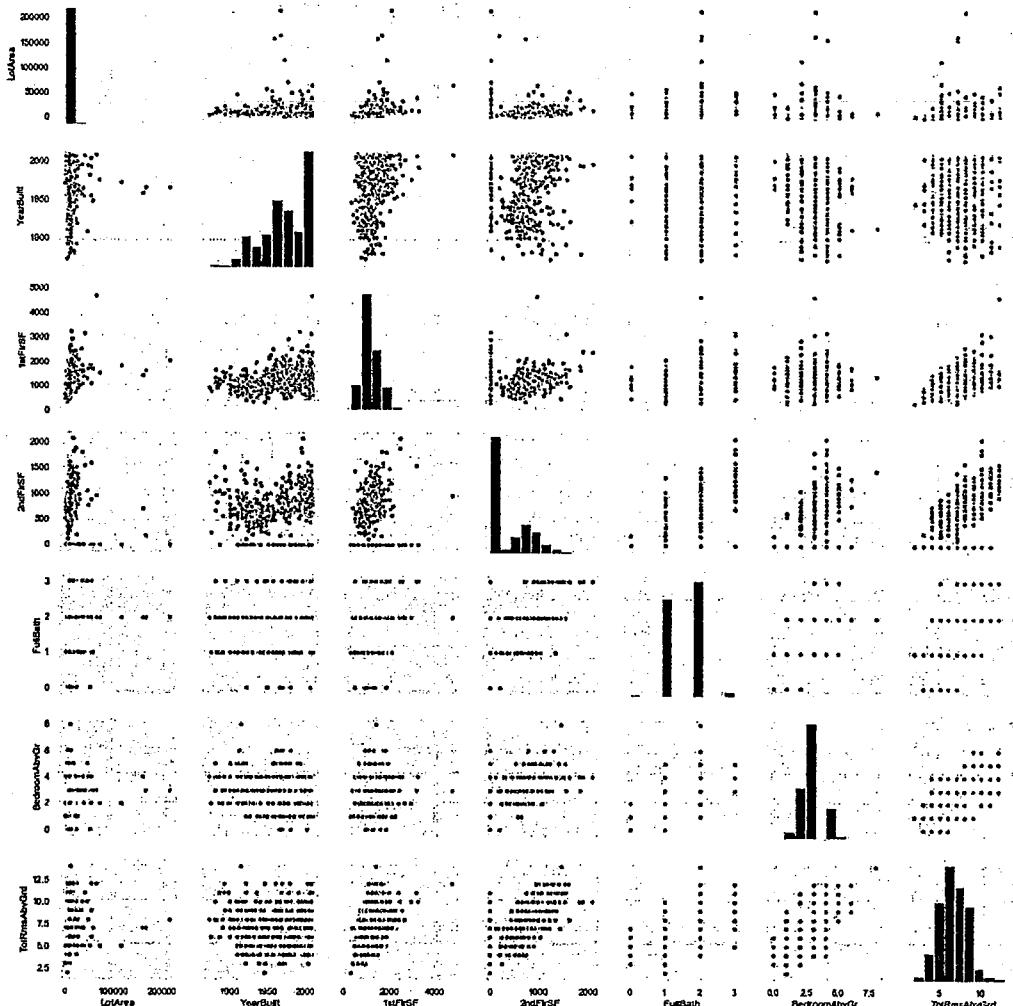
7/24/2019

Ex1\_house\_pricing-print



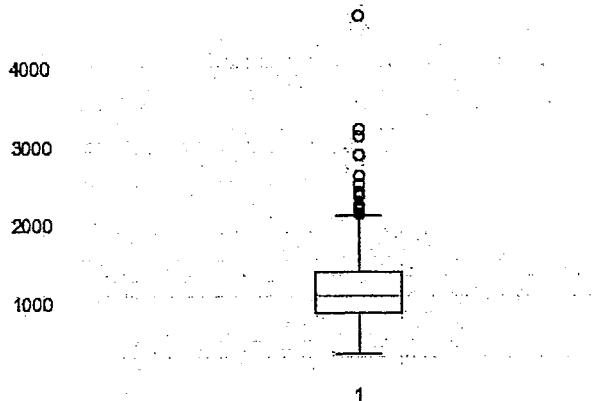
In [58]: `sns.pairplot(df[features])`

Out[58]: <seaborn.axisgrid.PairGrid at 0x1906aa54c50>





```
In [59]: plt.boxplot(df['1stFlrSF'])
plt.show()
```



```
In [60]: Q11 = np.percentile(df['1stFlrSF'], 25)
Q11
```

```
Out[60]: 882.0
```

```
In [61]: Q31 = np.percentile(df['1stFlrSF'], 75)
Q31
```

```
Out[61]: 1391.25
```

```
In [62]: IQR1 = scipy.stats.iqr(df['1stFlrSF'])
IQR1
```

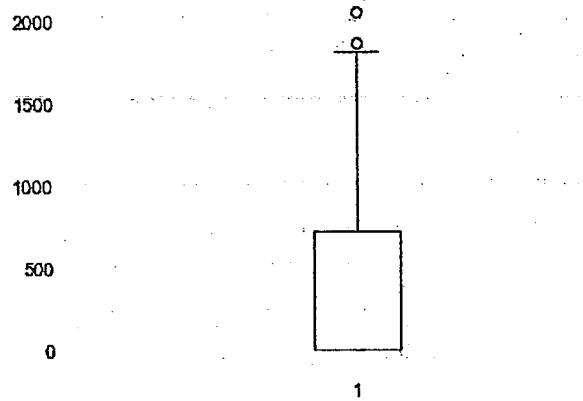
```
Out[62]: 509.25
```

```
In [63]: # Number of upper outliers
n_0_upper1 = df[df['1stFlrSF'] > (Q31 + 1.5*IQR1)].shape[0]
n_0_upper1
```

Out[63]: 20 → Vô khép bùn v) Vui vui 20 mìn / 1580 mìn -



```
In [64]: plt.boxplot(df['2ndFlrSF'])
plt.show()
```



```
In [65]: Q12 = np.percentile(df['2ndFlrSF'], 25)
Q12
```

Out[65]: 0.0

```
In [66]: Q32 = np.percentile(df['2ndFlrSF'], 75)
Q32
```

Out[66]: 728.0

```
In [67]: IQR2 = scipy.stats.iqr(df['2ndFlrSF'])
IQR2
```

Out[67]: 728.0

```
In [68]: # Number of upper outliers
n_0_upper2 = df[df['2ndFlrSF'] > (Q32 + 1.5*IQR2)].shape[0]
n_0_upper2
```

Out[68]: 2 → có 2 khé bát

```
In [69]: # Percentage of outliers
outliers_per = (n_0_lower + n_0_upper + n_0_upper1 + n_0_upper2 )/df.shape[0]
outliers_per
```

Out[69]: 0.06232876712328767

```
In [70]: # Có thể drop outliers của '2ndFlrSF', '1stFlrSF', 'LotArea': vì tổng số outliers
# Cũng có thể không cần drop thay vào đó khi áp dụng Machine Learning thì dùng thư
```

→ Tùy / tùy  
lâm các bài  
vấn đề.

```
In [71]: df_now = df_now[(df_now['2ndFlrSF'] <= (Q32 + 1.5*IQR2))] Xem lại
```

```
In [72]: df_now = df_now[(df_now['1stFlrSF'] <= (Q31 + 1.5*IQR1))] Xem lại
```

7/24/2019

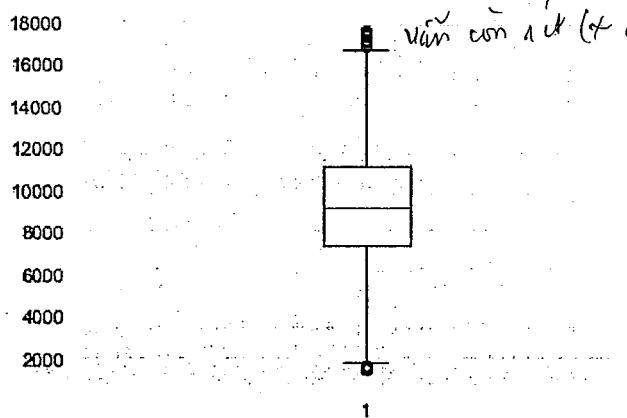
Ex1\_house\_pricing-print



```
In [73]: df_now = df_now[(df_now.LotArea >= (Q1 - 1.5*LotArea_iqr)) & (df_now.LotArea <= (Q3 + 1.5*LotArea_iqr))]
```

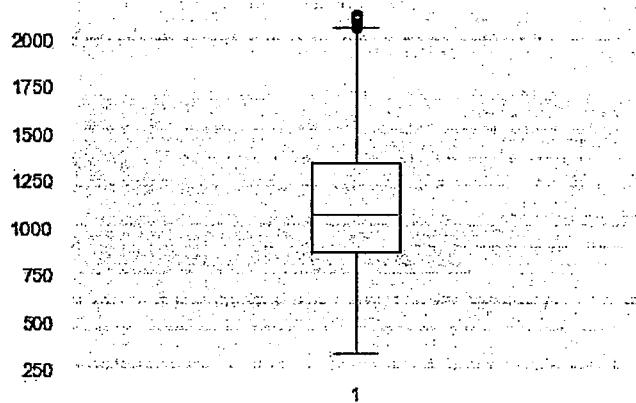
```
In [89]: plt.boxplot(df_now['LotArea'])
plt.show()
```

KQ Gau  
nhép kinh nghiem

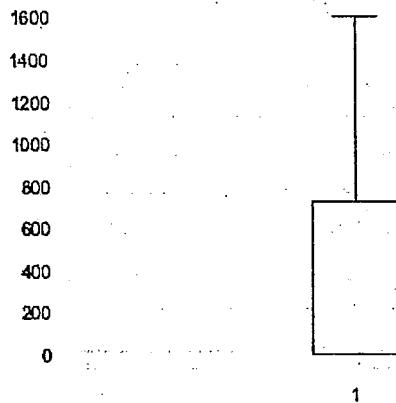


vẫn còn 1 số (tại quan max → trung)

```
In [75]: plt.boxplot(df_now['1stFlrSF'])
plt.show()
```



```
In [76]: plt.boxplot(df_now['2ndFlrSF'])
plt.show()
```



In [77]: # Xem xét thêm biên ngoài: major  
<https://m.wikihow.com/Calculate-Outliers> → Tham khảo thêm để xác định nên dùng biên  
# Công thức:  $O_u = Q3 + 3*IQR$ ,  $O_L = Q1 - 3*IQR$  Ngoci hay biên trong  
# hoặc tính trung bình trước và sau khi loại bỏ outlier + biên ngoài 3\*IQR → các giá trị bất thường, max, min, max → họ lý

```
In [78]: df_now.LotArea.mean()
```

Out[78]: 9228.756168359942

```
In [79]: df.LotArea.mean()
```

Out[79]: 10516.828082191782

In [80]: # nên loại bỏ ngoại lệ vì chênh lệch lớn

```
In [81]: df_now['1stFlrSF'].mean()
```

Out[81]: 1132.8526850507983

```
In [82]: df['1stFlrSF'].mean()
```

Out[82]: 1162.626712328767

In [83]: # không nhất thiết phải loại bỏ ngoại lệ vì chênh lệch nhỏ

```
In [84]: df_now['2ndFlrSF'].mean()
```

Out[84]: 340.67851959361394

```
In [85]: df['2ndFlrSF'].mean()
```

Out[85]: 346.99246575342465



In [86]: # không nhất thiết phải loại bỏ ngoại lệ vì chênh lệch nhỏ

In [87]: # vì 3 phân phối này đều không là phân phối chuẩn => không drop theo z-score  
ma nên dùng IQR

In [88]: # hoặc có thể xem xét chỉnh dữ liệu Log(cột)



## DATA PRE-PROCESSING AND ANALYSIS

### Bài 3: Data Pre-processing – Cleaning data

Phòng LT & Mạng

ĐỀ TẬP: Dữ liệu ô nhiễm và cách Data Pre-processing để  
đoán kết quả

2019



### Giới thiệu

- Làm sạch dữ liệu (Data cleansing/cleaning) là quá trình phát hiện và sửa chữa (hoặc loại bỏ) các mảnh bị hỏng hoặc không chính xác từ bộ dữ liệu, bảng hoặc cơ sở dữ liệu và đề cập đến việc xác định các phần không đầy đủ, không chính xác hoặc không liên quan của dữ liệu, sau đó thay thế, sửa đổi, hoặc xóa dữ liệu không phù hợp.
- Việc làm sạch dữ liệu có thể được thực hiện bằng các công cụ hay thông qua việc lập trình.
- Sau khi làm sạch, bộ dữ liệu phải phù hợp với các bộ dữ liệu tương tự khác trong hệ thống.



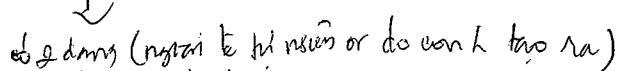
## Nội dung

1. Khám phá dữ liệu (Exploring data)
2. Thu gọn dữ liệu (Tidying data)
3. Kết hợp dữ liệu (Combining data)
4. Làm sạch dữ liệu (Cleaning data)



## Khám phá dữ liệu (Exploring data)

**Với bộ dữ liệu hoàn toàn mới và đang  
muốn bắt đầu khám phá, vậy chúng ta phải  
bắt đầu từ đâu và làm thế nào để có thể  
chắc chắn rằng bộ dữ liệu này sạch?  
Chúng ta sẽ tìm hiểu cách khám phá dữ  
liệu trực quan để chẩn đoán các vấn đề  
như ngoại lệ, giá trị bị thiếu và trùng lặp.**

  
outliers (outlier là từ tiếng Anh or do con lừa tạo ra)



## Khám phá dữ liệu (Exploring data)

### Chuẩn đoán dữ liệu

- Làm sạch dữ liệu là việc chuẩn bị dữ liệu cho việc phân tích
- Tuy nhiên, dữ liệu ban đầu hầu như không bao giờ sạch  
=> Cần chẩn đoán dữ liệu để tìm ra các vấn đề.



## Khám phá dữ liệu (Exploring data)

### Các vấn đề thường gặp

- Tên các cột không thống nhất
- Dữ liệu bị thiếu
- Ngoại lệ
- Dòng dữ liệu trùng lặp
- Không gọn gàng (Ví): *Emi đines nhưng bai tais ra 3 cat: Nam, Nữ, ≠)*
- Các loại cột có thể chứa các giá trị dữ liệu không mong muốn





## Khám phá dữ liệu (Exploring data)

- Ví dụ các vấn đề thường gặp

- Dữ liệu NaN
- Tên cột bị thừa ký tự khoảng trắng ở đầu hay cuối
- Cột số chứa chuỗi và ngược lại
- ...



## Khám phá dữ liệu (Exploring data)

- Khám phá dữ liệu

- Tần số đếm: Đếm số lượng giá trị duy nhất trong dữ liệu
- Kiểu dữ liệu của các cột
- Xem thống kê tóm tắt
  - Các cột dữ liệu số
  - Ngoại lệ (outlier): cao hơn hay thấp hơn đáng kể, yêu cầu kiểm tra thêm



## Khám phá dữ liệu (Exploring data)

### • Ví dụ:

Cho dữ liệu dob\_job\_application\_filings\_subset.csv

```
# Print the value counts for 'Site Fill'  
print(df['Site Fill'].value_counts(dropna=False))
```

|                     |             |
|---------------------|-------------|
| NOT APPLICABLE      | 7806        |
| <u>Nan</u>          | <u>4205</u> |
| ON-SITE             | 519         |
| OFF-SITE            | 186         |
| USE UNDER 300 CU.YD | 130         |

Name: Site Fill, dtype: int64

```
# Print the value counts for 'Borough'  
print(df['Borough'].value_counts(dropna=False))
```

|               |      |
|---------------|------|
| MANHATTAN     | 6310 |
| BROOKLYN      | 2866 |
| QUEENS        | 2121 |
| BRONX         | 974  |
| STATEN ISLAND | 575  |

Name: Borough, dtype: int64

Nan

Đếm bao nhiêu dòng chứa 'NaN' = True

9

## Khám phá dữ liệu (Exploring data)

### ☐ Trực quan dữ liệu khám phá

- Cách tuyệt vời để phát hiện các ngoại lệ và lỗi một cách rõ ràng
- Không chỉ để tìm kiếm các pattern
- Lập kế hoạch các bước làm sạch dữ liệu



## Khám phá dữ liệu (Exploring data)

### • Thường sử dụng:

- Barplot biểu diễn số lượng dữ liệu rời rạc
- Histogram biểu diễn số lượng dữ liệu liên tục, xem tần số
- Boxplot: biểu diễn min, max, 1Q, median, 3Q, outliers
- Scatter plot: thể hiện mối quan hệ giữa 2 biến số, đánh dấu dữ liệu lỗi khi không thể thấy bằng quan sát 1 biến



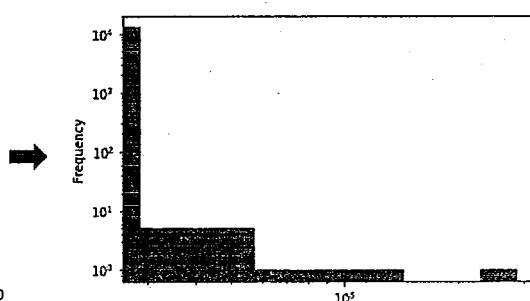
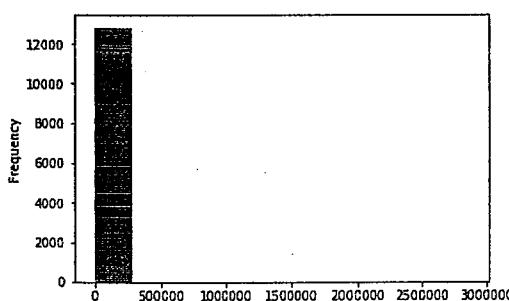
## Khám phá dữ liệu (Exploring data)

### • Ví dụ:

```
# Plot the histogram  
df['Existing Zoning Sqft'].plot.hist()  
# Display the histogram  
plt.show()
```

đổi lại thang đo log(x), log(y) để thể hiện các giá trị  
có khe konkav tăng

```
# Plot the histogram  
df['Existing Zoning Sqft'].plot.hist(logx=True, logy=True)  
# Display the histogram  
plt.show()
```



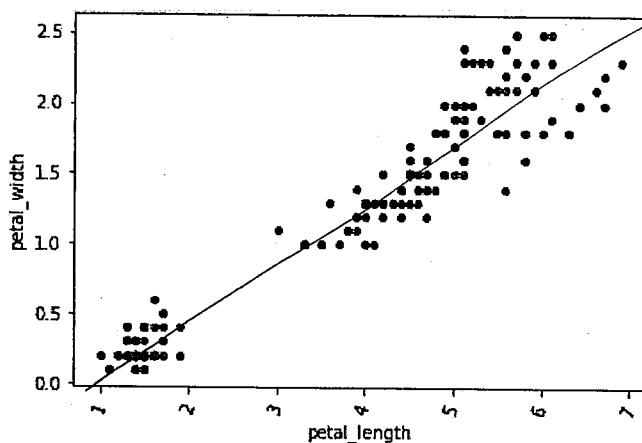
Có sự khác biệt rất lớn giữa các giá trị tối thiểu và tối đa, và biểu đồ cần phải được điều chỉnh cho phù hợp. Trong những trường hợp như vậy, nên xem xét biểu đồ trên thang đo log: tham số logx = True hoặc logy = True có thể được truyền vào .plot() tùy thuộc vào trực ta muốn rescale lại.



## Khám phá dữ liệu (Exploring data)

### • Ví dụ:

```
# Create and display the first scatter plot  
iris.plot.scatter(x='petal_length', y='petal_width', rot=70)  
plt.show()
```



VD: Chiều cao cân nặng → thể hình  
chi số BMI → chọn racai  
để bắt they.

Data Pre-processing and Analysis

13

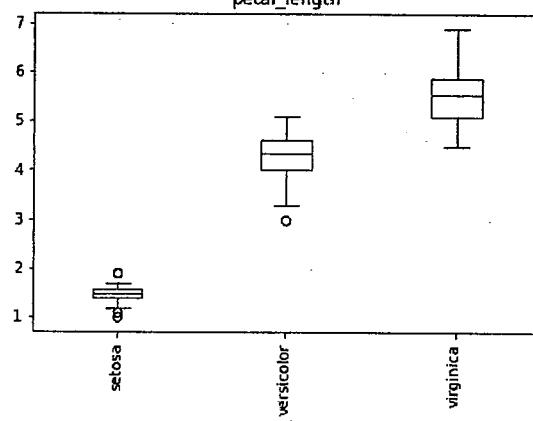
## Khám phá dữ liệu (Exploring data)

### • Ví dụ:

```
# Create the boxplot  
plt.figure(figsize=(6, 8))  
iris.boxplot(column='petal_length', by='species', rot=90)  
plt.show()
```

<Figure size 432x576 with 0 Axes>

Boxplot grouped by species  
petal\_length



outlier:

- phân bố chuẩn → Z-score
- xét đến p<sup>2</sup>:
  - ± 1.5 IQR
  - ± 3 IQR



14

## Khám phá dữ liệu (Exploring data)

### • Xác định error

- » Không phải tất cả các outlier đều là điểm dữ liệu xấu
- » Một số có thể là lỗi, nhưng một số khác có thể là các giá trị hợp lệ



## Nội dung

1. Khám phá dữ liệu (Exploring data)
2. Thu gọn dữ liệu (Tidying data)
3. Kết hợp dữ liệu (Combining data)
4. Làm sạch dữ liệu (Cleaning data)



## Thu gọn dữ liệu (Tidying data)

**Mục tiêu ở đây không phải là phân tích các bộ dữ liệu mà là chuẩn bị chúng theo cách chuẩn hóa trước khi phân tích.**



## Thu gọn dữ liệu (Tidying data)

**Một số loại dữ liệu lộn xộn mà chúng ta cần giải quyết:**

- Tiêu đề cột là giá trị, không phải tên biến. (vd: giá trị mã lai kieu  
SEX; Nam, Nữ #).
- Nhiều biến được lưu trữ trong một cột.
- Các biến được lưu trữ trong cả hàng và cột.
- Nhiều đơn vị mẫu được lưu trữ trong cùng một bảng
- Một mẫu quan sát duy nhất được lưu trữ trong nhiều table/file.



## Thu gọn dữ liệu (Tidying data)

- **Dữ liệu gọn gàng (tiny data)** là dữ liệu thu được do kết quả của một quá trình gọi thu gọn dữ liệu (data tidying). Đây là một trong những quy trình làm sạch quan trọng trong quá trình xử lý dữ liệu lớn và là bước được công nhận trong thực tiễn khoa học dữ liệu. **Bộ dữ liệu gọn gàng** có cấu trúc và làm việc với chúng rất dễ dàng: dễ dàng thao tác, mô hình hóa và trực quan. **Các tập dữ liệu gọn gàng** được sắp xếp sao cho mỗi biến là một cột và mỗi quan sát (hoặc trường hợp) là một hàng.



Theo: [https://en.wikipedia.org/wiki/Tidy\\_data](https://en.wikipedia.org/wiki/Tidy_data)

Data Pre-processing and Analysis

19

## Thu gọn dữ liệu (Tidying data)

### □ Đặc điểm của tiny data

- Mỗi biến đo lường phải ở trong một cột.
- Mỗi mẫu khác nhau của biến đó nên ở một hàng khác nhau.
- Cần có một bảng cho mỗi "loại" biến.
- Nếu có nhiều bảng, chúng nên có một cột trong bảng cho phép chúng liên kết.



Theo “Jeff Leek - The Elements of Data Analytic Style”: [https://en.wikipedia.org/wiki/The\\_Elements\\_of\\_Data\\_Analytic\\_Style](https://en.wikipedia.org/wiki/The_Elements_of_Data_Analytic_Style)

Data Pre-processing and Analysis

20

## Thu gọn dữ liệu (Tidying data)

### • Ví dụ:

- Dữ liệu này không phải là “tiny data” vì tên cột chứa thông tin về phép đo được thực hiện: số lượng xi lanh (cyl), công suất ngựa (hp) và số thùng bộ chế hòa khí (carb).

| maker    | cyl | hp  | carb |
|----------|-----|-----|------|
| Delorian | 4   | 160 | 4    |
| Fantom   | 2   | 80  | 2    |

Chuyển thành  
“tiny data”

| maker    | metric | value |
|----------|--------|-------|
| Delorian | cyl    | 4     |
| Fantom   | cyl    | 2     |
| Delorian | hp     | 160   |
| Fantom   | hp     | 80    |
| Delorian | carb   | 4     |
| Fantom   | carb   | 2     |



## Thu gọn dữ liệu (Tidying data)

### □ Tiny data

- Chính thức hóa cách chúng ta mô tả hình dạng của dữ liệu
- Cung cấp cho chúng ta một mục tiêu khi định dạng dữ liệu
- “Cách tiêu chuẩn” để tổ chức các giá trị dữ liệu trong một tập dữ liệu
- Tiny data giúp dễ dàng sửa chữa các vấn đề dữ liệu phổ biến



## Thu gọn dữ liệu (Tidying data)

### □ Chuyển dữ liệu thành Tiny data

- Vấn đề về dữ liệu cần khắc phục:

- Cột chứa giá trị thay vì chứa biến

- Giải pháp:

- Dùng pd.melt()



## Thu gọn dữ liệu (Tidying data)

df

|   | marker   | cyl | hp  | carb |
|---|----------|-----|-----|------|
| 0 | Delorian | 4   | 160 | 4    |
| 1 | Fantom   | 2   | 80  | 2    |

df\_new = pd.melt(frame=df, id\_vars="marker", value\_vars=["cyl", "hp", "carb"], var\_name="metric")

chỗ đổi tên để hàng id- vars

Lý do: nếu id\_vars = ["marker", ...] .

|   | marker   | metric | value |
|---|----------|--------|-------|
| 0 | Delorian | cyl    | 4     |
| 1 | Fantom   | cyl    | 2     |
| 2 | Delorian | hp     | 160   |
| 3 | Fantom   | hp     | 80    |
| 4 | Delorian | carb   | 4     |
| 5 | Fantom   | carb   | 2     |

## Thu gọn dữ liệu (Tidying data)

### ☐ Pivoting data (un-melting data)

#### • Ngược lại với melting data

- Trong melting data, chúng ta chuyển các cột thành các dòng
- Trong pivoting data: chuyển các giá trị duy nhất thành các cột riêng biệt
- Dùng để tạo các báo cáo
- Vì phạm nguyên tắc của tidy data: các dòng chứa các mẫu
  - Nhiều biến được lưu trữ trong cùng một cột

Data Pre-processing and Analysis

25

## Thu gọn dữ liệu (Tidying data)

df\_new

|   | marker   | metric | value |
|---|----------|--------|-------|
| 0 | Delorian | cyl    | 4     |
| 1 | Fantom   | cyl    | 2     |
| 2 | Delorian | hp     | 160   |
| 3 | Fantom   | hp     | 80    |
| 4 | Delorian | carb   | 4     |
| 5 | Fantom   | carb   | 2     |

df\_pivot = df\_new.pivot(index = 'marker', columns='metric', values='value')

|          | metric | carb | cyl | hp |
|----------|--------|------|-----|----|
| marker   |        |      |     |    |
| Delorian | 4      | 4    | 160 |    |
| Fantom   | 2      | 2    | 80  |    |

Thay đổi kết quả.

Data Pre-processing and Analysis

26



## Thu gọn dữ liệu (Tidying data)

```
df_new = df_new.append({'marker': 'Fantom', 'metric': 'carb', 'value': 4}, ignore_index=True)
df_new
```

|   | marker   | metric | value |
|---|----------|--------|-------|
| 0 | Delorian | cyl    | 4     |
| 1 | Fantom   | cyl    | 2     |
| 2 | Delorian | hp     | 160   |
| 3 | Fantom   | hp     | 80    |
| 4 | Delorian | carb   | 4     |
| 5 | Fantom   | carb   | 2     |
| 6 | Fantom   | carb   | 4     |

đtrung  $\rightarrow$  jai data tham so vao cuoi day : mean, max ...

```
df_pivot_2 = df_new.pivot(index = 'marker', columns='metric', values='value')
df_pivot_2
```

ValueError: Index contains duplicate entries, canrot reshape



## Thu gọn dữ liệu (Tidying data)

### ☐ Pivot table

- Có thêm một tham số chỉ định cách xử lý trùng lặp giá trị
  - Ví dụ: Có thể tổng hợp các giá trị trùng lặp bằng cách lấy Trung bình cộng



## Thu gọn dữ liệu (Tidying data)

df\_new

|   | marker   | metric | value |
|---|----------|--------|-------|
| 0 | Delorian | cyl    | 4     |
| 1 | Fantom   | cyl    | 2     |
| 2 | Delorian | hp     | 160   |
| 3 | Fantom   | hp     | 80    |
| 4 | Delorian | carb   | 4     |
| 5 | Fantom   | carb   | 2     |
| 6 | Fantom   | carb   | 4     |

```
# Pivot table  
df_pivot_3 = df_new.pivot_table(index = 'marker', columns='metric',  
values='value', aggfunc= np.mean) → hùng (lấy giá trung)
```

|          | metric | carb | cyl | hp  |
|----------|--------|------|-----|-----|
| marker   |        |      |     |     |
| Delorian | 4      | 4    | 4   | 160 |
| Fantom   | 3      | 2    | 2   | 80  |

Data Pre-processing and Analysis

29

## Nội dung

1. Khám phá dữ liệu (Exploring data)
2. Thu gọn dữ liệu (Tidying data)
3. Kết hợp dữ liệu (Combining data)
4. Làm sạch dữ liệu (Cleaning data)



Data Pre-processing and Analysis

30

## Kết hợp dữ liệu (Combining data)

### ☐ Giới thiệu

- Dữ liệu có thể không phải lúc nào cũng được lưu trong 1 tệp lớn
  - Ví dụ: 5 triệu dòng dữ liệu có thể được chia thành 5 bộ dữ liệu riêng biệt
  - Dễ dàng lưu trữ và chia sẻ
  - Có thể có dữ liệu mới mỗi ngày
  - Có thể kết hợp sau đó làm sạch, hoặc ngược lại



## Kết hợp dữ liệu (Combining data)

### ☐ Nối dữ liệu: pd.concat([df1, df2,...],

mặc định = False ignore\_index=True) → Nếu định là "False" → giữ lại index ban đầu  
có nghĩa là giữ lại index tăng dần, và đổi lại m index

- Sử dụng để nối các DataFrame cùng cấu trúc, bỏ qua index đang có, tạo bộ index mới



## Kết hợp dữ liệu (Combining data)

```
iris_setosa.info()           iris_versicolor.info()          iris_virginica.info()
<class 'pandas.core.frame.DataFrame'>    <class 'pandas.core.frame.DataFrame'>    <class 'pandas.core.frame.DataFrame'>
Int64Index: 50 entries, 0 to 49          Int64Index: 50 entries, 50 to 99          Int64Index: 50 entries, 100 to 149
Data columns (total 5 columns):          Data columns (total 5 columns):          Data columns (total 5 columns):
sepal_length    50 non-null float64      sepal_length    50 non-null float64      sepal_length    50 non-null float64
sepal_width     50 non-null float64      sepal_width     50 non-null float64      sepal_width     50 non-null float64
petal_length    50 non-null float64      petal_length    50 non-null float64      petal_length    50 non-null float64
petal_width     50 non-null float64      petal_width     50 non-null float64      petal_width     50 non-null float64
species        50 non-null object       species        50 non-null object       species        50 non-null object
dtypes: float64(4), object(1)          dtypes: float64(4), object(1)          dtypes: float64(4), object(1)
memory usage: 2.3+ KB                  memory usage: 2.3+ KB                  memory usage: 2.3+ KB

df = pd.concat([iris_setosa, iris_versicolor, iris_virginica])

df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal_length    150 non-null float64
sepal_width     150 non-null float64
petal_length    150 non-null float64
petal_width     150 non-null float64
species        150 non-null object
dtypes: float64(4), object(1)
memory usage: 7.0+ KB
```

33

## Kết hợp dữ liệu (Combining data)

Phím tắt

### ☐ Nối nhiều tập tin.

- Tận dụng các tính năng của Python để làm sạch dữ liệu với Pandas:
  - Đễ nối các DataFrames
  - Chúng phải ở trong một danh sách
  - Có thể tải riêng lẻ nếu chỉ có một vài bộ dữ liệu
- Nhưng nếu có hàng ngàn bộ dữ liệu thì sao?
  - Giải pháp: sử dụng glob function để tìm tập tin dựa trên một pattern



## Kết hợp dữ liệu (Combining data)



### □ Pattern khớp với file names

- Ký tự đại diện: \* ? \* : Ký tự đại diện giáng đế.
- Mọi file csv: \*.csv
- Ký tự đơn: file\_?.csv

### □ Trả về: danh sách file names

### □ Có thể sử dụng danh sách để tải vào các DataFrame riêng



## Kết hợp dữ liệu (Combining data)



### □ Thực hiện

- Tải tập tin từ glob vào pandas
- Thêm DataFrame vào danh sách
- Ghép nhiều bộ dữ liệu cùng một lúc



## Kết hợp dữ liệu (Combining data)

### • Ví dụ:

```
# Find and concatenate
import glob

csv_files = glob.glob('iris/*.csv')

print(csv_files)
['iris\\setosa.csv', 'iris\\versicolor.csv', 'iris\\virginica.csv']
```



## Kết hợp dữ liệu (Combining data)

```
list_data = []
for filename in csv_files:
    data = pd.read_csv(filename, index_col=0)
    list_data.append(data)
iris_now = pd.concat(list_data, ignore_index=True)
```

Chu vien sd toi da s vong lặp "for"

```
iris_now.shape
```

```
(150, 5)
```

```
iris_now.head()
```

```
iris_now.tail()
```

|   | sepal_length | sepal_width | petal_length | petal_width | species |     | sepal_length | sepal_width | petal_length | petal_width | species   |
|---|--------------|-------------|--------------|-------------|---------|-----|--------------|-------------|--------------|-------------|-----------|
| 0 | 5.1          | 3.5         | 1.4          | 0.2         | setosa  | 145 | 6.7          | 3.0         | 5.2          | 2.3         | virginica |
| 1 | 4.9          | 3.0         | 1.4          | 0.2         | setosa  | 146 | 6.3          | 2.5         | 5.0          | 1.9         | virginica |
| 2 | 4.7          | 3.2         | 1.3          | 0.2         | setosa  | 147 | 6.5          | 3.0         | 5.2          | 2.0         | virginica |
| 3 | 4.6          | 3.1         | 1.5          | 0.2         | setosa  | 148 | 6.2          | 3.4         | 5.4          | 2.3         | virginica |
| 4 | 5.0          | 3.6         | 1.4          | 0.2         | setosa  | 149 | 5.9          | 3.0         | 5.1          | 1.8         | virginica |



## Kết hợp dữ liệu (Combining data)

### ☐ Trộn dữ liệu pd.merge()

- Tương tự như phép join các bảng trong CSDL
- Kết hợp các bộ dữ liệu khác nhau dựa trên các cột chung
- Tham khảo:

[https://www.tutorialspoint.com/python\\_pandas/as/python\\_pandas\\_merging\\_joining.htm](https://www.tutorialspoint.com/python_pandas/as/python_pandas_merging_joining.htm)



## Kết hợp dữ liệu (Combining data)

- Ví dụ:

students\_info

|   | name | age |
|---|------|-----|
| 0 | John | 17  |
| 1 | Lucy | 17  |
| 2 | Lily | 18  |

students\_address

|   | name | address     |
|---|------|-------------|
| 0 | John | 123 rue N10 |
| 1 | Lucy | 345 rue N6  |
| 2 | Lily | 234 rue N12 |

students = pd.merge(left=students\_info, right=students\_address)  
students

|   | name | age | address     |
|---|------|-----|-------------|
| 0 | John | 17  | 123 rue N10 |
| 1 | Lucy | 17  | 345 rue N6  |
| 2 | Lily | 18  | 234 rue N12 |



## Kết hợp dữ liệu (Combining data).

### • Ví dụ: one-to-one data merge

site

|   | name  | lat    | long    |
|---|-------|--------|---------|
| 0 | DR-1  | -49.85 | -128.57 |
| 1 | DR-3  | -47.15 | -126.72 |
| 2 | MSK-4 | -48.87 | -123.40 |

visited

|   | ident | site  | dated      |
|---|-------|-------|------------|
| 0 | 619   | DR-1  | 1927-02-08 |
| 1 | 734   | DR-3  | 1939-01-07 |
| 2 | 837   | MSK-4 | 1932-01-14 |

# Merge the DataFrames: o2o

```
o2o = pd.merge(left=site, right=visited, left_on='name', right_on='site')
```

o2o

|   | name  | lat    | long    | ident | site  | dated      |
|---|-------|--------|---------|-------|-------|------------|
| 0 | DR-1  | -49.85 | -128.57 | 619   | DR-1  | 1927-02-08 |
| 1 | DR-3  | -47.15 | -126.72 | 734   | DR-3  | 1939-01-07 |
| 2 | MSK-4 | -48.87 | -123.40 | 837   | MSK-4 | 1932-01-14 |



## Kết hợp dữ liệu (Combining data)

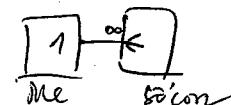
### • Ví dụ: many-to-1 data merge

site

|   | name  | lat    | long    |
|---|-------|--------|---------|
| 0 | DR-1  | -49.85 | -128.57 |
| 1 | DR-3  | -47.15 | -126.72 |
| 2 | MSK-4 | -48.87 | -123.40 |

visited

|   | ident | site  | dated      |
|---|-------|-------|------------|
| 0 | 619   | DR-1  | 1927-02-08 |
| 1 | 622   | DR-1  | 1927-02-10 |
| 2 | 734   | DR-3  | 1939-01-07 |
| 3 | 735   | DR-3  | 1939-01-12 |
| 4 | 751   | DR-3  | 1930-02-26 |
| 5 | 752   | DR-3  | NaN        |
| 6 | 837   | MSK-4 | 1932-01-14 |
| 7 | 844   | DR-1  | 1932-03-22 |



# Merge the DataFrames: m2o

```
m2o = pd.merge(left=site, right=visited, left_on='name', right_on='site')
```

m2o

|   | name  | lat    | long    | ident | site  | dated      |
|---|-------|--------|---------|-------|-------|------------|
| 0 | DR-1  | -49.85 | -128.57 | 619   | DR-1  | 1927-02-08 |
| 1 | DR-1  | -49.85 | -128.57 | 622   | DR-1  | 1927-02-10 |
| 2 | DR-1  | -49.85 | -128.57 | 844   | DR-1  | 1932-03-22 |
| 3 | DR-3  | -47.15 | -126.72 | 734   | DR-3  | 1939-01-07 |
| 4 | DR-3  | -47.15 | -126.72 | 735   | DR-3  | 1930-01-12 |
| 5 | DR-3  | -47.15 | -126.72 | 751   | DR-3  | 1930-02-26 |
| 6 | DR-3  | -47.15 | -126.72 | 752   | DR-3  | NaN        |
| 7 | MSK-4 | -48.87 | -123.40 | 837   | MSK-4 | 1932-01-14 |



## Kết hợp dữ liệu (Combining data)

• Ví dụ: many-to-many data merge

site

|   | name  | lat    | long    |
|---|-------|--------|---------|
| 0 | DR-1  | 49.85  | 128.57  |
| 1 | DR-3  | -47.15 | -126.72 |
| 2 | MSK-4 | -48.87 | -123.40 |

visited

|   | ident | site  | dated      |
|---|-------|-------|------------|
| 0 | 619   | DR-1  | 1927-02-08 |
| 1 | 622   | DR-1  | 1927-02-10 |
| 2 | 734   | DR-3  | 1939-01-07 |
| 3 | 735   | DR-3  | 1930-01-12 |
| 4 | 751   | DR-3  | 1930-02-26 |
| 5 | 752   | DR-3  | Nan        |
| 6 | 837   | MSK-4 | 1932-01-14 |
| 7 | 844   | DR-1  | 1932-03-22 |

# Merge site and visited: m2m

```
m2m = pd.merge(left=site, right=visited, left_on='name', right_on='site')
print(m2m)
```

m2m

|   | name  | lat    | long    | ident | site  | dated      |
|---|-------|--------|---------|-------|-------|------------|
| 0 | DR-1  | -49.85 | -128.57 | 619   | DR-1  | 1927-02-08 |
| 1 | DR-1  | -49.85 | -128.57 | 622   | DR-1  | 1927-02-10 |
| 2 | DR-1  | -49.85 | -128.57 | 844   | DR-1  | 1932-03-22 |
| 3 | DR-3  | -47.15 | -126.72 | 734   | DR-3  | 1939-01-07 |
| 4 | DR-3  | -47.15 | -126.72 | 735   | DR-3  | 1930-01-12 |
| 5 | DR-3  | -47.15 | -126.72 | 751   | DR-3  | 1930-02-26 |
| 6 | DR-3  | -47.15 | -126.72 | 752   | DR-3  | Nan        |
| 7 | MSK-4 | -48.87 | -123.40 | 837   | MSK-4 | 1932-01-14 |

## Kết hợp dữ liệu (Combining data)

m2m

|   | name  | lat    | long    | ident | site  | dated      |
|---|-------|--------|---------|-------|-------|------------|
| 0 | DR-1  | -49.85 | -128.57 | 619   | DR-1  | 1927-02-08 |
| 1 | DR-1  | -49.85 | -128.57 | 622   | DR-1  | 1927-02-10 |
| 2 | DR-1  | -49.85 | -128.57 | 844   | DR-1  | 1932-03-22 |
| 3 | DR-3  | -47.15 | -126.72 | 734   | DR-3  | 1939-01-07 |
| 4 | DR-3  | -47.15 | -126.72 | 735   | DR-3  | 1930-01-12 |
| 5 | DR-3  | -47.15 | -126.72 | 751   | DR-3  | 1930-02-26 |
| 6 | DR-3  | -47.15 | -126.72 | 752   | DR-3  | Nan        |
| 7 | MSK-4 | -48.87 | -123.40 | 837   | MSK-4 | 1932-01-14 |

survey.head(10)

|   | taker | person | quart | reading |
|---|-------|--------|-------|---------|
| 0 | 619   | dyer   | rad   | 9.82    |
| 1 | 619   | dyer   | sal   | 0.13    |
| 2 | 622   | dyer   | rad   | 7.88    |
| 3 | 622   | dyer   | sal   | 0.09    |
| 4 | 734   | pb     | rad   | 8.41    |
| 5 | 734   | lake   | sal   | 0.05    |
| 6 | 734   | pb     | temp  | -21.50  |
| 7 | 735   | pb     | rad   | 7.22    |
| 8 | 735   | Nan    | sal   | 0.06    |
| 9 | 735   | Nan    | temp  | -26.00  |

43

44

## Kết hợp dữ liệu (Combining data)

### • Ví dụ: many-to-many data merge

```
# Merge m2m and survey: m2m
m2m = pd.merge(left=m2m, right=survey, left_on='ident', right_on='taken')
# Print the first 10 lines of m2m
print(m2m.head(10))
```

|   | name | lat    | long    | ident | site | dated      | taken | person | quant | reading |
|---|------|--------|---------|-------|------|------------|-------|--------|-------|---------|
| 0 | DR-1 | -49.85 | -128.57 | 619   | DR-1 | 1927-02-08 | 619   | dyer   | rad   | 9.82    |
| 1 | DR-1 | -49.85 | -128.57 | 619   | DR-1 | 1927-02-08 | 619   | dyer   | sal   | 0.13    |
| 2 | DR-1 | -49.85 | -128.57 | 622   | DR-1 | 1927-02-10 | 522   | dyer   | rad   | 7.80    |
| 3 | DR-1 | -49.85 | -128.57 | 622   | DR-1 | 1927-02-10 | 522   | dyer   | sal   | 0.89    |
| 4 | DR-1 | -49.85 | -128.57 | 844   | DR-1 | 1932-03-22 | 844   | roe    | rad   | 11.25   |
| 5 | DR-3 | -47.15 | -126.72 | 734   | DR-3 | 1939-01-07 | 734   | pb     | rad   | 8.41    |
| 6 | DR-3 | -47.15 | -126.72 | 734   | DR-3 | 1939-01-07 | 734   | lake   | sal   | 0.65    |
| 7 | DR-3 | -47.15 | -126.72 | 734   | DR-3 | 1939-01-07 | 734   | pb     | temp  | -21.50  |
| 8 | DR-3 | -47.15 | -126.72 | 735   | DR-3 | 1939-01-12 | 735   | pb     | rad   | 7.22    |
| 9 | DR-3 | -47.15 | -126.72 | 735   | DR-3 | 1939-01-12 | 735   | NaN    | sal   | 0.86    |



## Nội dung

1. Khám phá dữ liệu (Exploring data)
2. Thu gọn dữ liệu (Tidying data)
3. Kết hợp dữ liệu (Combining data)
4. Làm sạch dữ liệu (Cleaning data)



## Làm sạch dữ liệu (Cleaning data)

### ☐ Kiểu dữ liệu (Data type)

- Theo yêu cầu, có thể phải chuyển dữ liệu từ kiểu này sang kiểu khác

    - Cột số có thể chứa chuỗi và ngược lại

- Chuyển đổi kiểu dữ liệu cho cột: Dùng  
 $df['tên_cột'].astype(kiểu_dữ_liệu)$

- Hoặc chuyển dữ liệu thành số:

$pd.to_numeric(df['tên_cột'])$

*một kiểu số là ok*



## Làm sạch dữ liệu (Cleaning data)

### ● Ví dụ

```
initial_cost['Initial Cost'] = initial_cost['Initial Cost'].apply(lambda x:x[1:]).astype(float)
initial_cost.head()
```

initial\_cost.head()

|   | Initial Cost |
|---|--------------|
| 0 | \$75000.00   |
| 1 | \$0.00       |
| 2 | \$30000.00   |
| 3 | \$1500.00    |
| 4 | \$19500.00   |



|   | Initial Cost |
|---|--------------|
| 0 | 75000.0      |
| 1 | 0.0          |
| 2 | 30000.0      |
| 3 | 1500.0       |
| 4 | 19500.0      |

```
type((initial_cost['Initial Cost'].values)[0])
str
```

type((initial\_cost['Initial Cost'].values)[0])
numpy.float64



## Làm sạch dữ liệu (Cleaning data)

### □ Dữ liệu phân loại (categorical data)

- Chuyển categorical data thành 'category'

dtype:

- Có thể làm cho DataFrame giảm được kích thước trong memory
- Có thể làm cho chúng được sử dụng dễ dàng bởi các thư viện Python khác



## Làm sạch dữ liệu (Cleaning data)

### • Ví dụ

|   | name | age | address     | sex    |
|---|------|-----|-------------|--------|
| 0 | John | 17  | 123 rue N10 | male   |
| 1 | Lucy | 17  | 345 rue N6  | female |
| 2 | Lyly | 18  | 234 rue N12 | male   |

students.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 0 to 2
Data columns (total 4 columns):
 name    3 non-null object
 age     3 non-null int64
 address 3 non-null object
 sex     3 non-null object
dtypes: int64(1), object(3)
memory usage: 120.0+ bytes
```

# Convert the sex column to type 'category'  
students.sex = students.sex.astype('category')

|   | name | age | address     | sex    |
|---|------|-----|-------------|--------|
| 0 | John | 17  | 123 rue N10 | male   |
| 1 | Lucy | 17  | 345 rue N6  | female |
| 2 | Lyly | 18  | 234 rue N12 | male   |

students.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 0 to 2
Data columns (total 4 columns):
 name    3 non-null object
 age     3 non-null int64
 address 3 non-null object
 sex     3 non-null category
dtypes: category(1), int64(1), object(2)
memory usage: 195.0+ bytes
```



## Làm sạch dữ liệu (Cleaning data)

### ☐ Thao tác trên chuỗi

- Phần lớn việc làm sạch dữ liệu liên quan đến thao tác chuỗi
- Hầu hết dữ liệu trên thế giới là văn bản không có cấu trúc
- Cũng phải thực hiện thao tác chuỗi để làm cho các bộ dữ liệu phù hợp với nhau



## Làm sạch dữ liệu (Cleaning data)

- Có rất nhiều thư viện hỗ trợ built-in và thư viện bên ngoài
- Sử dụng thư viện `'re'` để áp dụng regular expressions
  - Cách chỉ định mẫu cho chuỗi
  - Chứa một chuỗi các ký tự
- So khớp mẫu
  - Tương tự như glob



## Làm sạch dữ liệu (Cleaning data)

### Sử dụng regular expression

vD: đ/c là đ/c pass:  
v/c số  
v/c tên  
v/c ngày  
v/c số điện thoại  
v/c địa chỉ

- Biên dịch mẫu
- Sử dụng mẫu đã biên dịch để khớp với các giá trị → Thoát → OK.
- Giúp mẫu được sử dụng nhiều lần
- Hữu ích vì có thể so khớp mẫu với lần lượt từng giá trị của cả cột



## Làm sạch dữ liệu (Cleaning data)

### • Ví dụ

```
# Import the regular expression module
import re

# Compile the pattern: prog
prog = re.compile('\d{3}-\d{3}-\d{4}') ✓

# See if the pattern matches
result = prog.match('123-456-7890')
print(bool(result))

# See if the pattern matches
result2 = prog.match('1123-456-7890')
print(bool(result2)) false
True
False
```



## Làm sạch dữ liệu (Cleaning data)

### • Ví dụ:

```
# Write the first pattern: A telephone number of the format xxx-xxx-xxxx  
pattern1 = bool(re.match(pattern='^\\d{3}-\\d{3}-\\d{4}', string='123-456-7890'))  
print(pattern1)  
  
# Write the second pattern:  
# A dollar sign, an arbitrary number of digits, a decimal point, 2 digits  
pattern2 = bool(re.match(pattern='^\\$\\d*\\.\\d{2}', string='$123.45'))  
print(pattern2)  
  
# Write the third pattern:  
# A capital letter, followed by an arbitrary number of alphanumeric characters.  
pattern3 = bool(re.match(pattern='[A-Z]\\w*', string='Australia'))  
print(pattern3)  
  
True  
True  
True
```

$\rightarrow$   $\backslash d^+ \rightarrow \backslash d\{1, 4\}$

## Làm sạch dữ liệu (Cleaning data)

### • Ví dụ: Lọc chuỗi số từ chuỗi

```
# Find the numeric values: matches  
matches = re.findall('\\d+', 'She has 10 apples, 3 pears and 5 plums.')  
# Print the matches  
print(matches)  
  
['10', '3', '5']
```

## Làm sạch dữ liệu (Cleaning data)

### ☐ Xử lý các ký tự không phù hợp

- Sử dụng các phương thức str.xxx()
- Sau đó chuyển kiểu dữ liệu phù hợp:  
.astype('kiểu\_dl')

## Làm sạch dữ liệu (Cleaning data)

### ● Ví dụ

```
so_survey_df['RawSalary'].head()          # đang ở dạng chứa kí tự không đúng
                                            # đúng là dạng chứa số thập phân
                                            # → thay dấu ',' thành '.'

# Remove the commas in the column
so_survey_df['RawSalary'] = so_survey_df['RawSalary'].str.replace(',', '') → thay dấu ',' thành ''

# Remove the dollar signs, euro signs in the column
so_survey_df['RawSalary'] = so_survey_df['RawSalary'].str.replace('$', '')
so_survey_df['RawSalary'] = so_survey_df['RawSalary'].str.replace('£', '')

so_survey_df['RawSalary'] = so_survey_df['RawSalary'].astype('float') → chuyển sang dạng số thập phân
                                            # float

so_survey_df['RawSalary'].head()
                                            # → thay dấu ',' thành ''
                                            # → thay dấu '$' thành ''
                                            # → thay dấu '£' thành ''
```

0 NaN  
1 70,841.00  
2 NaN  
3 21,426.00  
4 £41,671.00

Name: RawSalary, dtype: object

## Làm sạch dữ liệu (Cleaning data)

- Tìm các ký tự khác: dùng pd. to\_numeric(df['tên\_cột']), errors= 'coerce')
- Ví dụ:

coerced\_vals = pd. to\_numeric(so\_survey\_df['RawSalary'], errors='coerce')  
so\_survey\_df[coerced\_vals.isna()].head()

| vertedSalary | Hobby | Country      | StackOverflowJobsRecommend | VersionControl | Age | Years<br>Experience | Gender | RawSalary | Paid_Job | equal_binned | boundary_binned |
|--------------|-------|--------------|----------------------------|----------------|-----|---------------------|--------|-----------|----------|--------------|-----------------|
| NaN          | Yes   | South Africa |                            | NaN            | Git | 21                  | 13     | Male      | NaN      | 0            | NaN             |
| NaN          | No    | Sweeden      |                            | 8.0            | Git | 45                  | 11     | NaN       | NaN      | 0            | NaN             |
| NaN          | Yes   | UK           |                            | NaN            | Git | 34                  | 11     | Male      | NaN      | 0            | NaN             |
| NaN          | No    | South Africa |                            | 10.0           | Git | 23                  | 17     | Male      | NaN      | 0            | NaN             |
| NaN          | Yes   | USA          |                            | 10.0           | Git | 42                  | 0      | NaN       | NaN      | 0            | NaN             |

Data Pre-processing and Analysis

59

## Làm sạch dữ liệu (Cleaning data)

### □ Sử dụng “chuỗi” các phương thức

- Khi áp dụng nhiều thao tác trên cùng một cột (như các phần trước), chúng ta đã thực hiện các thay đổi trong một số bước, gán lại kết quả trong mỗi bước. Tuy nhiên, khi áp dụng nhiều hoạt động liên tiếp trên cùng một cột, chúng ta có thể "xâu chuỗi" các hoạt động này lại với nhau để rõ ràng và dễ quản lý bằng cách dùng:

# Method chaining      A  
df['column'] = df['column'].method1().method2().method3()  
                        A      A      A  
  
# Same as  
df['column'] = df['column'].method1()  
df['column'] = df['column'].method2()  
df['column'] = df['column'].method3()



60

## Xử lý dữ liệu lỏn xộn (messy data)

### • Ví dụ

```
# Use method chaining  
so_survey_df['RawSalary'] = so_survey_df['RawSalary']\  
    .str.replace(',', '')\  
    .str.replace('$', '')\  
    .str.replace('f', '')\  
    .astype('float')  
  
# Print the RawSalary column  
print(so_survey_df['RawSalary'].head())
```



## Làm sạch dữ liệu (Cleaning data)

### Sử dụng function

- Làm sạch dữ liệu đòi hỏi nhiều bước:
  - Trích xuất số từ chuỗi
  - Thực hiện việc chuyển đổi trên số trích xuất
  - ...
- Giải pháp: sử dụng Python function



## Làm sạch dữ liệu (Cleaning data)

### • Các function

- df.apply()
- Regular expression re.compile()
- User defined function



## Làm sạch dữ liệu (Cleaning data)

- Ví dụ: Viết function chuyển giới tính từ female => 0, male => 1, không có => np.nan

```
# Define recode_gender()
def recode_gender(gender):
    # Return 0 if gender is 'female'
    if gender == 'female':
        return 0
    # Return 1 if gender is 'male'
    elif gender == 'male':
        return 1
    # Return np.nan
    else:
        return np.nan
```

C1

```
# Apply the function to the sex column
students.sex = students.sex.apply(recode_gender)
students
```

C2

|   | name | age | address     | sex    |
|---|------|-----|-------------|--------|
| 0 | John | 17  | 123 rue N10 | male   |
| 1 | Lucy | 17  | 345 rue N6  | female |
| 2 | Lyly | 18  | 234 rue N12 | male   |

↓

|   | name | age | address     | sex |
|---|------|-----|-------------|-----|
| 0 | John | 17  | 123 rue N10 | 1   |
| 1 | Lucy | 17  | 345 rue N6  | 0   |
| 2 | Lyly | 18  | 234 rue N12 | 1   |



## Làm sạch dữ liệu (Cleaning data)

### Dữ liệu trùng lắp (duplicate data)

- Có thể làm sai lệch kết quả
- Giải pháp: dùng df.drop\_duplicates()
- Tham khảo:

[https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop\\_duplicates.html](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop_duplicates.html)



## Làm sạch dữ liệu (Cleaning data)

### Ví dụ

```
df2 = pd.DataFrame({'A':["foo", "foo", "foo", "bar"],  
                    "B":[0,1,1,1], "C":["A","A","B","A"]})  
df2
```

|   | A   | B | C |
|---|-----|---|---|
| 0 | foo | 0 | A |
| 1 | foo | 1 | A |
| 2 | foo | 1 | B |
| 3 | bar | 1 | A |

keep: first → lấy cột nào (nếu trùng)  
keep = first → lấy 1 đầu (1).

```
df2 = df2.drop_duplicates(subset=['A', 'C'], keep = False) → trả về 2.
```

|   | A   | B | C |
|---|-----|---|---|
| 2 | foo | 1 | B |
| 3 | bar | 1 | A |



## Làm sạch dữ liệu (Cleaning data)

### ❑ Dữ liệu bị thiếu (missing data)

- Bỏ đi các dữ liệu bị thiếu
- Sử dụng df.dropna()



## Làm sạch dữ liệu (Cleaning data)

### • Ví dụ

df3

```
# remove all column where all value is 'NaN' exists  
df3 = df3.dropna(axis=1, how='all')
```

|   | A   | B   | C   | D   |
|---|-----|-----|-----|-----|
| 0 | NaN | 2.0 | NaN | 5.0 |
| 1 | 3.0 | 4.0 | NaN | 1.0 |
| 2 | NaN | NaN | NaN | NaN |
| 3 | 3.0 | 4.0 | NaN | 1.0 |
| 4 | 3.0 | 4.0 | NaN | 1.0 |

df3

|   | A   | B   | D   |
|---|-----|-----|-----|
| 0 | NaN | 2.0 | 5.0 |
| 1 | 3.0 | 4.0 | 1.0 |
| 2 | NaN | NaN | NaN |
| 3 | 3.0 | 4.0 | 1.0 |
| 4 | 3.0 | 4.0 | 1.0 |

df3

```
#remove all row where all value is 'NaN' exists  
df3 = df3.dropna(axis=0, how='all')
```

|   | A   | B   | D   |
|---|-----|-----|-----|
| 0 | NaN | 2.0 | 5.0 |
| 1 | 3.0 | 4.0 | 1.0 |
| 3 | 3.0 | 4.0 | 1.0 |
| 4 | 3.0 | 4.0 | 1.0 |

Nguyên tắc: dòng nào có số  
NaN là số dòng cần  
xử lý. NaN là số dòng cần  
xử lý.



## Làm sạch dữ liệu (Cleaning data)

### ☐ Dữ liệu bị thiếu (missing data)

- Điền giá trị cho các dữ liệu bị thiếu
  - Điền bằng giá trị được cung cấp
  - Điền bằng giá trị có được từ summary statistics (describe)
  - Nếu cột dữ liệu có outlier thì nên điền median
- Sử dụng `df['tên_cột'].fillna(value)` hoặc `df[['tên_cột1', 'tên_cột2', ...]].fillna(value)`



## Làm sạch dữ liệu (Cleaning data)

### ● Ví dụ

```
# Calculate the mean of the A column: mean_A
mean_A = df3.A.mean()
# Replace all the missing values in A column with the mean
df3.A = df3.A.fillna(mean_A)
```

df3

|   | A   | B   | D   |
|---|-----|-----|-----|
| 0 | NaN | 2.0 | 5.0 |
| 1 | 3.0 | 4.0 | 1.0 |
| 3 | 3.0 | 4.0 | 1.0 |
| 4 | 3.0 | 4.0 | 1.0 |

df3

|   | A   | B   | D   |
|---|-----|-----|-----|
| 0 | 3.0 | 2.0 | 5.0 |
| 1 | 3.0 | 4.0 | 1.0 |
| 3 | 3.0 | 4.0 | 1.0 |
| 4 | 3.0 | 4.0 | 1.0 |



Data Pre-processing and Analysis

71



## Chapter 3

### Ex1: Tidying Data

#### Câu 1:

Cho dữ liệu BMI.xlsx. Bộ dữ liệu này ghi lại BMI của một số quốc gia trong một số năm.

- Đọc dữ liệu
- Xem xét vấn đề về dữ liệu cần khắc phục
- Chuẩn bị dữ liệu để khắc phục vấn đề trên

**Đặt yêu cầu ngược lại là cần phải tạo pivot table để xem thống kê theo country, year.**

- Hãy chuyển dữ liệu mới làm ở trên về dạng thống kê

#### Câu 2:

Cho dữ liệu student.xlsx. Bộ dữ liệu này ghi lại điểm các môn của sinh viên

- Đọc dữ liệu
- Xem xét vấn đề về dữ liệu cần khắc phục
- Chuẩn bị dữ liệu để khắc phục vấn đề trên

**Đặt yêu cầu ngược lại là cần phải tạo pivot table để xem thống kê theo từng student và subject.**

- Hãy chuyển dữ liệu mới làm ở trên về dạng thống kê

#### Câu 1: Gợi ý

In [1]: `import pandas as pd`

In [2]: `df = pd.read_excel("BMI.xlsx")  
df.head()`

Out[2]:

|   | Country     | Y1980 | Y1981 | Y1982 | Y1983 |
|---|-------------|-------|-------|-------|-------|
| 0 | Afghanistan | 21.48 | 21.46 | 21.45 | 21.44 |
| 1 | Albania     | 25.22 | 25.24 | 25.26 | 25.27 |
| 2 | Algeria     | 22.26 | 22.35 | 22.44 | 22.52 |

#### Vấn đề cần khắc phục:



- Tên cột chứa giá trị thay vì chứa biến

```
In [3]: # Melt df into new dataframe: df_melted
df_melted = pd.melt(df, id_vars=['Country'],
                     var_name="year",
                     value_name="bmi")
```

```
In [4]: df_melted
```

```
Out[4]:
```

|    | Country     | year  | bmi   |
|----|-------------|-------|-------|
| 0  | Afghanistan | Y1980 | 21.48 |
| 1  | Albania     | Y1980 | 25.22 |
| 2  | Algeria     | Y1980 | 22.26 |
| 3  | Afghanistan | Y1981 | 21.46 |
| 4  | Albania     | Y1981 | 25.24 |
| 5  | Algeria     | Y1981 | 22.35 |
| 6  | Afghanistan | Y1982 | 21.45 |
| 7  | Albania     | Y1982 | 25.26 |
| 8  | Algeria     | Y1982 | 22.44 |
| 9  | Afghanistan | Y1983 | 21.44 |
| 10 | Albania     | Y1983 | 25.27 |
| 11 | Algeria     | Y1983 | 22.52 |

```
In [5]: df_melted.year = df_melted.year.map(lambda x: int(x[1:]))


```

import os  
print (os.getcwd())  
→ Xem tập tin đang làm việc nằm ở đâu.

7/24/2019

Ex1



In [6]: df\_melted

Out[6]:

|    | Country     | year | bmi   |
|----|-------------|------|-------|
| 0  | Afghanistan | 1980 | 21.48 |
| 1  | Albania     | 1980 | 25.22 |
| 2  | Algeria     | 1980 | 22.26 |
| 3  | Afghanistan | 1981 | 21.46 |
| 4  | Albania     | 1981 | 25.24 |
| 5  | Algeria     | 1981 | 22.35 |
| 6  | Afghanistan | 1982 | 21.45 |
| 7  | Albania     | 1982 | 25.26 |
| 8  | Algeria     | 1982 | 22.44 |
| 9  | Afghanistan | 1983 | 21.44 |
| 10 | Albania     | 1983 | 25.27 |
| 11 | Algeria     | 1983 | 22.52 |

In [7]: df\_pivot = df\_melted.pivot(index='Country', columns='year', values='bmi')

In [8]: df\_pivot

Out[8]:

| Country     | year  | 1980  | 1981  | 1982  | 1983 |
|-------------|-------|-------|-------|-------|------|
| Afghanistan | 21.48 | 21.46 | 21.45 | 21.44 |      |
| Albania     | 25.22 | 25.24 | 25.26 | 25.27 |      |
| Algeria     | 22.26 | 22.35 | 22.44 | 22.52 |      |

## Câu 2

- Các bạn tự làm nhé.

In [ ]:

## Ex2: Tidying Data

### Câu 1: Pew Research Center

Cho dữ liệu: `pew-raw.csv`. Bộ dữ liệu này khám phá mối quan hệ giữa thu nhập và tôn giáo (income & religion)

- Đọc dữ liệu
- Xem xét vấn đề về dữ liệu cần khắc phục
- Chuẩn lại dữ liệu để khắc phục vấn đề trên

### Câu 2: Billboard Top 100

Cho dữ liệu `billboard.csv`. Bộ dữ liệu này đại diện cho thứ hạng hàng tuần của các bài hát kể từ thời điểm chúng lọt vào Top 100 của Billboard cho đến 75 tuần tiếp theo.

- Đọc dữ liệu
- Xem xét vấn đề về dữ liệu cần khắc phục
- Chuẩn lại dữ liệu để khắc phục vấn đề trên
- Sau khi chuẩn lại dữ liệu, có điều gì phát sinh cần khắc phục tiếp theo không? Nếu có thì hãy chuẩn lại dữ liệu mới có ở câu trên

### Câu 3: Tuberculosis

Cho dữ liệu `tb-raw.csv`. Bộ dữ liệu này ghi lại số ca bệnh lao được xác nhận theo quốc gia, năm, tuổi và giới tính.

- Đọc dữ liệu
- Xem xét vấn đề về dữ liệu cần khắc phục
- Chuẩn lại dữ liệu để khắc phục vấn đề trên

### Câu 4: Global Historical Climatology Network

Cho dữ liệu `weather-raw.csv`. Bộ dữ liệu này đại diện cho các bản ghi thời tiết hàng ngày cho một trạm thời tiết (MX17004) ở Mexico trong 5 tháng năm 2010.

- Đọc dữ liệu
- Xem xét vấn đề về dữ liệu cần khắc phục
- Chuẩn lại dữ liệu để khắc phục vấn đề trên

### Câu 1: Gợi ý

In [1]: `import pandas as pd`

In [2]: `df = pd.read_csv("pew-raw.csv")  
df`

Out[2]:

|   | religion                | <\$10k | \$10-20k | \$20-30k | \$30-40k | \$40-50k | \$50-75k |
|---|-------------------------|--------|----------|----------|----------|----------|----------|
| 0 | Agnostic                | 27     | 34       | 60       | 81       | 76       | 137      |
| 1 | Atheist                 | 12     | 27       | 37       | 52       | 35       | 70       |
| 2 | Buddhist                | 27     | 21       | 30       | 34       | 33       | 58       |
| 3 | Catholic                | 418    | 617      | 732      | 670      | 638      | 1116     |
| 4 | Dont know/refused       | 15     | 14       | 15       | 11       | 10       | 35       |
| 5 | Evangelical Prot        | 575    | 869      | 1064     | 982      | 881      | 1486     |
| 6 | Hindu                   | 1      | 9        | 7        | 9        | 11       | 34       |
| 7 | Historically Black Prot | 228    | 244      | 236      | 238      | 197      | 223      |
| 8 | Jehovahs Witness        | 20     | 27       | 24       | 24       | 21       | 30       |
| 9 | Jewish                  | 19     | 19       | 25       | 25       | 30       | 95       |

### Vấn đề cần khắc phục:

- Tên cột chứa giá trị thay vì chứa biến, trong từng cell chứa tần suất

In [3]: `df_after = pd.melt(frame=df,  
 id_vars = ["religion"],  
 var_name="income",  
 value_name="freq")  
#df_after = df_after.sort_values(by=["religion"])  
df_after.head(10)`

Out[3]:

|   | religion                | income | freq |
|---|-------------------------|--------|------|
| 0 | Agnostic                | <\$10k | 27   |
| 1 | Atheist                 | <\$10k | 12   |
| 2 | Buddhist                | <\$10k | 27   |
| 3 | Catholic                | <\$10k | 418  |
| 4 | Dont know/refused       | <\$10k | 15   |
| 5 | Evangelical Prot        | <\$10k | 575  |
| 6 | Hindu                   | <\$10k | 1    |
| 7 | Historically Black Prot | <\$10k | 228  |
| 8 | Jehovahs Witness        | <\$10k | 20   |
| 9 | Jewish                  | <\$10k | 19   |

## Câu 2: Gợi ý

```
In [4]: df1 = pd.read_csv("billboard.csv", encoding="mac_latin2")
df1.head()
```

Out[4]:

|   | year | artist.inverted     | track                                 | time | genre | date.entered | date.peaked | x1st.week | x2nd.week |
|---|------|---------------------|---------------------------------------|------|-------|--------------|-------------|-----------|-----------|
| 0 | 2000 | Destiny's Child     | Independent Women Part I              | 3:38 | Rock  | 2000-09-23   | 2000-11-18  | 78        | 63.0      |
| 1 | 2000 | Santana             | Maria, Maria                          | 4:18 | Rock  | 2000-02-12   | 2000-04-08  | 15        | 8.0       |
| 2 | 2000 | Savage Garden       | I Knew I Loved You                    | 4:07 | Rock  | 1999-10-23   | 2000-01-29  | 71        | 48.0      |
| 3 | 2000 | Madonna             | Music                                 | 3:45 | Rock  | 2000-08-12   | 2000-09-16  | 41        | 23.0      |
| 4 | 2000 | Aguilera, Christina | Come On Over Baby (All I Want Is You) | 3:38 | Rock  | 2000-08-05   | 2000-10-14  | 57        | 47.0      |

5 rows × 83 columns



### Vấn đề cần khắc phục:

- Tiêu đề cột chứa giá trị thay vì chứa biến, trong từng cell chứa thứ hạng
- Nếu một bài hát nằm trong Top 100 dưới 75 tuần, các cột còn lại chứa đầy các giá trị bị thiếu (NaN)

```
In [5]: df1_after = pd.melt(frame=df1,
                        id_vars = ["year", "artist.inverted", "track", "time", "genre",
                                   var_name="week", value_name="rank")

df1_after.head(10)
```

Out[5]:

|   | year | artist.inverted     | track                                 | time | genre   | date.entered | date.peaked | week      | rank |
|---|------|---------------------|---------------------------------------|------|---------|--------------|-------------|-----------|------|
| 0 | 2000 | Destiny's Child     | Independent Women Part I              | 3:38 | Rock    | 2000-09-23   | 2000-11-18  | x1st.week | 78.0 |
| 1 | 2000 | Santana             | Maria, Maria                          | 4:18 | Rock    | 2000-02-12   | 2000-04-08  | x1st.week | 15.0 |
| 2 | 2000 | Savage Garden       | I Knew I Loved You                    | 4:07 | Rock    | 1999-10-23   | 2000-01-29  | x1st.week | 71.0 |
| 3 | 2000 | Madonna             | Music                                 | 3:45 | Rock    | 2000-08-12   | 2000-09-16  | x1st.week | 41.0 |
| 4 | 2000 | Aguilera, Christina | Come On Over Baby (All I Want Is You) | 3:38 | Rock    | 2000-08-05   | 2000-10-14  | x1st.week | 57.0 |
| 5 | 2000 | Janet               | Doesn't Really Matter                 | 4:17 | Rock    | 2000-06-17   | 2000-08-26  | x1st.week | 59.0 |
| 6 | 2000 | Destiny's Child     | Say My Name                           | 4:31 | Rock    | 1999-12-25   | 2000-03-18  | x1st.week | 83.0 |
| 7 | 2000 | Iglesias, Enrique   | Be With You                           | 3:36 | Latin   | 2000-04-01   | 2000-06-24  | x1st.week | 63.0 |
| 8 | 2000 | Sisqo               | Incomplete                            | 3:52 | Rock    | 2000-06-24   | 2000-08-12  | x1st.week | 77.0 |
| 9 | 2000 | Lonestar            | Amazed                                | 4:25 | Country | 1999-06-05   | 2000-03-04  | x1st.week | 81.0 |

```
In [6]: # Chuẩn hóa dữ liệu
#_ Thay chuỗi trong tuần bằng số
#_ Xóa bỏ các dòng dữ liệu chứa NaN
#_ Xem kiểu của rank, nếu chưa là số thì chuyển thành số
```

```
In [7]: df1_after["week"] = df1_after['week'].str.extract('(\d+)', expand=False).astype(int)
```

```
In [8]: # Cleaning out unnecessary rows
df1_after = df1_after.dropna()
```

```
In [9]: df1_after["rank"] = df1_after["rank"].astype(int)
```

7/24/2019

Ex2

In [10]: df1\_after.head()

Out[10]:

|   | year | artist.inverted     | track                                 | time | genre | date.entered | date.peaked | week | rank |
|---|------|---------------------|---------------------------------------|------|-------|--------------|-------------|------|------|
| 0 | 2000 | Destiny's Child     | Independent Women Part I              | 3:38 | Rock  | 2000-09-23   | 2000-11-18  | 1    | 78   |
| 1 | 2000 | Santana             | Maria, Maria                          | 4:18 | Rock  | 2000-02-12   | 2000-04-08  | 1    | 15   |
| 2 | 2000 | Savage Garden       | I Knew I Loved You                    | 4:07 | Rock  | 1999-10-23   | 2000-01-29  | 1    | 71   |
| 3 | 2000 | Madonna             | Music                                 | 3:45 | Rock  | 2000-08-12   | 2000-09-16  | 1    | 41   |
| 4 | 2000 | Aguilera, Christina | Come On Over Baby (All I Want Is You) | 3:38 | Rock  | 2000-08-05   | 2000-10-14  | 1    | 57   |

**Vấn đề mới phát sinh:**

- Nhiều đơn vị quan sát (song & rank của nó) trong một bảng duy nhất.

In [11]: # Tạo một dataframe songs và các thông tin của nó  
# Tạo một dataframe ranks chứa id của song và week kèm theo rank

In [12]: songs\_cols = ["year", "artist.inverted", "track", "time", "genre"]  
songs = df1\_after[songs\_cols].drop\_duplicates()  
songs = songs.reset\_index(drop=True)  
songs["song\_id"] = songs.index  
songs.head(10)

Out[12]:

|   | year | artist.inverted     | track                                 | time | genre   | song_id |
|---|------|---------------------|---------------------------------------|------|---------|---------|
| 0 | 2000 | Destiny's Child     | Independent Women Part I              | 3:38 | Rock    | 0       |
| 1 | 2000 | Santana             | Maria, Maria                          | 4:18 | Rock    | 1       |
| 2 | 2000 | Savage Garden       | I Knew I Loved You                    | 4:07 | Rock    | 2       |
| 3 | 2000 | Madonna             | Music                                 | 3:45 | Rock    | 3       |
| 4 | 2000 | Aguilera, Christina | Come On Over Baby (All I Want Is You) | 3:38 | Rock    | 4       |
| 5 | 2000 | Janet               | Doesn't Really Matter                 | 4:17 | Rock    | 5       |
| 6 | 2000 | Destiny's Child     | Say My Name                           | 4:31 | Rock    | 6       |
| 7 | 2000 | Iglesias, Enrique   | Be With You                           | 3:36 | Latin   | 7       |
| 8 | 2000 | Sisqo               | Incomplete                            | 3:52 | Rock    | 8       |
| 9 | 2000 | Lonestar            | Amazed                                | 4:25 | Country | 9       |

```
In [13]: ranks = pd.merge(df1_after, songs, on=["year", "artist.inverted", "track", "time"],  
ranks = ranks[["song_id", "date.entered", "date.peaked", "week", "rank"]]  
ranks.head(10)
```

Out[13]:

|   | song_id | date.entered | date.peaked | week | rank |
|---|---------|--------------|-------------|------|------|
| 0 | 0       | 2000-09-23   | 2000-11-18  | 1    | 78   |
| 1 | 0       | 2000-09-23   | 2000-11-18  | 2    | 63   |
| 2 | 0       | 2000-09-23   | 2000-11-18  | 3    | 49   |
| 3 | 0       | 2000-09-23   | 2000-11-18  | 4    | 33   |
| 4 | 0       | 2000-09-23   | 2000-11-18  | 5    | 23   |
| 5 | 0       | 2000-09-23   | 2000-11-18  | 6    | 15   |
| 6 | 0       | 2000-09-23   | 2000-11-18  | 7    | 7    |
| 7 | 0       | 2000-09-23   | 2000-11-18  | 8    | 5    |
| 8 | 0       | 2000-09-23   | 2000-11-18  | 9    | 1    |
| 9 | 0       | 2000-09-23   | 2000-11-18  | 10   | 1    |

In [14]: # Tạo cột date  
ranks['date'] = pd.to\_datetime(ranks['date.entered']) + pd.to\_timedelta(ranks['week'],  
unit='w') - pd.DateOffset(weeks=1) ???

In [15]: ranks.head()

Out[15]:

|   | song_id | date.entered | date.peaked | week | rank | date       |
|---|---------|--------------|-------------|------|------|------------|
| 0 | 0       | 2000-09-23   | 2000-11-18  | 1    | 78   | 2000-09-23 |
| 1 | 0       | 2000-09-23   | 2000-11-18  | 2    | 63   | 2000-09-30 |
| 2 | 0       | 2000-09-23   | 2000-11-18  | 3    | 49   | 2000-10-07 |
| 3 | 0       | 2000-09-23   | 2000-11-18  | 4    | 33   | 2000-10-14 |
| 4 | 0       | 2000-09-23   | 2000-11-18  | 5    | 23   | 2000-10-21 |

In [16]: ranks = ranks.drop(["date.entered", "date.peaked"], axis=1)

In [17]: ranks.head()

Out[17]:

|   | song_id | week | rank | date       |
|---|---------|------|------|------------|
| 0 | 0       | 1    | 78   | 2000-09-23 |
| 1 | 0       | 2    | 63   | 2000-09-30 |
| 2 | 0       | 3    | 49   | 2000-10-07 |
| 3 | 0       | 4    | 33   | 2000-10-14 |
| 4 | 0       | 5    | 23   | 2000-10-21 |

### Câu 3: Gợi ý

In [18]: `df2 = pd.read_csv("tb-raw.csv")  
df2`

Out[18]:

|   | country | year | m014  | m1524 | m2534  | m3544 | m4554 | m5564 | m65   | mu  | f014  |
|---|---------|------|-------|-------|--------|-------|-------|-------|-------|-----|-------|
| 0 | AD      | 2000 | 0.0   | 0.0   | 1.0    | 0.0   | 0     | 0     | 0.0   | NaN | NaN   |
| 1 | AE      | 2000 | 2.0   | 4.0   | 4.0    | 6.0   | 5     | 12    | 10.0  | NaN | 3.0   |
| 2 | AF      | 2000 | 52.0  | 228.0 | 183.0  | 149.0 | 129   | 94    | 80.0  | NaN | 93.0  |
| 3 | AG      | 2000 | 0.0   | 0.0   | 0.0    | 0.0   | 0     | 0     | 1.0   | NaN | 1.0   |
| 4 | AL      | 2000 | 2.0   | 19.0  | 21.0   | 14.0  | 24    | 19    | 16.0  | NaN | 3.0   |
| 5 | AM      | 2000 | 2.0   | 152.0 | 130.0  | 131.0 | 63    | 26    | 21.0  | NaN | 1.0   |
| 6 | AN      | 2000 | 0.0   | 0.0   | 1.0    | 2.0   | 0     | 0     | 0.0   | NaN | 0.0   |
| 7 | AO      | 2000 | 186.0 | 999.0 | 1003.0 | 912.0 | 482   | 312   | 194.0 | NaN | 247.0 |
| 8 | AR      | 2000 | 97.0  | 278.0 | 594.0  | 402.0 | 419   | 368   | 330.0 | NaN | 121.0 |
| 9 | AS      | 2000 | NaN   | NaN   | NaN    | NaN   | 1     | 1     | NaN   | NaN | NaN   |

### Vấn đề cần khắc phục:

- Tên cột chứa giá trị thay vì chứa biến, ngoài ra nó còn chứa 2 thông tin là giới tính và độ tuổi
- Trong cell chứa hỗn hợp giá trị (value, 0 và NaN)

In [19]: `# Chuyển dữ liệu theo định dạng country, year, và sex_and_age  
df2_after = pd.melt(df2, id_vars=["country", "year"], value_name="cases", var_name="sex_and_age")  
df2_after.head()`

Out[19]:

|   | country | year | sex_and_age | cases |
|---|---------|------|-------------|-------|
| 0 | AD      | 2000 | m014        | 0.0   |
| 1 | AE      | 2000 | m014        | 2.0   |
| 2 | AF      | 2000 | m014        | 52.0  |
| 3 | AG      | 2000 | m014        | 0.0   |
| 4 | AL      | 2000 | m014        | 2.0   |

có kè chí \ df1,2  
156f2-55'f'c'c'

In [20]: `# Tách sex_and_age thành 2 cột: sex và age  
# Extract Sex, Age Lower bound and Age upper bound group  
tmp_df = df2_after["sex_and_age"].str.extract("(\\D)(\\d+)(\\d{2})")  
# Name columns  
tmp_df.columns = ["sex", "age_lower", "age_upper"]  
  
# Create `age` column based on `age_Lower` and `age_upper`  
tmp_df["age"] = tmp_df["age_lower"] + "-" + tmp_df["age_upper"]`

```
In [21]: # Merge
df2_after = pd.concat([df2_after, tmp_df], axis=1)
df2_after.head()
```

Out[21]:

|   | country | year | sex_and_age | cases | sex | age_lower | age_upper | age  |
|---|---------|------|-------------|-------|-----|-----------|-----------|------|
| 0 | AD      | 2000 | m014        | 0.0   | m   | 0         | 14        | 0-14 |
| 1 | AE      | 2000 | m014        | 2.0   | m   | 0         | 14        | 0-14 |
| 2 | AF      | 2000 | m014        | 52.0  | m   | 0         | 14        | 0-14 |
| 3 | AG      | 2000 | m014        | 0.0   | m   | 0         | 14        | 0-14 |
| 4 | AL      | 2000 | m014        | 2.0   | m   | 0         | 14        | 0-14 |

## Câu 4: Gợi ý

```
In [22]: df3 = pd.read_csv("weather-raw.csv")
df3
```

Out[22]:

|   | id      | year | month | element | d1  | d2   | d3   | d4  | d5   | d6  | d7  | d8  |
|---|---------|------|-------|---------|-----|------|------|-----|------|-----|-----|-----|
| 0 | MX17004 | 2010 | 1     | tmax    | NaN | NaN  | NaN  | NaN | NaN  | NaN | NaN | NaN |
| 1 | MX17004 | 2010 | 1     | tmin    | NaN | NaN  | NaN  | NaN | NaN  | NaN | NaN | NaN |
| 2 | MX17004 | 2010 | 2     | tmax    | NaN | 27.3 | 24.1 | NaN | NaN  | NaN | NaN | NaN |
| 3 | MX17004 | 2010 | 2     | tmin    | NaN | 14.4 | 14.4 | NaN | NaN  | NaN | NaN | NaN |
| 4 | MX17004 | 2010 | 3     | tmax    | NaN | NaN  | NaN  | NaN | 32.1 | NaN | NaN | NaN |
| 5 | MX17004 | 2010 | 3     | tmin    | NaN | NaN  | NaN  | NaN | 14.2 | NaN | NaN | NaN |
| 6 | MX17004 | 2010 | 4     | tmax    | NaN | NaN  | NaN  | NaN | NaN  | NaN | NaN | NaN |
| 7 | MX17004 | 2010 | 4     | tmin    | NaN | NaN  | NaN  | NaN | NaN  | NaN | NaN | NaN |
| 8 | MX17004 | 2010 | 5     | tmax    | NaN | NaN  | NaN  | NaN | NaN  | NaN | NaN | NaN |
| 9 | MX17004 | 2010 | 5     | tmin    | NaN | NaN  | NaN  | NaN | NaN  | NaN | NaN | NaN |

## Vấn đề cần khắc phục

- Các biến được lưu trữ trong cả các dòng (tmin, tmax) và các cột (days).
- Tên cột chứa giá trị day thay vì chứa biến
- Trong từng cell chứa temperature nhưng có rất nhiều giá trị NaN  
=> Để làm cho tập dữ liệu này gọn gàng => di chuyển ba biến bị đặt sai (tmin, tmax và day) thành ba cột riêng lẻ: tmin, tmax và date.

7/24/2019

Ex2

In [23]: `df3_after = pd.melt(df3, id_vars=["id", "year", "month", "element"], var_name="day_raw")  
df3_after.head(10)`

Out[23]:

|   | <b>id</b> | <b>year</b> | <b>month</b> | <b>element</b> | <b>day_raw</b> | <b>value</b> |
|---|-----------|-------------|--------------|----------------|----------------|--------------|
| 0 | MX17004   | 2010        | 1            | tmax           | d1             | NaN          |
| 1 | MX17004   | 2010        | 1            | tmin           | d1             | NaN          |
| 2 | MX17004   | 2010        | 2            | tmax           | d1             | NaN          |
| 3 | MX17004   | 2010        | 2            | tmin           | d1             | NaN          |
| 4 | MX17004   | 2010        | 3            | tmax           | d1             | NaN          |
| 5 | MX17004   | 2010        | 3            | tmin           | d1             | NaN          |
| 6 | MX17004   | 2010        | 4            | tmax           | d1             | NaN          |
| 7 | MX17004   | 2010        | 4            | tmin           | d1             | NaN          |
| 8 | MX17004   | 2010        | 5            | tmax           | d1             | NaN          |
| 9 | MX17004   | 2010        | 5            | tmin           | d1             | NaN          |

In [24]: `# Tạo cột day chỉ chứa ngày  
df3_after["day"] = df3_after["day_raw"].str.extract("d(\d+)", expand=False)`

Out[24]:

|   | <b>id</b> | <b>year</b> | <b>month</b> | <b>element</b> | <b>day_raw</b> | <b>value</b> | <b>day</b> |
|---|-----------|-------------|--------------|----------------|----------------|--------------|------------|
| 0 | MX17004   | 2010        | 1            | tmax           | d1             | NaN          | 1          |
| 1 | MX17004   | 2010        | 1            | tmin           | d1             | NaN          | 1          |
| 2 | MX17004   | 2010        | 2            | tmax           | d1             | NaN          | 1          |
| 3 | MX17004   | 2010        | 2            | tmin           | d1             | NaN          | 1          |
| 4 | MX17004   | 2010        | 3            | tmax           | d1             | NaN          | 1          |

In [26]: `# Chuyển dữ liệu thành số  
df3_after[["year", "month", "day"]] = df3_after[["year", "month", "day"]].apply(lambda x:  
pd.to_numeric(x, errors='ignore'))`

Out[26]:

|   | <b>id</b> | <b>year</b> | <b>month</b> | <b>element</b> | <b>day_raw</b> | <b>value</b> | <b>day</b> |
|---|-----------|-------------|--------------|----------------|----------------|--------------|------------|
| 0 | MX17004   | 2010        | 1            | tmax           | d1             | NaN          | 1          |
| 1 | MX17004   | 2010        | 1            | tmin           | d1             | NaN          | 1          |
| 2 | MX17004   | 2010        | 2            | tmax           | d1             | NaN          | 1          |
| 3 | MX17004   | 2010        | 2            | tmin           | d1             | NaN          | 1          |
| 4 | MX17004   | 2010        | 3            | tmax           | d1             | NaN          | 1          |

In [28]: # Tạo cột date chứa cả day, month, year  
 import datetime  
*Với hàm chuyển đổi day sang datetime.*  
 def create\_date\_from\_year\_month\_day(row):  
 return datetime.datetime(year=row["year"], month=int(row["month"]), day=row["day"])

In [29]: df3\_after["date"] = df3\_after.apply(lambda row: create\_date\_from\_year\_month\_day(row), axis=1)

In [30]: # bỏ các cột thừa  
 df3\_after = df3\_after.drop(['year', "month", "day", "day\_raw"], axis=1)

In [31]: # Bỏ các dòng dữ liệu NaN  
 df3\_after = df3\_after.dropna()

In [32]: df3\_after.head()

Out[32]:

|    |         | id | element | value | date       |
|----|---------|----|---------|-------|------------|
| 12 | MX17004 |    | tmax    | 27.3  | 2010-02-02 |
| 13 | MX17004 |    | tmin    | 14.4  | 2010-02-02 |
| 22 | MX17004 |    | tmax    | 24.1  | 2010-02-03 |
| 23 | MX17004 |    | tmin    | 14.4  | 2010-02-03 |
| 44 | MX17004 |    | tmax    | 32.1  | 2010-03-05 |

In [ ]:



## Ex3: Combining Data

### Câu 1: Illinois Male Baby Names

Cho dữ liệu 201\*-baby-names-illinois.csv. Bộ dữ liệu này thống kê tần suất các tên gọi được đặt.

- Đọc dữ liệu
- Xem xét vấn đề về dữ liệu cần khắc phục
- Chuẩn bị dữ liệu để khắc phục vấn đề trên

```
In [1]: import matplotlib.pyplot as plt
import pandas as pd
import glob
```

#### Vấn đề cần khắc phục

- Dữ liệu được lưu trữ trên nhiều table/file.
- Biến "Year" được để trong tên tập tin

```
gõ lệnh
In [2]: allFiles = glob.glob("201*-baby-names-illinois.csv")
frame = pd.DataFrame()
df_list= []
for file_ in allFiles:
    df = pd.read_csv(file_,index_col=None, header=0)
    print(file_, df.shape)
    df.columns = map(str.lower, df.columns)
    df["year"] = file_[0:4]
    df_list.append(df)
```

2014-baby-names-illinois.csv (101, 4)  
2015-baby-names-illinois.csv (100, 4)

```
In [3]: df4 = pd.concat(df_list)
df4.head(5)
```

Out[3]:

|   | rank | name      | frequency | sex  | year |
|---|------|-----------|-----------|------|------|
| 0 | 1    | Noah      | 837       | Male | 2014 |
| 1 | 2    | Alexander | 747       | Male | 2014 |
| 2 | 3    | William   | 687       | Male | 2014 |
| 3 | 4    | Michael   | 680       | Male | 2014 |
| 4 | 5    | Liam      | 670       | Male | 2014 |



## Ex4: Combining Data

### Câu 1: Cho dữ liệu employees1.csv và employees2.csv

- Đọc dữ liệu từ 2 tập tin trên
- Kết hợp 2 dữ liệu trên thành 1 DataFrame

### Câu 2: Cho dữ liệu department.csv

- Đọc dữ liệu từ tập tin trên
- Kết hợp dữ liệu này với dữ liệu kết quả từ câu 1

### Câu 3: Cho dữ liệu skills.csv

- Đọc dữ liệu từ tập tin trên
- Kết hợp dữ liệu này với dữ liệu kết quả từ câu 2

### Câu 4: Cho dữ liệu salary.csv

- Đọc dữ liệu từ tập tin trên
- Kết hợp dữ liệu này với dữ liệu ở câu 1 (gợi ý: dùng right\_on & left\_on khi merge vì trong salary có cột name, còn df ở câu 1 lại có cột employee trùng nội dung, bỏ cột name)

### Câu 1+: Cho dữ liệu employees1.csv và employees2.csv

- Đọc dữ liệu từ 2 tập tin trên vào 2 DataFrame với index của các DataFrame là 'employee'
- Kết hợp 2 DataFrame trên thành 1 DataFrame dùng chung 1 index là 'employee' (gợi ý: dùng left\_index hoặc/right\_index hoặc dùng dataframe1.join(dataframe2))

### Câu 5: Cho dữ liệu như sau:

1. df6 = pd.DataFrame({'name': ['Peter', 'Paul', 'Mary'], 'food': ['fish', 'beans', 'bread']}, columns=['name', 'food'])
  2. df7 = pd.DataFrame({'name': ['Mary', 'Joseph'], 'drink': ['wine', 'beer']}, columns=['name', 'drink'])
- Kết hợp 2 bộ dữ liệu này với tham số how='inner', how='outer', how='left', how='right'. Quan sát kết quả trong từng trường hợp.



## Câu 6: Cho dữ liệu như sau:

1. df8 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'], 'rank': [1, 2, 3, 4]})
2. df9 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'], 'rank': [3, 1, 4, 2]})
- Kết hợp 2 bộ dữ liệu này với tham số `on='tên_cột_trùng_lắp'` và/hoặc `suffixes=['_L', '_R']`

## Câu 1: Gợi ý

In [1]: `import pandas as pd`

In [2]: `df1 = pd.DataFrame({'employee': ['Bob', 'Jake', 'Lisa', 'Sue', 'John', 'Billy'], 'group': ['Accounting', 'Engineering', 'Engineering', 'HR', 'IT', 'Marketing']})`  
~~997~~  
`df2 = pd.DataFrame({'employee': ['Lisa', 'Bob', 'Jake', 'Sue', 'John', 'Billy'], 'hire_date': [2004, 2008, 2012, 2014, 2010, 2015]})`

In [3]: `df1.to_csv("employees1.csv")`  
~~997~~  
`df2.to_csv("employees2.csv")`

In [4]: `e1 = pd.read_csv("employees1.csv", index_col=0)`  
`e2 = pd.read_csv("employees2.csv", index_col=0)`

In [5]: `e1`

Out[5]:

|   | employee | group       |
|---|----------|-------------|
| 0 | Bob      | Accounting  |
| 1 | Jake     | Engineering |
| 2 | Lisa     | Engineering |
| 3 | Sue      | HR          |
| 4 | John     | IT          |
| 5 | Billy    | HR          |

In [6]: `e2`

Out[6]:

|   | employee | hire_date |
|---|----------|-----------|
| 0 | Lisa     | 2004      |
| 1 | Bob      | 2008      |
| 2 | Jake     | 2012      |
| 3 | Sue      | 2014      |
| 4 | John     | 2010      |
| 5 | Billy    | 2015      |



In [7]: `df = pd.merge(df1, df2)`  
`df`

Out[7]:

|   | employee | group       | hire_date |
|---|----------|-------------|-----------|
| 0 | Bob      | Accounting  | 2008      |
| 1 | Jake     | Engineering | 2012      |
| 2 | Lisa     | Engineering | 2004      |
| 3 | Sue      | HR          | 2014      |
| 4 | John     | IT          | 2010      |
| 5 | Billy    | HR          | 2015      |

## Câu 2: Gợi ý

In [8]: `df4 = pd.DataFrame({'group': ['Accounting', 'Engineering', 'HR', 'IT'], 'supervisor': ['Carly', 'Guido', 'Steve', 'Jame']})`

9/2

In [9]: `df4.to_csv("department.csv")`

In [10]: `d = pd.read_csv("department.csv", index_col=0)`

In [11]: `d`

Out[11]:

|   | group       | supervisor |
|---|-------------|------------|
| 0 | Accounting  | Carly      |
| 1 | Engineering | Guido      |
| 2 | HR          | Steve      |
| 3 | IT          | Jame       |

In [12]: `df_with_dept = pd.merge(df, d)`

In [13]: `df_with_dept`

Out[13]:

|   | employee | group       | hire_date | supervisor |
|---|----------|-------------|-----------|------------|
| 0 | Bob      | Accounting  | 2008      | Carly      |
| 1 | Jake     | Engineering | 2012      | Guido      |
| 2 | Lisa     | Engineering | 2004      | Guido      |
| 3 | Sue      | HR          | 2014      | Steve      |
| 4 | Billy    | HR          | 2015      | Steve      |
| 5 | John     | IT          | 2010      | Jame       |



### Câu 3: Gợi ý

```
In [14]: df5 = pd.DataFrame({'group': ['Accounting', 'Accounting',
   'Engineering', 'Engineering', 'HR', 'HR', 'IT', 'IT',
   'skills': ['math', 'spreadsheets', 'coding', 'linux',
   'spreadsheets', 'organization', 'coding', 'math']})
```

```
In [15]: df5.to_csv("skills.csv")
```

```
In [16]: skills = pd.read_csv("skills.csv", index_col=0)
```

```
In [17]: skills
```

Out[17]:

|   | group       | skills       |
|---|-------------|--------------|
| 0 | Accounting  | math         |
| 1 | Accounting  | spreadsheets |
| 2 | Engineering | coding       |
| 3 | Engineering | linux        |
| 4 | HR          | spreadsheets |
| 5 | HR          | organization |
| 6 | IT          | coding       |
| 7 | IT          | math         |

```
In [18]: df_with_dept_skills = pd.merge(df_with_dept, skills)
```

```
In [19]: df_with_dept_skills
```

Out[19]:

|    | employee | group       | hire_date | supervisor | skills       |
|----|----------|-------------|-----------|------------|--------------|
| 0  | Bob      | Accounting  | 2008      | Carly      | math         |
| 1  | Bob      | Accounting  | 2008      | Carly      | spreadsheets |
| 2  | Jake     | Engineering | 2012      | Guido      | coding       |
| 3  | Jake     | Engineering | 2012      | Guido      | linux        |
| 4  | Lisa     | Engineering | 2004      | Guido      | coding       |
| 5  | Lisa     | Engineering | 2004      | Guido      | linux        |
| 6  | Sue      | HR          | 2014      | Steve      | spreadsheets |
| 7  | Sue      | HR          | 2014      | Steve      | organization |
| 8  | Billy    | HR          | 2015      | Steve      | spreadsheets |
| 9  | Billy    | HR          | 2015      | Steve      | organization |
| 10 | John     | IT          | 2010      | Jame       | coding       |
| 11 | John     | IT          | 2010      | Jame       | math         |



## Câu 4: Gợi ý

```
In [20]: df3 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue', 'John', 'Billy'],
   'salary': [70000, 80000, 120000, 90000, 125000, 92000]})
```

```
In [21]: df3.to_csv("salary.csv")
```

```
In [22]: salary = pd.read_csv("salary.csv", index_col=0)
salary
```

Out[22]:

|   | name  | salary |
|---|-------|--------|
| 0 | Bob   | 70000  |
| 1 | Jake  | 80000  |
| 2 | Lisa  | 120000 |
| 3 | Sue   | 90000  |
| 4 | John  | 125000 |
| 5 | Billy | 92000  |

```
In [23]: df_em_salary = pd.merge(df, salary, left_on="employee", right_on="name").drop('name')
```

```
In [24]: df_em_salary
```

Out[24]:

|   | employee | group       | hire_date | salary |
|---|----------|-------------|-----------|--------|
| 0 | Bob      | Accounting  | 2008      | 70000  |
| 1 | Jake     | Engineering | 2012      | 80000  |
| 2 | Lisa     | Engineering | 2004      | 120000 |
| 3 | Sue      | HR          | 2014      | 90000  |
| 4 | John     | IT          | 2010      | 125000 |
| 5 | Billy    | HR          | 2015      | 92000  |

## Câu 1+: Gợi ý

```
In [25]: e1a = pd.read_csv("employees1.csv", index_col=0)
e2a = pd.read_csv("employees2.csv", index_col=0)
e1a = e1a.set_index('employee')
e2a = e2a.set_index('employee')
```



In [26]: `display(e1a, e2a)`

group  
employee  
Bob Accounting  
Jake Engineering  
Lisa Engineering  
Sue HR  
John IT  
Billy HR

hire\_date  
employee  
Lisa 2004  
Bob 2008  
Jake 2012  
Sue 2014  
John 2010  
Billy 2015

In [27]: `df_merge = pd.merge(e1a, e2a, left_index=True, right_index=True)`  
`df_merge`

Out[27]:

group hire\_date  
employee  
Bob Accounting 2008  
Jake Engineering 2012  
Lisa Engineering 2004  
Sue HR 2014  
John IT 2010  
Billy HR 2015



```
In [28]: df_join = e1a.join(e2a)
df_join
```

Out[28]:

| group    | hire_date   |      |
|----------|-------------|------|
| employee |             |      |
| Bob      | Accounting  | 2008 |
| Jake     | Engineering | 2012 |
| Lisa     | Engineering | 2004 |
| Sue      | HR          | 2014 |
| John     | IT          | 2010 |
| Billy    | HR          | 2015 |

## Câu 5: Gợi ý

```
In [29]: df6 = pd.DataFrame({'name': ['Peter', 'Paul', 'Mary'],
                           'food': ['fish', 'beans', 'bread']},
                           columns=['name', 'food'])
df7 = pd.DataFrame({'name': ['Mary', 'Joseph'],
                           'drink': ['wine', 'beer']},
                           columns=['name', 'drink'])
```

```
In [30]: display(df6, df7)
```

|   | name  | food  |
|---|-------|-------|
| 0 | Peter | fish  |
| 1 | Paul  | beans |
| 2 | Mary  | bread |

|   | name   | drink |
|---|--------|-------|
| 0 | Mary   | wine  |
| 1 | Joseph | beer  |

```
In [31]: df67_merge = pd.merge(df6, df7)
df67_merge
```

Out[31]:

|   | name | food  | drink |
|---|------|-------|-------|
| 0 | Mary | bread | wine  |



In [32]: `df67_inner = pd.merge(df6, df7, how='inner')`  
`df67_inner`

Out[32]:

|   | name | food  | drink |
|---|------|-------|-------|
| 0 | Mary | bread | wine  |

In [33]: `df67_outer = pd.merge(df6, df7, how='outer')`  
`df67_outer`

Out[33]:

|   | name   | food  | drink |
|---|--------|-------|-------|
| 0 | Peter  | fish  | NaN   |
| 1 | Paul   | beans | NaN   |
| 2 | Mary   | bread | wine  |
| 3 | Joseph | NaN   | beer  |

In [34]: `df67_left = pd.merge(df6, df7, how='left')`  
`df67_left`

Out[34]:

|   | name  | food  | drink |
|---|-------|-------|-------|
| 0 | Peter | fish  | NaN   |
| 1 | Paul  | beans | NaN   |
| 2 | Mary  | bread | wine  |

In [35]: `df67_right = pd.merge(df6, df7, how='right')`  
`df67_right`

Out[35]:

|   | name   | food  | drink |
|---|--------|-------|-------|
| 0 | Mary   | bread | wine  |
| 1 | Joseph | NaN   | beer  |

## Câu 6: Gợi ý

In [36]: `df8 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'],`  
`'rank': [1, 2, 3, 4]})`  
`df9 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'],`  
`'rank': [3, 1, 4, 2]})`



In [37]: `display(df8, df9)`

|   | name | rank |
|---|------|------|
| 0 | Bob  | 1    |
| 1 | Jake | 2    |
| 2 | Lisa | 3    |
| 3 | Sue  | 4    |

|   | name | rank |
|---|------|------|
| 0 | Bob  | 3    |
| 1 | Jake | 1    |
| 2 | Lisa | 4    |
| 3 | Sue  | 2    |

In [38]: `df89_on = pd.merge(df8, df9, on='name')`  
`df89_on`

Out[38]:

|   | name | rank_x | rank_y |
|---|------|--------|--------|
| 0 | Bob  | 1      | 3      |
| 1 | Jake | 2      | 1      |
| 2 | Lisa | 3      | 4      |
| 3 | Sue  | 4      | 2      |

In [39]: `df89_on_suff = pd.merge(df8, df9, on='name', suffixes=['_L', '_R'])`  
`df89_on_suff`

Out[39]:

|   | name | rank_L | rank_R |
|---|------|--------|--------|
| 0 | Bob  | 1      | 3      |
| 1 | Jake | 2      | 1      |
| 2 | Lisa | 3      | 4      |
| 3 | Sue  | 4      | 2      |

In [ ]:



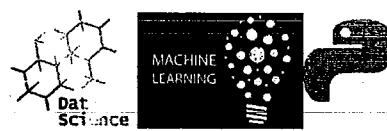
# DATA PRE-PROCESSING AND ANALYSIS

## Bài 4: Data Pre-processing – Data Standardization

Phòng LT & Mạng

[https://drive.google.com/file/d/1DwvPzXfVQmIaLJyvZDp-Preprocessing-and-Analysis\\_198](https://drive.google.com/file/d/1DwvPzXfVQmIaLJyvZDp-Preprocessing-and-Analysis_198)

2019



## Nội dung

1. Giới thiệu
2. Log normalization
3. Feature Scaling



## Giới thiệu

### □ Chuẩn hóa dữ liệu - Data Standardization

- Chuẩn hóa dữ liệu là một quy trình xử lý dữ liệu thực hiện việc chuyển đổi cấu trúc của các bộ dữ liệu khác nhau thành định dạng dữ liệu chung (Common Data Format).
- Là một phần của việc chuẩn bị dữ liệu, chuẩn hóa dữ liệu liên quan đến việc chuyển đổi các bộ dữ liệu sau khi dữ liệu được lấy từ các hệ thống nguồn và trước khi nó được tải vào các hệ thống đích.

Data Pre-processing and Analysis

## Giới thiệu

- Chuẩn hóa dữ liệu cho phép người dùng phân tích và sử dụng dữ liệu một cách nhất quán. Thông thường, khi dữ liệu được tạo và lưu trữ trong hệ thống nguồn, nó được cấu trúc theo một cách cụ thể mà người dùng không biết. Hơn nữa, các bộ dữ liệu có thể được lưu trữ và trình bày khác nhau, do đó gây khó khăn cho người dùng khi tổng hợp hoặc so sánh các bộ dữ liệu.
- Các bộ dữ liệu trong thế giới thực không chứa các tính năng rất khác nhau về cường độ, đơn vị và phạm vi. Chuẩn hóa dữ liệu nên được thực hiện khi thang đo của một tính năng không liên quan hoặc gây hiểu lầm và không nên chuẩn hóa khi thang đo có ý nghĩa.

Data Pre-processing and Analysis



## Giới thiệu

- Chuẩn hóa là cách làm cho dữ liệu của chúng ta phù hợp với các giả định và cải thiện hiệu suất của thuật toán. Đảm bảo dữ liệu phù hợp nghĩa là, mỗi loại dữ liệu có cùng loại nội dung và định dạng.
- Chuẩn hóa dữ liệu là một phần quan trọng của nghiên cứu và là thứ mà người sử dụng dữ liệu nên cân nhắc trước khi thu thập, dọn dẹp hoặc phân tích dữ liệu của họ.
- Có nhiều phương pháp được sử dụng để chuẩn hóa dữ liệu như square root, cube root, logarithmic, binning...



## Giới thiệu

### Khi nào cần chuẩn hóa dữ liệu?

- Khi xây dựng mô hình (model):
  - Mô hình trong không gian tuyến tính (linear space) Nếu sử dụng thuật toán linear regression → cần chuẩn hóa dữ liệu → tính toán. Nên L2
  - Các tính năng dữ liệu có phương sai cao
  - Các tính năng dữ liệu liên tục và trên các đơn vị đo (scale) khác nhau
  - Giả định tuyến tính



## Giới thiệu

- Nhu cầu chuyển đổi dữ liệu có thể phụ thuộc vào phương pháp mô hình hóa mà chúng ta dự định sử dụng.
  - Ví dụ, đối với linear regression và logistic regression, tốt nhất là đảm bảo rằng mỗi quan hệ giữa các biến đầu vào và biến đầu ra xấp xỉ tuyến tính, nghĩa là biến đầu vào xấp xỉ chuẩn (normal) trong phân phối và biến đầu ra là phương sai không đổi (nghĩa là phương sai của biến đầu ra không phụ thuộc vào biến đầu vào). Vì thế ta có thể cần phải chuyển đổi một số biến đầu vào của mình để đáp ứng tốt hơn cho các giả định này.

Data Pre-processing and Analysis

## Nội dung

1. Giới thiệu
2. Log normalization
3. Feature Scaling

Data Pre-processing and Analysis

## Log normalization

- Log của một biến là phương thức biến đổi phổ biến được sử dụng để thay đổi dạng phân phối của biến trên biểu đồ phân phối.
- Nó thường được sử dụng để giảm độ lệch phải (right skewness) của các biến.
- Mặc dù vậy, nó cũng có thể được áp dụng cho các giá trị 0 hoặc âm.



## Log normalization

- Áp dụng chuyển đổi log
- Log tự nhiên sử dụng hằng số e (2.718)
- Nắm bắt những thay đổi tương đối, cường độ thay đổi và giữ mọi thứ trong không gian dương (positive space)
- Chia dữ liệu vào phạm vi nhỏ hơn theo log
- Dùng cho Skewed và Wide Distribution  
dữ liệu có xu hướng



## **Log normalization**

### **☐ Nên sử dụng khi nào?**

- Dữ liệu có độ lệch dương.
- Chúng ta nghi ngờ một thành phần theo cấp số nhân trong dữ liệu.
- Dữ liệu có thể được phân loại tốt nhất theo thứ tự của độ lớn.



## **Log normalization**

### **☐ Ưu điểm**

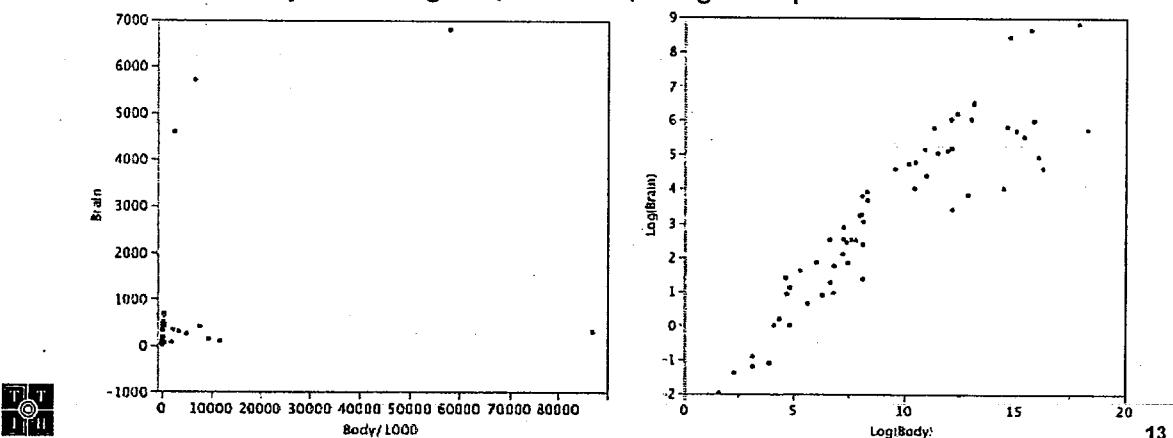
- Việc sử dụng log normalization có thể được sử dụng để làm cho các bản phân phối có độ lệch cao ít bị sai lệch. Nó cũng giúp cho các mẫu trong dữ liệu dễ hiểu hơn và giúp đáp ứng các giả định của thống kê suy luận.



## Log normalization

### • Ví dụ:

- Hình dưới cho thấy một ví dụ về cách chuyển đổi log có thể làm cho các mẫu hiển thị rõ hơn. Cả hai biểu đồ biểu thị trọng lượng não của động vật như một function của trọng lượng não. Các trọng số thô được hiển thị trong hình trái; các trọng số chuyển đổi log được hiển thị trong hình phải.



13

## Log normalization

### • Ví dụ: Sử dụng np.log() áp dụng log normalization

```
wine_sub.head()
```

|   | Alcohol | Proline | Proanthocyanins | Ash  |
|---|---------|---------|-----------------|------|
| 0 | 14.23   | 1065    | 2.29            | 2.43 |
| 1 | 13.20   | 1050    | 1.28            | 2.14 |
| 2 | 13.16   | 1185    | 2.81            | 2.67 |
| 3 | 14.37   | 1480    | 2.18            | 2.50 |
| 4 | 13.24   | 735     | 1.82            | 2.87 |

```
# dữ liệu cột cần được chuẩn hóa? Proline
print("The variance of the Alcohol column:", wine_sub["Alcohol"].var())
print("The variance of the Proline column:", wine_sub["Proline"].var())
wine_sub["Proline_log"] = np.log(wine_sub["Proline"])

# Check the variance of the Proline column again
print("The variance of the Proline column after log:", wine_sub["Proline_log"].var())
```

```
The variance of the Alcohol column: 0.6590623278105763
```

```
The variance of the Proline column: 99166.71735542428
```

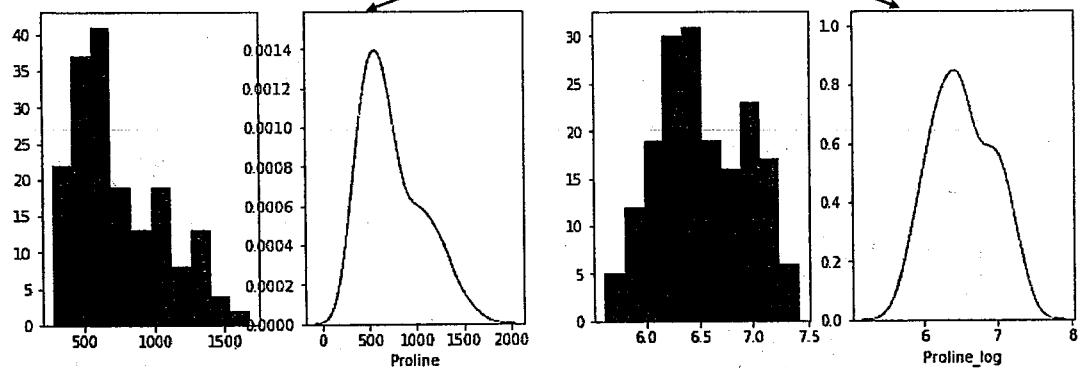
```
The variance of the Proline column after log: 0.17231366191842018
```

14

## Log normalization

uine\_sub.head()

|   | Alcohol | Proline | Proanthocyanins | Ash  | Proline_log |
|---|---------|---------|-----------------|------|-------------|
| 0 | 14.23   | 1065    |                 | 2.29 | 2.43        |
| 1 | 13.20   | 1050    |                 | 1.28 | 2.14        |
| 2 | 13.16   | 1185    |                 | 2.81 | 2.67        |
| 3 | 14.37   | 1480    |                 | 2.18 | 2.50        |
| 4 | 13.24   | 735     |                 | 1.82 | 2.87        |
|   |         |         |                 |      | 6.970730    |
|   |         |         |                 |      | 6.956545    |
|   |         |         |                 |      | 7.077498    |
|   |         |         |                 |      | 7.299797    |
|   |         |         |                 |      | 6.599870    |



Data Pre-processing and Analysis

15

## Nội dung

1. Giới thiệu
2. Log normalization
3. Feature Scaling



## Feature Scaling

### □ Feature Scaling

- Cũng là công việc chuẩn hóa dữ liệu
- Áp dụng cho các biến/ tính năng của dữ liệu.
- Về cơ bản nó giúp chuẩn hóa hóa dữ liệu trong một phạm vi cụ thể. Đôi khi, nó cũng giúp tăng tốc các tính toán trong một thuật toán.



## Feature Scaling

### □ Đặc điểm

- Các tính năng/ thuộc tính có thể ở trên các thang đo khác nhau
- Các mô hình có đặc điểm tuyến tính
- Tính năng trung tâm ở khoảng 0<sup>mean =</sup> và biến đổi thành phương sai đơn vị.()
- Chuyển sang normal distribution



Khoảng cách giữa 2 điểm:

$$1) \text{ Khoảng cách Euclidean: } d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2)

3)



## Feature Scaling

☐ Các thuật toán sử dụng thước đo Khoảng cách Euclidean rất nhạy cảm với Độ lớn. Vì vậy feature scaling giúp cân đối tất cả các tính năng như nhau. Như vậy, nếu một tính năng trong bộ dữ liệu có quy mô lớn so với các tính năng khác thì trong các thuật toán được đo bằng khoảng cách Euclidean, tính năng có tỷ lệ lớn sẽ trở nên thống trị và cần được chuẩn hóa.



Thiết kế theo xác suất  $\rightarrow$  Độ chuẩn hóa dữ liệu  
Data Pre-processing and Analysis

19

## Feature Scaling

Đối với các thuật toán này cần phải scale dữ liệu

• Ví dụ các thuật toán có thể áp dụng Feature Scaling

K-Means

▪ K-Means sử dụng đo khoảng cách Euclidean.

▪ K-Nearest-Neighbor cũng yêu cầu chuyển đổi dữ liệu. KNN

▪ PCA: Cố gắng để có được tính năng với phương sai tối đa, do đó cũng cần phải scaling.

▪ Gradient Descent: Tốc độ tính toán tăng khi tính toán  $\Theta$   $\rightarrow m, b$  trở nhanh hơn sau khi scaling.



## Feature Scaling

### ☐ Chú ý Không cần scale RC

- Naive Bayes, Linear Discriminant Analysis,

*không bị ảnh hưởng bởi feature scaling. Do đó, các thuật toán không dựa trên khoảng cách thì không cần phải áp dụng feature scaling.*

đo lường  
thuần túy  
(tính biệt)  
quan trọng  
+ Decision Tree  
+ Random Forest



## Feature Scaling

### ☐ Sử dụng thư viện: sklearn.preprocessing

- Là thư viện cung cấp một số hàm tiện ích phổ biến và các lớp biến đổi để thay đổi các vector tính năng thành biểu diễn phù hợp hơn cho các công cụ ước tính xuôi dòng.

- Một số Scaler thông dụng

- StandardScaler
- MinMaxScaler
- RobustScaler
- Normalizer

▪ ...



## Feature Scaling

### □ StandardScaler

- StandardScaler là một kỹ thuật hữu ích để biến đổi các thuộc tính có phân phối Gaussian và các giá trị trung bình (mean) & độ lệch chuẩn (std) khác nhau thành phân phối Gaussian tiêu chuẩn với giá trị trung bình là 0 và độ lệch chuẩn là 1. Và tuân theo công thức sau cho mỗi tính năng/ thuộc tính:

$$\frac{x_i - \text{mean}(x)}{\text{stdev}(x)} \rightarrow \text{gía trị sau khiScaler (StandardScaler)}$$

Data Pre-processing and Analysis

$$\frac{1,64 - 1,55}{0,03} = 3.$$

23

## Feature Scaling

- StandardScalser phù hợp nhất cho các kỹ thuật giả định phân phối Gaussian trong các biến đầu vào và hoạt động tốt hơn với dữ liệu được chuẩn hóa lại, như hồi quy tuyến tính, hồi quy logistic và phân tích phân biệt tuyến tính (linear discriminate analysis).
- Nếu dữ liệu không được phân phối chuẩn thì đây không phải là cách chia tỷ lệ tốt nhất để sử dụng.



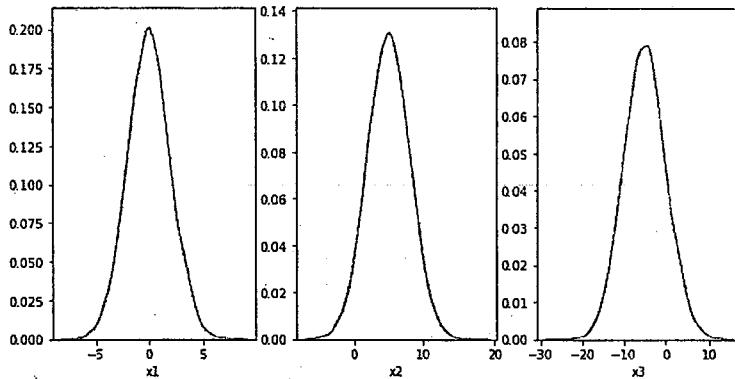
## Feature Scaling

### • Ví dụ:

```
np.random.seed(1)
df = pd.DataFrame({
    'x1': np.random.normal(0, 2, 10000),
    'x2': np.random.normal(5, 3, 10000),
    'x3': np.random.normal(-5, 5, 10000)
})
```

```
df.head()
```

|   | x1        | x2       | x3         |
|---|-----------|----------|------------|
| 0 | 3.248691  | 4.632578 | -14.657819 |
| 1 | -1.223513 | 5.684509 | -5.802131  |
| 2 | -1.056344 | 3.943085 | -9.161098  |
| 3 | -2.145937 | 2.508340 | -6.030558  |
| 4 | 1.730815  | 4.216731 | 0.131275   |



Phân phối Gausian

=> StandardScaler

Data Pre-processing and Analysis

25

## Feature Scaling

```
df2.head()
```

|   | x1        | x2       | x3         |
|---|-----------|----------|------------|
| 0 | 3.248691  | 4.632578 | -14.657819 |
| 1 | -1.223513 | 5.684509 | -5.802131  |
| 2 | -1.056344 | 3.943085 | -9.161098  |
| 3 | -2.145937 | 2.508340 | -6.030558  |
| 4 | 1.730815  | 4.216731 | 0.131275   |

```
scaler = preprocessing.StandardScaler()
```

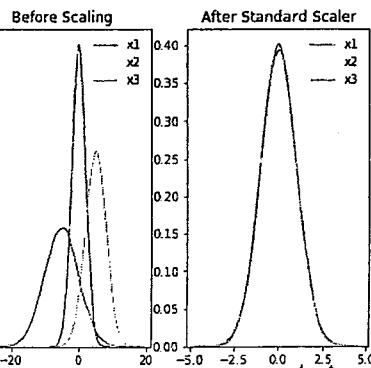
```
scaled_df = scaler.fit_transform(df2) → hàm biến đổi dữ liệu
```

```
scaled_df = pd.DataFrame(scaled_df, columns=['x1', 'x2', 'x3'])
```

```
scaled_df.head()
```

|   | x1        | x2        | x3        |
|---|-----------|-----------|-----------|
| 0 | 1.616535  | -0.131753 | -1.924757 |
| 1 | -0.622285 | 0.218475  | -0.146416 |
| 2 | -0.538598 | -0.361311 | -0.820942 |
| 3 | -1.084057 | -0.838991 | -0.192287 |
| 4 | 0.856675  | -0.270204 | 1.045092  |

thực hiện fit và điểm vào



Scalor fit: đưa data (data-train) → để dữ liệu TO

transform train new: *Scalor*.transform (data-train)

test new: *Scalor*.transform (data-test)

→ hàn lâm Scalor này.

New → chỉ định báu cho tập dữ liệu mới → *Scalor*.transform(new) → new -Scalor → predict

## Feature Scaling

### ◻ MinMaxScaler

- MinMaxScaler có thể được xem là thuật toán chia tỷ lệ nỗi tiếng nhất và tuân theo công thức sau cho mỗi tính năng/ thuộc tính:

$$\frac{x_i - \min(x)}{\max(x) - \min(x)} = \frac{165 - 160}{180 - 160} = \frac{5}{21} = 0.24$$

Nam (chiều cao TB).

- Về cơ bản, nó thu hẹp phạm vi sao cho phạm vi mới nằm trong khoảng từ 0 đến 1 (hoặc -1 đến 1 nếu có các giá trị âm).



## Feature Scaling

nhược điểm

- Bộ chia tỷ lệ này hoạt động tốt hơn trong các trường hợp mà bộ chia tỷ lệ tiêu chuẩn (standard scaler) có thể không hoạt động tốt.
- Nếu phân phối không phải là Gaussian hoặc độ lệch chuẩn là rất nhỏ, min-max scaler hoạt động tốt hơn.
- Tuy nhiên, nó rất nhạy cảm với các ngoại lệ (outlier), vì vậy nếu có các ngoại lệ trong dữ liệu, chúng ta có thể áp dụng Robust Scaler.



## Feature Scaling

- Động lực để sử dụng tỷ lệ này bao gồm độ lệch chuẩn của các tính năng rất nhỏ và duy trì các entry có giá trị 0 trong dữ liệu thưa thớt.

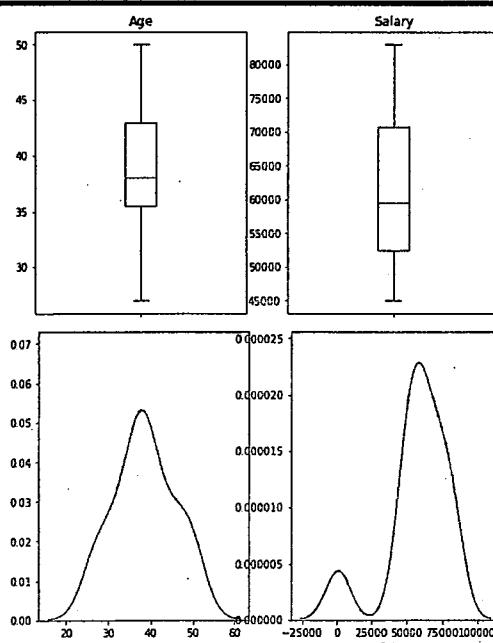


## Feature Scaling

- Ví dụ:

x

|   | Age  | Salary  |
|---|------|---------|
| 0 | 44.0 | 72000.0 |
| 1 | 27.0 | 48000.0 |
| 2 | 30.0 | 54000.0 |
| 3 | 38.0 | 61000.0 |
| 4 | 40.0 | 45000.0 |
| 5 | 35.0 | 58000.0 |
| 6 | 38.0 | 52000.0 |
| 7 | 48.0 | 79000.0 |
| 8 | 50.0 | 83000.0 |
| 9 | 37.0 | 67000.0 |



- Phân phối không phải là Gausian  
- Không có outlier  
MinMax Scaler



## Feature Scaling

### • Ví dụ:

X

```
Age   Salary  from sklearn import preprocessing  
0  44.0  72000.0  min_max_scaler = preprocessing.MinMaxScaler() # feature_range =(0, 1)  
1  27.0  48000.0  X_after_min_max_scaler = min_max_scaler.fit_transform(X)  
2  30.0  54000.0  X_after_min_max_scaler = pd.DataFrame(X_after_min_max_scaler, columns=['Age', 'Salary'])  
3  38.0  61000.0  print("After min max Scaling :")  
4  40.0  45000.0  After min max Scaling :  
5  35.0  58000.0  X_after_min_max_scaler  
6  38.0  52000.0  
7  48.0  79000.0  
8  50.0  83000.0  
9  37.0  67000.0
```

Age Salary

|   |          |          |
|---|----------|----------|
| 0 | 0.739130 | 0.7:0526 |
| 1 | 0.000000 | 0.078947 |
| 2 | 0.130435 | 0.236842 |
| 3 | 0.478261 | 0.421053 |
| 4 | 0.565217 | 0.000000 |
| 5 | 0.347826 | 0.342105 |
| 6 | 0.478261 | 0.184211 |
| 7 | 0.913043 | 0.894737 |
| 8 | 1.000000 | 1.000000 |
| 9 | 0.434783 | 0.578947 |

Data Pre-processing and Analysis

31

## Feature Scaling

### □ Robust Scaler

- RobustScaler sử dụng một phương pháp tương tự như Min-Max Scaler nhưng nó sử dụng interquartile range thay cho min-max, do đó nó vượt trội hơn. Tuân theo công thức sau cho mỗi tính năng/đặc tính:

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

                    
                    
IQR

(dùng IQR để robust hơn, không bị ảnh hưởng bởi outliers)



## Feature Scaling

- Dĩ nhiên, điều này có nghĩa là RobustScaler đang sử dụng ít dữ liệu hơn khi chia tỷ lệ để nó phù hợp hơn khi có các ngoại lệ trong dữ liệu.
- Chú ý: sau khi áp dụng Robust scaling, các phân phối được dựa vào cùng một tỷ lệ và trùng lặp, nhưng các ngoại lệ vẫn nằm ngoài phần lớn của các bản phân phối mới.



## Feature Scaling

### Normalizer (dùng cho dữ liệu thừa thớt)

- Normalizer định cỡ lại mỗi quan sát (hàng) để có độ dài 1 (được gọi là unit norm trong đại số tuyến tính).

VD:  
- Tên bài hát xuất hiện  
- Thời gian nào đó trong ngày  
- Điểm rating cho phim

- Quá trình tiền xử lý này có thể hữu ích cho các bộ dữ liệu thừa thớt (chứa nhiều 0) với các thuộc tính có tỷ lệ khác nhau khi sử dụng các thuật toán có trọng số đầu vào như Neuron network và thuật toán sử dụng các thước đo khoảng cách như K-Nearest Neighbors.



## Feature Scaling

- Normalizer chia tỷ lệ từng giá trị bằng cách chia mỗi giá trị cho độ lớn của nó trong không gian n chiều cho n tính năng.
- Giả sử bộ dữ liệu có ba tính năng là x, y và z. Cartesian phối hợp giá trị tỷ lệ cho x sẽ là:

$$\frac{x_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}$$

VD: hôm nay trời có nhiều mây  
hôm nay đường không thoáng  
Sáng nay trời rất ấm áp

Data Pre-processing and Analysis

|   | 0    | 1    | 2    | 3    |
|---|------|------|------|------|
| 0 | 1/16 | 1/16 | 1/16 | 1/16 |
| 1 | 1/16 | 1/16 | 1/16 | 1/16 |
| 2 | 1/16 | 1/16 | 1/16 | 1/16 |

|   | Thời gian | Điểm | Độ cao | Độ cao |
|---|-----------|------|--------|--------|
| 0 |           | 1/16 |        |        |
| 1 |           | 1/16 | 2/16   | 2/16   |

$$\frac{1}{\sqrt{1^2 + 1^2 + 1^2 + 2^2}} = \frac{1}{\sqrt{6}}$$

### Feature Scaling

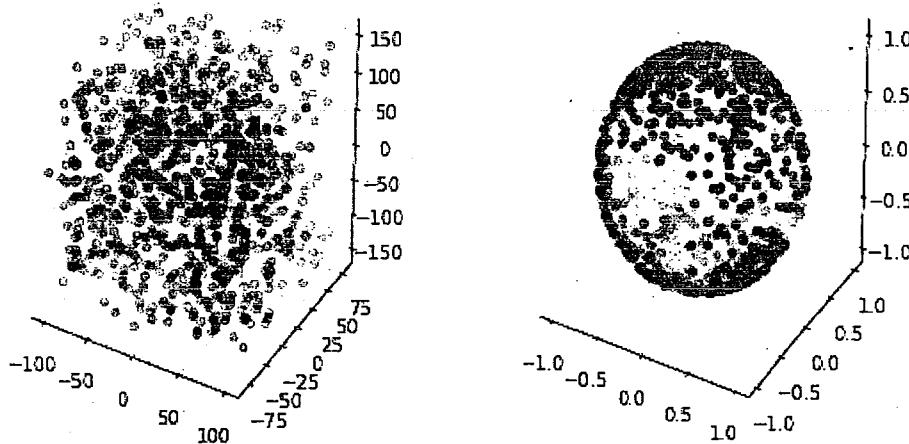
#### • Ví dụ

```
df4 = pd.DataFrame({
    'x1': np.random.randint(-100, 100, 1000).astype(float),
    'y1': np.random.randint(-80, 80, 1000).astype(float),
    'z1': np.random.randint(-150, 150, 1000).astype(float),
})
df4.head()
    # Mở file dữ liệu
    # Đầu file đầu tiên
    # Mua hàng (chỉ mua 1 Vai áo)
    # Văn bản
scaler = preprocessing.Normalizer()
scaled_df = scaler.fit_transform(df4)
scaled_df = pd.DataFrame(scaled_df, columns=df4.columns)
scaled_df.head()
```

|   | x1    | y1    | z1     |
|---|-------|-------|--------|
| 0 | 31.0  | 26.0  | -20.0  |
| 1 | -82.0 | 17.0  | -57.0  |
| 2 | -18.0 | -57.0 | 88.0   |
| 3 | 79.0  | 38.0  | -104.0 |
| 4 | -62.0 | 23.0  | -94.0  |

|   | x1        | y1        | z1        |
|---|-----------|-----------|-----------|
| 0 | 0.686857  | 0.576073  | -0.443133 |
| 1 | -0.809465 | 0.167816  | -0.562677 |
| 2 | -0.169203 | -0.535808 | 0.827212  |
| 3 | 0.580804  | 0.279374  | -0.764603 |
| 4 | -0.539457 | 0.200121  | -0.817886 |

## Feature Scaling



Lưu ý rằng tất cả các điểm được đưa vào trong một hình cầu cách tối đa 1 so với điểm gốc. Ngoài ra, các trục trước đây là các thang đo khác nhau, còn bây giờ đều là cùng một thang đo.

## Feature Scaling

### Binarizer

- Ví: ~~TB 2 lít/suất~~  
Khi ~~4/1000 /kg~~  
 $\rightarrow$  ~~đơn vị lượng~~  $\rightarrow$  ~~khối~~  
 $\rightarrow$  ~~đơn vị khối~~  $\rightarrow$   $\checkmark$  ①
- Có thể chuyển đổi dữ liệu bằng cách dùng binary threshold. Tất cả các giá trị trên threshold được thay bằng 1 và các giá trị  $\leq$  threshold được thay bằng 0.
  - Nó có thể hữu ích khi chúng ta có các xác suất mà ta muốn tạo ra các giá trị rõ nét.
  - Nó cũng hữu ích khi feature engineering và chúng ta muốn thêm các tính năng mới có ý nghĩa.

## Feature Scaling

### • Ví dụ:

- Ví dụ 1: Dữ liệu liên tục của các giá trị pixel của ảnh với grayscale 8-bit có các giá trị nằm trong khoảng từ 0 (đen) đến 255 (trắng) và người ta cần có màu đen và trắng. Vì vậy, bằng cách sử dụng Binarizer (), người ta có thể đặt ngưỡng chuyển đổi các giá trị pixel từ 0 - 127 thành 0 và 128 - 255 thành 1.
- Ví dụ 2: Một bản ghi của một máy có thuộc tính "Success Percentage". Các giá trị này liên tục nằm trong khoảng từ 10% đến 99% nhưng một nhà nghiên cứu chỉ muốn sử dụng dữ liệu này để dự đoán tình trạng vượt qua hoặc thất bại cho máy dựa trên các tham số đã cho khác.



## Feature Scaling

### • Ví dụ:

```
Y = [[ 1., -1.,  2.],
      [ 2.,  0.,  0.],
      [ 0.,  1., -1.],
      [ 3., -2.,  1. ]]

scaler = preprocessing.Binarizer(threshold=0)
scaled_Y = scaler.fit_transform(Y)
scaled_Y

array([[1., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 1.]])
```





Data Pre-processing and Analysis

41



## Chapter 4

### Ex1: Standardizing Data

**Cho dữ liệu mammals.csv chứa thông tin về mammals.**

- Phân tích thông tin sơ bộ về dữ liệu trên hai thuộc tính BrainWt, BodyWt, xem xét mối quan hệ của 2 thuộc tính này. Trực quan hóa dữ liệu.
- Để dự đoán BrainWt dựa trên BodyWt cần phải kiểm tra và chuẩn hóa dữ liệu. Hãy chọn một phương pháp để chuẩn hóa dữ liệu dựa trên thông tin nêu trên. Trực quan hóa dữ liệu sau khi chuẩn hóa.

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
data = pd.read_csv("mammals.csv", index_col=0)
data.head()
```

Out[2]:

|   | Species               | BodyWt    | BrainWt | NonDreaming | Dreaming | TotalSleep | LifeSpan | Gestation |
|---|-----------------------|-----------|---------|-------------|----------|------------|----------|-----------|
| 1 | Africanelephant       | 6654.0000 | 5712.0  | NaN         | NaN      | 3.3        | 38.6     | 645       |
| 2 | Africagiantpouchedrat | 1.000     | 6.6     | 6.3         | 2.0      | 8.3        | 4.5      | 42        |
| 3 | ArcticFox             | 3.385     | 44.5    | NaN         | NaN      | 12.5       | 14.0     | 60        |
| 4 | Arcticgroundsquirrel  | 0.920     | 5.7     | NaN         | NaN      | 16.5       | NaN      | 25        |
| 5 | Asianelephant         | 2547.0000 | 4603.0  | 2.1         | 1.8      | 3.9        | 69.0     | 624       |

In [3]:

```
data.describe()
```

Out[3]:

|       | BodyWt      | BrainWt     | NonDreaming | Dreaming  | TotalSleep | LifeSpan   | Gestation  | Pre  |
|-------|-------------|-------------|-------------|-----------|------------|------------|------------|------|
| count | 62.000000   | 62.000000   | 48.000000   | 50.000000 | 58.000000  | 58.000000  | 58.000000  | 62.0 |
| mean  | 198.789984  | 283.134194  | 8.672917    | 1.972000  | 10.532759  | 19.877586  | 142.353448 | 2.0  |
| std   | 899.158011  | 930.278942  | 3.666452    | 1.442651  | 4.606760   | 18.206255  | 146.805039 | 1.4  |
| min   | 0.005000    | 0.140000    | 2.100000    | 0.000000  | 2.600000   | 2.000000   | 12.000000  | 1.0  |
| 25%   | 0.600000    | 4.250000    | 6.250000    | 0.900000  | 8.050000   | 6.625000   | 35.750000  | 2.0  |
| 50%   | 3.342500    | 17.250000   | 8.350000    | 1.800000  | 10.450000  | 15.100000  | 79.000000  | 3.0  |
| 75%   | 48.202500   | 166.000000  | 11.000000   | 2.550000  | 13.200000  | 27.750000  | 207.500000 | 4.0  |
| max   | 6654.000000 | 5712.000000 | 17.900000   | 6.600000  | 19.900000  | 100.000000 | 645.000000 | 5.0  |



```
In [4]: body_range = data.BodyWt.ptp()  
body_range
```

```
Out[4]: 6653.995
```

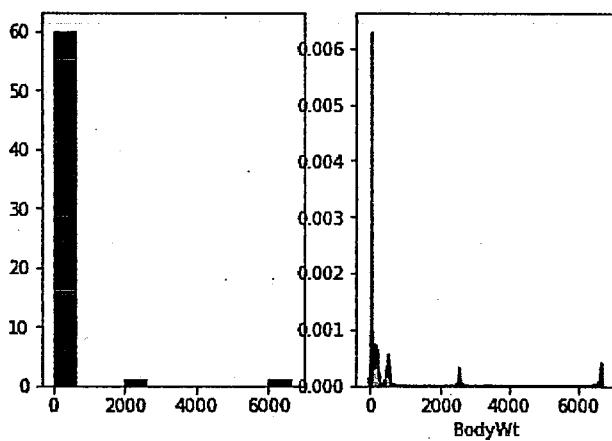
```
In [5]: brain_range = data.BrainWt.ptp()  
brain_range
```

```
Out[5]: 5711.86
```

```
In [6]: # Có một khoảng Lớn giữa min và max
```

```
In [7]: import matplotlib.pyplot as plt  
import seaborn as sns
```

```
In [8]: plt.subplot(1,2,1)  
plt.hist(data.BodyWt)  
plt.subplot(1,2,2)  
sns.distplot(data.BodyWt)  
plt.show()
```

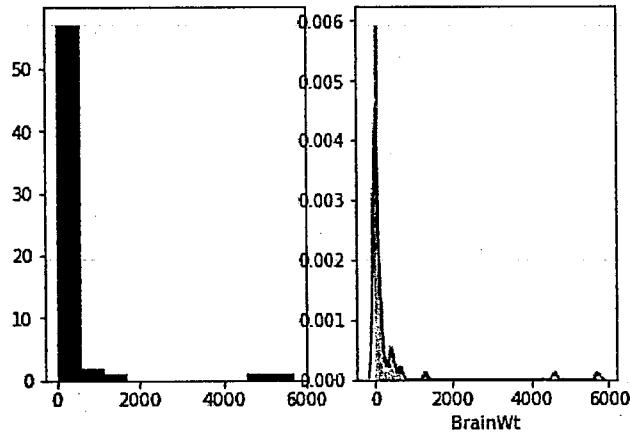


```
In [9]: data.BodyWt.skew() # phân phối Lệch phái
```

```
Out[9]: 6.563608062833757
```



```
In [10]: plt.subplot(1,2,1)
plt.hist(data.BrainWt)
plt.subplot(1,2,2)
sns.distplot(data.BrainWt)
plt.show()
```

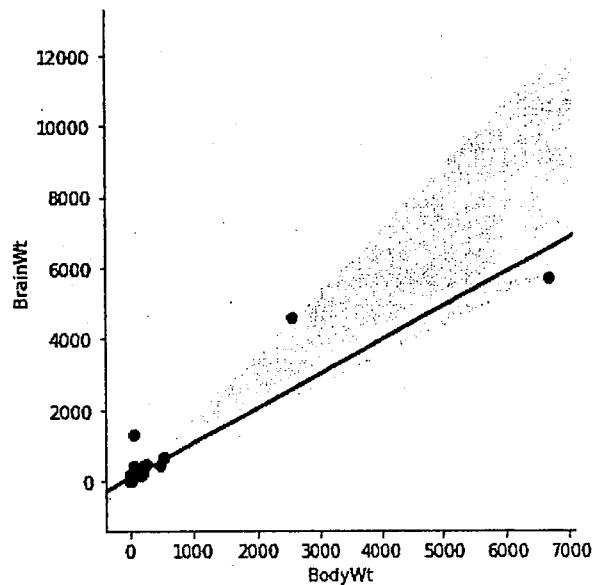


```
In [11]: data.BrainWt.skew() # phân phối Lệch phải
```

Out[11]: 5.071589456939673

```
In [12]: # Xem xét mối quan hệ
sns.lmplot(data=data, x='BodyWt', y='BrainWt')
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0x21520687e48>



```
In [13]: # Quan hệ tuyến tính
# Chọn phương pháp chuẩn hóa Là log normalization
```

7/24/2019

Ex1



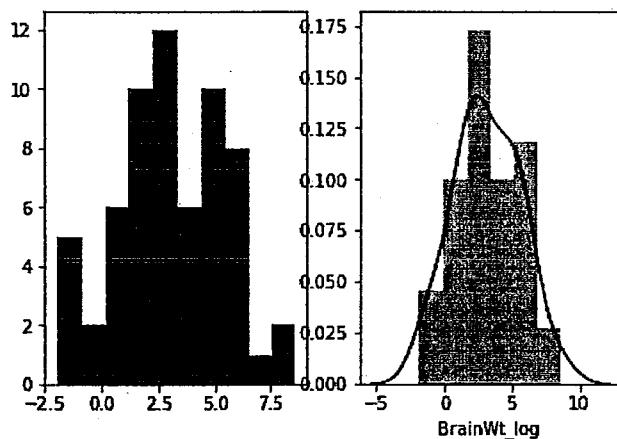
```
In [14]: data['BodyWt_log'] = np.log(data.BodyWt)
data['BrainWt_log'] = np.log(data.BrainWt)
```

```
In [15]: data.head()
```

Out[15]:

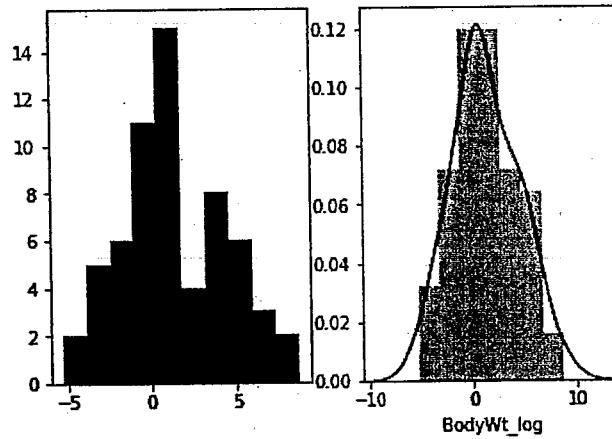
|   | Species                | BodyWt   | BrainWt | NonDreaming | Dreaming | TotalSleep | LifeSpan | Gestatic |
|---|------------------------|----------|---------|-------------|----------|------------|----------|----------|
| 1 | Africanelephant        | 6654.000 | 5712.0  | NaN         | NaN      | 3.3        | 38.6     | 645      |
| 2 | Africangiantpouchedrat | 1.000    | 6.6     | 6.3         | 2.0      | 8.3        | 4.5      | 42       |
| 3 | ArcticFox              | 3.385    | 44.5    | NaN         | NaN      | 12.5       | 14.0     | 60       |
| 4 | Arcticgroundsquirrel   | 0.920    | 5.7     | NaN         | NaN      | 16.5       | NaN      | 25       |
| 5 | Asianelephant          | 2547.000 | 4603.0  | 2.1         | 1.8      | 3.9        | 69.0     | 624      |

```
In [16]: plt.subplot(1,2,1)
plt.hist(data.BrainWt_log)
plt.subplot(1,2,2)
sns.distplot(data.BrainWt_log)
plt.show()
```



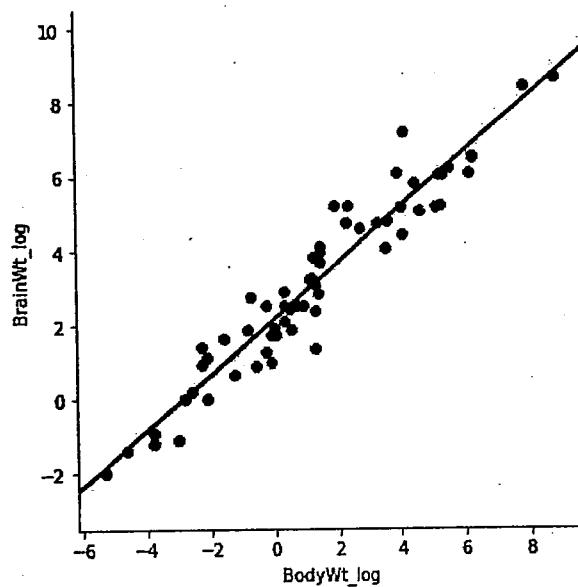


```
In [17]: plt.subplot(1,2,1)
plt.hist(data.BodyWt_log)
plt.subplot(1,2,2)
sns.distplot(data.BodyWt_log)
plt.show()
```



```
In [18]: # Xem xét mối quan hệ
sns.lmplot(data=data, x='BodyWt_log', y='BrainWt_log')
```

```
Out[18]: <seaborn.axisgrid.FacetGrid at 0x215334ef470>
```



```
In [ ]:
```



## Ex 2: Feature Scaling

**Cho dữ liệu titanic3.csv chứa thông tin về các hành khách trên con tàu Titanic**

- Một trong những thông tin quan trọng để dự đoán một hành khách còn sống hay đã chết là 'age', 'fare'. Kiểm tra xem dữ liệu trên 2 cột này có null hay không, nếu có hãy xóa bỏ các dòng null. Phân tích thông tin sơ bộ về dữ liệu trên hai thuộc tính này. Trực quan hóa dữ liệu.
- Để việc dự đoán tốt hơn cần phải kiểm tra và chuẩn hóa dữ liệu. Hãy chọn một phương pháp để chuẩn hóa dữ liệu dựa trên thông tin nêu trên.

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: data = pd.read_csv("titanic3.csv")
data.head()
```

Out[2]:

|   | pclass | survived | name   | sex    | age     | sibsp | parch | ticket | fare     | cabin      | embarked |
|---|--------|----------|--|--------|---------|-------|-------|--------|----------|------------|----------|
| 0 | 1      | 1        | Allen,<br>Miss.<br>Elisabeth<br>Walton                         | female | 29.0000 | 0     | 0     | 24160  | 211.3375 | B5         | S        |
| 1 | 1      | 1        | Allison,<br>Master.<br>Hudson<br>Trevor                        | male   | 0.9167  | 1     | 2     | 113781 | 151.5500 | C22<br>C26 | S        |
| 2 | 1      | 0        | Allison,<br>Miss.<br>Helen<br>Lorraine                         | female | 2.0000  | 1     | 2     | 113781 | 151.5500 | C22<br>C26 | S        |
| 3 | 1      | 0        | Allison,<br>Mr.<br>Hudson<br>Joshua<br>Creighton               | male   | 30.0000 | 1     | 2     | 113781 | 151.5500 | C22<br>C26 | S        |
| 4 | 1      | 0        | Allison,<br>Mrs.<br>Hudson J<br>C (Bessie<br>Waldo<br>Daniels) | female | 25.0000 | 1     | 2     | 113781 | 151.5500 | C22<br>C26 | S        |

7/24/2019

Ex2



In [3]: `data = data[['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']]  
data.head()`

Out[3]:

|   | pclass | sex    | age     | sibsp | parch | fare     | embarked |
|---|--------|--------|---------|-------|-------|----------|----------|
| 0 | 1      | female | 29.0000 | 0     | 0     | 211.3375 | S        |
| 1 | 1      | male   | 0.9167  | 1     | 2     | 151.5500 | S        |
| 2 | 1      | female | 2.0000  | 1     | 2     | 151.5500 | S        |
| 3 | 1      | male   | 30.0000 | 1     | 2     | 151.5500 | S        |
| 4 | 1      | female | 25.0000 | 1     | 2     | 151.5500 | S        |

In [4]: `data = data.dropna()`

In [5]: `data.describe()`

Out[5]:

|       | pclass      | age         | sibsp       | parch       | fare        |
|-------|-------------|-------------|-------------|-------------|-------------|
| count | 1043.000000 | 1043.000000 | 1043.000000 | 1043.000000 | 1043.000000 |
| mean  | 2.209012    | 29.813199   | 0.504314    | 0.421860    | 36.603024   |
| std   | 0.840685    | 14.366261   | 0.913080    | 0.840655    | 55.753648   |
| min   | 1.000000    | 0.166700    | 0.000000    | 0.000000    | 0.000000    |
| 25%   | 1.000000    | 21.000000   | 0.000000    | 0.000000    | 8.050000    |
| 50%   | 2.000000    | 28.000000   | 0.000000    | 0.000000    | 15.750000   |
| 75%   | 3.000000    | 39.000000   | 1.000000    | 1.000000    | 35.077100   |
| max   | 3.000000    | 80.000000   | 8.000000    | 6.000000    | 512.329200  |

In [6]: `data.age.ptp()`

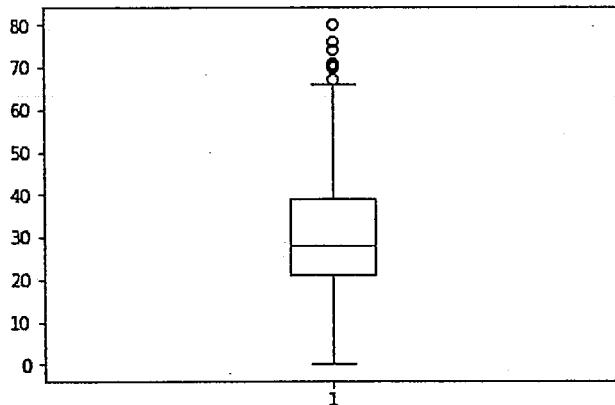
Out[6]: 79.8333

In [7]: `data.fare.ptp()`

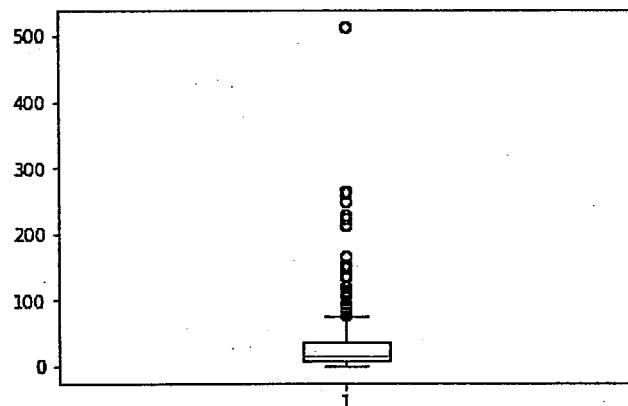
Out[7]: 512.3292



```
In [8]: plt.boxplot(data.age)  
plt.show()
```



```
In [9]: plt.boxplot(data.fare)  
plt.show()
```



```
In [10]: # Có khoảng cách Lớn giữa min và max  
# 2 thang đo cho 2 cột khác nhau
```

```
In [11]: data.age.skew()
```

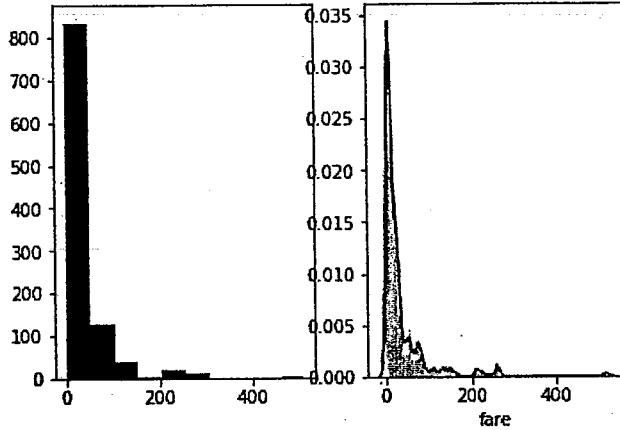
```
Out[11]: 0.40688028266803467
```

```
In [12]: data.fare.skew()
```

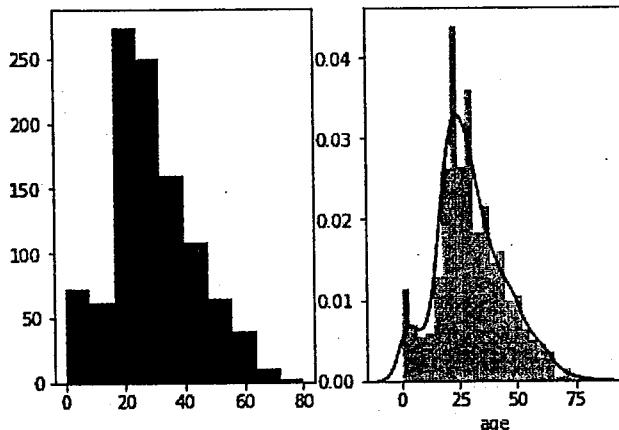
```
Out[12]: 4.122508729348891
```



```
In [13]: plt.subplot(1,2,1)
plt.hist(data.fare)
plt.subplot(1,2,2)
sns.distplot(data.fare)
plt.show()
```



```
In [14]: plt.subplot(1,2,1)
plt.hist(data.age)
plt.subplot(1,2,2)
sns.distplot(data.age)
plt.show()
```



```
In [15]: # Nhìn biểu đồ trên và giá trị skew ta thấy age tương đối chuẩn, còn fare Lệch phâ
# Có outlier
# => Chọn Robust Scaler ??? Standard Scaler ??? => Sẽ kiểm chứng khi áp dụng mô hình
```

```
In [16]: from sklearn import preprocessing
```



```
In [17]: age_fare = data[['age', 'fare']].astype('float64')
age_fare.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1043 entries, 0 to 1308
Data columns (total 2 columns):
age      1043 non-null float64
fare     1043 non-null float64
dtypes: float64(2)
memory usage: 24.4 KB
```

## Robust Scaler

```
In [18]: scaler = preprocessing.RobustScaler()
robust_scaler = scaler.fit_transform(age_fare)
df = pd.DataFrame(robust_scaler, columns=['age_scaler', 'fare_scaler'])
df.head()
```

Out[18]:

|   | age_scaler | fare_scaler |
|---|------------|-------------|
| 0 | 0.055556   | 7.236718    |
| 1 | -1.504628  | 5.024586    |
| 2 | -1.444444  | 5.024586    |
| 3 | 0.111111   | 5.024586    |
| 4 | -0.166667  | 5.024586    |

In [19]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1043 entries, 0 to 1042
Data columns (total 2 columns):
age_scaler    1043 non-null float64
fare_scaler   1043 non-null float64
dtypes: float64(2)
memory usage: 16.4 KB
```

```
In [20]: data['age_scaler'] = df.age_scaler.values
data['fare_scaler'] = df.fare_scaler.values
data.head()
```

Out[20]:

|   | pclass | sex    | age     | sibsp | parch | fare     | embarked | age_scaler | fare_scaler |
|---|--------|--------|---------|-------|-------|----------|----------|------------|-------------|
| 0 | 1      | female | 29.0000 | 0     | 0     | 211.3375 | S        | 0.055556   | 7.236718    |
| 1 | 1      | male   | 0.9167  | 1     | 2     | 151.5500 | S        | -1.504628  | 5.024586    |
| 2 | 1      | female | 2.0000  | 1     | 2     | 151.5500 | S        | -1.444444  | 5.024586    |
| 3 | 1      | male   | 30.0000 | 1     | 2     | 151.5500 | S        | 0.111111   | 5.024586    |
| 4 | 1      | female | 25.0000 | 1     | 2     | 151.5500 | S        | -0.166667  | 5.024586    |



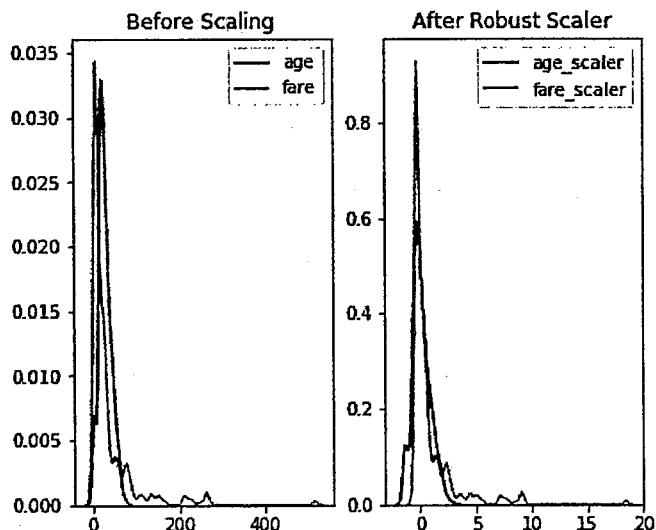
In [21]: `data.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1043 entries, 0 to 1308
Data columns (total 9 columns):
pclass      1043 non-null int64
sex         1043 non-null object
age         1043 non-null float64
sibsp       1043 non-null int64
parch       1043 non-null int64
fare         1043 non-null float64
embarked    1043 non-null object
age_scaler   1043 non-null float64
fare_scaler  1043 non-null float64
dtypes: float64(4), int64(3), object(2)
memory usage: 81.5+ KB
```

In [22]: `fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6, 5))
ax1.set_title('Before Scaling')
sns.kdeplot(data['age'], ax=ax1)
sns.kdeplot(data['fare'], ax=ax1)

ax2.set_title('After Robust Scaler')
sns.kdeplot(data['age_scaler'], ax=ax2)
sns.kdeplot(data['fare_scaler'], ax=ax2)

plt.show()`



In [23]: `### Standard Scaler` → Nhập chuẩn hóa dữ liệu trước khi train.

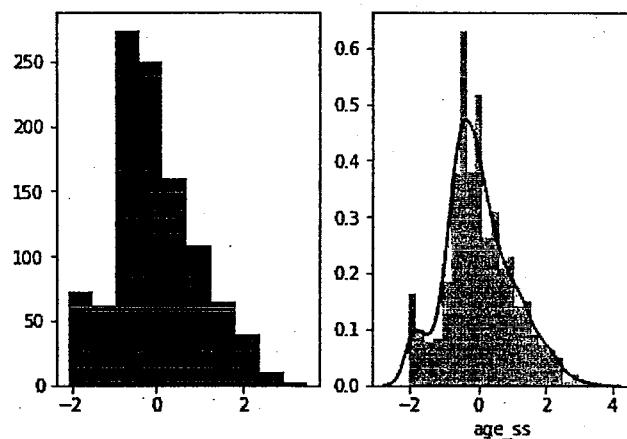


```
In [24]: scaler1 = preprocessing.StandardScaler()
ss_scaler = scaler1.fit_transform(data[['age', 'fare']].astype('float64'))
df1 = pd.DataFrame(ss_scaler, columns=['age_ss', 'fare_ss'])
df1.head()
```

Out[24]:

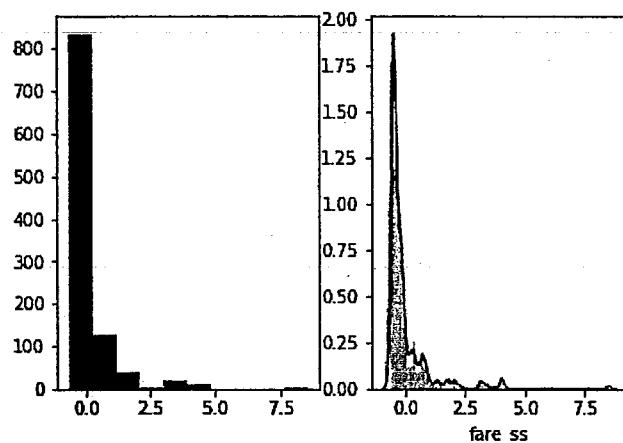
|   | age_ss    | fare_ss  |
|---|-----------|----------|
| 0 | -0.056632 | 3.135549 |
| 1 | -2.012379 | 2.062683 |
| 2 | -1.936937 | 2.062683 |
| 3 | 0.013009  | 2.062683 |
| 4 | -0.335196 | 2.062683 |

```
In [25]: plt.subplot(1,2,1)
plt.hist(df1.age_ss)
plt.subplot(1,2,2)
sns.distplot(df1.age_ss)
plt.show()
```





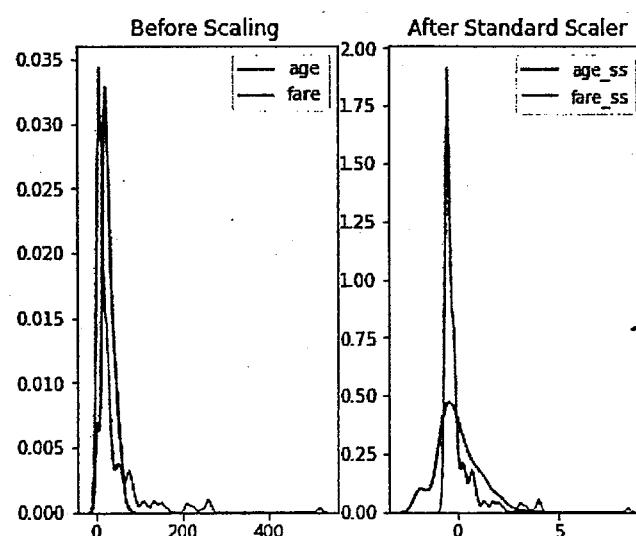
```
In [26]: plt.subplot(1,2,1)
plt.hist(df1.fare_ss)
plt.subplot(1,2,2)
sns.distplot(df1.fare_ss)
plt.show()
```



```
In [27]: fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6, 5))
ax1.set_title('Before Scaling')
sns.kdeplot(data['age'], ax=ax1)
sns.kdeplot(data['fare'], ax=ax1)

ax2.set_title('After Standard Scaler')
sns.kdeplot(df1['age_ss'], ax=ax2)
sns.kdeplot(df1['fare_ss'], ax=ax2)

plt.show()
```



```
In [28]: # Link tham khảo: https://jorisvandenbossche.github.io/blog/2018/05/28/scikit-Lear
```



## Ex3: Feature Scaling

**Cho dữ liệu trong thư mục loan\_prediction-1 chứa thông tin giao dịch cho vay của một ngân hàng**

- Trong phạm vi bài này, chúng ta chỉ xem xét tập tin X\_train.csv và Y\_train, dùng để huấn luyện mô hình dự đoán cho vay hay không cho vay. Phân tích thông tin sơ bộ về dữ liệu X\_train (với các dữ liệu numeric như: ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term, Credit\_History). Trực quan hóa dữ liệu.
- Để việc dự đoán tốt hơn cần phải kiểm tra và chuẩn hóa dữ liệu. Hãy chọn một phương pháp để chuẩn hóa dữ liệu dựa trên thông tin nêu trên.

In [1]: `# Link tham khảo: https://www.analyticsvidhya.com/blog/2016/07/practical-guide-data-mining/`

In [2]: `import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns`

In [3]: `train_data = pd.read_csv("loan_prediction-1/X_train.csv")  
train_data.head()`

Out[3]:

|   | Loan_ID  | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coapplica |
|---|----------|--------|---------|------------|-----------|---------------|-----------------|-----------|
| 0 | LP001032 | Male   | No      | 0          | Graduate  | No            | 4950            |           |
| 1 | LP001824 | Male   | Yes     | 1          | Graduate  | No            | 2882            |           |
| 2 | LP002928 | Male   | Yes     | 0          | Graduate  | No            | 3000            |           |
| 3 | LP001814 | Male   | Yes     | 2          | Graduate  | No            | 9703            |           |
| 4 | LP002244 | Male   | Yes     | 0          | Graduate  | No            | 2333            |           |

In [4]: `train_data_sub = train_data[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',  
train_data_sub.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 384 entries, 0 to 383
Data columns (total 5 columns):
ApplicantIncome    384 non-null int64
CoapplicantIncome  384 non-null float64
LoanAmount         384 non-null int64
Loan_Amount_Term   384 non-null int64
Credit_History     384 non-null int64
dtypes: float64(1), int64(4)
memory usage: 15.1 KB
```

In [5]: # không có dữ liệu null

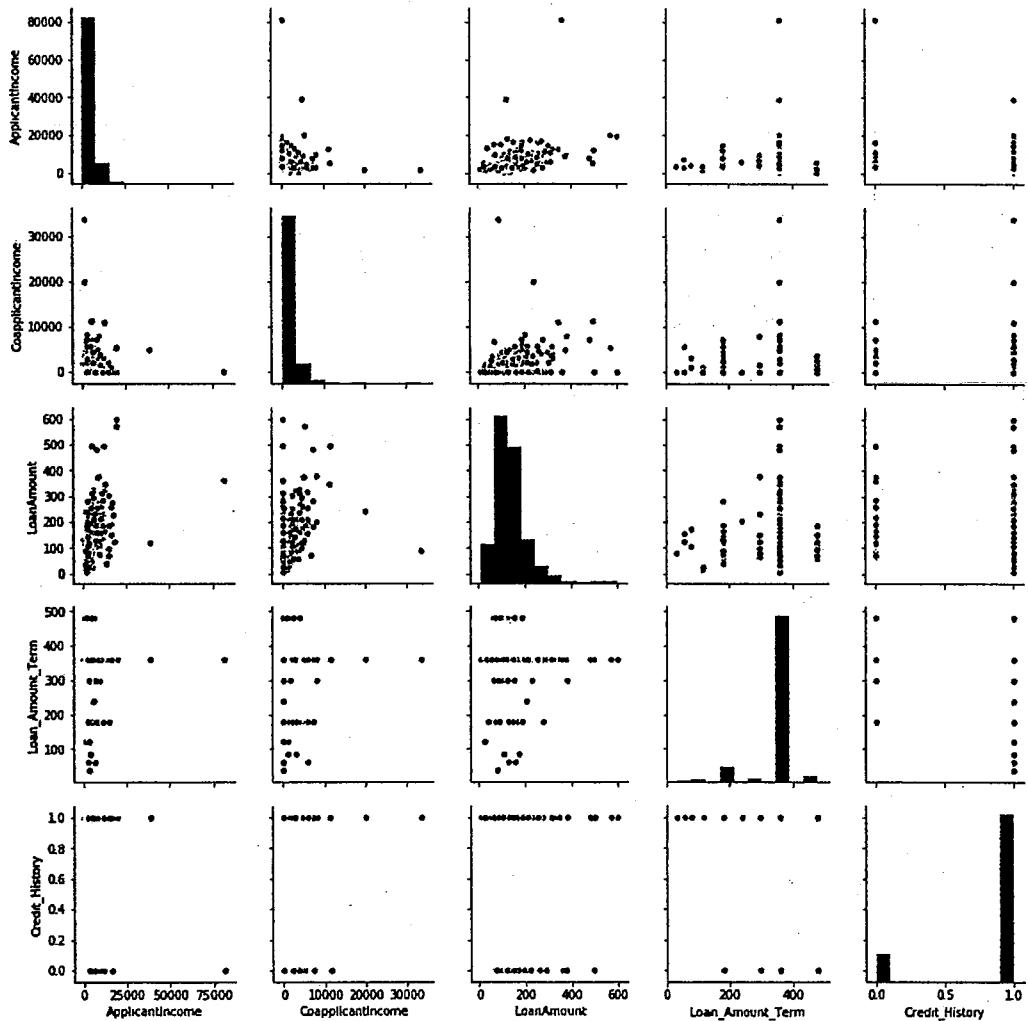
In [6]: # Credit\_History là dữ liệu nhị phân nên không cần chuẩn hóa

# Loan\_Amount\_Term: đơn vị là tháng

# 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount' với đơn vị là USD

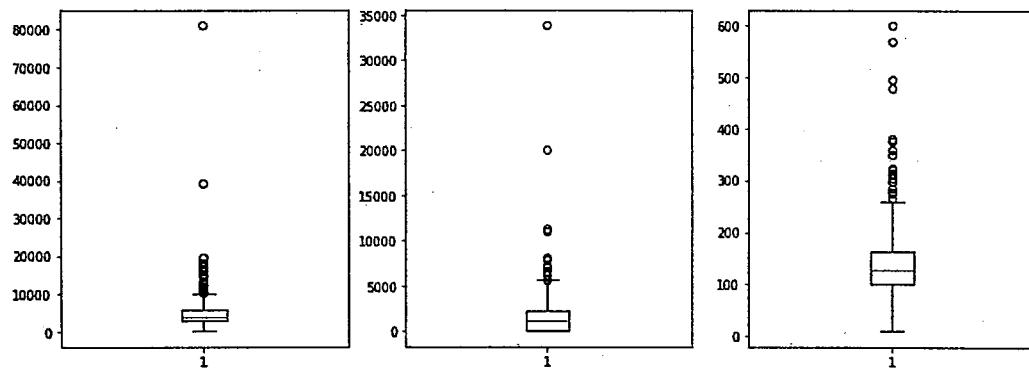
In [7]: sns.pairplot(train\_data\_sub)

Out[7]: <seaborn.axisgrid.PairGrid at 0x1dcb0257e48>

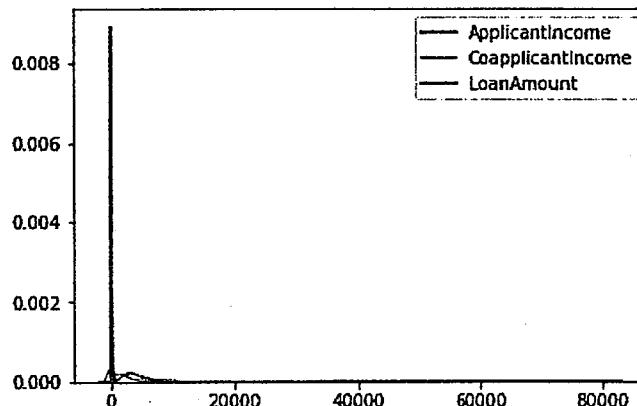




```
In [8]: plt.figure(figsize=(14,5))
plt.subplot(1,3,1)
plt.boxplot(train_data_sub.ApplicantIncome)
plt.subplot(1,3,2)
plt.boxplot(train_data_sub.CoapplicantIncome)
plt.subplot(1,3,3)
plt.boxplot(train_data_sub.LoanAmount)
plt.show()
```



```
In [9]: sns.kdeplot(train_data_sub.ApplicantIncome)
sns.kdeplot(train_data_sub.CoapplicantIncome)
sns.kdeplot(train_data_sub.LoanAmount)
plt.show()
```



In [10]: # Có một thang dữ liệu khác nhau như sau: ApplicantIncome: 0 -> 80000USD, LoanAmou

In [11]: # Dữ liệu không phân phối chuẩn => xem xét các outliers => Sau đó áp dụng MinMaxScaler  
# Cũng có thể dùng RobustScaler nếu muốn giữ lại outliers  
# (Vận dụng kiến thức của các bài trước để ra quyết định)  
# Vô duy bog

In [12]: from sklearn.preprocessing import MinMaxScaler

In [13]: min\_max=MinMaxScaler()



```
In [14]: X_train_minmax=min_max.fit_transform(train_data_sub)
```

```
c:\program files\python36\lib\site-packages\sklearn\preprocessing\data.py:323:  
DataConversionWarning: Data with input dtype int64, float64 were all converted  
to float64 by MinMaxScaler.  
    return self.partial_fit(X, y)
```

```
In [15]: X_train_minmax[:4]
```

```
Out[15]: array([[0.0593692 , 0.          , 0.1962775 , 0.72972973, 1.        ],  
                 [0.03379097, 0.054467 , 0.1928934 , 1.          , 1.        ],  
                 [0.03525046, 0.10095458, 0.07952623, 0.32432432, 1.        ],  
                 [0.11815708, 0.          , 0.17428088, 0.72972973, 1.        ]])
```

```
In [16]: df_train_minmax = pd.DataFrame(X_train_minmax, columns=['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History'])  
df_train_minmax.head()
```

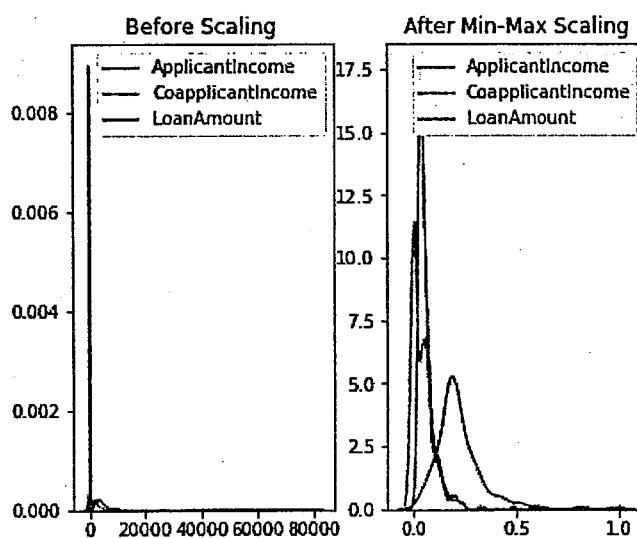
|   | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History |
|---|-----------------|-------------------|------------|------------------|----------------|
| 0 | 0.059369        | 0.000000          | 0.196277   | 0.729730         | 1.0            |
| 1 | 0.033791        | 0.054467          | 0.192893   | 1.000000         | 1.0            |
| 2 | 0.035250        | 0.100955          | 0.079526   | 0.324324         | 1.0            |
| 3 | 0.118157        | 0.000000          | 0.174281   | 0.729730         | 1.0            |
| 4 | 0.027001        | 0.071431          | 0.214890   | 0.729730         | 1.0            |



```
In [17]: fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6, 5))
ax1.set_title('Before Scaling')
sns.kdeplot(train_data_sub.ApplicantIncome, ax=ax1)
sns.kdeplot(train_data_sub.CoapplicantIncome, ax=ax1)
sns.kdeplot(train_data_sub.LoanAmount, ax=ax1)

ax2.set_title('After Min-Max Scaling')
sns.kdeplot(df_train_minmax.ApplicantIncome, ax=ax2)
sns.kdeplot(df_train_minmax.CoapplicantIncome, ax=ax2)
sns.kdeplot(df_train_minmax.LoanAmount, ax=ax2)

plt.show()
```



```
In [ ]:
```



## Ex4

- Cho dữ liệu baseball\_hw.csv chứa thông tin chiều cao, cân nặng của các vận động viên
- Phân tích thông tin sơ bộ về dữ liệu. Trực quan hóa dữ liệu.
- Để việc dự đoán tốt hơn cần phải kiểm tra và chuẩn hóa dữ liệu. Hãy chọn một phương pháp để chuẩn hóa dữ liệu dựa trên thông tin nêu trên.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
data = pd.DataFrame([[74, 180], [74, 215], [72, 210], [72, 210], [73, 188], [69, 180]])
data
```

In [3]:

```
data.head()
```

Out[3]:

|   | 0  | 1   |
|---|----|-----|
| 0 | 74 | 180 |
| 1 | 74 | 215 |
| 2 | 72 | 210 |
| 3 | 72 | 210 |
| 4 | 73 | 188 |

In [4]:

```
data.columns = ['height', 'weight']
```

In [5]:

```
data.head()
```

Out[5]:

|   | height | weight |
|---|--------|--------|
| 0 | 74     | 180    |
| 1 | 74     | 215    |
| 2 | 72     | 210    |
| 3 | 72     | 210    |
| 4 | 73     | 188    |



In [6]:

```
data.height = data.height * 0.0254
data.weight = data.weight * 0.453592
data.head()
```

Out[6]:

|   | height | weight    |
|---|--------|-----------|
| 0 | 1.8796 | 81.646560 |
| 1 | 1.8796 | 97.522280 |
| 2 | 1.8288 | 95.254320 |
| 3 | 1.8288 | 95.254320 |
| 4 | 1.8542 | 85.275296 |

In [7]:

```
data.shape
```

Out[7]:

```
(1015, 2)
```

In [8]:

```
data.to_csv("baseball_hw.csv")
```

In [9]:

```
# Đọc dữ liệu từ tập tin baseball_hw.csv
df = pd.read_csv("baseball_hw.csv"; index_col=0)
df.head()
```

Out[9]:

|   | height | weight    |
|---|--------|-----------|
| 0 | 1.8796 | 81.646560 |
| 1 | 1.8796 | 97.522280 |
| 2 | 1.8288 | 95.254320 |
| 3 | 1.8288 | 95.254320 |
| 4 | 1.8542 | 85.275296 |

In [10]:

```
df.describe()
```

Out[10]:

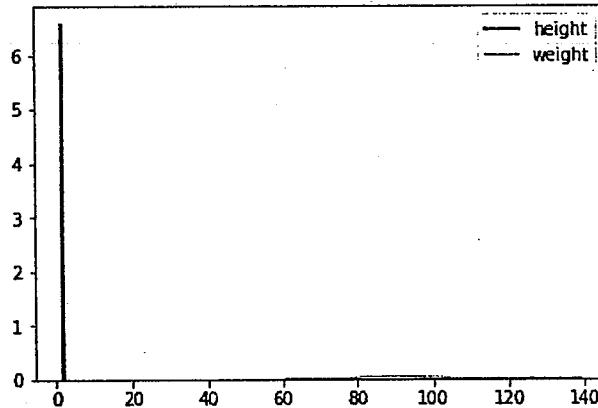
|       | height      | weight      |
|-------|-------------|-------------|
| count | 1015.000000 | 1015.000000 |
| mean  | 1.871717    | 91.330191   |
| std   | 0.058774    | 9.445198    |
| min   | 1.701800    | 68.038800   |
| 25%   | 1.828800    | 84.368112   |
| 50%   | 1.879600    | 90.718400   |
| 75%   | 1.905000    | 97.522280   |
| max   | 2.108200    | 131.541680  |

In [11]:

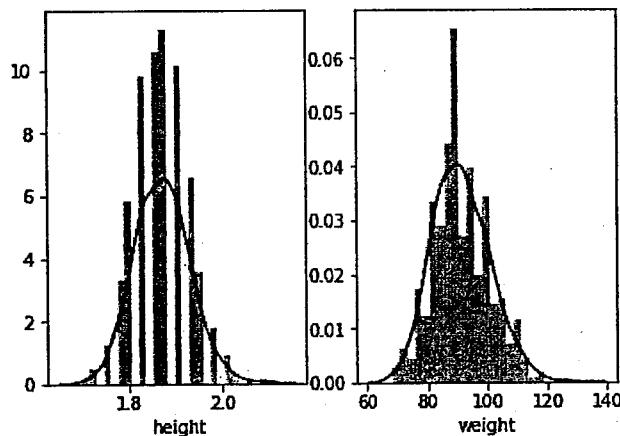
```
import seaborn as sns
```



In [12]: # height và weight có thang đo khác nhau, phân phối khác nhau  
 sns.kdeplot(df.height)  
 sns.kdeplot(df.weight)  
 plt.show()



In [13]: plt.subplot(1,2,1)  
 sns.distplot(df.height)  
 plt.subplot(1,2,2)  
 sns.distplot(df.weight)  
 plt.show()



In [14]: # Xem phân phối:  
 df.height.skew()

Out[14]: 0.23353939068968324

In [15]: df.weight.skew()

Out[15]: 0.34318597598127093

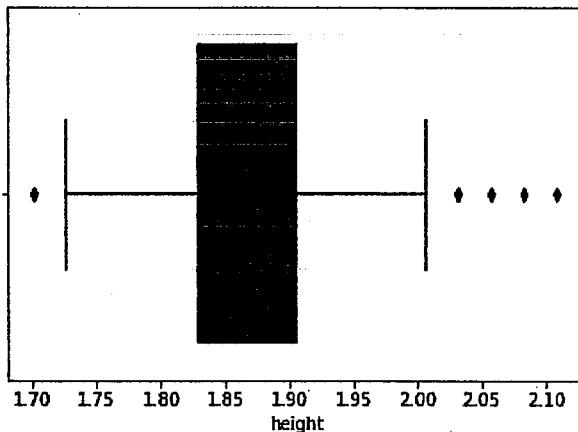
In [16]: # Phân phối Lệch phải

7/24/2019

Ex4

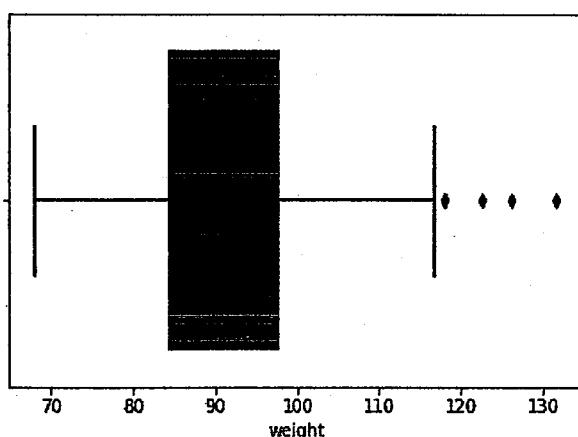
In [17]: # height và weight có thang đo khác nhau.  
sns.boxplot(df.height)

Out[17]: <matplotlib.axes.\_subplots.AxesSubplot at 0x1bc16cf7438>



In [18]: # height và weight có thang đo khác nhau.  
sns.boxplot(df.weight)

Out[18]: <matplotlib.axes.\_subplots.AxesSubplot at 0x1bc16e36630>



In [19]: # Loại bỏ outliers (HV tự làm)  
# không là phân phối chuẩn  
# => Áp dụng MinMaxScaler

In [20]: Q1h = np.percentile(df.height, 25)  
Q1h

Out[20]: 1.8288



```
In [21]: Q3h = np.percentile(df.height, 75)
Q3h
```

Out[21]: 1.905

```
In [22]: Q1w = np.percentile(df.weight, 25)
Q1w
```

Out[22]: 84.368112

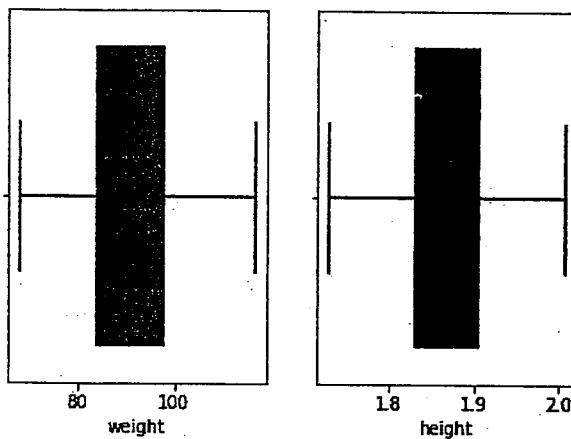
```
In [23]: Q3w = np.percentile(df.weight, 75)
Q3w
```

Out[23]: 97.52228000000001

```
In [24]: df = df[(df.height >= (Q1h - 1.5*(Q3h - Q1h))) & (df.height <= (Q3h + 1.5*(Q3h-Q1h))]
```

```
In [25]: df = df[(df.weight >= (Q1w - 1.5*(Q3w - Q1w))) & (df.weight <= (Q3w + 1.5*(Q3w-Q1w))]
```

```
In [26]: plt.subplot(1,2,1)
sns.boxplot(df.weight)
plt.subplot(1,2,2)
sns.boxplot(df.height)
plt.show()
```



```
In [27]: from sklearn.preprocessing import MinMaxScaler
```

```
In [28]: min_max=MinMaxScaler()
```

```
In [29]: X_minmax=min_max.fit_transform(df)
```



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
**TRUNG TÂM TIN HỌC**

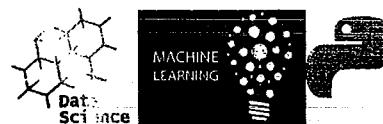
# DATA PRE-PROCESSING AND ANALYSIS

Bài 5: Data Pre-processing – Feature Engineering

Phòng LT & Mạng

<https://cor.scs.vt.edu/~hawkinsl/Data-Pre-processing-and-Feature-Selection.html>

2019



## **Giới thiệu**

- Feature engineering là quá trình sử dụng kiến thức miền (domain knowledge) về dữ liệu để tạo ra các tính năng giúp thuật toán máy học (machine learning algorithms) hoạt động. Feature engineering là nền tảng cho ứng dụng machine learning, nó vừa khó khăn vừa tốn kém. Nhu cầu về feature engineering thủ công có thể được giảm bớt bằng cách học tự động.
  - Feature engineering là công việc thiết yếu trong machine learning ứng dụng.



## Giới thiệu

### ☒ Đặc điểm

- Feature engineering là việc tạo các tính năng đầu vào mới từ những tính năng hiện có của bộ dữ liệu.
- Đây là một trong những nhiệm vụ có giá trị nhất mà một nhà khoa học dữ liệu có thể làm để cải thiện hiệu suất mô hình, vì 3 lý do lớn:
  - Chúng ta có thể cô lập và làm nổi bật thông tin chính, giúp thuật toán "tập trung" vào những gì quan trọng.
  - Chúng ta có thể vận dụng domain knowledge để có tính năng thích hợp.
  - Đồng thời, chúng ta cũng có thể đem đến cho người khác kiến thức về domain knowledge.

VD: có dữ liệu đầu vào  
chiều cao, cân nặng  
⇒ Phân loại (hình  
nặng/mỏng)



## Giới thiệu

### ☐ Domain Knowledge

- Chúng ta có thể thiết kế các thuộc tính thông tin (informative feature) bằng cách dựa trên kiến thức chuyên môn của mình (hoặc những người khác) về lĩnh vực làm việc.
- Ở đây, với một thông tin cụ thể mà ta có thể muốn cô lập, ta sẽ có rất nhiều "tự do sáng tạo".
- Như chúng ta biết, "domain knowledge" rất rộng mở. Có khi, ta sẽ gặp khó khăn hoặc cạn kiệt ý tưởng. Và cũng sẽ có một số phương pháp phỏng đoán cụ thể giúp gợi mở nhiều hơn.



## Nội dung

1. Tạo thuộc tính (feature)
2. Xử lý dữ liệu văn bản (Text Data)



## Tạo thuộc tính (feature)

- Trong phần này, chúng ta sẽ tìm hiểu các kỹ thuật liên quan đến thuộc tính năng (feature engineering) và cách áp dụng nó vào dữ liệu trong thế giới thực.  
Phục quan hóa dữ liệu text → Word cloud.
- Chúng ta sẽ tải, khám phá và trực quan hóa bộ dữ liệu, tìm hiểu về các loại dữ liệu cơ bản của nó và lý do tại sao chúng có ảnh hưởng đến cách chúng ta thiết kế các thuộc tính của mình.



## Tạo thuộc tính (feature)

### ☐ Các kiểu dữ liệu khác nhau

- Continuous: dữ liệu liên tục có thể là integer hoặc float
- Categorical: dữ liệu phân loại, là tập hợp giá trị giới hạn ví dụ như: gender, country of birth
- Ordinal: giá trị xếp hạng, thường không có chi tiết về khoảng cách giữa chúng
- Boolean: giá trị True/False
- Datetime: giá trị thời gian



## Tạo thuộc tính (feature)

### ☐ Các thuộc tính thông dụng

- df.columns: xem danh sách tên cột

```
print(so_survey_df.columns)

Index(['SurveyDate', 'FormalEducation', 'ConvertedSalary', 'Hobby', 'Country',
       'StackOverflowJobsRecommend', 'VersionControl', 'Age',
       'Years Experience', 'Gender', 'RawSalary'],
      dtype='object')
```



## Tạo thuộc tính (feature)

- df.dtypes: xem danh sách kiểu dữ liệu của cột

```
# Print the data type of each column
print(so_survey_df.dtypes)
```

|                            |         |
|----------------------------|---------|
| SurveyDate                 | object  |
| FormalEducation            | object  |
| ConvertedSalary            | float64 |
| Hobby                      | object  |
| Country                    | object  |
| StackOverflowJobsRecommend | float64 |
| VersionControl             | object  |
| Age                        | int64   |
| Years Experience           | int64   |
| Gender                     | object  |
| RawSalary                  | object  |
| dtype: object              |         |



## Tạo thuộc tính (feature)

- Chọn các cột có loại dữ liệu cụ thể

```
# Create subset of only the numeric columns
so_numeric_df = so_survey_df.select_dtypes(include=['int', 'float'])

# Print the column names contained in so_survey_df_num
print(so_numeric_df.columns)

Index(['ConvertedSalary', 'StackOverflowJobsRecommend'], dtype='object')
```



## Tạo thuộc tính (feature)

### Mã hóa thuộc tính phân loại

#### (Encoding categorical feature)

- Để sử dụng các biến phân loại trong mô hình Machine Learning, trước tiên chúng ta cần biểu diễn chúng theo dạng định lượng. Một số cách tiếp cận phổ biến là one hot encoder/dummy encoder hay label encoder.



## Tạo thuộc tính (feature)

### Categorical Data

- Categorical data là các biến chứa giá trị nhãn (label)
- Số lượng các giá trị thường được giới hạn (còn vô hạn) trong một bộ cố định. Ví: Nam/Nữ; mua/kö mua; màu sắc: Đỏ/Xanh...
- Biến phân loại (Categorical variable) thường được gọi là nominal.
- Ví dụ:
  - “pet” variable: gồm “dog” và “cat”.
  - “color” variable gồm: “red”, “green” và “blue”.
  - “place” variable gồm: “first”, “second” và “third”.
- Mỗi giá trị đại diện cho một loại khác nhau.



## Tạo thuộc tính (feature)

- Một số loại có thể có mối quan hệ tự nhiên với nhau, chẳng hạn như một trật tự tự nhiên.  
→ first, second, third...
  - “place” variable ở trên có một thứ tự tự nhiên của các giá trị. Loại biến phân loại này được gọi là biến thứ tự (ordinal variable).



## Tạo thuộc tính (feature)

### □ Vấn đề với Categorical Data

- Một số thuật toán có thể làm việc với dữ liệu phân loại trực tiếp.

▪ Ví dụ, decision tree có thể được huấn luyện trực tiếp từ dữ liệu phân loại mà không cần chuyển đổi dữ liệu (điều này phụ thuộc vào việc triển khai cụ thể).

→ Để giải bài toán + đều gốc chuyên để số liệu đầu vào.



## Tạo thuộc tính (feature)

- Nhiều thuật toán Machine Learning không thể hoạt động trực tiếp trên dữ liệu nhãn. Chúng yêu cầu tất cả các biến đầu vào và biến đầu ra là số.
- Điều này có nghĩa là dữ liệu phân loại phải được chuyển đổi thành dạng số. Nếu biến phân loại là biến đầu ra, chúng ta cũng có thể chuyển đổi dự đoán của mô hình thành dạng phân loại để trình bày chúng hoặc sử dụng chúng trong một số ứng dụng.

Data Pre-processing and Analysis

15

## Tạo thuộc tính (feature)

### Integer encoder/ label encoder

- Mỗi giá trị phân loại duy nhất được gán một giá trị nguyên (integer value)
  - Ví dụ: "red" là 1, "green" là 2, "blue" là 3. → label encoder.
- Việc này được gọi là label encoding hay integer encoding và có thể đảo ngược dễ dàng.
- Đối với một số biến, có thể áp dụng cách này

Data Pre-processing and Analysis

16

## Tạo thuộc tính (feature)

- Các giá trị số nguyên có mối quan hệ được sắp xếp tự nhiên với nhau và các thuật toán Machine Learning có thể có thể hiểu và khai thác mối quan hệ này.
  - \* Ví dụ: "place" variable trên có thể là một ví dụ điển hình trong đó label encoder là phù hợp.
- Hoặc thuộc tính Target trong dataset



## Tạo thuộc tính (feature)

- Ví dụ: Label encoder

```
# Stage
label_encoder_s = preprocessing.LabelEncoder()
students['Stage_new'] = label_encoder_s.fit_transform(students['Stage'])
students
```

When 1 cat → first value  
⇒ using Label encoder

+ 1 cat → many value ⇒ using Dummy

|     |   |   |
|-----|---|---|
| Red | 0 | 0 |
| 0   | 1 | 0 |
| 0   | 0 | 1 |

Red green blue

| Name   | Sex    | Age | Stage  | Stage_new |
|--------|--------|-----|--------|-----------|
| 0 John | Male   | 16  | Second | 1         |
| 1 Lucy | Female | 17  | First  | 0         |
| 2 Lyly | Other  | 16  | Third  | 2         |
| 3 Mark | Male   | 17  | Second | 1         |



## Tạo thuộc tính (feature)

### • Ví dụ: Label encoder

```
# Label_encoder object knows how to understand word labels.  
label_encoder = preprocessing.LabelEncoder()  
# Encode labels in column 'species'.  
iris['species']= label_encoder.fit_transform(iris['species'])  
iris['species'].unique()
```

```
array([0, 1, 2], dtype=int64)
```

```
iris.head()
```

|   | sepal_length | sepal_width | petal_length | petal_width | species |
|---|--------------|-------------|--------------|-------------|---------|
| 0 | 5.1          | 3.5         | 1.4          | 0.2         | 0       |
| 1 | 4.9          | 3.0         | 1.4          | 0.2         | 0       |
| 2 | 4.7          | 3.2         | 1.3          | 0.2         | 0       |
| 3 | 4.6          | 3.1         | 1.5          | 0.2         | 0       |
| 4 | 5.0          | 3.6         | 1.4          | 0.2         | 0       |

Data Pre-processing and Analysis

19

## Tạo thuộc tính (feature)

### One hot encoder/ Dummy encoder

- Đối với các biến phân loại không tồn tại mối quan hệ thứ tự, integer encoder là không đủ.
- Tùy thuộc vào dữ liệu chúng ta có, chúng ta có thể gặp phải tình huống, sau khi Label encoder, có thể nhầm lẫn mô hình vì nghĩ rằng một cột có dữ liệu với thứ tự hoặc thứ bậc nào đó.
- Trong trường hợp này, one-hot encoder/dummy encoder có thể được áp dụng thay cho biểu diễn số nguyên.

Data Pre-processing and Analysis

20

## Tạo thuộc tính (feature)

### • One hot encoder

- Biến được mã hóa số nguyên được loại bỏ và biến nhị phân mới được thêm vào cho mỗi giá trị số nguyên duy nhất.
- Với one hot encoder, các thuộc tính có thể giải thích rõ ràng
  - Ví dụ: "color" variable, có 3 loại vậy sẽ có 3 biến nhị phân (binary variable): giá trị "1" được thay thế trong binary variable cho color và giá trị "0" cho các color còn lại

| red | green | blue |
|-----|-------|------|
| 1   | 0     | 0    |
| 0   | 1     | 0    |
| 0   | 0     | 1    |

## Tạo thuộc tính (feature)

Ví dụ:  
students

|   | Name | Sex    | Age |
|---|------|--------|-----|
| 0 | John | Male   | 16  |
| 1 | Lucy | Female | 17  |
| 2 | Lily | Other  | 16  |
| 3 | Mark | Male   | 17  |

bên kia có khái niệm one hot encoder  
n loại  $\rightarrow$  n cột

student\_one\_hot = pd.get\_dummies(students, columns=['Sex'], prefix='S')  
student\_one\_hot

|   | Name | Age | S_Female | S_Male | S_Other |
|---|------|-----|----------|--------|---------|
| 0 | John | 16  | 0        | 1      | 0       |
| 1 | Lucy | 17  | 1        | 0      | 0       |
| 2 | Lily | 16  | 0        | 0      | 1       |
| 3 | Mark | 17  | 0        | 1      | 0       |

có khía cạnh 1 cột Male/Female

## Tạo thuộc tính (feature)

- Dummy encoder  $\rightarrow n \text{ loại} \rightarrow (n-1 \text{ bit}) \rightarrow$  dùng Dummy encoder

- Tương tự như one hot encoder
- Chứa thông tin cần thiết mà không trùng lặp
- Ví dụ

```
students_dummies = pd.get_dummies(students, columns=['Sex'], prefix='S', drop_first=True)
```

|   | Name | Age | S_Male | S_Other |
|---|------|-----|--------|---------|
| 0 | John | 16  | 1      | 0       |
| 1 | Lucy | 17  | 0      | 0       |
| 2 | Lily | 16  | 0      | 1       |
| 3 | Mark | 17  | 1      | 0       |



## Tạo thuộc tính (feature)

### Xử lý các danh mục không phổ biến

#### (uncommon category)

Ví: Một khung khép kín sẽ bao gồm  
tất cả các giá trị có thể xuất hiện

- Một số tính năng có thể có nhiều loại khác nhau nhưng phân phối rất không đồng đều về sự xuất hiện của chúng.

- Lấy ví dụ các ngôn ngữ yêu thích của Data Science để mã hóa, một số lựa chọn phổ biến là Python, R và Julia, nhưng có thể có các cá nhân với các lựa chọn riêng biệt, như FORTRAN, C, v.v. Trong những trường hợp này, chúng ta có thể không muốn tạo một tính năng cho mỗi giá trị mà có thể gộp lại thành một tính năng cho các giá trị ít xuất hiện.



## Tạo thuộc tính (feature)

### • Ví dụ:

```
# Create a series out of the Country column
countries = so_survey_df.Country

# Get the counts of each category
country_counts = countries.value_counts()

# Print the count values for each category
print(country_counts)

South Africa    166
USA             164
Spain           134
Sweden          119
France          115
Russia           97
UK              95
India            95
Ukraine          9
Ireland           5
```

các quốc gia & (14)

```
Name: Country, dtype: int64
```



## Tạo thuộc tính (feature)

```
# Create a mask for only categories that occur less than 10 times
mask = countries.isin(country_counts[country_counts < 10].index)

# Print the top 5 rows in the mask series
print(mask.head())

0    False
1    False
2    False
3    False
4    False
Name: Country, dtype: bool
```

```
# Label all other categories as Other
countries[mask] = 'Other'

# Print the updated category counts
print(pd.value_counts(countries))

South Africa    166
USA             164
Spain           134
Sweden          119
France          115
Russia           97
UK              95
India            95
Other            14
Name: Country, dtype: int64
```

```
# Change Country to 'Other' at mask = True
so_survey_df['Country'][mask] = 'Other'
```



## Tạo thuộc tính (feature)

### ☐ Tạo cột nhị phân (Binarizing column)

- Mặc dù hầu hết các giá trị số thường được sử dụng mà không cần bất kỳ feature engineering nào, sẽ có trường hợp một số hình thức thao tác có thể hữu ích.
  - Ví dụ: Trong một số trường hợp, chúng ta có thể ~~75% vs~~ không quan tâm đến ~~độ lớn~~ <sup>66/Không</sup> của giá trị mà chỉ quan tâm đến hướng của nó hoặc nó tồn tại <sup>66/Không</sup> hay không. Trong những tình huống này, ta có thể muốn tạo thành một cột mới.



## Tạo thuộc tính (feature)

- Ví dụ: Trong dữ liệu so\_survey\_df, có một số lượng lớn người trả lời khảo sát đang làm việc tự nguyện (không phải trả tiền). Ta sẽ tạo một cột mới có tiêu đề Paid\_Job với mỗi người được trả tiền (mức lương của họ lớn hơn 0) sẽ = 1.

```
# Create the Paid_Job column filled with zeros
so_survey_df['Paid_Job'] = 0 → Tạo ra all gr/tu/ 0

# Replace all the Paid_Job values where ConvertedSalary is > 0
so_survey_df.loc[so_survey_df['ConvertedSalary'] > 0, 'Paid_Job'] = 1 → đổi bậc cù 1 nei/ có tíc

# Print the first five rows of the columns
print(so_survey_df[['Paid_Job', 'ConvertedSalary']].head())
```

|   | Paid_Job | ConvertedSalary |
|---|----------|-----------------|
| 0 | 0        | NAN             |
| 1 | 1        | 70841.0         |
| 2 | 0        | NAN             |
| 3 | 1        | 21426.0         |
| 4 | 1        | 41671.0         |



## Tạo thuộc tính (feature)

### Binning value

- Đối với nhiều giá trị liên tục, chúng ta có thể sẽ quan tâm ít hơn về giá trị chính xác của một cột kiểu số, thay vào đó chúng ta quan tâm đến nhóm mà nó rơi vào. Điều này hữu ích khi vẽ các giá trị hoặc đơn giản hóa các mô hình Machine Learning. Nó chủ yếu được sử dụng trên các biến liên tục trong đó độ chính xác không phải là mối quan tâm lớn nhất, ví dụ: tuổi, chiều cao, tiền lương.  
→ tạo khoảng trống, khoảng chiều cao, khung lồng



## Tạo thuộc tính (feature)

### Đặc điểm

- Động lực chính của việc tạo bin là làm cho mô hình mạnh mẽ hơn và ngăn ngừa overfitting, tuy nhiên, nó có chi phí hiệu suất cao hơn. Mỗi khi chúng ta sử dụng bin một cột nào đó, chúng ta chấp nhận bớt thông tin.
- Sự đánh đổi giữa hiệu suất và overfitting là điểm mấu chốt của quy trình tạo bin. Nhìn chung, đối với các cột số, ngoại trừ một số trường hợp quá rõ ràng, việc tạo bin có thể là dư thừa đối với một số loại thuật toán, do ảnh hưởng của nó đến hiệu suất mô hình.

*mô hình chỉ tốt khi nó ở nhóm train  
kết quả không bị test*





## Tạo thuộc tính (feature)

- Tuy nhiên, đối với các cột phân loại, các nhãn có tần số thấp có thể ảnh hưởng tiêu cực đến độ mạnh của các mô hình thống kê. Do đó, việc gán một danh mục chung cho các giá trị ít thường xuyên này sẽ giúp duy trì sự mạnh mẽ của mô hình.
- Ví dụ: nếu kích thước dữ liệu là 100.000 dòng, thì có thể là một lựa chọn tốt để hợp nhất các nhãn có số lượng nhỏ hơn 100 đến một danh mục mới như Other.



## Tạo thuộc tính (feature)

- Các bin được tạo ra bằng cách sử dụng:  
cắt bao nhiêu khoảng  
`pd.cut(df['column_name'], bins)`
- Trong đó, bins là integer chỉ định số lượng bin cách đều nhau hoặc danh sách các ranh giới của bin.



## Tạo thuộc tính (feature)

### \* Ví dụ

```
# Bin the continuous variable ConvertedSalary into 5 bins
so_survey_df['equal_binned'] = pd.cut(so_survey_df['ConvertedSalary'], 5)
so_survey_df['equal_binned'].unique()

[NaN, (-2000.0, 400000.0], (800000.0, 1200000.0], (400000.0, 800000.0], (1600000.0, 2000000.0]]
Categories (4, interval[float64]): [(-2000.0, 400000.0] < (800000.0, 1200000.0] < (1600000.0, 2000000.0]
0.0 - 2000.0, 34444 < 400000.0 < 799999 < 800000.0 < 1600000.0 < 2000000.0

# Print the first 5 rows of the equal_binned column
print(so_survey_df[['equal_binned', 'ConvertedSalary']].head())

  equal_binned  ConvertedSalary
0            NaN
1  (-2000.0, 400000.0]  70841.0
2            NaN
3  (-2000.0, 400000.0]  21426.0
4  (-2000.0, 400000.0]  41671.0
```

## Tạo thuộc tính (feature)

### \* Ví dụ: Tạo danh sách nhãn cho các bin

```
# Import numpy
import numpy as np

# Specify the boundaries of the bins
bins = [-np.inf, 10000, 50000, 100000, 150000, np.inf]

# Bin labels
labels = ['Very low', 'Low', 'Medium', 'High', 'Very high'] → Thuật toán "label encoder" hoặc đơn giản là "label encoder" hoặc đơn giản là "label encoder"

# Bin the continuous variable ConvertedSalary using these boundaries
so_survey_df['boundary_binned'] = pd.cut(so_survey_df['ConvertedSalary'], bins, labels = labels)

# Print the first 5 rows of the boundary_binned column
print(so_survey_df[['boundary_binned', 'ConvertedSalary']].head())
```

| boundary_binned | ConvertedSalary |
|-----------------|-----------------|
| NaN             | NaN             |
| Medium          | 70841.0         |
| NaN             | NaN             |
| Low             | 21426.0         |
| Low             | 41671.0         |



## Nội dung

1. Tạo thang đo tính (Metric)
2. Xử lý dữ liệu văn bản (Text Data)



## Xử lý dữ liệu văn bản (Text Data)

### Chuẩn hóa text

- Dữ liệu văn bản phi cấu trúc có thể được sử dụng trực tiếp trong hầu hết các phân tích. Cần thực hiện nhiều bước để chuyển từ một chuỗi biểu mẫu dài sang một tập hợp các cột số theo đúng định dạng có thể được mô hình Machine Learning sử dụng.
- Bước đầu tiên của quy trình này là chuẩn hóa dữ liệu và loại bỏ bất kỳ ký tự nào có thể gây ra sự cố sau này trong việc phân tích của bạn.



## Xử lý dữ liệu văn bản (Text Data)

### • Loại bỏ các ký tự không mong muốn

#### ▪ Dùng regular expression:

- [ a-zA-Z] : tất cả các ký tự chữ
- [ ^a-zA-Z] : tất cả các ký tự không phải là chữ

#### ▪ Ví dụ

```
speech_df['text'][0][:100]
```

'Fellow-Citizens of the Senate and of the House of Representatives: AMONG the vicissitudes incident '

```
speech_df['text'] = speech_df['text'].str.replace('[^a-zA-Z]', ' ')
```

*Thay thế tất cả ký tự không phải là chữ bằng khoảng trống*

```
speech_df['text'][0][:100]
```

'Fellow Citizens of the Senate and of the House of Representatives AMONG the vicissitudes incident '



## Xử lý dữ liệu văn bản (Text Data)

### • Chuẩn hóa loại chữ

#### ▪ Dùng str.lower(): chuyển sang chữ thường

```
speech_df['text'][0][:100]
```

'Fellow Citizens of the Senate and of the House of Representatives AMONG the vicissitudes incident '

```
speech_df['text'] = speech_df['text'].str.lower()  
speech_df['text'][0][:100]
```

'fellow citizens of the senate and of the house of representatives among the vicissitudes inciment '



## Xử lý dữ liệu văn bản (Text Data)

### • Thuộc tính văn bản cấp cao (High level text feature)

▪ Khi văn bản đã được làm sạch và chuẩn hóa, chúng ta có thể bắt đầu tạo các tính năng từ dữ liệu. Thông tin cơ bản nhất ta có thể tính toán về văn bản phi biểu mẫu là kích thước của nó, chẳng hạn như độ dài và số lượng từ.

- Dùng .str.len() để biết chiều dài chuỗi
- Dùng .str.split() để cắt chuỗi thành các phần tử chứa trong list



## Xử lý dữ liệu văn bản (Text Data)

### ▪ Ví dụ

```
# Find the length of each text
speech_df['char_cnt'] = speech_df['text_clean'].str.len()

# Count the number of words in each text
speech_df['word_cnt'] = speech_df['text_clean'].str.split().str.len()

# Find the average length of word
speech_df['avg_word_length'] = speech_df['char_cnt'] / speech_df['word_cnt']

# Print the first 5 rows of these columns
speech_df[['text_clean', 'char_cnt', 'word_cnt', 'avg_word_length']].head()
```

|   |   | text_clean | char_cnt | word_cnt | avg_word_length            |          |
|---|---|------------|----------|----------|----------------------------|----------|
| 0 | fellow citizens of the senate and of the house... | 8616 ký tự | 1432 từ  | 6.016760 | dung lượng 1 bài 6 ký tự → | đơn giản |
| 1 | fellow citizens i am again called upon by th...   | 787        | 135      | 5.829630 |                            | trên tay |
| 2 | when it was first perceived in early times t...   | 13871      | 2323     | 5.971158 |                            |          |
| 3 | friends and fellow citizens called upon to u...   | 10144      | 1736     | 5.843318 |                            |          |
| 4 | proceeding fellow citizens to that qualifica...   | 12902      | 2169     | 5.948363 |                            |          |



## Xử lý dữ liệu văn bản (Text Data)

### Word Count Representation

- Khi thông tin cấp cao đã được ghi lại, chúng ta có thể bắt đầu tạo các thuộc tính dựa trên nội dung thực tế của từng văn bản. Một cách để làm là tiếp cận nó theo cách tương tự như cách đã làm việc với các biến phân loại.
  - ☞ Đổi với mỗi từ duy nhất trong tập dữ liệu tạo ra một cột.
  - ☞ Đổi với entry, số lần từ này xuất hiện được đếm và giá trị đếm được nhập vào cột tương ứng.
- Các cột "count" này có thể được sử dụng để huấn luyện các mô hình machine learning.



## Xử lý dữ liệu văn bản (Text Data)

- Kỹ thuật này còn được gọi là bag of words
- Ví dụ

"citizens of the senate and of the house of representatives"



| Index | citizens | of | the | senate | and | house | representatives |
|-------|----------|----|-----|--------|-----|-------|-----------------|
| 1     | 1        | 3  | 2   | 1      | 1   | 1     | 1               |

Nhập Sắp xếp theo thứ tự ABC.



# Xử lý dữ liệu văn bản (Text Data)

## • Các bước thực hiện

### ▪ Bước 1: Khởi tạo CountVectorizer

```
# Import CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
# Instantiate CountVectorizer
cv = CountVectorizer() → có bao nhiêu sd hết, ko giới hạn.
cv

CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```



# Xử lý dữ liệu văn bản (Text Data)

### ▪ Bước 2: fit

```
# Fit the vectorizer
cv.fit(speech_df['text_clean'])

CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

```
# Print feature names
print(cv.get_feature_names())
```

→ 'tự' duy nhất tìm ra theo abc .  
['abandon', 'abandoned', 'abandonment', 'abate', 'abdicated', 'abeyance', 'abhorring', 'abide', 'abiding', 'abilities', 'ability', 'abject', 'able', 'ably', 'abnormal', 'abode', 'abolish', 'abolished', 'abolishing', 'aboriginal', 'aborigines', 'abound', 'abounding', 'abounds', 'about', 'above', 'abraham', 'abreast', 'abridging', 'abroad', 'absence', 'absent', 'absolute', 'absolutely', 'absolutism', 'absorb', 'absorbed', 'absorbing', 'absorbs', 'abstain', 'abstaining', 'abstract', 'abstractions', 'absurd', 'abundance', 'abundant', 'abundantly', 'abuse', 'abused', 'abuses', 'academies', 'accept', 'acceptance', 'accepted', 'accepting', 'accepts', 'access', 'accessible', 'accession', 'accident', 'accidental', 'accidents', 'acclaim', 'accommodation', 'accommodations', 'accompanied', 'accompany', 'accomplish', 'accomplished', 'accomplishing', 'accomplishment', 'accomplishments', 'accord', 'accordance', 'accorded', 'according', 'accordingly', 'accords', 'account', 'accountability', 'accountable', 'accounted', 'accrue', 'accrued', 'accruing', 'accumulate', 'accumulated', 'accumulation', 'accurately', 'accustom', 'achieve', 'achieved', 'achievement', 'achievements', 'achieving', 'acknowledge', 'acknowledged', 'acknowledging',



## Xử lý dữ liệu văn bản (Text Data)

### Bước 3: Chuyển đổi văn bản

```
# Apply the vectorizer
cv_transformed = cv.transform(speech_df['text_clear'])

# Print the full array
cv_array = cv_transformed.toarray() Chuyển array
print(cv_array)

[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 ...
 [0 1 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]

# Print the shape of cv_array
print(cv_array.shape)

(58, 9043)
```



## Xử lý dữ liệu văn bản (Text Data)

### Bước 4: Chuyển kết quả thành các cột trong DataFrame → tạo DataFrame.

```
cv_df = pd.DataFrame(cv_array, columns=cv.get_feature_names()).add_prefix('Counts_')
cv_df.head()
```

|   | Counts_abandon | Counts_abandoned | Counts_abandonment | Counts_abate | Counts_abdicated | Counts_abeyance | Counts_abhorring | Counts_abide | Counts_at |
|---|----------------|------------------|--------------------|--------------|------------------|-----------------|------------------|--------------|-----------|
| 0 | 0              | 0                | 0                  | 0            | 0                | 0               | 0                | 0            | 0         |
| 1 | 0              | 0                | 0                  | 0            | 0                | 0               | 0                | 0            | 0         |
| 2 | 0              | 1                | 0                  | 0            | 0                | 0               | 0                | 0            | 0         |
| 3 | 1              | 0                | 0                  | 0            | 0                | 0               | 0                | 0            | 0         |
| 4 | 0              | 0                | 0                  | 0            | 0                | 0               | 0                | 0            | 0         |

5 rows x 9043 columns



## Xử lý dữ liệu văn bản (Text Data)

### • CountVectorizer bổ sung tham số min\_df và max\_df

- Như ta đã thấy, sử dụng CountVectorizer mặc định sẽ tạo ra một tính năng cho mỗi từ đơn lẻ trong kho văn bản. Điều này có thể tạo ra quá nhiều thuộc tính, bao gồm các thuộc tính cung cấp rất ít giá trị phân tích.
- Vì thế, CountVectorizer khi bổ sung các tham số giúp giảm số lượng thuộc tính:

min\_df: Chỉ sử dụng các từ xuất hiện nhiều hơn tỷ lệ phần trăm tài liệu này. Việc này có thể được sử dụng để loại bỏ các từ ít hơn không khái quát trên các văn bản.

max\_df: Chỉ sử dụng các từ xuất hiện ít hơn tỷ lệ phần trăm tài liệu này. Việc này rất hữu ích, giúp loại bỏ các từ rất phổ biến xảy ra trong mọi kho văn bản mà không cần thêm giá trị như "and" hoặc "the".



## Xử lý dữ liệu văn bản (Text Data)

### • Ví dụ

```
# Import CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer

# Specify arguments to limit the number of features generated
cv = CountVectorizer(min_df=0.2, max_df=0.8) → giới hạn min max (tùy quantifiable)
# Fit, transform, and convert into array
cv_transformed = cv.fit_transform(speech_df['text_clean'])
cv_array = cv_transformed.toarray()

# Print the array shape
print(cv_array.shape)
(58, 818) hoặc 9043
```



## Xử lý dữ liệu văn bản (Text Data)

### Bước 5: Gộp DataFrame

```
speech_df = pd.concat([speech_df, cv_df], axis=1 , sort=False)  
print(speech_df.shape)  
(58, 9051)
```



## Xử lý dữ liệu văn bản (Text Data)

### Tf-Idf (Term frequency-inverse document frequency)

- Xử lý ngôn ngữ tự nhiên là một kỹ thuật quan trọng nhằm giúp máy tính hiểu được ngôn ngữ của con người, qua đó hướng dẫn máy tính thực hiện và giúp đỡ con người trong những công việc có liên quan đến ngôn ngữ như: dịch thuật, phân tích dữ liệu văn bản, nhận dạng tiếng nói, tìm kiếm thông tin, tóm tắt văn bản...

- Một trong những kỹ thuật để xử lý ngôn ngữ tự nhiên là TF-IDF (Tần suất xuất hiện của từ-nghịch đảo tần suất của văn bản)



## Xử lý dữ liệu văn bản (Text Data)

- Mặc dù số lần xuất hiện của các từ có thể có ích khi xây dựng các mô hình, các từ xuất hiện nhiều lần có thể làm sai lệch kết quả một cách không mong muốn. Để hạn chế những từ phổ biến này từ việc áp đảo mô hình, một hình thức chuẩn hóa có thể được sử dụng. TF-IDF có tác dụng làm giảm giá trị của các từ phổ biến, đồng thời tăng trọng số của các từ không xảy ra trong nhiều tài liệu.



## Xử lý dữ liệu văn bản (Text Data)

- TF-IDF là trọng số của một từ trong văn bản thu được qua thống kê, nó thể hiện mức độ quan trọng của từ này trong một văn bản, với bối cảnh văn bản đang xét nằm trong một tập hợp các văn bản.
- IF-IDF thường được sử dụng vì: trong ngôn ngữ luôn có những từ xảy ra thường xuyên với các từ khác.





```
In [7]: # chuyên chuỗi số sang số
dataset.age = pd.to_numeric(dataset.age)
dataset.bp = pd.to_numeric(dataset.bp)
dataset.sg = pd.to_numeric(dataset.sg)
dataset.al = pd.to_numeric(dataset.al)
dataset.su = pd.to_numeric(dataset.su)
dataset.bgr = pd.to_numeric(dataset.bgr)
dataset.bu = pd.to_numeric(dataset.bu)
dataset.sc = pd.to_numeric(dataset.sc)
dataset.sod = pd.to_numeric(dataset.sod)
dataset.pot = pd.to_numeric(dataset.pot)
dataset.hemo = pd.to_numeric(dataset.hemo)
dataset.pcv = pd.to_numeric(dataset.pcv)
dataset.wc = pd.to_numeric(dataset.wc)
dataset.rc = pd.to_numeric(dataset.rc)
```

```
In [8]: # dataset.info()
```

```
In [9]: dataset.head()
```

```
Out[9]:
```

|   | age  | bp   | sg   | al  | su  | rbc    | pc       | pcc        | ba         | bgr   | ... | pcv  | wc     | rc  |
|---|------|------|------|-----|-----|--------|----------|------------|------------|-------|-----|------|--------|-----|
| 0 | 48.0 | 80.0 | 1.02 | 1.0 | 0.0 | NaN    | normal   | notpresent | notpresent | 121.0 | ... | 44.0 | 7800.0 | 5.2 |
| 1 | 7.0  | 50.0 | 1.02 | 4.0 | 0.0 | NaN    | normal   | notpresent | notpresent | NaN   | ... | 38.0 | 6000.0 | NaN |
| 2 | 62.0 | 80.0 | 1.01 | 2.0 | 3.0 | normal | normal   | notpresent | notpresent | 423.0 | ... | 31.0 | 7500.0 | NaN |
| 3 | 48.0 | 70.0 | 1.01 | 4.0 | 0.0 | normal | abnormal | present    | notpresent | 117.0 | ... | 32.0 | 6700.0 | 3.9 |
| 4 | 51.0 | 80.0 | 1.01 | 2.0 | 0.0 | normal | normal   | notpresent | notpresent | 106.0 | ... | 35.0 | 7300.0 | 4.6 |

5 rows × 25 columns

```
In [10]: dataset = dataset.dropna() → để xử lý, (nên dropna)
dataset.head()
```

```
Out[10]:
```

|    | age  | bp   | sg   | al  | su  | rbc      | pc       | pcc        | ba         | bgr   | ... | pcv  | wc      |
|----|------|------|------|-----|-----|----------|----------|------------|------------|-------|-----|------|---------|
| 3  | 48.0 | 70.0 | 1.01 | 4.0 | 0.0 | normal   | abnormal | present    | notpresent | 117.0 | ... | 32.0 | 6700.0  |
| 9  | 53.0 | 90.0 | 1.02 | 2.0 | 0.0 | abnormal | abnormal | present    | notpresent | 70.0  | ... | 29.0 | 12100.0 |
| 11 | 63.0 | 70.0 | 1.01 | 3.0 | 0.0 | abnormal | abnormal | present    | notpresent | 380.0 | ... | 32.0 | 4500.0  |
| 14 | 68.0 | 80.0 | 1.01 | 3.0 | 2.0 | normal   | abnormal | present    | present    | 157.0 | ... | 16.0 | 11000.0 |
| 20 | 61.0 | 80.0 | 1.02 | 2.0 | 0.0 | abnormal | abnormal | notpresent | notpresent | 173.0 | ... | 24.0 | 9200.0  |

5 rows × 25 columns



In [11]: # Categorical boolean mask → Turn cat object  
 categorical\_feature\_mask = dataset.dtypes==object  
 # filter categorical columns using mask and turn it into a List  
 categorical\_cols = dataset.columns[categorical\_feature\_mask].tolist() → change shape list -  
 categorical\_cols

Out[11]: ['rbc', 'pc', 'pcc', 'ba', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane', 'class']

In [12]: dataset = pd.get\_dummies(data=dataset, columns=categorical\_cols, drop\_first=True)

In [13]: dataset.head()

Out[13]:

|    | age  | bp   | sg   | al  | su  | bgr   | bu    | sc  | sod   | pot | ... | pc_normal | pcc_present | ba_pres |
|----|------|------|------|-----|-----|-------|-------|-----|-------|-----|-----|-----------|-------------|---------|
| 3  | 48.0 | 70.0 | 1.01 | 4.0 | 0.0 | 117.0 | 56.0  | 3.8 | 111.0 | 2.5 | ... | 0         | 1           |         |
| 9  | 53.0 | 90.0 | 1.02 | 2.0 | 0.0 | 70.0  | 107.0 | 7.2 | 114.0 | 3.7 | ... | 0         | 1           |         |
| 11 | 63.0 | 70.0 | 1.01 | 3.0 | 0.0 | 380.0 | 60.0  | 2.7 | 131.0 | 4.2 | ... | 0         | 1           |         |
| 14 | 68.0 | 80.0 | 1.01 | 3.0 | 2.0 | 157.0 | 90.0  | 4.1 | 130.0 | 6.4 | ... | 0         | 1           |         |
| 20 | 61.0 | 80.0 | 1.02 | 2.0 | 0.0 | 173.0 | 148.0 | 3.9 | 135.0 | 5.2 | ... | 0         | 0           |         |

5 rows × 25 columns

In [ ]:



```
In [10]: # Fit the vectorizer
cv.fit(source)
```

```
Out[10]: CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                        lowercase=True, max_df=1.0, max_features=None, min_df=1,
                        ngram_range=(1, 1), preprocessor=None, stop_words='english',
                        strip_accents=None, token_pattern='(\\u)\\b\\w\\w+\\b',
                        tokenizer=None, vocabulary=None)
```

```
In [11]: # Print feature names
# print(cv.get_feature_names())
```

```
In [12]: cv.vocabulary_
{'30th': 494,
 'august': 1257,
 'areyouunique': 1160,
 'joined': 4186,
 'league': 4426,
 'touch': 7595,
 'deal': 2409,
 'personal': 5608,
 'finally': 3123,
 'completed': 2124,
 'course': 2246,
 'suggest': 7166,
 'stays': 7027,
 'able': 857,
 'ors': 5424,
 'stool': 7062,
 'settled': 6566,
 'wishin': 8169,
 'mrng': 5023,
 'hav': 3665}
```

```
In [13]: # Apply the vectorizer
cv_transformed = cv.transform(source)
# Print the full array
cv_array = cv_transformed.toarray()
```

```
In [14]: cv_array.shape
```

```
Out[14]: (5572, 8404)
```

```
In [15]: from scipy import sparse
```



In [16]: `a0 = sparse.csr_matrix(cv_array[0])` → mảng dense chỉ có các indices  
thứ tự.

```
(0, 1051)    1
(0, 1271)    1
(0, 1701)    1
(0, 1703)    1
(0, 1994)    1
(0, 2271)    1
(0, 3494)    1
(0, 3534)    1
(0, 4224)    1
(0, 4349)    1
(0, 5741)    1
(0, 8026)    1
(0, 8227)    1
```

## Với Tf-Idf

In [17]:

```
# Import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
# Instantiate TfidfVectorizer
tv = TfidfVectorizer(max_features=500, stop_words='english')
tv
```

Out[17]: `TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',  
dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',  
lowercase=True, max_df=1.0, max_features=500, min_df=1,  
ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf=True,  
stop_words='english', strip_accents=None, sublinear_tf=False,  
token_pattern='(\\w+\\W+\\w+)', tokenizer=None, use_idf=True,  
vocabulary=None)`

In [18]:

```
# Fit the vectorizer
tv.fit_transform(source)
```

Out[18]: <5572x500 sparse matrix of type '<class 'numpy.float64'>'  
with 23808 stored elements in Compressed Sparse Row format>

In [19]:

```
# print(tv.get_feature_names())
```



In [20]: `tv.vocabulary_`

```
Out[20]: {'available': 42,
          'great': 177,
          'world': 484,
          'got': 175,
          'wat': 461,
          'ok': 301,
          'lar': 224,
          'wif': 472,
          'free': 157,
          'entry': 143,
          'win': 475,
          'final': 150,
          'text': 405,
          'receive': 345,
          'question': 338,
          'txt': 441,
          'rate': 340,
          'apply': 36,
          'dun': 135,
          ...}
```

In [21]: *# Transform the data*  
`tv_transformed = tv.transform(source)`  
`tv_array = tv_transformed.toarray()`

In [22]: `print(tv_array)`

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

In [23]: *# Create a DataFrame with these features*  
`tv_df = pd.DataFrame(tv_transformed.toarray(),`  
 `columns=tv.get_feature_names()).add_prefix('TFIDF_')`  
`tv_df.head()`

Out[23]:

|   | TFIDF_000 | TFIDF_10 | TFIDF_100 | TFIDF_1000 | TFIDF_10p | TFIDF_11 | TFIDF_12hrs | TFIDF_150 | 1   |
|---|-----------|----------|-----------|------------|-----------|----------|-------------|-----------|-----|
| 0 | 0.0       | 0.0      | 0.0       | 0.0        | 0.0       | 0.0      | 0.0         | 0.0       | 0.0 |
| 1 | 0.0       | 0.0      | 0.0       | 0.0        | 0.0       | 0.0      | 0.0         | 0.0       | 0.0 |
| 2 | 0.0       | 0.0      | 0.0       | 0.0        | 0.0       | 0.0      | 0.0         | 0.0       | 0.0 |
| 3 | 0.0       | 0.0      | 0.0       | 0.0        | 0.0       | 0.0      | 0.0         | 0.0       | 0.0 |
| 4 | 0.0       | 0.0      | 0.0       | 0.0        | 0.0       | 0.0      | 0.0         | 0.0       | 0.0 |

5 rows × 500 columns



```
In [24]: examine_row = tv_df.iloc[0]
print(examine_row. sort_values(ascending=False).head())
```

```
TFIDF_available    0.549238
TFIDF_world        0.496702
TFIDF_wat          0.410286
TFIDF_great         0.405632
TFIDF_got           0.344604
Name: 0, dtype: float64
```

## Với TF-IDF và N-grams

In [25]: `tv_bi_gram_vec = TfidfVectorizer(ngram_range = (1, 2), stop_words='english')
# Fit and apply bigram vectorizer
tv_bi_gram = tv_bi_gram_vec.fit_transform(source)`

In [26]: `# Print the bigram features
# print(tv_bi_gram_vec.get_feature_names())`

In [27]: `tv_bi_gram_vec.vocabulary_
'week word': 35352,
'word like': 36079,
'like fun': 18589,
'fun tb': 12515,
'tb ok': 31004,
'ok xxx': 23194,
'xxx std': 36528,
'std chgs': 29972,
'chgs send': 6359,
'send 50': 27896,
'50 rcv': 1489,
'brother': 5157,
'speak': 29541,
'treat': 32981,
'aids': 2571,
'patent': 23906,
'brother like': 5164,
'like speak': 18672,
'speak treat': 29553,
'treat like': 32985,`

In [\*]: `# Create a DataFrame with the Counts features
tv_df = pd.DataFrame(tv_bi_gram.toarray(), columns=tv_bi_gram_vec.get_feature_name
tv_sums = tv_df.sum()`

In [\*]: `print(tv_sums.head())`

In [ ]:



## Ex4: Movie Review Sentiment Analysis

Cho dữ liệu train.tsv, test.tsv nằm trong thư mục movies - Rotten Tomatoes movie review dataset . Bộ dữ liệu này được sử dụng để dự đoán tình cảm của người dùng dành cho các phim.

Có các nhãn "sentiment" sau:

- 0 - negative
- 1 - somewhat negative
- 2 - neutral
- 3 - somewhat positive
- 4 - positive

### Yêu cầu:

1. Đọc dữ liệu
2. Phân tích sơ bộ dữ liệu
3. Tạo wordcloud của Positive và Negative review. In danh sách 20 từ có trọng số lớn (chữ to) trong word cloud. Trực quan hóa dữ liệu
4. Lọc lại dữ liệu, chỉ giữ lại những mẫu có phần Phrase nhận xét từ 50 ký tự trở lên. Chia dữ liệu thành 2 bộ source và target  
Đo chiều, từ comment là câu trả lời → để xem SL từ
5. Chọn phương pháp để chuẩn hóa dữ liệu và thực hiện việc chuẩn hóa (với dữ liệu đã lọc ở phần 4)

### Đọc dữ liệu

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [2]: # Dữ Liệu được download từ: https://www.kaggle.com/c/movie-review-sentiment-analysis
```



In [3]: # Đọc dữ liệu.  
df\_train = pd.read\_csv("movie\_review/train.tsv", sep='\t')  
df\_train.head()

Out[3]:

| PhraseId | SentenceId | Phrase  | Sentiment |
|----------|------------|---|-----------|
| 0        | 1          | A series of escapades demonstrating the adage ... | 1         |
| 1        | 2          | A series of escapades demonstrating the adage ... | 2         |
| 2        | 3          | A series  | 2         |
| 3        | 4          | A   | 2         |
| 4        | 5          | series  | 2         |

In [4]: df\_train.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156060 entries, 0 to 156059
Data columns (total 4 columns):
PhraseId    156060 non-null int64
SentenceId  156060 non-null int64
Phrase      156060 non-null object
Sentiment   156060 non-null int64
dtypes: int64(3), object(1)
memory usage: 4.8+ MB
```

In [5]: df\_test = pd.read\_csv("movie\_review/test.tsv", sep='\t')  
df\_test.head()

Out[5]:

| PhraseId | SentenceId | Phrase  |
|----------|------------|---|
| 0        | 156061     | An intermittently pleasing but mostly routine ... |
| 1        | 156062     | An intermittently pleasing but mostly routine ... |
| 2        | 156063     | An  |
| 3        | 156064     | Intermittently pleasing but mostly routine effort |
| 4        | 156065     | intermittently pleasing but mostly routine        |

In [6]: df\_test.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66292 entries, 0 to 66291
Data columns (total 3 columns):
PhraseId    66292 non-null int64
SentenceId  66292 non-null int64
Phrase      66292 non-null object
dtypes: int64(2), object(1)
memory usage: 1.5+ MB
```

## Phân tích sơ bộ dữ liệu

Các bạn suy nghĩ và thực hiện nhé.

phản ánh dữ liệu  
visualize  
Tạo word cloud.



# DATA PRE-PROCESSING AND ANALYSIS

## Bài 6:Tổng quan Data Analysis

Phòng LT & Mạng

[https://csc.edu.vn/lap-trinh-va-csic/Data-Pre-processing-and-Analysis\\_196](https://csc.edu.vn/lap-trinh-va-csic/Data-Pre-processing-and-Analysis_196)

2019



## Nội dung

1. Giới thiệu
2. Quy trình Data Analysis
3. Exploratory Data Analysis





## Giới thiệu

- **Data analysis** - Phân tích dữ liệu là một quá trình kiểm tra, làm sạch, chuyển đổi và mô hình hóa dữ liệu với mục tiêu khám phá thông tin hữu ích, thông báo kết luận và hỗ trợ ra quyết định. Phân tích dữ liệu có nhiều khía cạnh và cách tiếp cận, bao gồm các kỹ thuật đa dạng dưới nhiều tên khác nhau và được sử dụng trong các lĩnh vực kinh doanh, khoa học và khoa học xã hội... Trong kinh doanh, phân tích dữ liệu đóng vai trò giúp đưa ra quyết định khoa học hơn và giúp doanh nghiệp hoạt động hiệu quả hơn.

Theo: [https://en.wikipedia.org/wiki/Data\\_analysis](https://en.wikipedia.org/wiki/Data_analysis)

Data Pre-processing and Analysis

3



## Giới thiệu

- Mục tiêu của bất kỳ việc phân tích dữ liệu là để có được thông tin hành động liên quan đến công việc/doanh nghiệp... Đối với các nhà quản lý làm việc trên miền dữ liệu, để hỗ trợ các quyết định và hành động kinh doanh, việc hiểu các lý thuyết và kỹ thuật trong các lĩnh vực phân tích dữ liệu như thống kê, khai thác dữ liệu và phân tích dự đoán (statistics, data mining & predictive analytics) là rất quan trọng.



Data Pre-processing and Analysis

4



## Giới thiệu

### □ Một số kỹ thuật phân tích dữ liệu

#### • Phân tích tương quan - Correlation Analysis

▪ Correlation là một kỹ thuật thống kê có thể cho thấy mối quan hệ giữa hai biến. Nó thường được sử dụng khi hai biến có một sự tiến hóa tương tự.

▪ Ví dụ: Tốc độ lái xe và số vụ tai nạn giao thông có liên quan không? Mối quan hệ này không hoàn hảo vì mối quan hệ chính xác giữa tốc độ và sự cố phụ thuộc vào nhiều yếu tố. Tuy nhiên, theo một nghĩa chung, nếu trên đường, tốc độ trở nên cao hơn, tỷ lệ va chạm cũng sẽ tăng.

Data Pre-processing and Analysis

5



## Giới thiệu



#### • Phân tích hồi quy - Regression Analysis

độn biến or đại biến  
 $y = f(x)$ .  
Trực tuyến  
Phù tuyến

▪ Đây là một trong những kỹ thuật phân tích dữ liệu thống kê điều tra mối quan hệ giữa các biến khác nhau. Nó được sử dụng khi ngờ rằng một trong các biến có thể ảnh hưởng đến (các) biến khác. Phân tích hồi quy có thể được sử dụng khi cố gắng tạo dự báo hoặc phân tích mối quan hệ giữa các biến.

▪ Ví dụ: ảnh hưởng của giá cả tăng theo nhu cầu.



Data Pre-processing and Analysis

6



## Giới thiệu

### • Trực quan hóa dữ liệu - Data Visualization

- Trực quan hóa dữ liệu giúp mọi người hiểu được thông tin quan trọng trong dữ liệu bằng cách hiển thị nó trong bối cảnh trực quan. Đây là một trong những kỹ thuật phân tích quan trọng nhất hiện nay khi thế giới đầy dữ liệu. Nó đặc biệt hữu ích khi chúng ta tìm cách nắm bắt những hiểu biết sâu sắc từ một khối lượng lớn dữ liệu một cách nhanh chóng.



Data Pre-processing and Analysis

7

## Giới thiệu



### • Phân tích kịch bản - Scenario Analysis

Chỉa học → Sẽ hđ khi làm  
luôn vẩn TN

- Phân tích kịch bản là phân tích các sự kiện có thể xảy ra trong tương lai với kết quả thay thế. Nó được sử dụng khi chúng ta có một số lựa chọn thay thế tiềm năng nhưng không chắc chắn về quyết định đưa ra.
- Phân tích kịch bản là một công cụ quan trọng trong nhiều doanh nghiệp và được sử dụng rộng rãi để đưa ra dự đoán cho tương lai.



Data Pre-processing and Analysis

8

## Giới thiệu



### • Khai thác dữ liệu - Data mining học thêm về mảng machine learning

- Khai thác dữ liệu, đôi khi được gọi là khám phá dữ liệu/ kiến thức, là quá trình phân tích dữ liệu được thiết kế để làm việc với khối lượng dữ liệu lớn để phát hiện các mẫu, mối quan hệ hoặc thông tin có liên quan có thể cải thiện hiệu suất.
- Ví dụ: Nhà bán lẻ sử dụng khai thác dữ liệu để công ty thẻ tín dụng đề xuất sản phẩm cho chủ thẻ dựa trên phân tích chi tiêu hàng tháng của họ.



## Giới thiệu



### • Mạng neuron - Neural Networks Data Science Khoa học

- Mạng neuron là kỹ thuật lấy cảm hứng từ cách các mạng thần kinh sinh học, như não, để xử lý thông tin. Mục đích của các mạng này là mô phỏng quá trình học tập của bộ não con người trên máy tính để tạo điều kiện cho việc ra quyết định trong trí tuệ nhân tạo (AI).
- Kỹ thuật Neural Networks có thể được sử dụng để trích xuất các mẫu và phát hiện các xu hướng quá phức tạp để được xác định bởi con người hoặc các kỹ thuật máy tính khác. Mạng lưới thần kinh được đào tạo có thể xem như một chuyên gia có khả năng đưa ra các dự đoán với các tình huống đã cho và trả lời câu hỏi “what if”.





## Giới thiệu

### • A/B Testing

A/B Testing còn được gọi là thử nghiệm phân tách. Đây là một phương pháp so sánh hai phiên bản của một trang web hoặc ứng dụng với nhau để xác định phiên bản nào hoạt động tốt hơn. Kỹ thuật A/B Testing thường được sử dụng trong tiếp thị kỹ thuật số để kiểm tra phản ứng của người dùng đối với một tin nhắn và xem cái nào hoạt động tốt nhất, kiểm tra các giả thuyết trong việc ra mắt một sản phẩm mới, một chiến dịch quảng cáo hoặc một thông điệp quảng cáo.



Data Pre-processing and Analysis

(11)

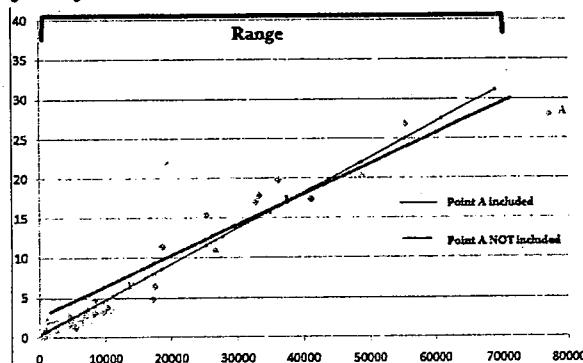
## Giới thiệu



### □ Công cụ dành cho phân tích dữ liệu

#### • Hai công cụ chính tạo nên phân tích dữ liệu là dòng và bảng.

Ví dụ: có thể tạo một line plot với phương trình hồi quy tuyến tính.



12



## Giới thiệu

- Tạo bảng phân phối tần suất (frequency distribution table) để hiển thị dữ liệu.

- Ví dụ:

| Sex    | female | male |
|--------|--------|------|
| Pclass |        |      |
| 1      | 94     | 122  |
| 2      | 76     | 108  |
| 3      | 144    | 347  |



## Giới thiệu

### ❑ Ba quy tắc phân tích dữ liệu

- Sử dụng ba quy tắc cơ bản có thể giúp chúng ta tránh đưa ra tuyên bố không chính xác về dữ liệu của mình:

- 1. Nhìn vào dữ liệu và suy nghĩ về những gì chúng ta muốn biết. Đặt câu hỏi và đóng khung câu hỏi và xem là giả thuyết. Ví dụ: Chúng ta có muốn chứng minh Trái đất hình cầu?
- 2. Ước tính một xu hướng trung tâm (Central Tendency) cho dữ liệu. Ví dụ: mean, median. Cái nào chúng sử dụng sẽ phụ thuộc vào giả thuyết của trong Bước 1. Ví dụ: nếu muốn chứng minh Trái đất hình cầu, có thể chọn khối lượng trung bình hoặc chu vi trung bình.

quý ta  
+ độ biến chia





## Giới thiệu

### ■ 3. Xem xét các ngoại lệ cho xu hướng trung tâm.

Nếu đã đo trung bình, hãy nhìn vào các số liệu không phải là trung bình. Nếu đã đo được một trung vị, hãy nhìn vào những con số mà không đáp ứng kỳ vọng đó. Ngoại lệ có thể giúp bạn phát hiện vấn đề với kết luận.

- Ví dụ: Điểm số trung bình môn Toán ở khối 6 là 70. Không tệ, phải không? Nhưng nếu nhìn vào các ngoại lệ, bạn có thể thấy họ đang nhận được 100 trong ba lớp, và 40 trong ba lớp khác. Trong trường hợp này, trung bình là hoàn toàn sai lệch.



## Giới thiệu

### □ Vấn đề của Phân tích dữ liệu

- Tại sao nhiều trường hợp phân tích dữ liệu kết thúc với tuyên bố bị lỗi? Một trong những lý do chính là phân tích dữ liệu là một quá trình phức tạp và tẻ nhạt. Nó không bao giờ dễ dàng như đưa số vào máy tính.



## Giới thiệu



• Một số vấn đề có thể dẫn đến phân tích dữ liệu bị lỗi bao gồm:

- Không có kỹ năng phân tích đúng.
- Sử dụng các công cụ sai để phân tích dữ liệu. Ví dụ: sử dụng z-score khi dữ liệu của không có phân phối chuẩn.
- Đề bias ảnh hưởng đến kết quả.
- Không tìm ra ý nghĩa thống kê.
- Nói không chính xác null hypothesis và alternate hypothesis.
- Sử dụng graph và chart không chính xác gây hiểu lầm.



## Nội dung



1. Giới thiệu
2. Quy trình Data Analysis
3. Exploratory Data Analysis





## Quy trình Data Analysis

- Phân tích đề cập đến việc chia nhỏ toàn bộ thành các thành phần riêng biệt của để kiểm tra cá nhân.
- Phân tích dữ liệu là một quá trình để có được dữ liệu thô và chuyển đổi nó thành thông tin hữu ích cho việc ra quyết định của người dùng. Dữ liệu được thu thập và phân tích để trả lời các câu hỏi, giả thuyết kiểm tra hoặc các lý thuyết không được chứng minh.



[https://en.wikipedia.org/wiki/Data\\_analysis#The\\_process\\_of\\_data\\_analysis](https://en.wikipedia.org/wiki/Data_analysis#The_process_of_data_analysis)

Data Pre-processing and Analysis

19



## Quy trình Data Analysis

- Nhà thống kê học John Tukey đã định nghĩa “data analysis” năm 1961 như sau: “Procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data.”
- Tạm dịch: “Các phương pháp phân tích dữ liệu, các kỹ thuật diễn giải kết quả của các phương pháp đó, cách lập kế hoạch thu thập dữ liệu để phân tích dữ liệu dễ dàng hơn, chính xác hơn và tất cả công cụ và kết quả thống kê (toán học) áp dụng cho phân tích dữ liệu.”



[https://en.wikipedia.org/wiki/Data\\_analysis#The\\_process\\_of\\_data\\_analysis](https://en.wikipedia.org/wiki/Data_analysis#The_process_of_data_analysis)

Data Pre-processing and Analysis

20

# Quy trình Data Analysis



## □ Quy trình.

### • Data requirements

- Dữ liệu là cần thiết để làm đầu vào cho phân tích, được chỉ định dựa trên yêu cầu của những người chỉ đạo phân tích hoặc khách hàng (những người sẽ sử dụng thành phẩm của phân tích). Mẫu mà dữ liệu sẽ được thu thập được gọi là một đơn vị thử nghiệm (ví dụ: một người hoặc quần thể). Các biến cụ thể liên quan đến người (ví dụ: tuổi và thu nhập) có thể được chỉ định và thu được. Dữ liệu có thể là numerical hoặc categorical.

Data Pre-processing and Analysis

21

# Quy trình Data Analysis



### • Data collection

- Dữ liệu được thu thập từ nhiều nguồn khác nhau. Các yêu cầu có thể được các nhà phân tích truyền đạt tới người giám sát dữ liệu, chẳng hạn như nhân viên công nghệ thông tin trong một tổ chức. Dữ liệu cũng có thể được thu thập từ các cảm biến trong môi trường, chẳng hạn như camera giao thông, vệ tinh, thiết bị ghi... Nó cũng có thể được lấy qua các cuộc phỏng vấn, tải xuống từ các nguồn trực tuyến hoặc tài liệu đọc.

Data Pre-processing and Analysis

22



## Quy trình Data Analysis

### • Data processing

- Dữ liệu thu được ban đầu phải được xử lý hoặc tổ chức để phân tích. Ví dụ: một việc liên quan có thể là đặt dữ liệu vào các hàng và cột theo định dạng bảng (dữ liệu có cấu trúc) để phân tích thêm...



## Quy trình Data Analysis

### • Data cleaning

- Sau khi được xử lý và sắp xếp, dữ liệu có thể không đầy đủ, chứa các bản sao hoặc chứa lỗi. Nhu cầu làm sạch dữ liệu sẽ phát sinh từ các vấn đề trong cách dữ liệu được nhập và lưu trữ. Làm sạch dữ liệu là quá trình ngăn ngừa và sửa chữa các lỗi này. Các tác vụ phổ biến bao gồm đối sánh bản ghi, xác định tính không chính xác của dữ liệu, chất lượng tổng thể của dữ liệu hiện có, sao chép và phân đoạn cột. Các vấn đề dữ liệu như vậy cũng có thể được xác định thông qua một loạt các kỹ thuật phân tích.

- Ví dụ, với thông tin tài chính, tổng số của các biến cụ thể có thể được so sánh với các số được công bố riêng biệt được cho là đáng tin cậy. Số tiền bất thường trên hoặc dưới ngưỡng xác định trước cũng có thể được xem xét.





## Quy trình Data Analysis

### • Data cleaning

- Có một số loại làm sạch dữ liệu phụ thuộc vào loại dữ liệu như số điện thoại, địa chỉ email, nhà tuyển dụng... Phương pháp dữ liệu định lượng (Quantitative data) để phát hiện ngoại lệ có thể được sử dụng để loại bỏ dữ liệu nhập sai.



## Quy trình Data Analysis

### • Exploratory data analysis

- Một khi dữ liệu được làm sạch, nó có thể được phân tích. Các nhà phân tích có thể áp dụng một loạt các kỹ thuật được gọi là phân tích dữ liệu thăm dò để bắt đầu hiểu các thông điệp có trong dữ liệu. Quá trình thăm dò có thể dẫn đến việc làm sạch dữ liệu bổ sung hoặc yêu cầu bổ sung cho dữ liệu, do đó, các hoạt động này có thể lặp đi lặp lại về bản chất. Thống kê mô tả, như trung bình hoặc trung vị, có thể được tạo để giúp hiểu dữ liệu. Trực quan hóa dữ liệu cũng có thể được sử dụng để kiểm tra dữ liệu ở định dạng đồ họa, để có được cái nhìn sâu sắc về các thông điệp trong dữ liệu.





## Quy trình Data Analysis

### • Modeling & algorithms

- Các công thức hoặc mô hình toán học được gọi là thuật toán có thể được áp dụng cho dữ liệu để xác định mối quan hệ giữa các biến, chẳng hạn như tương quan hoặc quan hệ nhân quả. Nói chung, các mô hình có thể được phát triển để đánh giá một biến cụ thể trong dữ liệu dựa trên (các) biến khác trong dữ liệu, với một số lỗi còn lại tùy thuộc vào độ chính xác của mô hình (Data = Model + Error).



## Quy trình Data Analysis

- Thống kê suy luận bao gồm các kỹ thuật để đo lường mối quan hệ giữa các biến cụ thể.
  - Ví dụ: phân tích hồi quy có thể được sử dụng để mô hình hóa liệu thay đổi trong quảng cáo (biến độc lập X) giải thích sự thay đổi trong doanh số (biến phụ thuộc Y). Theo thuật ngữ toán học, Y (bán hàng) là một chức năng của X (quảng cáo). Nó có thể được mô tả là  $Y = aX + b + \text{error}$ , trong đó mô hình được thiết kế sao cho a và b giảm thiểu lỗi khi mô hình dự đoán Y cho một phạm vi giá trị nhất định của X.
- Các nhà phân tích có thể cố gắng xây dựng các mô hình mô tả dữ liệu để đơn giản hóa phân tích và truyền đạt kết quả.

Những từ là mô hình sử dụng biến ít nhất  
chính xác.



## Quy trình Data Analysis



### • Data product

→ Khi gán trên web  
|| app  
} vai trò dev

- Một sản phẩm dữ liệu là một ứng dụng máy tính nhận dữ liệu đầu vào và tạo đầu ra, đưa chúng trở lại môi trường. Nó có thể dựa trên một mô hình hoặc thuật toán. Một ví dụ là một ứng dụng phân tích dữ liệu về lịch sử mua hàng của khách hàng và khuyến nghị các giao dịch mua khác mà khách hàng có thể được hưởng.



## Quy trình Data Analysis



### • Communication

- Sau khi dữ liệu được phân tích, nó có thể được báo cáo theo nhiều định dạng cho người dùng để hỗ trợ các yêu cầu của họ. Người dùng có thể có phản hồi, dẫn đến phân tích bổ sung. Như vậy, phần lớn chu trình phân tích là lặp lại.





## Quy trình Data Analysis

- Khi xác định cách truyền đạt kết quả, nhà phân tích có thể xem xét các kỹ thuật trực quan hóa dữ liệu để giúp truyền đạt thông điệp rõ ràng và hiệu quả đến khán giả. Trực quan hóa dữ liệu sử dụng hiển thị thông tin (như bảng và biểu đồ) để giúp truyền đạt các thông điệp chính có trong dữ liệu. Các bảng hữu ích cho người dùng có thể tra cứu các số cụ thể, trong khi các biểu đồ (ví dụ: barplot, line plot...) có thể giúp giải thích các thông điệp định lượng (quantitative messages) có trong dữ liệu



## Nội dung

- Giới thiệu
- Quy trình Data Analysis
- Exploratory Data Analysis (EDA) *giør dữ liệu khám phá*



## Exploratory Data Analysis (EDA)



### □ Giới thiệu

- EDA là một cách tiếp cận để phân tích dữ liệu. EDA là nơi nhà nghiên cứu có một cái nhìn về dữ liệu và có gắng hiểu ý nghĩa của nó.
- EDA đề cập đến quá trình quan trọng của việc thực hiện điều tra ban đầu về dữ liệu để khám phá các mẫu, phát hiện dị thường, kiểm tra giả thuyết và kiểm tra các giả định với sự trợ giúp của thống kê tóm tắt và biểu diễn đồ họa.



## Exploratory Data Analysis (EDA)



- EDA liên quan đến việc nhà phân tích cố gắng để có được một cảm giác của người dùng cho bộ dữ liệu, thường sử dụng phán đoán của chính họ để xác định các yếu tố quan trọng nhất trong bộ dữ liệu là gì.
  - Ví dụ, chia tỷ lệ đa chiều (multidimensional scaling) là một EDA sử dụng các biểu diễn trực quan về khoảng cách hoặc điểm tương đồng giữa các bộ đối tượng; Nó hỗ trợ người dùng giải thích chính xác những gì khoảng cách thể hiện.





## Exploratory Data Analysis (EDA)

### ❑ Mục đích của EDA

- Kiểm tra dữ liệu bị thiếu và các lỗi khác.
- Có được cái nhin sâu sắc tối đa về tập dữ liệu và cấu trúc cơ bản của nó.
- Khám phá một mô hình tốt, mô hình giải thích dữ liệu với số lượng biến dự đoán tối thiểu.
- Kiểm tra các giả định liên quan đến bất kỳ mô hình phù hợp hoặc kiểm tra giả thuyết.
- Tạo một danh sách các ngoại lệ hoặc dị thường khác.
- Tìm ước tính tham số và khoảng tin cậy (confidence interval) liên quan hoặc sai số.
- Xác định các biến có ảnh hưởng nhất.



## Exploratory Data Analysis (EDA)

- Kiến thức cụ thể khác có thể có được thông qua EDA như tạo danh sách xếp hạng các yếu tố liên quan. Có thể không nhất thiết phải có tất cả các mục trên trong phân tích dữ liệu.





## Data Pre-processing and Analysis

37



## CHAPTER 6: OVERVIEW OF ANALYSIS

1. The optimal stage for determining appropriate analyses procedures occurs late in the research process.
  - A. True
  - B. False
2. Statistical analysis advice should be obtained at the stage of initial planning in a study:
  - A. so that attribution of authorship can be decided
  - B. so that conflicts of interest could be identified
  - C. to better coordinate the selection of appropriate sampling methods and data collection instruments
  - D. how data will be archived can be planned
3. Lack of clearly defined and objective outcome measurements can be overcome by sophisticated and robust statistical analysis tools
  - A. True
  - B. False
4. Data analysis is defined as the:
  - A. process to ensure that research data, digital and traditional, is stored in a secure manner to ensure that procedural controls are in place and adhered to in order to protect the integrity of data.
  - B. convention whereby research findings are prepared and disseminated to the scientific community.
  - C. process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.
  - D. the process of systematically applying statistical and/or logical techniques to describe and illustrate, condense, recap, and evaluate data.



5. The chief aim of analysis is to distinguish between:
  - A. an event occurring as either reflecting a true effect versus a one occurring by chance
  - B. right from wrong
  - C. quality control and quality assurance
  - D. misrepresentation and plagiarism
6. Quantitative data refers to:
  - A. statistical analysis.
  - B. any data you present in your report.
  - C. graphs and tables.
  - D. numerical data that could usefully be quantified to help you answer your research question(s) and to meet your objectives.
7. Computers are essential for quantitative data analysis because:
  - A. they enable easy calculation for those of us not too good with figures.
  - B. they are fun to use.
  - C. increasingly data analysis software contain algorithms that check the data for obvious errors as it is entered.
  - D. they are so powerful.
8. A pie chart is:
  - A. an illustration where the data are divided into proportional segments according to the share each has of the total value of the data.
  - B. only used in catering management research.
  - C. any form of pictorial representation of data.
  - D. a chart demonstrating the increasing incidence of obesity in society.
9. Which one of these is not a way of measuring central tendency?
  - A. Regression analysis.



B. Measuring the middle value or mid-point after the data have been ranked (median).

C. Measuring the value that occurs most frequently (mode).

D. Measuring the value, often known as the average, that includes all data values in its calculation (mean).

10. Standard deviation is:

A. a way of describing those phenomena that are not the norm.

B. inappropriate in management and business research.

C. a way of measuring the extent of spread of quantifiable data.

D. a way of illustrating crime statistics.

11. A correlation coefficient enables you to:

A. measure the difference between two variables.

B. quantify the strength of the linear relationship between two ranked or quantifiable variables.

C. establish whether the data is telling you what you think it should tell you.

D. assess whether two variables measure the same phenomenon.

12. Which of these is not one of the four main reasons for missing data?

A. The data was not required from the respondent, perhaps because of a skip generated by a filter question in a survey.

B. The analyst ignored its presence on the data form.

C. The respondent may have missed a question by mistake.

D. The respondent did not know the answer or did not have an opinion.

13. Quantitative data can be divided into two distinct groups: categorical and quantifiable.

A. True

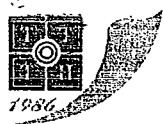
B. False

14. A bell-shaped curve is also known as the normal distribution.



- A. True
  - B. False
15. The mean is the value that occurs most frequently in a distribution.
- A. True
  - B. False
16. 24. Quartiles and percentiles are measures which state the difference between values.
- A. True
  - B. False





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

# DATA PRE-PROCESSING AND ANALYSIS

## Bài 7: EDA – Handling Imbalanced Dataset

Phòng LT & Mạng

[https://csc.edu.vn/lap-trinh-va-cSDL/Data-Pre-processing-and-Analysis\\_196](https://csc.edu.vn/lap-trinh-va-cSDL/Data-Pre-processing-and-Analysis_196)

2019



## Nội dung

1. Giới thiệu
2. Chiến thuật làm việc với dữ liệu mất cân bằng (Handling Imbalanced Dataset)



# Giới thiệu



## □ Đặt vấn đề

### • Vấn đề 1

- Bạn đang làm việc trên tập dữ liệu. Bạn tạo một mô hình phân loại (classification model) và nhận được độ chính xác 90% ngay lập tức. Tốt quá! Nhưng đến khi bạn tìm hiểu sâu hơn một chút thì phát hiện ra rằng 90% dữ liệu thuộc về một lớp. Rồi xong!
- Đây là một ví dụ về một bộ dữ liệu mất cân bằng và kết quả mà nó có thể gây ra.

Classification  
90% data belong to one class



# Giới thiệu



## □ Đặt vấn đề

### • Vấn đề 2

- Giả sử bạn đang làm việc trong một công ty và bạn được yêu cầu tạo một mô hình, dựa trên các phép đo khác nhau theo ý của bạn, mô hình sẽ dự đoán liệu sản phẩm có bị lỗi hay không. Bạn quyết định sử dụng trình phân loại yêu thích của mình, huấn luyện nó trên dữ liệu và thật tuyệt: bạn có độ chính xác 96,2%!
- Sếp của bạn rất ngạc nhiên và quyết định sử dụng mô hình của bạn mà không cần kiểm tra thêm. Vài tuần sau anh ta vào văn phòng của bạn và cho bạn "một trận" vì sự vô dụng của mô hình bạn đã làm. Sự thật là, mô hình bạn tạo không tìm thấy bất kỳ sản phẩm bị lỗi nào kể từ khi nó được sử dụng trong sản xuất. Sau khi kiểm tra lại, bạn phát hiện ra rằng chỉ có khoảng 3.8% sản phẩm do công ty của bạn sản xuất bị lỗi và mô hình của bạn luôn trả lời không bị lỗi, dẫn đến độ chính xác 96,2%. Các kết quả mà bạn thu được là do bộ dữ liệu không cân bằng mà bạn đang làm việc. Ác mộng!



## Giới thiệu



- Phân loại là một trong những vấn đề machine learning phổ biến nhất. Cách tốt nhất để tiếp cận bất kỳ vấn đề phân loại nào là bắt đầu bằng cách phân tích và khám phá bộ dữ liệu - Exploratory Data Analysis (EDA). Mục đích của công việc này là tạo ra càng nhiều insight và thông tin về dữ liệu càng tốt. EDA cũng được sử dụng để tìm ra các vấn đề có thể tồn tại trong bộ dữ liệu. Một trong những vấn đề phổ biến được tìm thấy trong các bộ dữ liệu được sử dụng để phân loại là vấn đề về các lớp không cân bằng (imbalanced classes).



Data Pre-processing and Analysis

Xem trang nhóm : bn mău  $\rightarrow$  nếu có tỷ chênh lệch lớn thì phải xem xét xử lý  
cân bằng data.  $\Rightarrow$  kiểm count value.



## Giới thiệu

- Mất cân bằng dữ liệu (Data Imbalance) là gì?



- Mất cân bằng dữ liệu thường phản ánh sự phân phối không đồng đều của các lớp trong một tập dữ liệu.

- Ví dụ 1: trong bộ dữ liệu phát hiện gian lận thẻ tín dụng, hầu hết các giao dịch thẻ tín dụng không phải là gian lận và rất ít là giao dịch gian lận. Điều này khiến chúng ta có tỷ lệ chênh lệch lớn (50:1... 1000:1...) giữa không gian lận : gian lận.

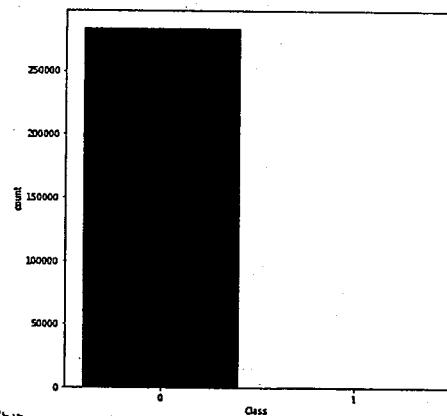


Data Pre-processing and Analysis



## Giới thiệu

- Như chúng ta có thể thấy, các giao dịch không gian lận vượt xa các giao dịch gian lận. Nếu chúng ta đào tạo một mô hình phân loại nhi phân mà không khắc phục vấn đề này, mô hình sẽ hoàn toàn sai lệch. Nó cũng tác động đến mối tương quan giữa các tính năng trong dataset.



Data Pre-processing and Analysis



## Giới thiệu

- Ví dụ 2: Bạn có thể có vấn đề phân loại 2 lớp (nhi phân) với 1000 trường hợp (dòng). Tổng cộng có 800 trường hợp được gắn nhãn Class-1 và 200 trường hợp còn lại được gắn nhãn Class-2. Đây là một bộ dữ liệu không cân bằng và tỷ lệ của các trường hợp Class-1 và Class-2 là 800:200 (hoặc 4:1).



## Giới thiệu



### ❑ Mất cân bằng là một vấn đề phổ biến

- Hầu hết các tập dữ liệu phân loại không có số lượng mẫu chính xác bằng nhau trong mỗi lớp, nhưng một sự chênh lệch nhỏ thường không quan trọng.
- Có những vấn đề mà sự mất cân bằng giữa các lớp rất phổ biến, hiển nhiên.
  - Ví dụ: trong các bộ dữ liệu đặc trưng cho các giao dịch bị mất cân bằng: phần lớn các giao dịch sẽ thuộc về lớp “Not-Fraud” và thiểu số rất nhỏ sẽ thuộc lớp “Fraud”
- Khi có sự mất cân bằng lớp 4: 1 như ví dụ trên => có thể gây ra vấn đề.



## Giới thiệu



### ❑ Nghịch lý chính xác (Accuracy Paradox)

- “Nghịch lý chính xác” là tên gọi cho trường hợp đo lường độ chính xác có kết quả chính xác tuyệt vời (ví dụ 90%), nhưng độ chính xác này chỉ phản ánh cho một lớp, mà không phải là tất cả.
- Nó rất phổ biến, bởi vì độ chính xác phân loại thường là thước đo đầu tiên chúng ta sử dụng khi đánh giá các mô hình phân loại.



## Nội dung



1. Giới thiệu
2. Chiến thuật làm việc với dữ liệu mất cân bằng (Handling Imbalanced Dataset)



## Chiến thuật làm việc với dữ liệu mất cân bằng



### □ Thu thập thêm dữ liệu

- Đây là một việc nên làm nhưng hầu như luôn bị bỏ qua.
- Hãy đặt cho mình câu hỏi: “Liệu có thể thu thập thêm dữ liệu về vấn đề của mình không không?” => Dành thời gian suy nghĩ và trả lời câu hỏi này.
- Một tập dữ liệu lớn hơn có thể cho thấy một quan điểm khác biệt và cân bằng hơn về các lớp.





## Chiến thuật làm việc với dữ liệu mất cân bằng

### □ Thay đổi Performance Metric

- Độ chính xác không phải là số liệu được dùng khi làm việc với bộ dữ liệu không cân bằng. Bởi vì nó bị sai lệch.
- Có những số liệu đã được thiết kế để cho chúng ta biết cách chân thực hơn khi làm việc với các lớp không cân bằng.

ham → 0 → spam

|                  |     |                            |
|------------------|-----|----------------------------|
| Confusion Matrix | → 0 | 1 → spam                   |
|                  | 0   | True Negatives (TN) : 460  |
|                  | 1   | False Positives (FP) : 200 |
|                  |     | False Negatives (FN) : 100 |

Tổng: 800 mail  
Nhìn: 2000 mail  
spam: 1000 mail

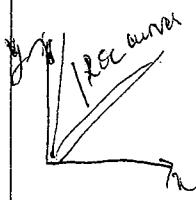
Data Pre-processing and Analysis

13



## Chiến thuật làm việc với dữ liệu mất cân bằng

- Các biện pháp hiệu suất sau đây có thể cung cấp cái nhìn sâu sắc hơn về độ chính xác của mô hình so với độ chính xác phân loại truyền thống:
  - Confusion Matrix: Phân tích các dự đoán thành một bảng hiển thị các dự đoán chính xác (trên đường chéo) và các loại dự đoán không chính xác (các lớp dự đoán không chính xác đã được chỉ định).
    - Precision: Thước đo độ chính xác của phân loại.
    - Recall: Thước đo của tính đầy đủ của phân loại
    - F1 Score (F-score): Trung bình của precision và recall.
  - ROC Curves: Giống như precision & recall, độ chính xác được chia thành độ nhạy (sensitivity) và độ đặc hiệu (specificity) và các mô hình có thể được chọn dựa trên ngưỡng cân bằng của các giá trị này.



Data Pre-processing and Analysis

14

## Chiến thuật làm việc với dữ liệu mất cân bằng

### • Confusion Matrix

- Cách tốt và đơn giản thường được sử dụng khi xử lý vấn đề phân loại là confusion matrix. Số liệu này cung cấp một cái nhìn tổng quan thú vị về việc một mô hình đang hoạt động tốt hay không. Vì vậy, nó là một điểm khởi đầu tuyệt vời cho bất kỳ đánh giá mô hình phân loại.



## Chiến thuật làm việc với dữ liệu mất cân bằng

Tóm tắt hầu hết các số liệu có thể được lấy từ confusion matrix như sau:

|                       |  | Predicted label<br>class 1 | Predicted label<br>class 2 |
|-----------------------|--|----------------------------|----------------------------|
| True label<br>class 1 | 600                                    | 600                        | 400                        |
|                       | 600                                    | 600                        | 400                        |
| True label<br>class 2 | 600                                    | 600                        | 400                        |
|                       | wrong<br>false positive<br>for class 1 |                            |                            |

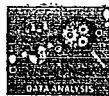
$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\times - \text{class 1} = \frac{[0,0]}{[0,0] + [1,0]} = \frac{8560}{800 + 8560} = 86.2\%$$

$$\begin{aligned}\text{class 1 precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{class 2 precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}\end{aligned}$$

$$\begin{aligned}\text{class 1 recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ \text{class 2 recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}\end{aligned}$$





## Chiến thuật làm việc với dữ liệu mất cân bằng

### ■ Giải thích

- Độ chính xác của mô hình về cơ bản là tổng số dự đoán đúng chia cho tổng số dự đoán.
- Độ chính xác (precision) của một lớp xác định mức độ tin cậy là kết quả khi mô hình trả lời một điểm thuộc về lớp đó.
- Recall của một lớp thể hiện mức độ tốt của mô hình có thể phát hiện lớp đó.
- F1-score của một lớp được tính bằng  $(2 \times \text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$ , nó kết hợp precision và recall của một lớp trong một metric.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{class 1 precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{class 1 recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{class 2 precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{class 2 recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$



$$F_1(\text{Class 0}) = \frac{2 \times 86,2\% \times 94,5\%}{86,2\% + 94,5\%}$$

17

$$= 90,4\%$$

$$F_1(\text{Class 1}) = 24,4\%$$



## Chiến thuật làm việc với dữ liệu mất cân bằng

- Đối với một lớp nhất định, các kết hợp recall và precision khác nhau có các ý nghĩa sau:

- high recall + high precision: lớp được xử lý hoàn hảo bởi mô hình
- low recall + high precision: mô hình có thể không phát hiện tốt lớp nhưng rất đáng tin cậy khi nó thực hiện
- high recall + low precision: lớp được phát hiện tốt nhưng mô hình cũng bao gồm các điểm của các lớp khác trong đó
- low recall + low precision: lớp được xử lý kém bởi mô hình



## Chiến thuật làm việc với dữ liệu mất cân bằng



- Với ví dụ công ty ở Vấn đề 2 phía trên, chúng ta có confusion matrix cho 10000 sản phẩm.

|                         |                                       | Predicted label<br>defective | Predicted label<br>not defective |
|-------------------------|---------------------------------------|------------------------------|----------------------------------|
| True label<br>defective | defective                             | 9620                         | 380                              |
|                         | not defective                         | 0                            | 9620                             |
|                         |                                       | đúng                         |                                  |
| accuracy =              | $\frac{9620 + 0}{9620 + 380 + 0 + 0}$ | defective =                  | $\frac{0}{0 + 380}$              |
| not defective =         | $\frac{9620}{380 + 9620}$             | not defective =              | $\frac{9620}{9620 + 0}$          |
| precision               |                                       | recall                       |                                  |

19

## Chiến thuật làm việc với dữ liệu mất cân bằng



- Độ chính xác là 96.2% như đã nói trước đó.
- Độ chính xác của lớp không bị lỗi là 96.2% và độ chính xác của lớp bị lỗi là không thể tính toán được.
- Recall của lớp không bị lỗi = 1.0 là hoàn hảo (tất cả các sản phẩm không bị lỗi đã được dán nhãn như vậy).
- Nhưng recall của lớp bị lỗi là 0.0, đây là trường hợp xấu hơn (không phát hiện thấy sản phẩm bị lỗi).
- Vì vậy, chúng ta có thể kết luận mô hình hoạt động không tốt cho lớp này. F1-score không thể tính được cho các sản phẩm bị lỗi, và F1-score = 0.981 cho các sản phẩm không bị lỗi.
- Trong ví dụ này, nhìn vào confusion matrix có thể dẫn đến suy nghĩ lại mô hình hoặc mục tiêu.



## Chiến thuật làm việc với dữ liệu mất cân bằng

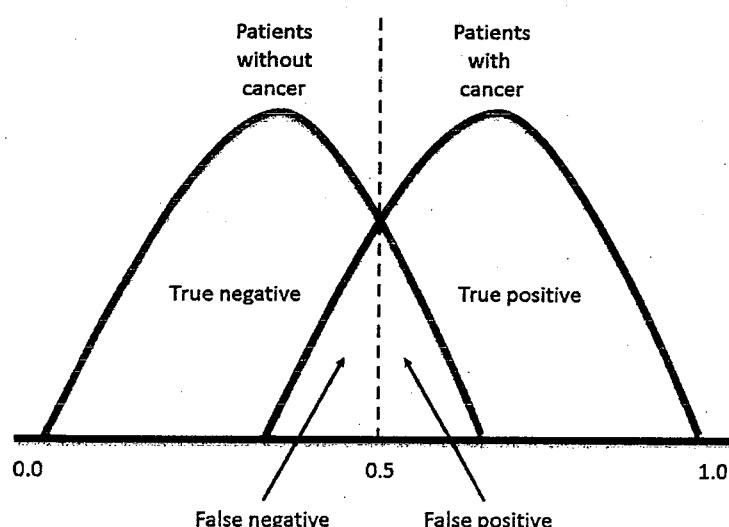


### • ROC Curves

- Hầu hết các phân loại tạo ra một điểm số, sau đó so với ngưỡng (threshold) để quyết định phân loại. Nếu một bộ phân loại tạo ra điểm số giữa 0.0 (chắc chắn là negative) và 1.0 (chắc chắn là positive với thông thường coi mọi thứ > 0.5 là positive).
- Tuy nhiên, bất kỳ ngưỡng nào cũng được áp dụng cho bộ dữ liệu (trong đó PP là positive population và NP là negative population) sẽ tạo ra true positive (TP), false positive (FP), true negative (TN) và false negative (FN) như hình tiếp theo. Chúng ta cần một phương pháp sẽ tính đến tất cả những con số này.



## Chiến thuật làm việc với dữ liệu mất cân bằng



Các bộ dữ liệu chòng chéo sẽ luôn tạo ra các kết quả false positive và false negative như true positive và true negative





## Chiến thuật làm việc với dữ liệu mất cân bằng

- Khi ta có số cho tất cả các đo lường này, một số số liệu hữu ích có thể được tính toán.

- $\text{Accuracy} = (1 - \text{Error}) = (\text{TP} + \text{TN}) / (\text{PP} + \text{NP}) = \text{Pr}(C)$ , xác suất phân loại chính xác
- $\text{Sensitivity} = \text{TP} / (\text{TP} + \text{FN}) = \text{TP} / \text{PP} = \text{kết quả dương}$  phát hiện “A” trong quần thể “A”
- $\text{Specificity} = \text{TN} / (\text{TN} + \text{FP}) = \text{TN} / \text{NP} = \text{kết quả âm}$  không phát hiện “A” trong quần thể “not A”.



## Chiến thuật làm việc với dữ liệu mất cân bằng

- Ví dụ: Nếu chúng ta có 100.000 bệnh nhân, trong đó 200 ( $20\%$ ) thực sự bị ung thư, chúng ta có thể thấy các kết quả xét nghiệm:

|                  | Test Positive | Test Negative | Total  |
|------------------|---------------|---------------|--------|
| Patient Diseased | 160           | 40            | 200    |
| Patient Healthy  | 29940         | 69860         | 99800  |
| Total            | 30100         | 69900         | 100000 |

$$\text{Sensitivity} = \text{TP} / (\text{TP} + \text{FN}) = 160 / (160 + 40) = 80.0\%$$

$$\text{Specificity} = \text{TN} / (\text{TN} + \text{FP}) = 69,860 / (69,860 + 29,940) = 70.0\%$$

Xét nghiệm xác định chính xác 80% người mắc bệnh, nhưng 30% người khỏe mạnh sẽ có test positive không chính xác. Bằng cách chỉ xem xét Sensitivity (hoặc accuracy) của thử nghiệm, thông tin quan trọng có khả năng sẽ bị mất. Bằng cách xem xét các kết quả sai cũng như kết quả chính xác, chúng ta hiểu rõ hơn về hiệu suất của mô hình phân loại.



## Chiến thuật làm việc với dữ liệu mất cân bằng



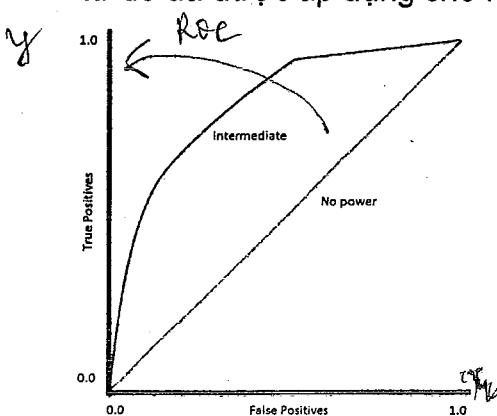
- Một cách để khắc phục vấn đề phải chọn cutoff là bắt đầu với ngưỡng 0.0, để mọi trường hợp được coi là positive. Chúng ta phân loại chính xác tất cả các trường hợp positive và phân loại không chính xác tất cả các trường hợp negative. Sau đó, di chuyển ngưỡng trên mọi giá trị trong khoảng từ 0.0 đến 1.0, giảm dần số lượng false positive và tăng số lượng true positive.



## Chiến thuật làm việc với dữ liệu mất cân bằng



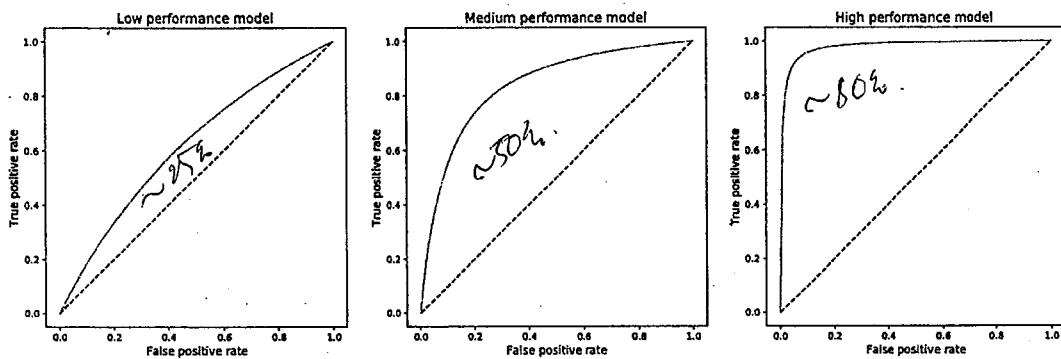
- TP (sensitivity) sau đó có thể được vẽ theo FP (1 – specificity) cho từng ngưỡng được sử dụng. Biểu đồ kết quả được gọi là Receiver Operating Characteristic (ROC) curve. Các đường cong ROC được phát triển để sử dụng trong việc phát hiện tín hiệu trở lại của radar năm 1950, và từ đó đã được áp dụng cho một loạt các vấn đề.



Với một phân loại hoàn hảo, đường cong ROC sẽ đi thẳng lên trục Y và sau đó dọc theo trục X. Một bộ phân loại no power (không có sức mạnh) sẽ nằm trên đường chéo, trong khi hầu hết các phân loại nằm đâu đó ở giữa trục Y và đường chéo.



## Chiến thuật làm việc với dữ liệu mất cân bằng



Các đường cong ROC có thể tùy thuộc vào hiệu quả của mô hình.  
Ở bên trái, model phải hy sinh rất nhiều precision để có được recall cao. Ở bên phải, mô hình có hiệu quả cao: nó có thể đạt được recall cao trong khi vẫn giữ precision cao.



## Chiến thuật làm việc với dữ liệu mất cân bằng



- Dựa trên đường cong ROC, chúng ta có thể xây dựng một số liệu khác, dễ sử dụng hơn, để đánh giá mô hình là AUROC (Area Under the ROC curve). AUROC hoạt động như một giá trị vô hướng tóm tắt toàn bộ đường cong ROC. AUROC hướng tới 1.0 cho trường hợp tốt nhất và hướng tới 0.5 cho trường hợp xấu nhất.
- Điểm AUROC tốt có nghĩa là mô hình mà chúng ta đang đánh giá không hy sinh nhiều precision để có được recall tốt về lớp được quan sát (thường là lớp thiểu số - minority class).





## Chiến thuật làm việc với dữ liệu mất cân bằng

### □ Lấy mẫu lại bộ dữ liệu (Resampling Dataset)

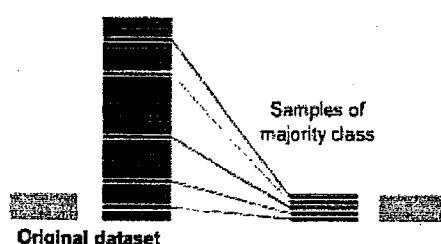
- Chúng ta có thể thay đổi tập dữ liệu mà ta sử dụng để xây dựng mô hình dự đoán, để có dữ liệu cân bằng hơn.
- Thay đổi này được gọi là lấy mẫu dữ liệu và có hai phương thức chính mà chúng ta có thể sử dụng:
  - Thêm các bản sao của các thể hiện từ lớp đại diện dưới mức (under-represented class) được gọi là over-sampling (hoặc lấy mẫu hơn chính thức với sự thay thế), hoặc
  - Có thể xóa các thể hiện khỏi lớp được đại diện quá mức (over-represented class), được gọi là under-sampling.



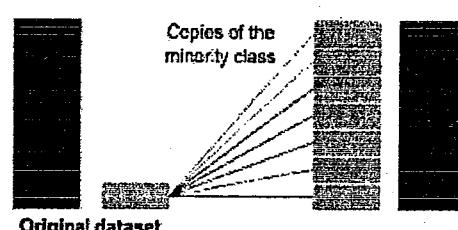
## Chiến thuật làm việc với dữ liệu mất cân bằng



**Undersampling**



**Oversampling**



## Chiến thuật làm việc với dữ liệu mất cân bằng



- Những cách tiếp cận này thường rất dễ thực hiện và nhanh chóng thực thi. Đây cũng là một giải pháp khởi đầu tốt.
- Trên thực tế, chúng ta nên thử cả hai cách tiếp cận trên cho tất cả các bộ dữ liệu mất cân bằng, để xem liệu nó có giúp chúng ta tăng cường các đo lường chính xác không.



Xem thêm: [https://en.wikipedia.org/wiki/Oversampling\\_and\\_undersampling\\_in\\_data\\_analysis](https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis)  
Data Pre-processing and Analysis

31

## Chiến thuật làm việc với dữ liệu mất cân bằng



### • Một số nguyên tắc:

- Cân nhắc kiểm tra việc lấy mẫu under-sampling khi có nhiều dữ liệu (hàng chục hoặc hàng trăm nghìn trường hợp trở lên)
- Cân nhắc kiểm tra việc lấy mẫu over-sampling khi không có nhiều dữ liệu (hàng chục nghìn trường hợp hoặc ít hơn)
- Xem xét thử nghiệm các scheme lấy mẫu ngẫu nhiên (random) và không ngẫu nhiên (non-random) (ví dụ: phân tầng).
- Xem xét thử nghiệm các tỷ lệ lấy mẫu khác nhau (ví dụ: ngoài tỷ lệ 1:1 trong bài toán phân loại nhị phân, hãy thử các tỷ lệ khác)



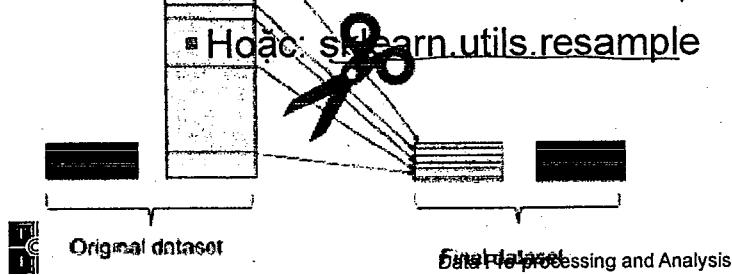


## Chiến thuật làm việc với dữ liệu mất cân bằng

### • Under-sampling

- Là quá trình xóa ngẫu nhiên một số mẫu khỏi lớp đa số (majority class) để khớp số lượng với lớp thiểu số (minority class).

- Lập trình: Sử dụng
  - `majority`
  - `imblearn.under_sampling.ClusterCentroids` hoặc `RandomUnderSample,...` (`pip install imblearn`)
  - Hoặc: `sklearn.utils.resample`



33



## Chiến thuật làm việc với dữ liệu mất cân bằng

### • Ví dụ:

```
from sklearn import datasets
import pandas as pd
import numpy as np
from collections import Counter
from sklearn.datasets import make_classification
import seaborn as sns
import matplotlib.pyplot as plt

X, y = make_classification(n_samples=15000, n_features=2, n_informative=2,
                           n_redundant=0, n_repeated=0, n_classes=2,
                           n_clusters_per_class=1,
                           weights=[0.05, 0.95],
                           class_sep=0.8, random_state=0)

print(sorted(Counter(y).items()))
[(0, 815), (1, 14185)]

type(X), type(y)
(numpy.ndarray, numpy.ndarray)
```



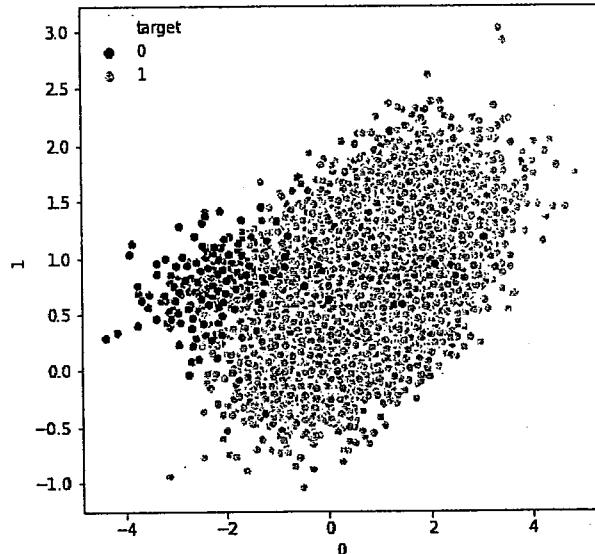
Data Pre-processing and Analysis

34

## Chiến thuật làm việc với dữ liệu mất cân bằng

```
data = pd.DataFrame(X)
data['target'] = y
data.head()
plt.figure(figsize=(6,6))
sns.scatterplot(data=data, x=0, y=1, hue='target')
<matplotlib.axes._subplots.AxesSubplot at 0x1ba27c70240>
```

|   | 0         | 1        | target |
|---|-----------|----------|--------|
| 0 | -2.473300 | 0.569620 | 0      |
| 1 | -1.283776 | 0.576974 | 1      |
| 2 | 1.428302  | 0.629649 | 1      |
| 3 | 2.071399  | 1.389588 | 1      |
| 4 | 1.247005  | 1.123625 | 1      |



Data Pre-processing and Analysis

35

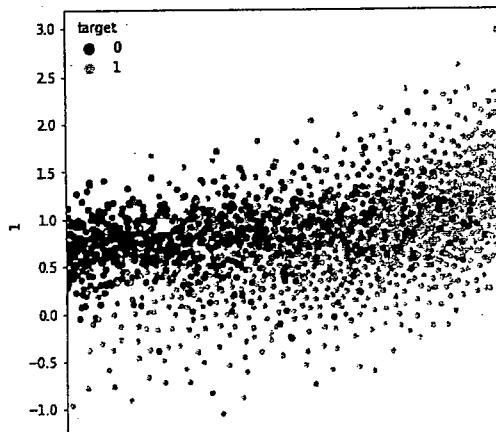
## Chiến thuật làm việc với dữ liệu mất cân bằng

### ClusterCentroids

```
# pip install imblearn
from imblearn.under_sampling import ClusterCentroids
cc = ClusterCentroids(random_state=0) → lấy mảng yêu cầu, sau đó, ta lấy mảng
X_resampled, y_resampled = cc.fit_resample(X, y)
print(sorted(Counter(y_resampled).items()))
[(0, 815), (1, 815)]
plt.figure(figsize=(6,6))
sns.swarmplot(data=data_new, x=0, y=1, hue='target')
<matplotlib.axes._subplots.AxesSubplot at 0x1ba29a02128>
```

```
data_new = pd.DataFrame(X_resampled)
data_new['target'] = y_resampled
data_new.head()
```

|   | 0         | 1        | target |
|---|-----------|----------|--------|
| 0 | -2.473300 | 0.569620 | 0      |
| 1 | -1.655427 | 1.168923 | 0      |
| 2 | -0.012906 | 0.989231 | 0      |
| 3 | -0.841488 | 0.212281 | 0      |
| 4 | 1.169471  | 1.035011 | 0      |



Data Pre-proce

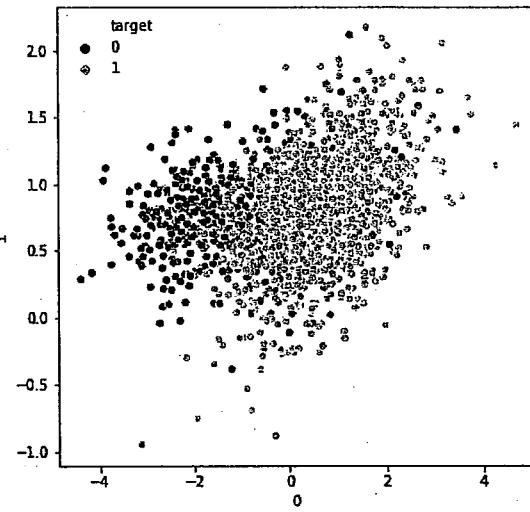
36

## Chiến thuật làm việc với dữ liệu mất cân bằng

### RandomUnderSample

```
# RandomUnderSampler
from imblearn.under_sampling import RandomUnderSampler
rs = RandomUnderSampler()
X_rs, y_rs = rs.fit_sample(X, y)
print(sorted(Counter(y_rs).items()))
[(0, 815), (1, 815)]
data_rs = pd.DataFrame(X_rs)
data_rs['target'] = y_rs
data_rs.head()
```

|   | 0         | 1        | target |
|---|-----------|----------|--------|
| 0 | -2.473300 | 0.569620 | 0      |
| 1 | -1.655427 | 1.168923 | 0      |
| 2 | -0.012906 | 0.989231 | 0      |
| 3 | -0.841488 | 0.212281 | 0      |
| 4 | 1.169471  | 1.035011 | 0      |



## Chiến thuật làm việc với dữ liệu mất cân bằng

### resample

```
data_0 = data[data.target==0]
data_1 = data[data.target==1]

data_0.shape
(815, 3)

data_1.shape
(14185, 3)

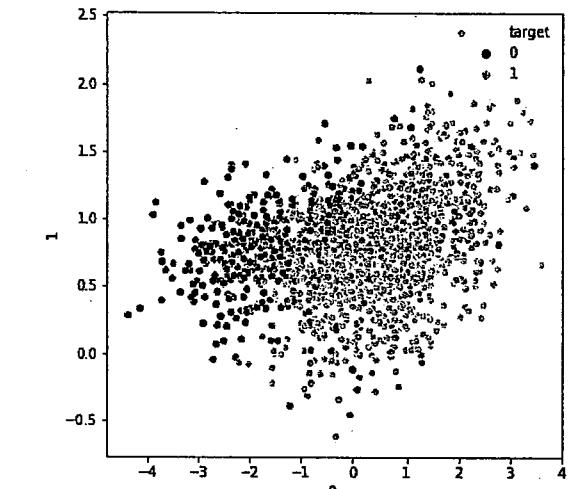
from sklearn.utils import resample

data_1_resample = resample(data_1,
                           replace = False, # sample without replacement
                           n_samples = data_0.shape[0], # match minority n
                           random_state = 27) # reproducible results
```

## Chiến thuật làm việc với dữ liệu mất cân bằng

```
downsampled = pd.concat([data_0, data_1_resample])  
downsampled.target.value_counts()  
1    815  
0    815  
Name: target, dtype: int64
```

resample



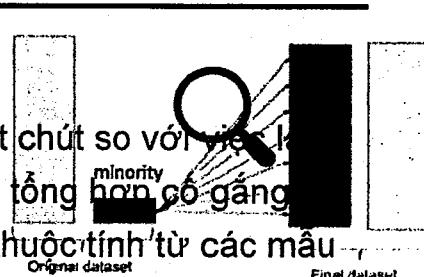
Link: <https://scikit-learn.org/stable/modules/generated/sklearn.utils.resample.html>  
Data Pre-processing and Analysis

39

## Chiến thuật làm việc với dữ liệu mất cân bằng

### • Over-sampling

- Quá trình này phức tạp hơn một chút so với việc lấp đầy mẫu. Đó là quá trình tạo dữ liệu tổng hợp có găng ngẫu nhiên một mẫu chứa các thuộc tính từ các mẫu trong lớp thiểu số. Có một số kỹ thuật được sử dụng trong đó kỹ thuật phổ biến nhất được gọi là SMOTE (Synthetic Minority Over-sampling Technique), nó xem xét không gian tính năng cho các điểm dữ liệu của nhóm thiểu số và xem xét k hàng xóm gần nhất (k nearest neighbor – KNN).
- Lập trình sử dụng imblearn.over\_sampling.SMOTE hoặc sklearn.utils.resample



Data Pre-processing and Analysis

40

## Chiến thuật làm việc với dữ liệu mất cân bằng

• Ví dụ:

```
from sklearn import datasets
import pandas as pd
import numpy as np
from collections import Counter
from sklearn.datasets import make_classification
import seaborn as sns
import matplotlib.pyplot as plt

X, y = make_classification(n_samples=1000, n_features=2, n_informative=2,
                           n_redundant=0, n_repeated=0, n_classes=2,
                           n_clusters_per_class=1,
                           weights=[0.1, 0.9],
                           class_sep=0.8, random_state=0)

print(sorted(Counter(y).items()))
[(0, 103), (1, 897)]
```

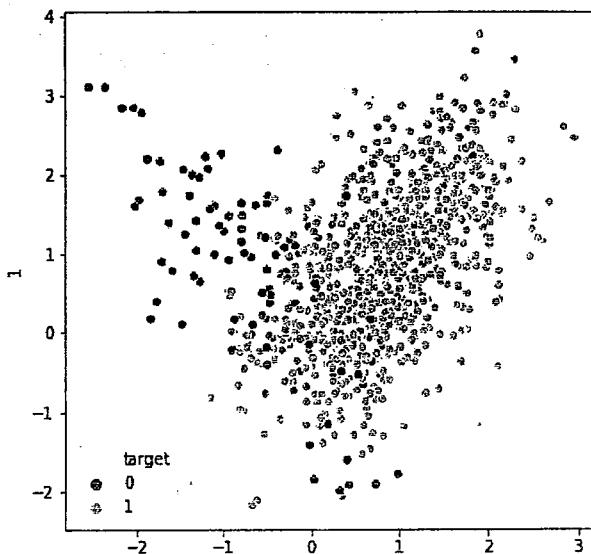


## Chiến thuật làm việc với dữ liệu mất cân bằng

```
data = pd.DataFrame(X)
data['target'] = y
data.head()
```

plt.figure(figsize=(6,6))
sns.scatterplot(data=data, x=0, y=1, hue='target')
<matplotlib.axes.\_subplots.AxesSubplot at 0x1da1490c9e8>

|   | 0        | 1         | target |
|---|----------|-----------|--------|
| 0 | 0.153184 | -1.013156 | 1      |
| 1 | 0.876102 | 0.980291  | 1      |
| 2 | 0.677376 | 0.060854  | 1      |
| 3 | 0.697991 | 2.207370  | 1      |
| 4 | 1.306738 | 2.383626  | 1      |



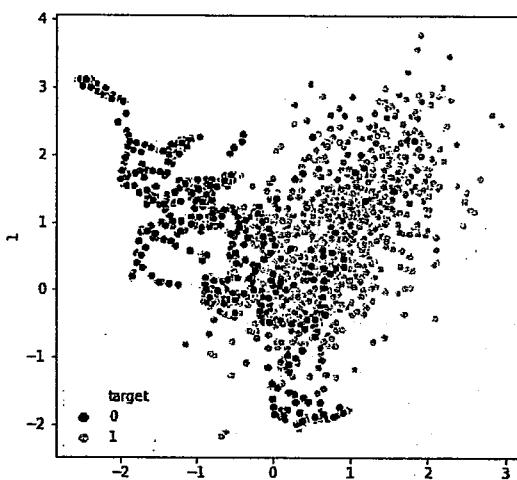
## Chiến thuật làm việc với dữ liệu mất cân bằng



```
from imblearn.over_sampling import SMOTE  
X_S, y_S = SMOTE().fit_resample(X, y)  
print(sorted(Counter(y_S).items()))
```

```
[(0, 897), (1, 897)]  
  
plt.figure(figsize=(6,6))  
sns.scatterplot(data=data_S, x=0, y=1, hue='target')  
<matplotlib.axes._subplots.AxesSubplot at 0x1da14deb0f0>
```

|   | 0        | 1         | target |
|---|----------|-----------|--------|
| 0 | 0.153184 | -1.013156 | 1      |
| 1 | 0.876102 | 0.980291  | 1      |
| 2 | 0.677376 | 0.060854  | 1      |
| 3 | 0.697991 | 2.207370  | 1      |
| 4 | 1.306738 | 2.383626  | 1      |



43

## Chiến thuật làm việc với dữ liệu mất cân bằng



```
data_0 = data[data.target==0]  
data_1 = data[data.target==1]
```

sklearn.utils.resample

```
data_0.shape
```

```
(103, 3)
```

```
data_1.shape
```

```
(897, 3)
```

```
from sklearn.utils import resample
```

```
data_0_resample = resample(data_0,  
                           replace=True, # sample with replacement  
                           n_samples=data_1.shape[0], # match number in majority class  
                           random_state=27) # reproducible results
```

```
data_0_resample.shape
```

```
(897, 3)
```

```
upsampled = pd.concat([data_0_resample, data_1])
```

```
upsampled.target.value_counts()
```

```
1    897  
0    897  
Name: target, dtype: int64
```



44

## Chiến thuật làm việc với dữ liệu mất cân bằng



### □ Thủ các thuật toán machine learning khác nhau

- Không nên sử dụng thuật toán yêu thích của mình cho mọi vấn đề. Ít nhất chúng ta nên kiểm tra cùng lúc (spot-checking) nhiều loại thuật toán khác nhau cho một vấn đề nhất định.
- Gợi ý: Decision - cây quyết định thường hoạt động tốt <sup>or random forest</sup>

Lien quan\_ trên các bộ dữ liệu mất cân bằng. Các quy tắc phân tách  
Cây, nồng <sup>↓</sup> dựa vào biến lớp được sử dụng trong việc tạo cây, có  
thể buộc cả hai lớp được xử lý. Cũng có thể sử dụng  
Random Forest trong trường hợp này.





## Chapter 7

### Ex1: Hacide

\* chỉ cần bằng train Data, ko cần bộ Test Data

- Cho 2 tập tin là hacide\_train.csv và hacide\_test.csv. Dữ liệu này được dùng để xây dựng model dự đoán và kiểm tra một mẫu là hiếm hay phổ biến.
- Đọc dữ liệu hacide\_train.csv, xem xét tính cân bằng giữa hai loại mẫu hiếm và phổ biến. Trực quan hóa. Nhận xét.
- Nếu 2 loại mẫu này không cân bằng, hãy chọn một phương pháp cân bằng dữ liệu và thực hiện. Giải thích lý do. Trực quan hóa kết quả.

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: train_data = pd.read_csv("hacide_train.csv")
```

```
In [3]: train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
Unnamed: 0    1000 non-null int64
cls           1000 non-null int64
x1            1000 non-null float64
x2            1000 non-null float64
dtypes: float64(2), int64(2)
memory usage: 31.3 KB
```

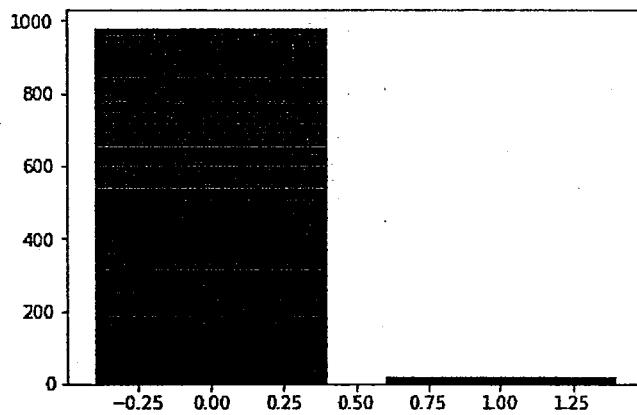
```
In [4]: # Đếm theo loại: hiếm, phổ biến
occ = train_data.cls.value_counts()
occ
```

```
Out[4]: 0    980
1     20
Name: cls, dtype: int64
```



```
In [5]: plt.bar(occ.index.values, occ.values)
```

```
Out[5]: <BarContainer object of 2 artists>
```



```
In [6]: # Print the ratio of fraud cases  
print(occ / len(train_data.index))
```

```
0    0.98  
1    0.02  
Name: cls, dtype: float64
```

```
In [7]: # Vì Lượng dữ Liệu class 1 rất ít => do đó ta sẽ áp dụng Oversampling để nâng số m
```

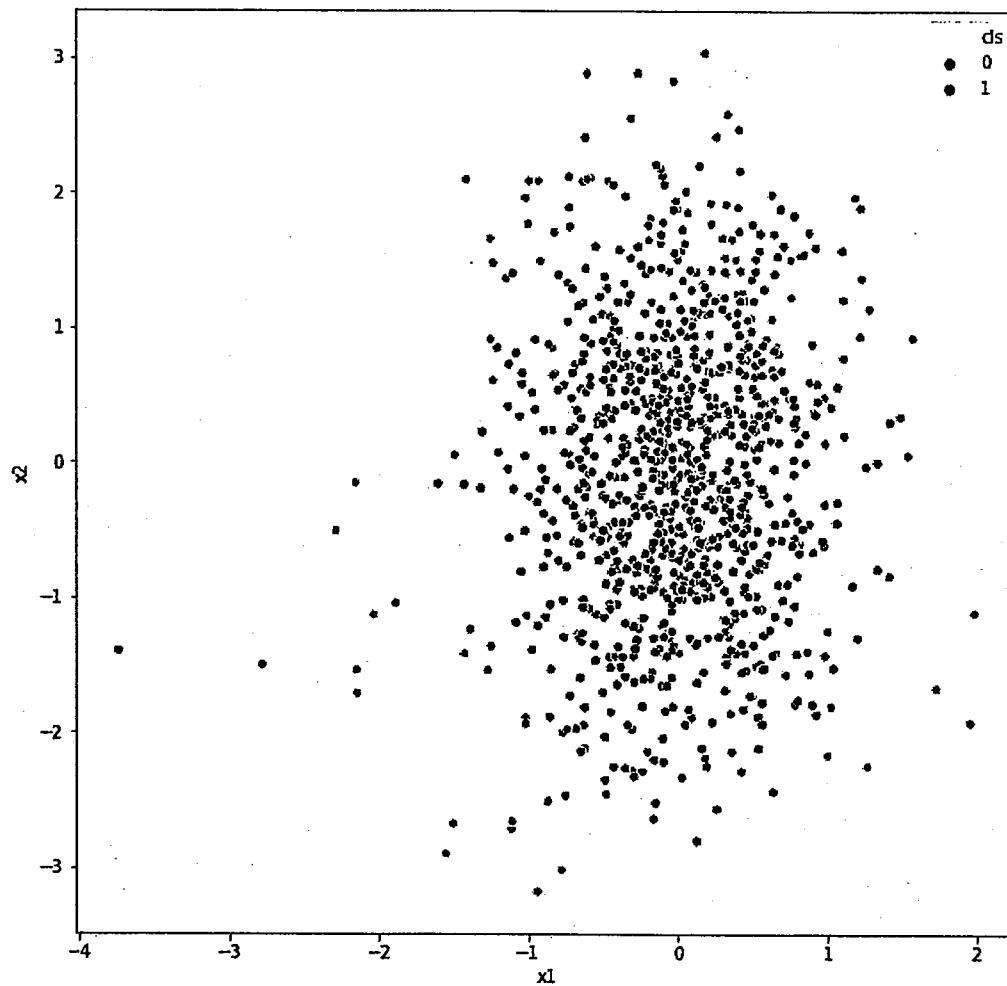
# Mô hình học máy

7/25/2019

Ex1



```
In [8]: plt.figure(figsize=(10,10))
sns.scatterplot(data=train_data, x="x1", y="x2", hue="cls")
plt.show()
```



```
In [9]: # chia ra 2 tập: X (input), y (output)
X = train_data[["x1", "x2"]]
X.head()
```

Out[9]:

|   | x1       | x2        |
|---|----------|-----------|
| 0 | 0.200798 | 0.678038  |
| 1 | 0.016620 | 1.576558  |
| 2 | 0.228725 | -0.559534 |
| 3 | 0.126379 | -0.093814 |
| 4 | 0.600821 | -0.298395 |



```
In [10]: y = train_data["cls"]
y.head()
```

```
Out[10]: 0    0
1    0
2    0
3    0
4    0
Name: cls, dtype: int64
```

```
In [11]: # Oversampling
from imblearn.over_sampling import SMOTE
```

```
In [12]: X_S, y_S = SMOTE().fit_resample(X, y)
```

```
In [13]: from collections import Counter
sorted(Counter(y_S).items())
```

```
Out[13]: [(0, 980), (1, 980)]
```

```
In [14]: data_S= pd.DataFrame(X_S)
data_S.columns = ["x1", "x2"]
data_S['cls'] = y_S
data_S.head()
```

```
Out[14]:
```

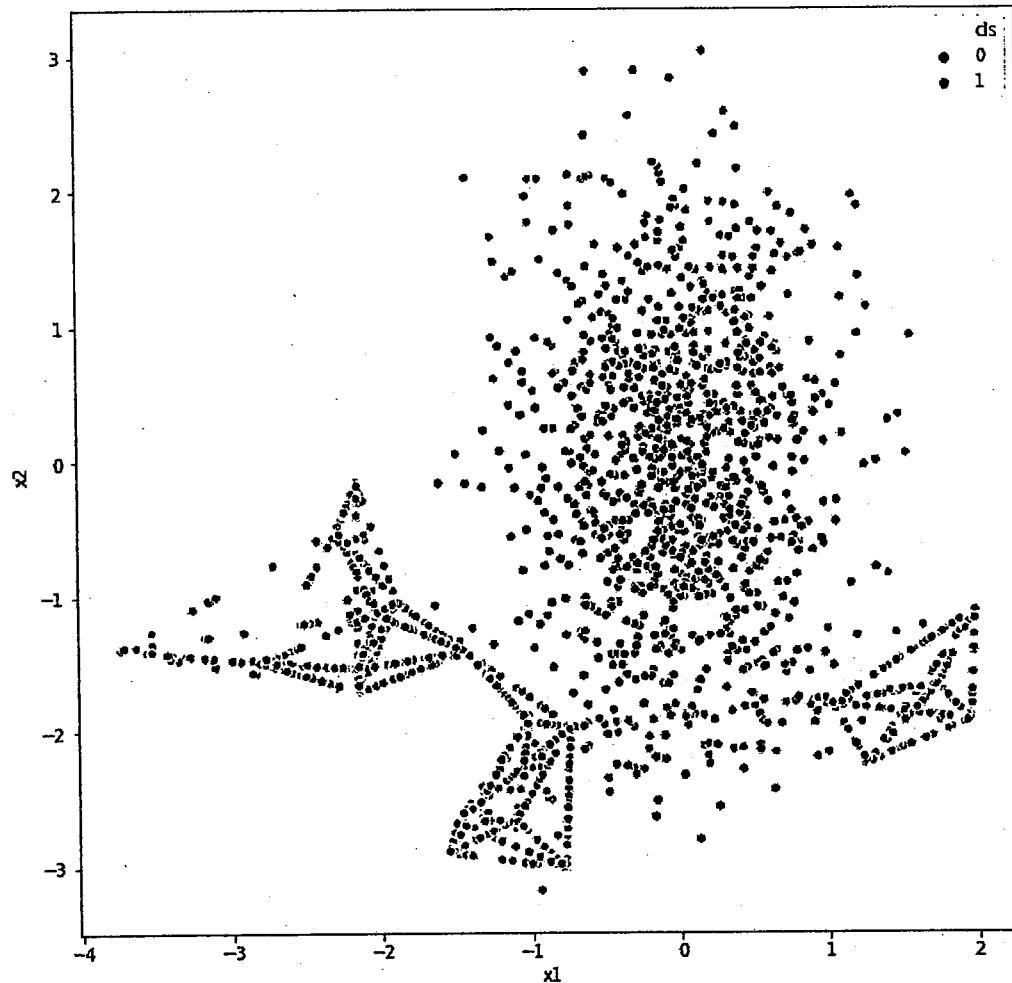
|   | x1       | x2        | cls |
|---|----------|-----------|-----|
| 0 | 0.200798 | 0.678038  | 0   |
| 1 | 0.016620 | 1.576558  | 0   |
| 2 | 0.228725 | -0.559534 | 0   |
| 3 | 0.126379 | -0.093814 | 0   |
| 4 | 0.600821 | -0.298395 | 0   |

7/25/2019

Ex1



```
In [15]: plt.figure(figsize=(10,10))
sns.scatterplot(data=data_S, x="x1", y="x2", hue="cls")
plt.show()
```



```
In [ ]:
```



## Ex2: Adult Dataset

- Adult Dataset được cung cấp bởi UCI (University of California, Irvine) được sử dụng để phát triển mô hình dự đoán Predictive Model Development.
- Bộ dữ liệu adult.data và adult.test chứa 48.842 mẫu và có 14 attributes/features. Dữ liệu này được dùng để xây dựng model dự đoán và kiểm tra một mẫu có thu nhập >50K USD hay không.

### Attribute Information:

- age: continuous.
- workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- fnlwgt: continuous.
- education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- education-num: continuous.
- marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- sex: Female, Male.
- capital-gain: continuous.
- capital-loss: continuous.
- hours-per-week: continuous.
- native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
- Class: >50K, <=50K.

### Yêu cầu:

- Đọc dữ liệu adult.data, tiền xử lý dữ liệu.
- Xem xét tính cân bằng giữa hai loại mẫu. Trực quan hóa. Nhận xét.
- Nếu 2 loại mẫu này không cân bằng, hãy chọn một phương pháp cân bằng dữ liệu và thực hiện. Trực quan hóa kết quả.

In [1]: # Link tham khảo: <https://towardsdatascience.com/under-sampling-a-performance-boos>



```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: # Đọc dữ liệu, kiểm tra sơ bộ bao đầu, trực quan hóa, tiền xử lý dữ liệu
adult_train = pd.read_csv("adult/adult.data", header=None)
```

```
In [4]: adult_train.head()
```

Out[4]:

|   | 0  | 1                | 2      | 3         | 4    | 5                  | 6                  | 7                 | 8       | 9      | 10   | 11 | 12 |    |
|---|----|------------------|--------|-----------|------|--------------------|--------------------|-------------------|---------|--------|------|----|----|----|
| 0 | 39 | State-gov        | 77516  | Bachelors | 13   | Never-married      | Adm-clerical       | Not-in-family     | White   | Male   | 2174 | 0  | 40 |    |
| 1 | 50 | Self-emp-not-inc | 83311  | Bachelors | 13   | Married-civ-spouse | Exec-managerial    | Husband           | White   | Male   | 0    | 0  | 13 |    |
| 2 | 38 | Private          | 215646 | HS-grad   | 9    | Divorced           | Handlers-cleaners  | Not-in-family     | White   | Male   | 0    | 0  | 40 |    |
| 3 | 53 | Private          | 234721 |           | 11th | 7                  | Married-civ-spouse | Handlers-cleaners | Husband | Black  | Male | 0  | 0  | 40 |
| 4 | 28 | Private          | 338409 | Bachelors | 13   | Married-civ-spouse | Prof-specialty     | Wife              | Black   | Female | 0    | 0  | 40 |    |

```
In [5]: adult_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
0    32561 non-null int64
1    32561 non-null object
2    32561 non-null int64
3    32561 non-null object
4    32561 non-null int64
5    32561 non-null object
6    32561 non-null object
7    32561 non-null object
8    32561 non-null object
9    32561 non-null object
10   32561 non-null int64
11   32561 non-null int64
12   32561 non-null int64
13   32561 non-null object
14   32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
In [6]: adult_train.to_csv("adult_data.csv")
```



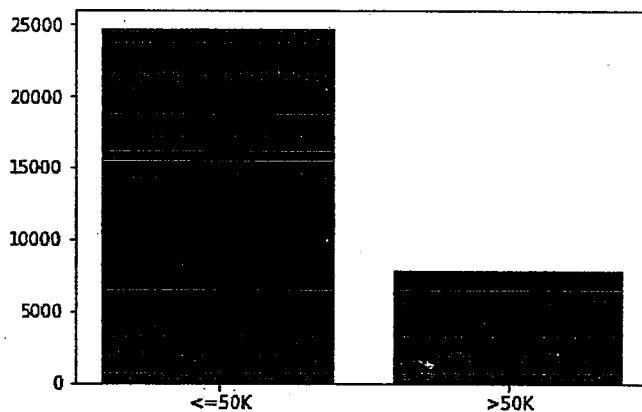
In [7]: # Không có dữ liệu null

In [8]: # Đếm theo loại: hiếm, phổ biến  
occ = adult\_train[14].value\_counts()  
occ

Out[8]: <=50K 24720  
>50K 7841  
Name: 14, dtype: int64

In [9]: plt.bar(occ.index.values, occ.values)

Out[9]: <BarContainer object of 2 artists>



In [10]: # Chuyển dữ liệu phân loại thành dạng numeric dùng Label encoder và dummy encoder

In [11]: y\_train = adult\_train[14]  
X\_train = adult\_train.drop([14], axis=1)

In [12]: X\_train.head(2)

Out[12]:

|   | 0  | 1                | 2     | 3         | 4  | 5                  | 6               | 7             | 8     | 9    | 10   | 11 | 12 |
|---|----|------------------|-------|-----------|----|--------------------|-----------------|---------------|-------|------|------|----|----|
| 0 | 39 | State-gov        | 77516 | Bachelors | 13 | Never-married      | Adm-clerical    | Not-in-family | White | Male | 2174 | 0  | 40 |
| 1 | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband       | White | Male | 0    | 0  | 13 |

In [13]: y\_train[:2]

Out[13]: 0 <=50K  
1 <=50K  
Name: 14, dtype: object



```
In [14]: from sklearn.preprocessing import LabelEncoder
```

```
In [15]: label_encoder = LabelEncoder()
y_train_1 = label_encoder.fit_transform(y_train)
```

```
In [16]: y_train_1[:2]
```

```
Out[16]: array([0, 0])
```

```
In [17]: # Categorical boolean mask
categorical_feature_mask = X_train.dtypes==object
# filter categorical columns using mask and turn it into a list
categorical_cols = X_train.columns[categorical_feature_mask].tolist()
categorical_cols
```

```
Out[17]: [1, 3, 5, 6, 7, 8, 9, 13]
```

```
In [18]: X_train_d = pd.get_dummies(data=X_train, columns=categorical_cols, drop_first=True)
```

```
In [19]: X_train_d.head(2)
```

```
Out[19]:
      0    2    4   10   11   12   Federal-gov   Local-gov   Never-worked   Private   ...   13_Portugal   13_Puerto-Rico   13_Scotland
0   39  77516   13  2174     0    40         0         0         0         0    ...        0         0         0
1   50  83311   13     0     0    13         0         0         0         0    ...        0         0         0
```

2 rows × 100 columns

```
In [20]: from collections import Counter
sorted(Counter(y_train_1).items())
```

```
Out[20]: [(0, 24720), (1, 7841)]
```

```
In [21]: # Vì lượng dữ liệu class 1 tương đối nhiều => do đó ta sẽ áp dụng Undersampling
# để giảm số mẫu của nhóm <=50k bằng với nhóm >50k
```

```
In [22]: from sklearn.utils import resample
```

```
In [23]: # có thể dùng cách resample
```

```
In [24]: data_train = X_train_d
data_train[14] = y_train_1
```

```
In [25]: data_0 = data_train[data_train[14]==0]
data_1 = data_train[data_train[14]==1]
```

7/25/2019

Ex2



In [26]: `display(data_0.shape, data_1.shape)`

(24720, 101)

(7841, 101)

In [27]: `from sklearn.utils import resample`

In [28]: `data_0_resample = resample(data_0,  
 replace = False, # sample without replacement  
 n_samples = data_1.shape[0], # match minority n  
 random_state = 27) # reproducible results`

In [29]: `downsampled = pd.concat([data_0_resample, data_1])  
downsampled.head()`

Out[29]:

|       | 0  | 2      | 4  | 10 | 11 | 12 | 1_Federal-gov | 1_Local-gov | 1_Never-worked | 1_Private | 13_Puerto-Rico | 13_Scotland | 13_South |
|-------|----|--------|----|----|----|----|---------------|-------------|----------------|-----------|----------------|-------------|----------|
| 31749 | 22 | 199426 | 10 | 0  | 0  | 17 | 0             | 0           | 0              | 1         | ...            | 0           | 0        |
| 24093 | 31 | 91964  | 13 | 0  | 0  | 40 | 0             | 0           | 0              | 1         | ...            | 0           | 0        |
| 21539 | 37 | 60313  | 9  | 0  | 0  | 40 | 0             | 0           | 0              | 1         | ...            | 0           | 0        |
| 24582 | 30 | 85708  | 9  | 0  | 0  | 40 | 0             | 0           | 0              | 1         | ...            | 0           | 0        |
| 622   | 65 | 109351 | 5  | 0  | 0  | 24 | 0             | 0           | 0              | 1         | ...            | 0           | 0        |

5 rows × 101 columns

In [30]: `display(data_0_resample.shape, data_1.shape)`

(7841, 101)

(7841, 101)



## Ex3: Customer Churn Analysis

- Cho dữ liệu WA\_Fn-UseC\_-Telco-Customer-Churn.csv chứa thông tin khách hàng. Bộ dữ liệu này được dùng để xây dựng mô hình dự đoán một khách hàng kết thúc mối quan hệ hay hủy/không gia hạn (churn) với doanh nghiệp hay không?
- Bộ dữ liệu gồm 7043 mẫu và 21 thuộc tính

### Yêu cầu:

- Đọc dữ liệu WA\_Fn-UseC\_-Telco-Customer-Churn.csv, tiền xử lý dữ liệu.
- Chia dữ liệu thành 2 bộ là train và test theo tỷ lệ 80-20.
- Xem xét tính cân bằng giữa hai loại mẫu ở train. Trực quan hóa. Nhận xét.
- Nếu 2 loại mẫu ở train này không cân bằng, hãy chọn một phương pháp cân bằng dữ liệu và thực hiện. Trực quan hóa kết quả.

In [1]: # Link tham khảo: <https://www.analyticsvidhya.com/blog/2017/03/imbalance-classification/>

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: data = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
```

```
In [4]: data.head()
```

Out[4]:

|   | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService  |
|---|------------|--------|---------------|---------|------------|--------|--------------|---------------|------------------|
| 0 | 7590-VHVEG | Female | 0             | Yes     | No         | 1      | No           | No            | No phone service |
| 1 | 5575-GNVDE | Male   | 0             | No      | No         | 34     | Yes          | No            | No               |
| 2 | 3668-QPYBK | Male   | 0             | No      | No         | 2      | Yes          | No            | No               |
| 3 | 7795-CFOCW | Male   | 0             | No      | No         | 45     | No           | No            | No phone service |
| 4 | 9237-HQITU | Female | 0             | No      | No         | 2      | Yes          | No            | No               |

5 rows × 21 columns



In [5]: `data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
customerID      7043 non-null object
gender          7043 non-null object
SeniorCitizen    7043 non-null int64
Partner          7043 non-null object
Dependents       7043 non-null object
tenure           7043 non-null int64
PhoneService     7043 non-null object
MultipleLines    7043 non-null object
InternetService  7043 non-null object
OnlineSecurity   7043 non-null object
OnlineBackup     7043 non-null object
DeviceProtection 7043 non-null object
TechSupport      7043 non-null object
StreamingTV      7043 non-null object
StreamingMovies   7043 non-null object
Contract          7043 non-null object
PaperlessBilling  7043 non-null object
PaymentMethod     7043 non-null object
MonthlyCharges   7043 non-null float64
TotalCharges     7043 non-null object
Churn             7043 non-null object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [6]: `# Đếm theo loại: hiém, phở biến  
occ = data.Churn.value_counts()  
occ`

Out[6]: No 5174  
Yes 1869  
Name: Churn, dtype: int64

In [7]: `# Print the ratio of fraud cases  
print(occ / len(data.index))`

```
No    0.73463
Yes   0.26537
Name: Churn, dtype: float64
```

In [8]: `X = data.drop(["customerID", "Churn"], axis=1)  
y = data.Churn`



In [9]: X.head()

Out[9]:

|   | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines    | InternetService |
|---|--------|---------------|---------|------------|--------|--------------|------------------|-----------------|
| 0 | Female | 0             | Yes     | No         | 1      | No           | No phone service | DSL             |
| 1 | Male   | 0             | No      | No         | 34     | Yes          | No               | DSL             |
| 2 | Male   | 0             | No      | No         | 2      | Yes          | No               | DSL             |
| 3 | Male   | 0             | No      | No         | 45     | No           | No phone service | DSL             |
| 4 | Female | 0             | No      | No         | 2      | Yes          | No               | Fiber optic     |

↑ ↓ ▶

In [10]: X.MonthlyCharges = X.MonthlyCharges.astype('float')

In [11]: X.TotalCharges = pd.to\_numeric(X.TotalCharges, errors='coerce')

In [12]: y.head()

Out[12]:

|   |     |
|---|-----|
| 0 | No  |
| 1 | No  |
| 2 | Yes |
| 3 | No  |
| 4 | Yes |

Name: Churn, dtype: object

In [13]:

```
# Chuẩn hóa y
# No: 0, Yes: 1
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y_new = label_encoder.fit_transform(y)
y_new[:5]
```

Out[13]: array([0, 0, 1, 0, 1])



```
In [14]: # Chuẩn hóa X; các cột là Category
# Categorical boolean mask
categorical_feature_mask = X.dtypes==object
# filter categorical columns using mask and turn it into a List
categorical_cols = X.columns[categorical_feature_mask].tolist()
categorical_cols
```

```
Out[14]: ['gender',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod']
```

```
In [15]: X_new = pd.get_dummies(data=X, columns=categorical_cols, drop_first=True)
```

```
In [16]: X_new.head()
```

```
Out[16]:
```

|   | SeniorCitizen | tenure | MonthlyCharges | TotalCharges | gender_Male | Partner_Yes | Dependents_Yes |
|---|---------------|--------|----------------|--------------|-------------|-------------|----------------|
| 0 | 0             | 1      | 29.85          | 29.85        | 0           | 1           | 0              |
| 1 | 0             | 34     | 56.95          | 1889.50      | 1           | 0           | 0              |
| 2 | 0             | 2      | 53.85          | 108.15       | 1           | 0           | 0              |
| 3 | 0             | 45     | 42.30          | 1840.75      | 1           | 0           | 0              |
| 4 | 0             | 2      | 70.70          | 151.65       | 0           | 0           | 0              |

5 rows × 30 columns

◀ ▶

```
In [17]: # Chia dữ liệu thành 2 bộ theo tỷ lệ 80:20
from sklearn.model_selection import train_test_split
```

```
In [18]: X_train, X_test, y_train, y_test = train_test_split(X_new, y_new, test_size = 0.2)
```

```
In [19]: #X_train.info()
```

```
In [20]: from collections import Counter
sorted(Counter(y_train).items())
```

```
Out[20]: [(0, 4141), (1, 1493)]
```



```
In [21]: # Vì Lượng dữ Liệu của mỗi Lớp đều khá nhiều và theo tỷ lệ 3:1
# Có thể cho undersampling hoặc oversampling
```

```
In [22]: # undersampling
from imblearn.under_sampling import RandomUnderSampler
```

```
In [23]: X_train = X_train.fillna(X_train.mean())
X_train.head()
```

Out[23]:

|      | SeniorCitizen | tenure | MonthlyCharges | TotalCharges | gender_Male | Partner_Yes | Dependents_ |
|------|---------------|--------|----------------|--------------|-------------|-------------|-------------|
| 5228 | 0             | 9      | 44.40          | 348.15       | 1           | 0           |             |
| 1473 | 0             | 24     | 98.75          | 2407.30      | 0           | 0           |             |
| 6606 | 0             | 38     | 105.00         | 4026.40      | 0           | 1           |             |
| 4747 | 0             | 35     | 20.60          | 754.00       | 1           | 0           |             |
| 1439 | 1             | 5      | 75.55          | 349.65       | 0           | 0           |             |

5 rows × 30 columns

1 [REDACTED] ▶

```
In [24]: rs = RandomUnderSampler()
X_train_resampled, y_train_resampled = rs.fit_resample(X_train, y_train)
```

```
In [25]: sorted(Counter(y_train_resampled).items())
```

Out[25]: [(0, 1493), (1, 1493)]

```
In [26]: # Oversampling
from imblearn.over_sampling import SMOTE
```

```
In [27]: X_train_S, y_train_S = SMOTE().fit_resample(X_train, y_train)
```

```
In [28]: sorted(Counter(y_train_S).items())
```

Out[28]: [(0, 4141), (1, 4141)]

In [ ]:



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

# DATA PRE-PROCESSING AND ANALYSIS

## Bài 8: Model

Phòng LT & Mạng

[https://csc.edu.vn/lap-trinh-va-cSDL/Data-Pre-processing-and-Analysis\\_196](https://csc.edu.vn/lap-trinh-va-cSDL/Data-Pre-processing-and-Analysis_196)

2019



## Nội dung

1. Giới thiệu
2. Model Deployment
3. Model Evaluation



## Giới thiệu



- Model deployment & evaluation là công đoạn nằm gần cuối trong quy trình Data Analysis



Data Pre-processing and Analysis

3

## Nội dung



1. Giới thiệu
2. Model Deployment
3. Model Evaluation



Data Pre-processing and Analysis

4

## Model Deployment



- Trong phân tích dữ liệu, chúng ta thường sử dụng phát triển mô hình (Model Development) để giúp dự đoán các quan sát (mẫu) trong tương lai từ dữ liệu mẫu hiện có.
- Một mô hình (model) sẽ giúp chúng ta hiểu mối quan hệ chính xác giữa các biến khác nhau và cách các biến này được sử dụng để dự đoán kết quả.



## Model Deployment



### Simple Linear Regression

- Simple Linear Regression là phương pháp giúp chúng ta hiểu mối quan hệ giữa hai biến:
  - Biến độc lập (predictor/independent variable): X
  - Biến phụ thuộc (response/dependent variable) là biến mà chúng ta muốn dự đoán: Y
- Kết quả của Linear Regression là một hàm tuyến tính (linear function) dự đoán biến phụ thuộc từ biến độc lập. *Là 1 đường thẳng*



## Model Deployment



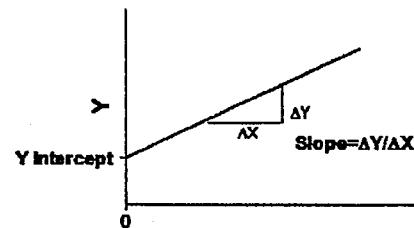
• Function:  $\hat{Y} = a + bX$

▪ Y: Response Variable

▪ X: Predictor Variables

▪ a: là intercept của regression line (giá trị của Y khi X bằng 0)

▪ b: là slope - độ dốc của regression line (giá trị mà Y thay đổi khi X tăng thêm 1 đơn vị)



## Model Deployment



• Ví dụ: Cho dữ liệu automobileEDA.csv, với  $\hat{Y} = a + bx$ .  
highway-mpg chúng ta dự đoán price của xe hơi.

```
import pandas as pd
```

```
df = pd.read_csv("automobileEDA.csv")  
df.head()
```

| make        | aspiration | num-of-doors | body-style  | drive-wheels | engine-location | wheel-base | length   | ...<br>compression-ratio | horsepower | peak-rpm | city-mpg | highway-mpg | price      |
|-------------|------------|--------------|-------------|--------------|-----------------|------------|----------|--------------------------|------------|----------|----------|-------------|------------|
| alfa-romero | std        | two          | convertible | rwd          | front           | 88.6       | 0.811148 | ...                      | 9.0        | 111.0    | 5000.0   | 21          | 27 13495.0 |
| alfa-romero | std        | two          | convertible | rwd          | front           | 88.6       | 0.811148 | ...                      | 9.0        | 111.0    | 5000.0   | 21          | 27 16500.0 |
| alfa-romero | std        | two          | hatchback   | rwd          | front           | 94.5       | 0.822681 | ...                      | 9.0        | 154.0    | 5000.0   | 19          | 26 16500.0 |
| audi        | std        | four         | sedan       | fwd          | front           | 99.8       | 0.848630 | ...                      | 10.0       | 102.0    | 5500.0   | 24          | 30 13950.0 |
| audi        | std        | four         | sedan       | 4wd          | front           | 99.4       | 0.848630 | ...                      | 8.0        | 115.0    | 5500.0   | 18          | 22 17450.0 |



# Model Deployment



- Load module, tạo đối tượng linear regression

```
from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
```

lm *tạo 1 model* *Mô hình*

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)
```

*→ huấn luyện model = cao đà*

- Chuẩn bị dữ liệu cho X, Y => fit model

*all dữ liệu có thể dùng*  
*model sklearn → fit huấn*  
*SD mẫu*

*→ X = df[['highway-mpg']]*

*→ Y = df['price']*

```
lm.fit(X,Y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)
```



# Model Deployment



- Dự đoán Y từ X

*Yhat=lm.predict(X) để áp dụng bằng dữ liệu, ta có*  
*Yhat[0:5] để đoán = dữ liệu mới hoàn toàn*

```
array([16236.50464347, 16236.50464347, 17058.23802179, 13771.3045085 ,  
20345.17153508])
```

*→ là giá trị liên tục*

- Xác định giá trị intercept (a) và slope (b)

```
a = lm.intercept_
```

```
b = lm.coef_[0]
```

```
(38423.3058581574, -821.7333783219254)
```

```
# price = 38423.31 - 821.73 x highway-mpg
```

Machine learning

N.L R

học rèn  
chỉ có X  
kết quả Y

inputs outputs

Classification

Regression

Y là 1

(Logistic Regression)

Y là -  
giá trị nằm  
trong biến ft

(Linear Regression)



## Model Deployment

### □ Multiple Linear Regression

- Nếu chúng ta muốn sử dụng nhiều biến hơn trong mô hình (X<sub>1</sub>, X<sub>2</sub>, ...) để dự đoán (Y), chúng ta có thể sử dụng Multiple Linear Regression. Multiple Linear Regression rất giống với Simple Linear Regression, nhưng phương pháp này được sử dụng để giải thích mối quan hệ giữa một biến phụ thuộc và hai hoặc nhiều biến độc lập.
- Hầu hết các mô hình hồi quy (regression model) trong thực tế liên quan đến nhiều yếu tố dự đoán.



## Model Deployment

• Function:  $Y = a + b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4 \dots$

▪ Y: Response Variable → output → biến phụ thuộc

▪ X<sub>1</sub>: Predictor Variable 1

▪ X<sub>2</sub>: Predictor Variable 2

▪ X<sub>3</sub>: Predictor Variable 3

▪ X<sub>4</sub>: Predictor Variable 4

▪ a: intercept

▪ b<sub>1</sub>: coefficients of Variable 1

▪ b<sub>2</sub>: coefficients of Variable 2

▪ b<sub>3</sub>: coefficients of Variable 3

▪ b<sub>4</sub>: coefficients of Variable 4

input → độc lập



## Model Deployment



- Ví dụ: Cho dữ liệu automobileEDA.csv, với horsepower, curb-weight, engine-size, và highway-mpg chúng ta dự đoán price của xe hơi.  $\rightarrow$  Nhập vào 4 biến  $\rightarrow$  dự đoán price

- Chuẩn bị dữ liệu
- Load module
- Tạo đối tượng linear regression

```
lm1 = LinearRegression()
```



## Model Deployment



- Chuẩn bị dữ liệu cho X, Y  $\Rightarrow$  fit model

```
df[4]  
x1 = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]  
y1 = df['price']
```

Jupyter

```
lm1.fit(x1, y1)  
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)
```

- Dự đoán Y từ X

$y_{hat1} = lm1.predict(x1) \rightarrow$  predict, hàm predict  $Y = a + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4$

$y_{hat1}[0:5]$

```
array([13699.11161184, 13699.11161184, 19051.65470233, 10620.36193015,  
15521.31420211])
```



## Model Deployment



- Xác định giá trị intercept (a) và slope (b1, b2, b3, b4...)

```
a1 = lm1.intercept_
```

```
b1 = lm1.coef_[0]
```

```
a1, b1
```

```
 $\alpha_1$   
(-15806.624626329223,  
array([53.49574423, 4.70770099, 81.53026382, 36.05748882]))  
 $b_1$        $b_2$        $b_3$        $b_4$   
 $a_1$        $b_1[0]$   
''' Price = -15806.625 + 53.496 x horsepower +  
                4.708 x curb-weight +  
                81.53 x engine-size +  
                36.057 x highway-mpg '''
```

Chi code thuật toán để công khai mô hình

Data Pre-processing and Analysis

15

$$y_p = a_1 + b_1[0] \times X['horsepower'] + b_2[0] \times X['curb-weight'] + b_3[0] \times X['engine-size'] + b_4[0] \times X['highway-mpg']$$

Thường hợp viết app/web để cần  $y_{hat} = f_{model}.predict()$

Model Deployment → chuyển model cho IT đưa lên web.



## Model Evaluation

- Sau khi đã phát triển một số mô hình, chúng ta cần đánh giá các mô hình để chọn được mô hình tốt nhất.
- Có thể sử dụng trực quan hóa dữ liệu để đánh giá (dùng matplotlib/ seaborn...)



Data Pre-processing and Analysis

16

trục quan trắc Linear Regression →

1. Regression plot

2. Residual Plot



## Model Deployment

### • Regression plot (số Regression plot chỉ dùng simple linear regression)

- Khi nói đến simple linear regression, một cách tốt để hình dung sự phù hợp của mô hình là sử dụng các biểu đồ hồi quy- regression plot.
- Biểu đồ này sẽ hiển thị kết hợp các điểm dữ liệu phân tán (scatter plot), cũng như đường hồi quy tuyến tính được tạo thành đi qua dữ liệu. Điều này sẽ cho chúng ta một ước tính hợp lý về mối quan hệ giữa hai biến, sức mạnh của mối tương quan (correlation), cũng như hướng (positive/negative correlation).



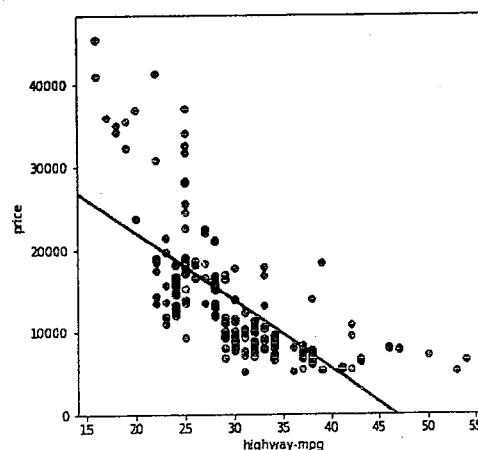
## Model Deployment



- Ví dụ: regression plot : highway-mpg và price

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6,6))
sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```





## Model Deployment

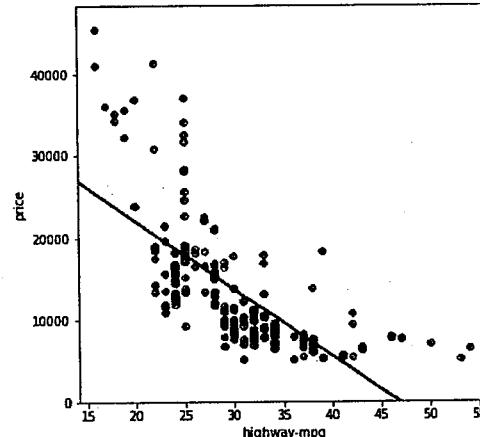
- Ví dụ: trực quan hóa highway-mpg và price

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6,6))
sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```

Quan sát biểu đồ ta thấy: price tương quan nghịch với highway-mpg, vì regression plot có giá trị âm.

Một điều cần lưu ý khi xem biểu đồ hồi quy là chú ý đến mức độ phân tán của các điểm dữ liệu xung quanh đường hồi quy. Điều này sẽ cung cấp cho chúng ta một dấu hiệu tốt về phương sai (variance) của dữ liệu và liệu mô hình tuyến tính có phù hợp nhất hay không. Nếu dữ liệu quá xa so với line, mô hình tuyến tính này có thể không phải là mô hình phù hợp cho dữ liệu này.



Data Pre-processing and Analysis

19

## Model Deployment



### • Residual Plot (Biểu đồ phần dư).

- Một cách tốt để trực quan phương sai của dữ liệu là sử dụng residual plot.

#### ▪ Residual (phần dư) là gì?

- Sự khác biệt giữa giá trị quan sát ( $y$ ) và giá trị dự đoán ( $\hat{Y}$ ) được gọi là phần dư ( $e$ ). Khi chúng ta xem xét một biểu đồ hồi quy, phần dư là khoảng cách từ điểm dữ liệu đến đường hồi quy (regression line) được fit.

#### ▪ Residual plot là gì?

- Biểu đồ dư là biểu đồ hiển thị phần dư trên trục y và biến độc lập trên trục x.



Data Pre-processing and Analysis

20



## Model Deployment

- Chúng ta chú ý điều gì khi quan sát residual plot?

Chúng ta xem xét sự lây lan của residual (phản dư)

- Nếu các điểm trong một residual plot được trãi ngẫu nhiên xung quanh trục x, thì mô hình tuyến tính phù hợp với dữ liệu.
- Tại sao? Vì ngẫu nhiên trãi residual có nghĩa là phương sai không đổi, và do đó mô hình tuyến tính phù hợp với dữ liệu này.

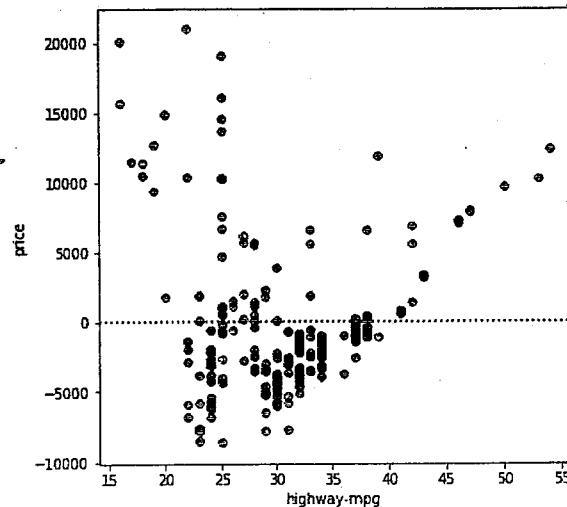


## Model Deployment

- Ví dụ: residual plot: highway-mpg và price

```
plt.figure(figsize=(6,6))
sns.residplot(df['highway-mpg'], df['price'])
plt.show()
```

Chúng ta có thể thấy từ residual plot này như sau: phản dư không được trãi ngẫu nhiên quanh trục x, điều này khiến chúng ta tin rằng có thể một mô hình phi tuyến tính (non-linear model) phù hợp hơn với dữ liệu này.



## Model Deployment



### □ Distribution plot ( $f(x)$ với $x_1, x_2, x_3, \dots$ )

- Làm thế nào để chúng ta trực quan hóa một mô hình cho Multiple Linear Regression? Điều này trở nên phức tạp hơn vì ta không thể hình dung nó bằng regression/ residual plot.
- Một cách để xem xét sự phù hợp của mô hình là bằng cách quan sát distribution plot (biểu đồ phân phối): có thể xem xét phân phối của các giá trị được dự đoán từ mô hình và so sánh nó với các giá trị thực tế.



Data Pre-processing and Analysis

23

## Model Deployment



- Ví dụ: distribution plot: y thực tế (actual value) và y ước tính (estimated value)

```
plt.figure(figsize=(8,8))
ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual Value")
sns.distplot(Yhat1, hist=False, color="b", label="Fitted Values", ax=ax1)

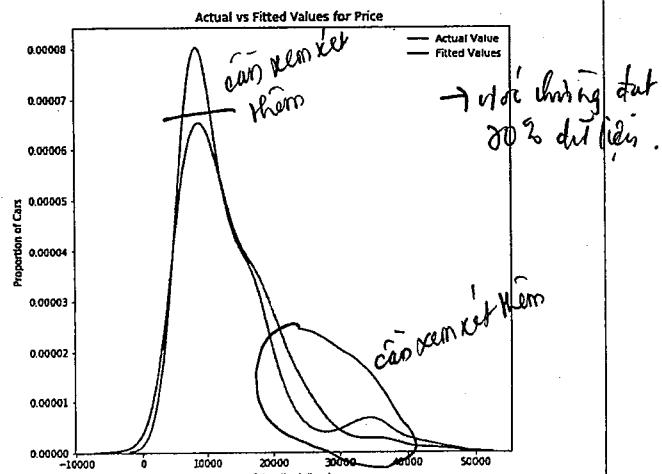
plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price (in dollars)')
plt.ylabel('Proportion of Cars')

plt.show()
plt.close()
```

Chúng ta có thể thấy rằng các giá trị được dự đoán gần với các giá trị thực tế, vì hai phân phối trùng nhau một phần. Tuy nhiên, chắc chắn có một số cần được cải thiện.



Data Pre-processing ar



→ để cải thiện dùng Polynomial Regression.



## Model Deployment

### Polynomial Regression

- Polynomial regression (hồi quy đa thức) là một trường hợp cụ thể của general linear regression model (mô hình hồi quy tuyến tính tổng quát) hoặc nhiều multiple linear regression model.

- Nếu  
mô hình tuyến  
nhé, ta OK  
→ (vẽ cùm làm  
kém bước này)  
hơn.
- Chúng ta có được các non-linear relationship (mối quan hệ phi tuyến) bằng cách bình phương hoặc đặt các biến dự đoán ở bậc cao



## Model Deployment

- Có nhiều thứ tự khác nhau của hồi quy đa thức:

càng nặng bao  
càng phức tạp

**Quadratic - 2nd order**

$$Yhat = a + b_1 X + b_2 X^2$$

**Cubic - 3rd order**

$$Yhat = a + b_1 X + b_2 X^2 + b_3 X^3$$

**Higher order:**

$$Yhat = a + b_1 X + b_2 X^2 + b_3 X^3 \dots$$





## Model Deployment

- Ví dụ: Trước đó, chúng ta đã thấy rằng linear model (mô hình tuyến tính) không cung cấp sự phù hợp nhất khi sử dụng highway-mpg làm biến dự đoán. Thay vào đó, hãy xem liệu chúng ta có thể thử điều chỉnh polynomial model (mô hình đa thức) cho dữ liệu hay không.



## Model Deployment

- Function plot data (chỉnh lại các giá trị tùy bài)

```
def PlotPolly(model, independent_variable, dependent_variable, Name):  
    1) x_new = np.linspace(15, 55, 100)  
        y_new = model(x_new)  
  
    plt.plot(independent_variable, dependent_variable, '.', x_new, y_new, '-')  
    2) plt.title('Polynomial Fit with Matplotlib for Price ~ Length')  
        ax = plt.gca()  
        ax.set_facecolor((0.898, 0.898, 0.898))  
        fig = plt.gcf()  
        plt.xlabel(Name)  
    3) plt.ylabel('Price of Cars')  
        plt.show()  
        plt.close()
```

\* 1,2,3 nên đưa lên trên để ý hợp cho n bài  
t老子 → đều không thư viện hàm hay sd.





## Model Deployment

- Chuẩn bị x, y => Sử dụng polyfit() để fit polynomial, sau đó dùng poly1d() để hiển thị polynomial function.

```
import numpy as np

x = df['highway-mpg']
y = df['price']

# Here we use a polynomial of the 3rd order (cubic)
f = np.polyfit(x, y, 3) bản đồ hàm số
p = np.poly1d(f)
p


$$-1.557x^3 + 284.8x^2 - 8965x + 1.379e+05$$


f
array([-1.55663829e+00, 2.04754306e+02, -8.96543312e+03, 1.37923594e+05])
```

Data Pre-processing and Analysis

29

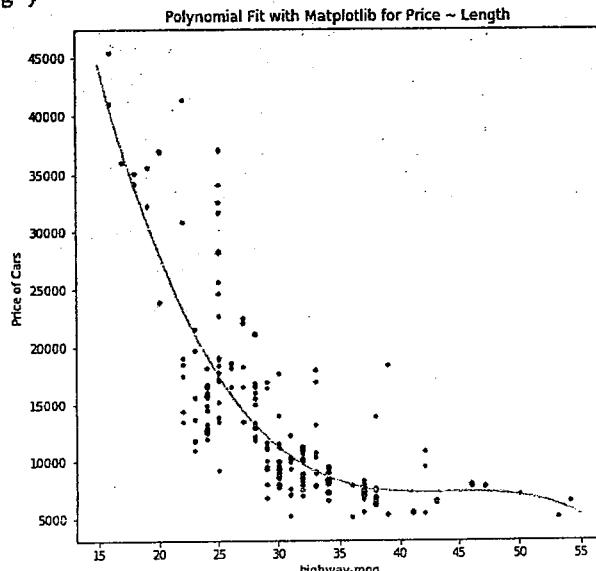


## Model Deployment

- Trực quan hóa kết quả

```
plt.figure(figsize=(8,8))
PlotPolly(p, x, y, 'highway-mpg')
```

Từ biểu đồ, chúng ta có thể thấy rằng mô hình đa thức này hoạt động tốt hơn mô hình tuyến tính. Điều này là do hàm đa thức được tạo "gần" nhiều điểm dữ liệu hơn.



30



Data Pr



## Model Deployment

### Multivariate Polynomial *(Ký` polynomial bậc 2, 3 hoặc độis thay đổi)*

- Biểu thức phân tích cho Multivariate Polynomial function (hàm đa thức đa biến) trở nên phức tạp.
- Ví dụ: Biểu thức cho đa thức bậc hai (degree = 2) với hai biến như sau:

$$Y_{hat} = a + b_1 X_1 + b_2 X_2 + b_3 X_1 X_2 + b_4 X_1^2 + b_5 X_2^2$$



## Model Deployment

- Chúng ta có thể thực hiện một phép biến đổi đa thức (polynomial transform) trên nhiều thuộc tính.

- Import module

```
from sklearn.preprocessing import PolynomialFeatures
```

- Khởi tạo **PolynomialFeatures** object với degree (ví dụ = 2)

```
pr=PolynomialFeatures(degree=2)
pr
PolynomialFeatures(degree=2, include_bias=True, interaction_only=False)
```



## Model Deployment



- Tạo dữ liệu mới: fit\_transform

X1\_pr=pr.fit\_transform(X1) (nếu dữ liệu chỉ có X mà không có y))  
fit-transform

X1.shape, X1\_pr.shape  
((201, 4), (201, 15))  
Nếu  $X_{new}$  chỉ 4 cột  $\rightarrow$  phải chia nhỏ  
theo dạng mới (15 cột)  
 $X_{new-p} = pr.transform(X_{new})$

- Dữ liệu gốc có 201 mẫu, 4 thuộc tính  $\Rightarrow$  dữ liệu transform có 201 mẫu, 15 thuộc tính



## Model Deployment



- Xây dựng model trên dữ liệu mới, dự đoán kết quả

```
lm1_t = LinearRegression()  
lm1_t.fit(X1_pr, Y1)  
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)  
  
Yhat_t = lm1_t.predict(X1_pr)  
  
a1_t = lm1_t.intercept_  
b1_t = lm1_t.coef_  
  
a1_t, b1_t  
(-21262.84827778044,  
array([ 0.00000000e+00, 3.79990747e+02, 9.53081616e+00, -1.06265103e+02,  
1.82168928e+02, 1.69423474e+00, -1.86864385e-01, -1.29043091e+00,  
-2.14860064e+00, 7.57100954e-04, 1.11910005e-01, -1.92007463e-01,  
8.95252693e-02, 4.23869258e-02, 4.66019878e+00]))
```





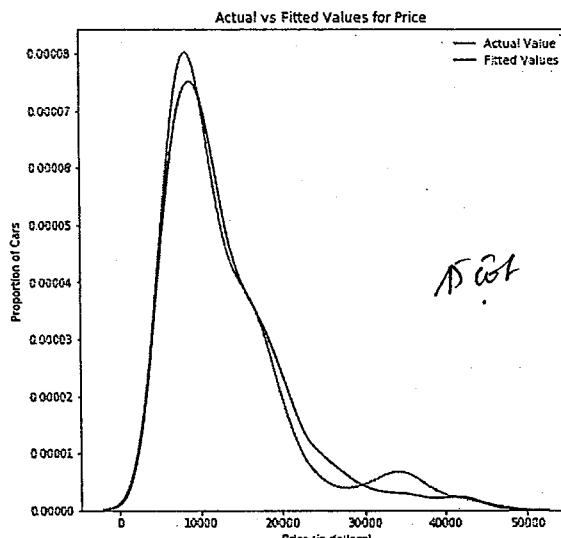
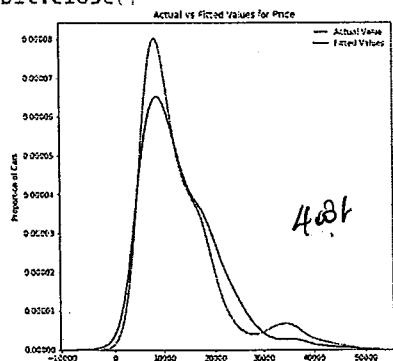
## Model Deployment

### ▪ Trực quan hóa kết quả

```
plt.figure(figsize=(8,8))
ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual Value")
sns.distplot(Yhat_t, hist=False, color="b", label="Fitted Values", ax=ax1)

plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price (in dollars)')
plt.ylabel('Proportion of Cars')

plt.show()
plt.close()
```



Multiple Linear Regression (original)

Data Pre-processing and Analysis

Multiple Linear Regression  
(polynomial transform)

35

## Model Deployment



### □ Pipeline

- Data Pipelines (Đường ống dữ liệu) giúp đơn giản hóa các bước xử lý dữ liệu.
- Chúng ta sử dụng module Pipeline để tạo pipeline và StandardScaler như một bước trong pipeline.





# Model Deployment

- Import thư viện
- Tạo pipeline, bằng cách tạo một danh sách các bộ dữ liệu bao gồm tên của mô hình hoặc công cụ ước tính và hàm khởi tạo tương ứng của nó.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
Input=[('scale',StandardScaler()), ('polynomial', PolynomialFeatures(include_bias=False)), ('model',LinearRegression())]

pipe=Pipeline(Input)
pipe
Pipeline(memory=None,
          steps=[('scale', StandardScaler(copy=True, with_mean=True, with_std=True)), ('polynomial', PolynomialFeatures(degree=2, include_bias=False, interaction_only=False)), ('model', LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False))])
```

*(gồm 3 bước chung 1 lần → lưu ý là không ját huỷ lời (nếu sai))*



# Model Deployment

- Chuẩn hóa dữ liệu, thực hiện chuyển đổi và điều chỉnh mô hình đồng thời.

pipe.fit(X1, Y1) (*fit 1 lần & chuẩn*)

```
Pipeline(memory=None,
          steps=[('scale', StandardScaler(copy=True, with_mean=True, with_std=True)), ('polynomial', PolynomialFeatures(degree=2, include_bias=False, interaction_only=False)), ('model', LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False))])
```

*polynomial mặc định 2*

- Dự đoán kết quả

```
y=pipe.predict(X1)
y=pipe[2:4]
array([13102.74784201, 13102.74784201, 18225.54572197, 10390.29636555])
```

*\* Ứng dụng hiện (lưu ý: lưu ý câu lệnh)*





## Model Deployment

### □ Measures for In-Sample Evaluation (Đo lường, tăng qua).

Các đo lường đánh giá trong mẫu

- Khi đánh giá các mô hình, chúng ta không chỉ trực quan hóa kết quả mà còn phải đo lường định lượng để xác định mức độ chính xác của mô hình.
- Có hai phép đo lường rất quan trọng thường được sử dụng trong Thống kê để xác định độ chính xác của mô hình là:

Thang đo  
đánh giá  
Regression

- Mean Squared Error (MSE) → Đo độ lỗi (độ lỗi càng nhỏ → model càng tốt)
- R<sup>2</sup> / R-squared → (0, 1) → càng lớn càng tốt, tuy nhiên kỹ thuật này không đồng

Data Pre-processing and Analysis

39

## Model Deployment



### • Mean Squared Error (MSE) → càng lớn lỗi càng nhiều.

- Lỗi bình phương trung bình đo trung bình của bình phương sai số, nghĩa là chênh lệch giữa giá trị thực, actual value (y) và giá trị ước tính, estimated value (.



# Model Deployment



## • R-squared (R<sup>2</sup>)

- R bình phương, còn được gọi là hệ số xác định (coefficient of determination), là thước đo để chỉ ra mức độ gần của dữ liệu với regression line được huấn luyện.
- Giá trị của R-squared là tỷ lệ phần trăm biến thiên của response variable (y) được giải thích bằng mô hình tuyến tính.

Tỷ lệ càng cao → càng nì điểm dự đoán vào con đường



# Model Deployment



## • Ví dụ Model 1: Simple Linear Regression

```
# Find the R^2
print('The R-square is: ', lm.score(X, Y))
The R-square is: 0.4965911884339176 , (chỗng 49,62% độ gần khía = model)
```

- Với kết quả trên, có thể nói rằng ~ 49.659% variation của price được giải thích bằng simple linear model này

```
# Find the MSE
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y, Yhat)
print('The mean square error of price and predicted value is: ', mse)

The mean square error of price and predicted value is: 31635042.944639883
```





## Model Deployment

### • Ví dụ Model 2: Multiple Linear Regression

```
# Find the R^2
print('The R-square is: ', lm1.score(X1, Y1))

The R-square is: 0.8093562806577457
```

- Với kết quả trên, có thể nói rằng ~ 80.093%. variation của price được giải thích bằng multiple linear model này

```
# Find MSE
print('The mean square error of price and predicted value using multifit is: ', \
      mean_squared_error(Y1, Yhat1))

The mean square error of price and predicted value using multifit is: 11980366.87072649
```



## Model Deployment

### • Ví dụ Model 3: Polynomial Fit

```
# Find R^2
from sklearn.metrics import r2_score
r_squared = r2_score(y, p(x))
print('The R-square value is: ', r_squared)

The R-square value is: 0.6741946663906515
```

- Với kết quả trên, có thể nói rằng ~ 67.419% variation của price được giải thích bằng polynomial fit này

```
# Find MSE
print('The mean square error of price and predicted value using Polinomial Fit: ', \
      mean_squared_error(y, p(x)))

The mean square error of price and predicted value using Polinomial Fit: 20474146.42636124
```



## Model Deployment



### • Ví dụ Model 4: Multiple Polynomial Fit

```
# Find R^2
from sklearn.metrics import r2_score
r_squared = r2_score(Y1, Yhat_t)
print('The R-square value is: ', r_squared)

The R-square value is: 0.846334195986473
```

- Với kết quả trên, có thể nói rằng ~84.633% variation của price được giải thích bằng multiple polynomial fit này

```
# Find MSE
print('The mean square error of price and predicted value using Polynomial Fit: ', \
      mean_squared_error(Y1, Yhat_t))

The mean square error of price and predicted value using Polynomial Fit: 9656613.403989587
```



## Model Deployment



### □ Decision Making

#### ● Prediction

- Sử dụng function predict() để dự đoán giá trị Y mới với input là X mới





## Model Deployment

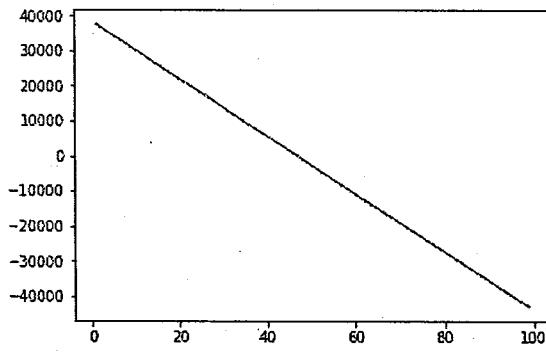
### ▪ Ví dụ: Model 1: Simple Linear Regression

```
# Create new input
new_input=np.arange(1, 100, 1).reshape(-1, 1)

yhat_new=lm.predict(new_input)
yhat_new[0:5]

array([37601.57247984, 36779.83910151, 35958.10572319, 35136.37234487,
       34314.63896655])

plt.plot(new_input, yhat_new)
plt.show()
```



Data Pre-processing and Analysis

47



## Model Deployment

### • Decision Making: Xác định mô hình phù hợp

- Chúng ta đã quan sát các mô hình khác nhau và tính các giá trị R-squared và MSE, vậy làm thế nào để chúng ta xác định một mô hình phù hợp?
  - Theo R-squared value: khi so sánh các mô hình, mô hình có giá trị R-squared cao hơn sẽ phù hợp hơn với dữ liệu.
  - Theo MSE: Khi so sánh các mô hình, mô hình có giá trị MSE nhỏ nhất phù hợp hơn với dữ liệu.



Data Pre-processing and Analysis

48



## Model Deployment

- Xem lại các giá trị cho các model khác nhau
  - Simple Linear Regression: sử dụng highway-mpg là Predictor Variable để dự đoán Price.
    - R-squared: 0.49659118843391759
    - MSE:  $3.16 \times 10^7$
  - Simple - Polynomial Fit: Sử dụng highway-mpg là Predictor Variable để dự đoán Price.
    - R-squared: 0.6741946663906514
    - MSE:  $2.05 \times 10^7$



## Model Deployment

- Multiple Linear Regression: Sử dụng horsepower, curb-weight, engine-size, và highway-mpg là Predictor Variables để dự đoán Price.
  - R-squared: 0.80896354913783497
  - MSE:  $1.2 \times 10^7$
- Multiple - Polynomial Fit: Sử dụng horsepower, curb-weight, engine-size, và highway-mpg là Predictor Variables để dự đoán Price.
  - R-squared: 0.846334196986473
  - MSE:  $9.66 \times 10^6$





## Model Deployment

- So sánh Simple Linear Regression model (SLR) và Multiple Linear Regression model (MLR)

▪ Thông thường, khi càng có nhiều biến số, mô hình sẽ càng được dự đoán tốt hơn, nhưng điều này không phải lúc nào cũng đúng. Đôi khi, chúng ta có thể không có đủ dữ liệu, hoặc có thể gặp phải các vấn đề về số hoặc nhiều biến cũng có thể không hữu ích và thậm chí hoạt động như nhiều (noise). Do đó, chúng ta phải luôn kiểm tra MSE và R-squared.



## Model Deployment

- So sánh Simple Linear Model (SLR) và Polynomial Fit
  - MSE: Chúng ta có thể thấy rằng Polynomial Fit đã giảm MSE, vì MSE này nhỏ hơn so với SLR.
  - R-squared: R-squared của Polinormial Fit lớn hơn R-squared của SLR, vì vậy Polynomial Fit cũng tăng R-squared.
  - Vì Polynomial Fit có MSE thấp hơn và R^2 cao hơn, chúng ta có thể kết luận rằng đây là mô hình phù hợp hơn so với SLR khi dự đoán price dựa trên highway-mpg.



## Model Deployment



- So sánh Multiple Linear Regression (MLR) và Polynomial Fit
  - MSE: MSE của MLR nhỏ hơn MSE của Polynomial Fit.
  - R-squared: R-squared của MLR cũng lớn hơn Polynomial Fit.



## Model Deployment



- So sánh Multiple Linear Regression (MLR) và Multiple - Polynomial Fit
  - MLR: MSE của MLR lớn hơn MSE của Multiple - Polynomial Fit.
  - R-squared: R-squared của MLR cũng nhỏ hơn Polynomial Fit.





## Model Deployment

### • Kết luận

- So sánh bốn mô hình này, chúng ta kết luận mô hình Multiple - Polynomial Fit là mô hình tốt nhất để có thể dự đoán giá từ bộ dữ liệu. Kết quả này có ý nghĩa, vì chúng ta có tổng cộng 27 biến và chúng ta biết rằng có nhiều hơn một biến trong số các biến đó là các yếu tố dự đoán của giá xe.



## Nội dung

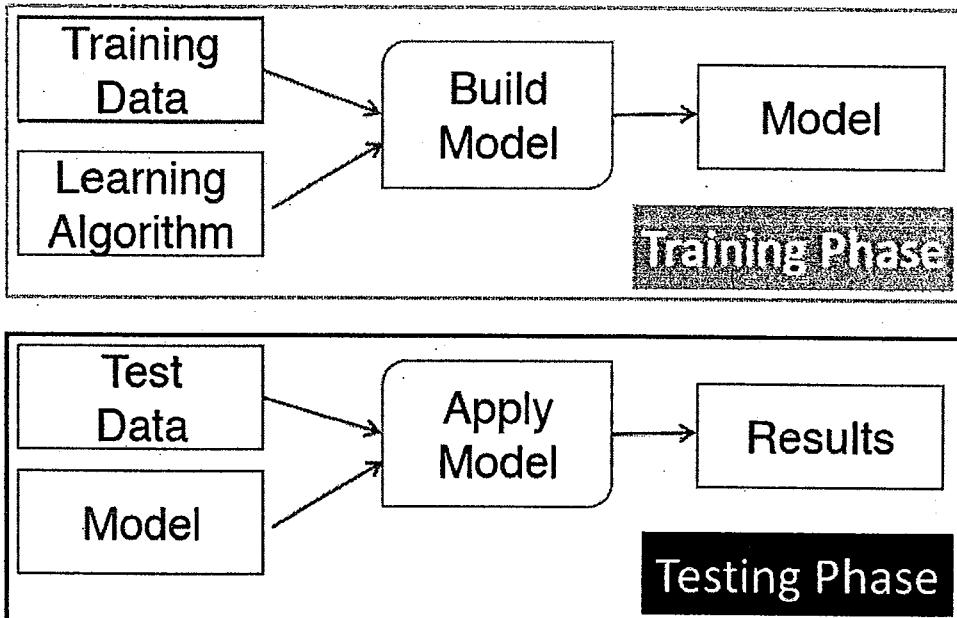
1. Giới thiệu
2. Model Deployment
3. Model Evaluation



## Model Evaluation



### □ Training & Testing



Data Pre-processing and Analysis

57

## Model Evaluation



- Một bước quan trọng trong việc kiểm tra mô hình là chia dữ liệu thành training data (dữ liệu huấn luyện) và testing data (dữ liệu kiểm tra)
- Sử dụng sklearn.model\_selection.train\_test\_split() để chia ngẫu nhiên dữ liệu thành training data và testing data



## Model Evaluation



- Ví dụ: Cho dữ liệu dự đoán price dựa trên horsepower

- Đọc dữ liệu

```
df = pd.read_csv("auto.csv")
df.head()
```

```
df=df._get_numeric_data()
df.head()
```

| boling | normalized-losses | wheel-base | length   | width    | height | curb-weight | ... | stroke | compression-ratio | horsepower | peak-rpm | city-mpg | highway-mpg | price   | city-L/100km | diesel | gas |
|--------|-------------------|------------|----------|----------|--------|-------------|-----|--------|-------------------|------------|----------|----------|-------------|---------|--------------|--------|-----|
| 3      | 122               | 88.6       | 0.811148 | 0.890278 | 48.8   | 2548        | ... | 2.68   | 9.0               | 111.0      | 5000.0   | 21       | 27          | 13495.0 | 11.190476    | 0      | 1   |
| 3      | 122               | 88.6       | 0.811148 | 0.890278 | 48.8   | 2548        | ... | 2.68   | 9.0               | 111.0      | 5000.0   | 21       | 27          | 16500.0 | 11.190476    | 0      | 1   |
| 1      | 122               | 94.5       | 0.822681 | 0.909722 | 52.4   | 2823        | ... | 3.47   | 9.0               | 154.0      | 5000.0   | 19       | 26          | 16500.0 | 12.368421    | 0      | 1   |
| 2      | 164               | 99.8       | 0.848630 | 0.919444 | 54.3   | 2337        | ... | 3.40   | 10.0              | 102.0      | 5500.0   | 24       | 30          | 13950.0 | 9.791667     | 0      | 1   |
| 2      | 164               | 99.4       | 0.848630 | 0.922222 | 54.3   | 2824        | ... | 3.40   | 8.0               | 115.0      | 5500.0   | 18       | 22          | 17450.0 | 13.055556    | 0      | 1   |



## Model Evaluation



- Chia dữ liệu thành training và testing data

```
y_data = df['price']

x_data=df.drop('price',axis=1)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.15, random_state=1)
print("number of test samples : ", x_test.shape[0])
print("number of training samples: ",x_train.shape[0])

number of test samples : 31
number of training samples: 170
```

15% bộ test  
giả nguyên nhử với khai random  
lần sau.



## Model Evaluation



- Huấn luyện mô hình với training data

```
from sklearn.linear_model import LinearRegression  
  
lre = LinearRegression()  
  
lre.fit(x_train[['horsepower']], y_train)  
  
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)
```

- Kiểm tra kết quả bằng cách so sánh  $R^2$  của mô hình khi áp dụng training data và testing data

```
#  $R^2$ , with training data  
lre.score(x_train[['horsepower']], y_train)  
0.6449517437659685  
  
#  $R^2$ , with testing data  
lre.score(x_test[['horsepower']], y_test)  
0.707688374146705
```

$R^2$  của training  
data <  $R^2$  của  
testing data

61

## Model Evaluation



### Cross-validation

- Đôi khi, chúng ta không có đủ dữ liệu thử nghiệm => chúng ta có thể thực hiện Cross-validation - xác thực chéo.

- Sử dụng

sklearn.model\_selection.cross\_val\_score để  
xác thực chéo





## Model Evaluation

### ▪ Ví dụ

```
from sklearn.model_selection import cross_val_score  
  
Rcross = cross_val_score(lre, x_data[['horsepower']], y_data, cv=4)  
  
Rcross  
array([0.7746232, 0.51716687, 0.74785353, 0.04839605])  
  
print("The mean of the folds are", Rcross.mean(), "and the standard deviation is", Rcross.std())  
The mean of the folds are 0.522009915042119 and the standard deviation is 0.2911839444756029
```

*Mô hình chưa tốt.*



## Model Evaluation

• Chúng ta cũng có thể sử dụng 'sklearn.model\_selection.cross\_val\_predict' để dự đoán output. Hàm chia dữ liệu thành số fold được chỉ định, sử dụng fold lần để dự đoán trong khi phần còn lại của các fold được sử dụng làm dữ liệu testing data.

### ▪ Ví dụ:

```
from sklearn.model_selection import cross_val_predict  
  
yhat = cross_val_predict(lre, x_data[['horsepower']], y_data, cv=4)  
yhat[0:5]  
  
array([14141.63807508, 14141.63807508, 20814.29423473, 12745.03562306,  
14762.35027598])
```



## Model Evaluation



### ❑ Overfitting, Underfitting & Model Selection

- Overfitting: cho biết dữ liệu thử nghiệm đôi khi được gọi là dữ liệu mẫu là một thước đo tốt hơn nhiều so với thực tế khi áp dụng mô hình.



## Model Evaluation



### ● Ví dụ:

- Tạo Multiple linear regression objects và huấn luyện mô hình sử dụng input là 'horsepower', 'curb-weight', 'engine-size' and 'highway-mpg'.

```
lr = LinearRegression()  
  
lr.fit(x_train[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']], y_train)  
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)
```





## Model Evaluation

- Dự đoán y cho training data và testing data

```
# dự đoán cho training data
yhat_train = lr.predict(x_train[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']])
yhat_train[0:5]

array([11927.70699817, 11236.71672034, 6436.91775515, 21890.22064982,
       16667.18254832])

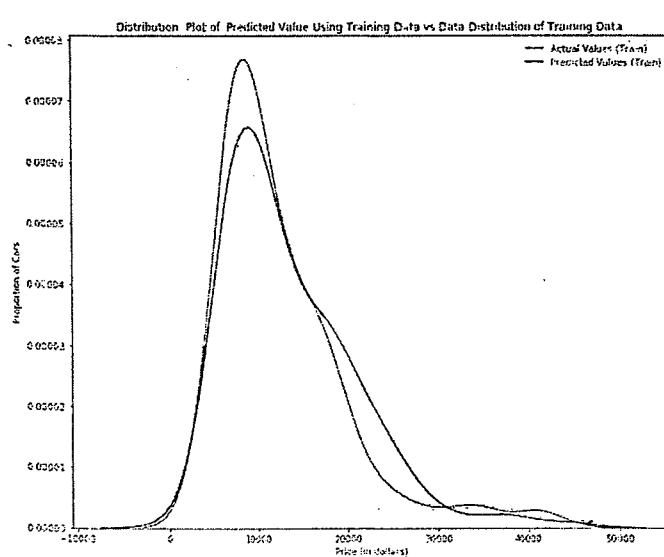
# dự đoán cho testing data
yhat_test = lr.predict(x_test[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']])
yhat_test[0:5]

array([11349.16502418, 5914.48335385, 11243.76325987, 6662.03197043,
       15555.76936275])
```



## Model Evaluation

- Thực hiện một số đánh giá mô hình bằng cách sử dụng riêng training data và testing data
  - Training data



Mô hình có vẻ hoạt động tốt với training dataset. Nhưng điều gì xảy ra khi mô hình gặp dữ liệu mới từ testing dataset?

Hình 1

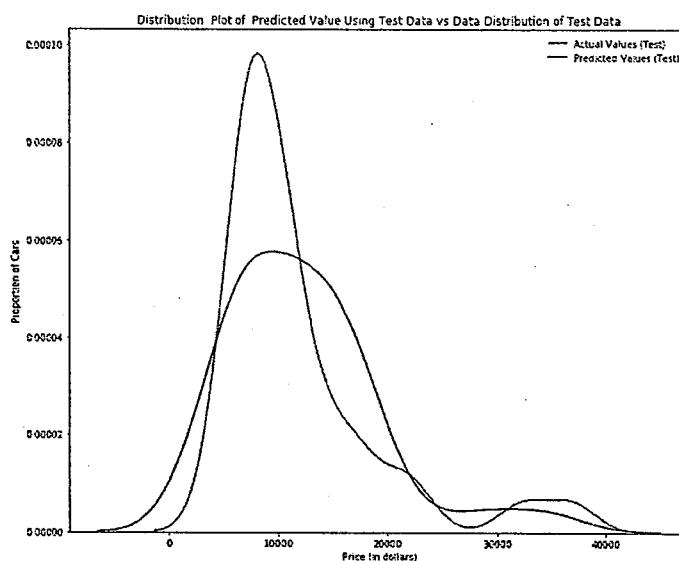


# Model Evaluation



## ▪ Testing data

Title="Distribution Plot of Predicted Value Using Test Data vs Data Distribution of Test Data"  
DistributionPlot(y\_test,yhat\_test,"Actual Values (Test)", "Predicted Values (Test)",Title)



Khi mô hình tạo các giá trị mới từ testing data, chúng ta thấy phân phối của các giá trị dự đoán (predicted values) khác nhiều so với các giá trị thực tế (actual values).

Hình 2

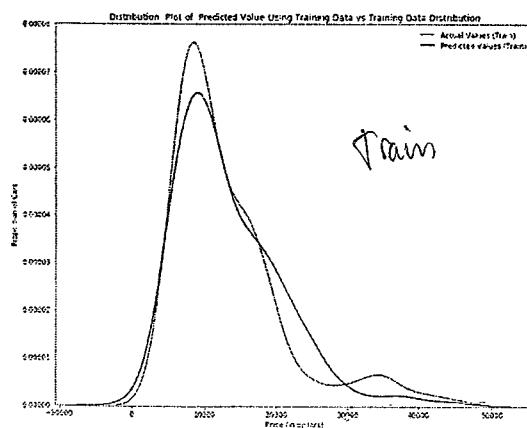


# Model Evaluation

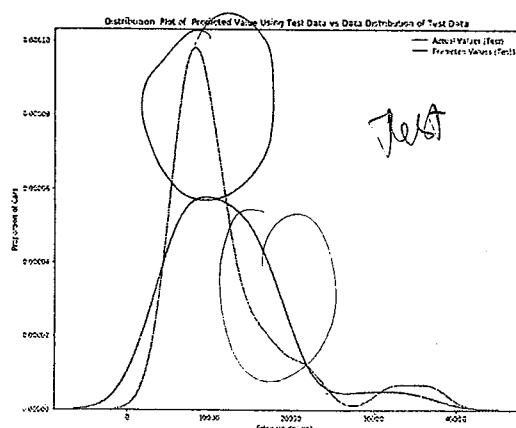


## ▪ Nhận xét

- So sánh Hình 1 và Hình 2, chúng ta thấy rõ ràng là sự phân phối dữ liệu thử nghiệm ở trong Hình 1 tốt hơn nhiều trong việc fit dữ liệu. Sự khác biệt này trong Hình 2 là rõ ràng dễ thấy trong các phạm vi từ 5000 đến 15 000. Đây là nơi hình dạng phân phối đặc biệt khác nhau.



Hình 1



Hình 2





## Model Evaluation

- Thực hiện với polynomial regression để xem độ chính xác ở testing data như thế nào.
  - Overfitting: xảy ra khi mô hình phù hợp nhiễu (noise). Do đó, khi kiểm tra mô hình bằng cách sử dụng testing data, mô hình không hoạt động tốt như mô hình nhiễu
  - Hãy tạo một mô hình đa thức bậc 5.



## Model Evaluation

- Ví dụ: Hãy tạo một mô hình đa thức bậc 5 để biến đổi thuộc tính 'horsepower'
  - Tạo dữ liệu x\_train\_pr và x\_test\_pr với PolynomialFeatures

```
from sklearn.preprocessing import PolynomialFeatures  
  
# Sử dụng 45% dataset làm testing data  
  
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.45, random_state=0)  
  
pr = PolynomialFeatures(degree=5)  
x_train_pr = pr.fit_transform(x_train[['horsepower']])  
x_test_pr = pr.fit_transform(x_test[['horsepower']])  
pr  
PolynomialFeatures(degree=5, include_bias=True, interaction_only=False)
```





## Model Evaluation

- Huấn luyện mô hình bằng dữ liệu mới

```

poly = LinearRegression()
poly.fit(x_train_pr, y_train)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
    
```

- Tìm predicted y bằng cách áp dụng mô hình cho x\_test\_pr

```

yhat = poly.predict(x_test_pr)
yhat[0:5]

array([ 6728.65571575, 7307.9879184 , 12213.78778443, 18893.24786348,
       19995.95173556])

print("Predicted values:", yhat[0:4])
print("True values:", y_test[0:4].values)

Predicted values: [ 6728.65571575 7307.9879184 12213.78778443 18893.24786348]
True values: [ 6295. 10698. 13860. 13499.]

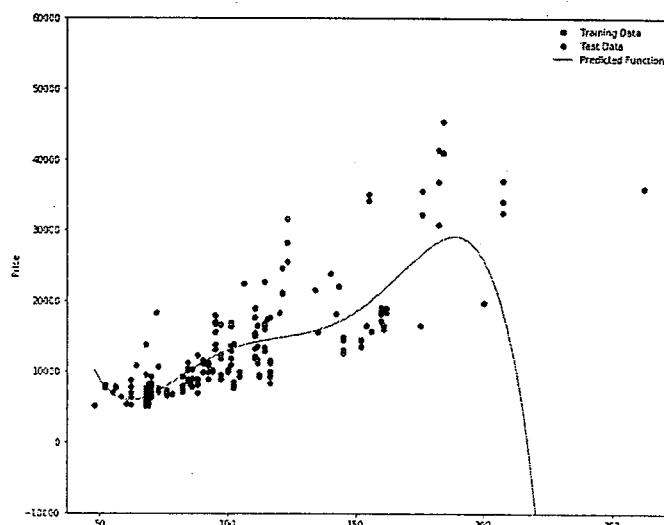
```



## Model Evaluation

- Trực quan hóa dữ liệu

```
PolyPlot(x_train[['horsepower']], x_test[['horsepower']], y_train, y_test, poly.pr)
```



**Polynomial regression model :** các chấm màu đỏ biểu thị dữ liệu huấn luyện, màu xanh lá biểu thị dữ liệu thử nghiệm và đường màu xanh biển biểu thị dự đoán từ mô hình. Ta thấy rằng chức năng ước tính xuất hiện để theo dõi dữ liệu nhưng khoảng từ 200 của horsepower, chức năng bắt đầu phân kỳ khỏi các điểm dữ liệu.





## Model Evaluation

- Tìm  $R^2$  của  $x_{train\_pr}$  và  $x_{test\_pr}$

```
# R^2 of the training data:  
poly.score(x_train_pr, y_train)
```

0.5567716902123356

```
# R^2 of the test data:  
poly.score(x_test_pr, y_test)
```

-29.87134086622682

Chúng ta thấy  $R^2$  cho dữ liệu huấn luyện là 0.5567 trong khi  $R^2$  trên dữ liệu thử nghiệm là -29,87.  $R^2$  càng thấp, mô hình càng tệ,  $R^2$  âm là dấu hiệu của overfitting hoặc  $\beta^2$  quá lớn.



## Model Evaluation

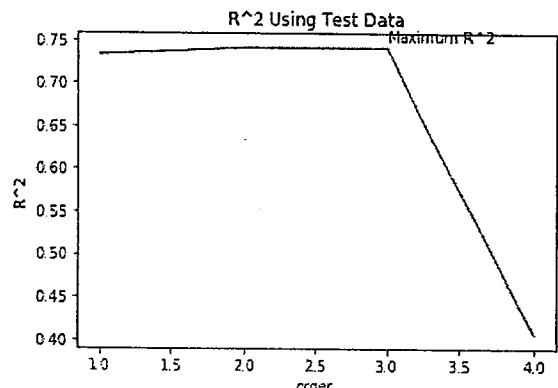


- Xem xét sự thay đổi của  $R^2$  trên dữ liệu thử nghiệm cho các đa thức bậc khác nhau và vẽ kết quả:

```
Rsqu_test = []  
order = [1, 2, 3, 4]  
lst = [0, 0]  
for n in order:  
    pr = PolynomialFeatures(degree=n)  
    x_train_pr = pr.fit_transform(x_train[['horsepower']])  
    x_test_pr = pr.fit_transform(x_test[['horsepower']])  
    lr.fit(x_train_pr, y_train)  
    score = lr.score(x_test_pr, y_test)  
    Rsqu_test.append(score)  
    if score > lst[1]:  
        lst = [n, score]  
print(n, lr.score(x_test_pr, y_test))  
  
plt.plot(order, Rsqu_test)  
plt.xlabel('order')  
plt.ylabel('R^2')  
plt.title('R^2 Using Test Data')  
plt.text(lst[0], lst[1], 'Maximum R^2')
```

Chọn như thế nào? qđ thử được?  
→ Thử chọn 1, 2, 3 đều đc.

1 0.7325101750243118  
2 0.74058587106999  
3 0.7406204232708171  
4 0.40639663604240456  
Text(3, 0.74062, 'Maximum R^2')



R<sup>2</sup> tăng dần cho đến đa thức bậc 3,  
sau đó, R<sup>2</sup> giảm dần kể từ bậc 4.







## Chapter 8

Bài vui phổi trẻ em

### Ex1: Lung Function in 1 to 10 Year Old Children

- Cho dữ liệu children\_lung.txt. Bộ dữ liệu có 345 trẻ em từ 1 đến 10 tuổi. Với output y = forced exhalation volume (FEV), thước đo lượng không khí mà ai đó có thể buộc phải thở ra từ phổi của họ và x = age in years. (Nguồn dữ liệu: Dữ liệu ở đây là một phần của bộ dữ liệu được đưa ra trong Kahn, Michael (2005). "An Exhalent Problem for Teaching Statistics", The Journal of Statistical Education, 13".

#### Yêu cầu:

- Đọc dữ liệu, tiền xử lý dữ liệu, tổng quan ban đầu về dữ liệu.
- Trực quan hóa dữ liệu. Quan sát và nhận xét. Có thấy vấn đề gì đặc biệt từ dữ liệu không? Nếu có thì đó là gì?
- Từ câu 2., xem xét việc tách bài toán này thành 2 phần có được không? Nếu được thì triển khai.

#### Gợi ý, dữ liệu phần 1

- Thực hiện Simple Linear Regression để dự đoán FEV từ age. Nhận xét kết quả. Trực quan hóa kết quả.
- Cho age lần lượt là: [2, 3, 4, 5]. Hãy cho biết FEV lần lượt là bao nhiêu?
- Đánh giá mô hình vừa xây dựng: có cần phải cải tiến gì không? Nếu cần thì thay đổi mô hình.
- Nhận xét kết quả. Trực quan hóa kết quả với mô hình mới thay đổi. So sánh với mô hình ban đầu. Mô hình sau có tốt hơn không? Quyết định chọn mô hình nào? Lý do?

#### Gợi ý, dữ liệu phần 2

- Thực hiện Multiple Linear Regression để dự đoán FEV từ age và ht. Nhận xét kết quả. Trực quan hóa kết quả.
- Cho age và ht in year lần lượt là: age = [5, 6, 7, 8, 9], ht = [49.5, 55, 57, 60, 62]. Hãy cho biết FEV lần lượt là bao nhiêu?
- Đánh giá mô hình vừa xây dựng: có cần phải cải tiến gì không? Nếu cần thì thay đổi mô hình.
- Nhận xét kết quả. Trực quan hóa kết quả với mô hình mới thay đổi. So sánh với mô hình ban đầu. Mô hình sau có tốt hơn không? Quyết định chọn mô hình nào? Lý do?

In [1]: # Link download dữ liệu: <https://newonlinecourses.science.psu.edu/stat462/node/101>

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```



In [3]: `### Đọc dữ liệu  
data = pd.read_csv("children_lung.txt", sep=" ")  
data.head()`

Out[3]:

|   | Unnamed:<br>0 | Unnamed:<br>1 | age | FEV   | ht   | Unnamed:<br>5 | Unnamed:<br>6 | sex | Unnamed:<br>8 | Unnamed:<br>9 | Unn. |
|---|---------------|---------------|-----|-------|------|---------------|---------------|-----|---------------|---------------|------|
| 0 | NaN           | NaN           | 9.0 | 1.708 | 57.0 | NaN           | NaN           | 0.0 | NaN           | NaN           |      |
| 1 | NaN           | NaN           | 8.0 | 1.724 | 67.5 | NaN           | NaN           | 0.0 | NaN           | NaN           |      |
| 2 | NaN           | NaN           | 7.0 | 1.720 | 54.5 | NaN           | NaN           | 0.0 | NaN           | NaN           |      |
| 3 | NaN           | NaN           | 9.0 | 1.558 | 53.0 | NaN           | NaN           | 1.0 | NaN           | NaN           |      |
| 4 | NaN           | NaN           | 9.0 | 1.895 | 57.0 | NaN           | NaN           | 1.0 | NaN           | NaN           |      |

In [4]: `data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 654 entries, 0 to 653
Data columns (total 15 columns):
Unnamed: 0      0 non-null float64
Unnamed: 1      345 non-null float64
age             654 non-null float64
FEV             654 non-null float64
ht              309 non-null float64
Unnamed: 5      0 non-null float64
Unnamed: 6      345 non-null float64
sex             309 non-null float64
Unnamed: 8      0 non-null float64
Unnamed: 9      0 non-null float64
Unnamed: 10     0 non-null float64
Unnamed: 11     345 non-null float64
smoke           309 non-null float64
Unnamed: 13     0 non-null float64
Unnamed: 14     0 non-null float64
dtypes: float64(15)
memory usage: 76.7 KB
```

In [5]: `data = data[["age", "FEV", "ht", "sex", "smoke"]]  
data.head()`

Out[5]:

|   | age | FEV   | ht   | sex | smoke |
|---|-----|-------|------|-----|-------|
| 0 | 9.0 | 1.708 | 57.0 | 0.0 | 0.0   |
| 1 | 8.0 | 1.724 | 67.5 | 0.0 | 0.0   |
| 2 | 7.0 | 1.720 | 54.5 | 0.0 | 0.0   |
| 3 | 9.0 | 1.558 | 53.0 | 1.0 | 0.0   |
| 4 | 9.0 | 1.895 | 57.0 | 1.0 | 0.0   |

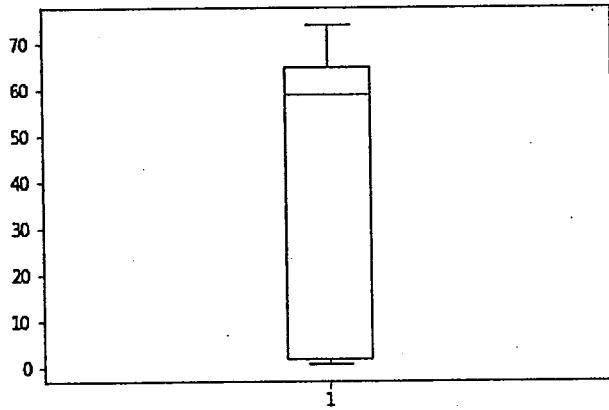
In [6]: `data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 654 entries, 0 to 653
Data columns (total 5 columns):
age    654 non-null float64
FEV    654 non-null float64
ht     309 non-null float64
sex    309 non-null float64
smoke  309 non-null float64
dtypes: float64(5)
memory usage: 25.6 KB
```

In [7]: `data.describe()`

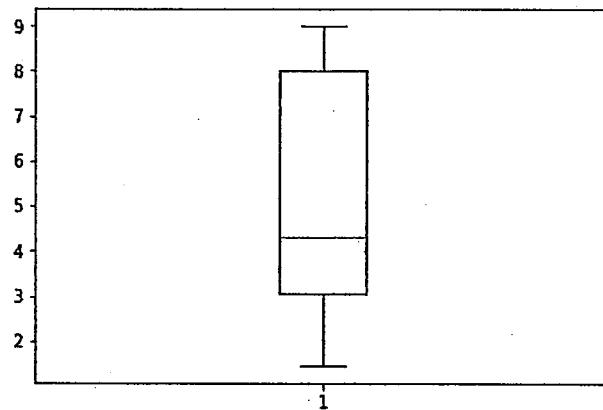
Out[7]:

|       | age        | FEV        | ht         | sex        | smoke      |
|-------|------------|------------|------------|------------|------------|
| count | 654.000000 | 654.000000 | 309.000000 | 309.000000 | 309.000000 |
| mean  | 5.205369   | 35.209546  | 56.922330  | 0.498382   | 0.003236   |
| std   | 2.429315   | 31.541283  | 4.408388   | 0.500808   | 0.056888   |
| min   | 1.458000   | 0.791000   | 46.000000  | 0.000000   | 0.000000   |
| 25%   | 3.048500   | 2.040500   | 53.500000  | 0.000000   | 0.000000   |
| 50%   | 4.330000   | 59.000000  | 57.000000  | 0.000000   | 0.000000   |
| 75%   | 8.000000   | 65.000000  | 60.000000  | 1.000000   | 0.000000   |
| max   | 9.000000   | 74.000000  | 69.000000  | 1.000000   | 1.000000   |

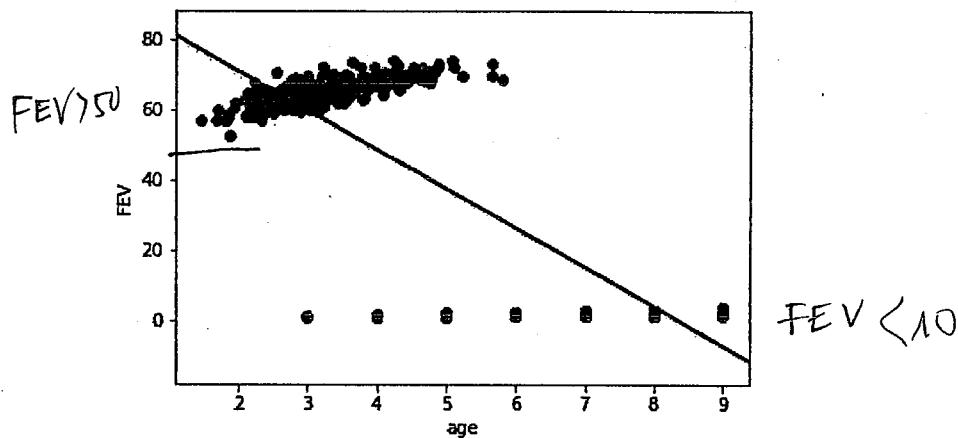
In [8]: `plt.boxplot(data.FEV)`  
`plt.show()`



```
In [9]: plt.boxplot(data.age)
plt.show()
```



```
In [10]: sns.regplot(data=data, x='age', y='FEV')
plt.show()
```



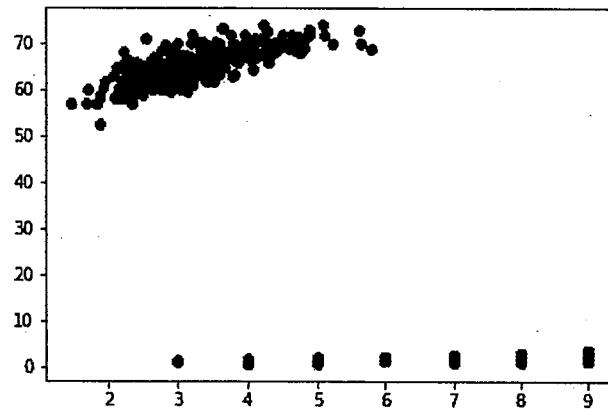
7/25/2019

Ex1



```
In [11]: plt.scatter(data.age, data.FEV)
```

```
Out[11]: <matplotlib.collections.PathCollection at 0x1960999e748>
```



```
In [12]: FEV_lower_10 = data[data.FEV < 10]  
FEV_lower_10.shape
```

```
Out[12]: (309, 5)
```

```
In [13]: FEV_lower_10.info()
```

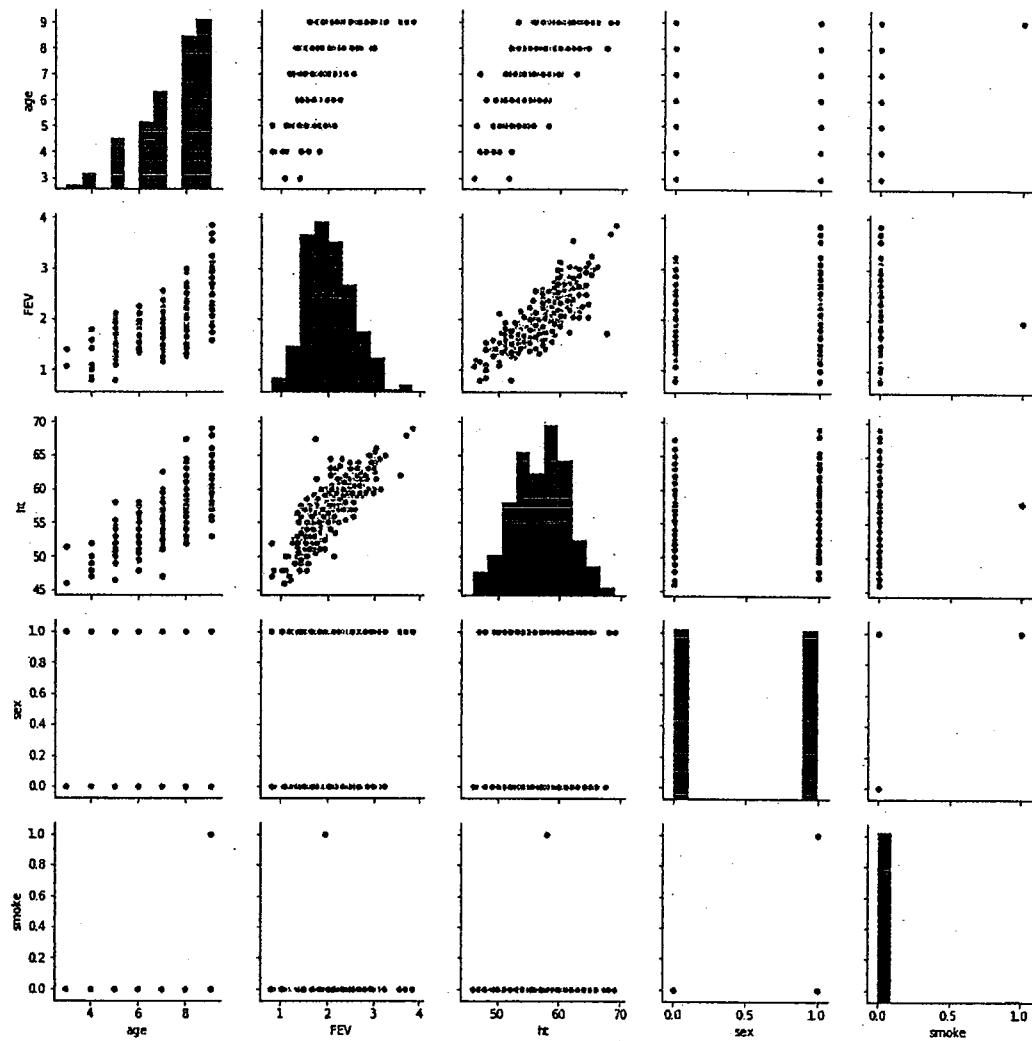
```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 309 entries, 0 to 308  
Data columns (total 5 columns):  
age      309 non-null float64  
FEV      309 non-null float64  
ht       309 non-null float64  
sex      309 non-null float64  
smoke    309 non-null float64  
dtypes: float64(5)  
memory usage: 14.5 KB
```

7/25/2019

Ex1

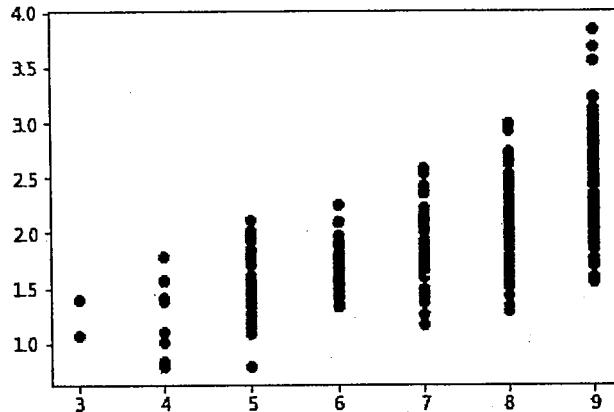


```
In [14]: sns.pairplot(FEV_lower_10)
plt.show()
```



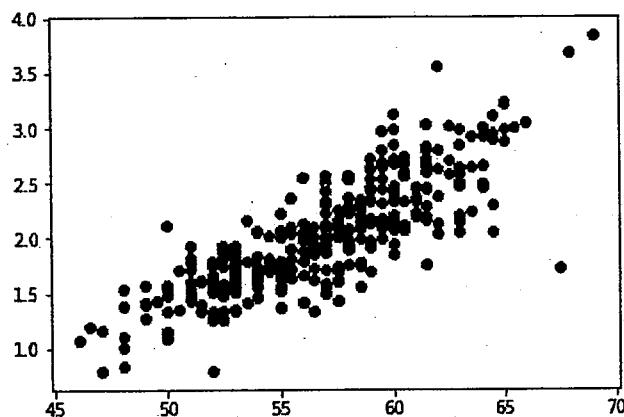


```
In [15]: plt.scatter(FEV_lower_10.age, FEV_lower_10.FEV)
plt.show()
```



```
In [16]: # Nhóm này có tuổi từ 3 đến 9, chủ yếu tập trung ở Lứa tuổi từ 5 trở lên. Có thể t
```

```
In [17]: plt.scatter(FEV_lower_10.ht, FEV_lower_10.FEV)
plt.show()
```



```
In [18]: # ht tương đối tuyến tính so với FEV
```

```
In [19]: # Smoke: trẻ em không hút thuốc
```

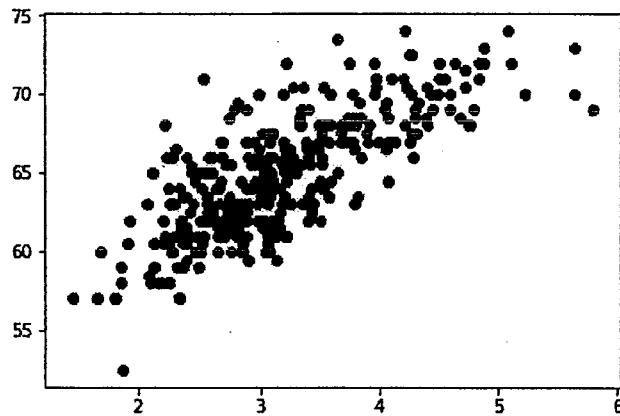
```
In [20]: FEV_upper_10 = data[data.FEV >= 10]
FEV_upper_10.shape
```

```
Out[20]: (345, 5)
```



In [21]: FEV\_upper\_10.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 345 entries, 309 to 653
Data columns (total 5 columns):
age      345 non-null float64
FEV      345 non-null float64
ht       0 non-null float64
sex      0 non-null float64
smoke    0 non-null float64
dtypes: float64(5)
memory usage: 16.2 KB
```

In [22]: plt.scatter(FEV\_upper\_10.age, FEV\_upper\_10.FEV)  
plt.show()

In [23]: # Nhóm này có tuổi từ 0 đến 6, tập trung nhiều từ 2 đến 5, tương đối tuyến tính

In [24]: # Chia làm 2 nhóm tương đương là FEV nhỏ hơn 10 và FEV lớn hơn 10.  
# Quan sát thấy với nhóm có FEV\_lower\_10 có 345 mẫu với đầy đủ dữ liệu,  
# Trong khi đó nhóm FEV\_upper\_10 chỉ có dữ liệu của age và FEV.  
# Liệu có nên chia bài toán thành 2 phần, dự đoán theo từng phần? Tham số của từng

## Phần 1: FEV\_upper\_10, dành cho dữ liệu chỉ có age

### Dự đoán FEV từ age

## Simple Linear Regression

In [25]: from sklearn.linear\_model import LinearRegression



```
In [26]: lm = LinearRegression()
lm
```

```
Out[26]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                           normalize=False)
```

```
In [27]: X_u = FEV_upper_10[['age']]
Y_u = FEV_upper_10['FEV']
```

```
In [28]: # Chia dữ liệu thành 2 phần, train và test
from sklearn.model_selection import train_test_split
```

```
In [29]: X_train_u, X_test_u, y_train_u, y_test_u = train_test_split(X_u, Y_u, test_size=0.)
```

```
In [30]: # Train model
lm.fit(X_train_u,y_train_u)
```

```
Out[30]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                           normalize=False)
```

```
In [31]: b = lm.intercept_
b
```

```
Out[31]: 52.17543272888112
```

```
In [32]: m = lm.coef_[0]
m
```

```
Out[32]: 3.9873414785499666
```

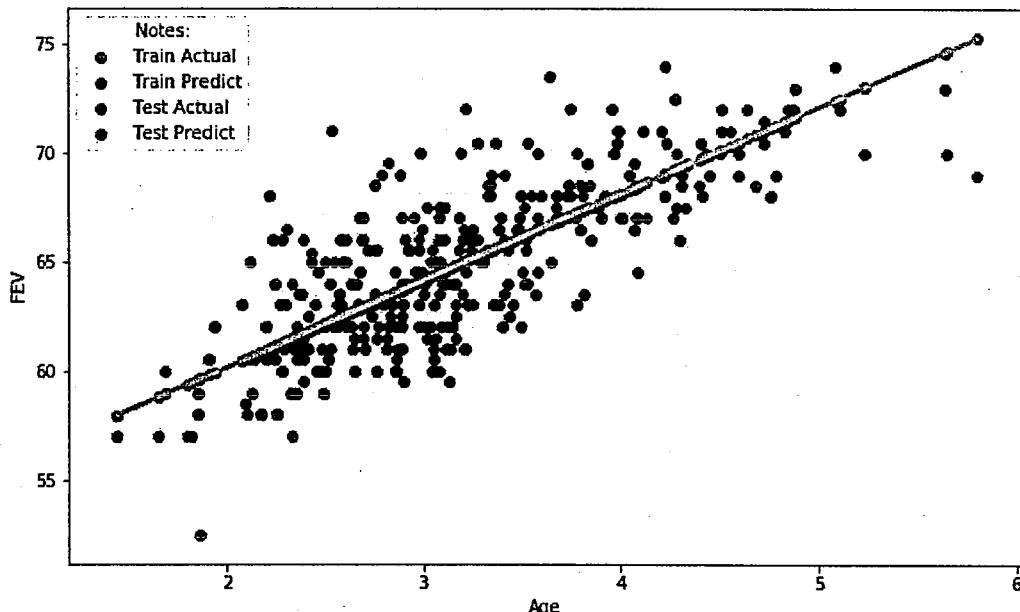
"y = mx + b "

```
In [33]: # Predict trên X_train_u
yHat_train = lm.predict(X_train_u)
```

```
In [34]: # Predict trên X_test_u
yHat_test = lm.predict(X_test_u)
```

```
In [35]: reg_line = [(m* float(x)) + b for x in np.array(X_u)]
```

```
In [36]: plt.figure(figsize=(10,6))
plt.plot(X_u, reg_line, color="orange", linewidth=3)
plt.scatter(X_train_u, y_train_u, color='red', label='Train Actual')
plt.scatter(X_train_u, yHat_train, color='green', label='Train Predict')
plt.scatter(X_test_u, y_test_u, color='black', label='Test Actual')
plt.scatter(X_test_u, yHat_test, color='blue', label='Test Predict')
plt.xlabel("Age")
plt.ylabel("FEV")
plt.legend(title="Notes:")
plt.show()
```



```
In [37]: # Find the MSE
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y_u, lm.predict(X_u))
print('The mean square error of FEV and predicted value is: ', mse)
```

The mean square error of FEV and predicted value is: 5.91756259463458

```
In [38]: # Find the R^2
print('The R-square is: ', lm.score(X_u, Y_u))
```

The R-square is: 0.57634645569363

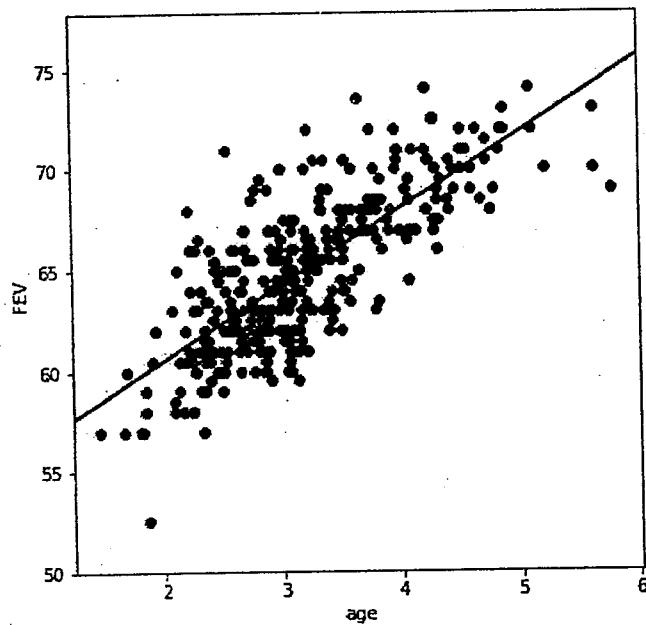
```
In [39]: # predict new data
X_new_u = pd.DataFrame(np.array([2, 3, 4, 5]))
y_new_u = lm.predict(X_new_u)
y_new_u
```

Out[39]: array([60.15011569, 64.13745716, 68.12479864, 72.11214012])



```
In [40]: # regression plot  
plt.figure(figsize=(6,6))  
sns.regplot(x="age", y="FEV", data=FEV_upper_10)  
plt.ylim(50,)
```

Out[40]: (50, 77.82593943908267)

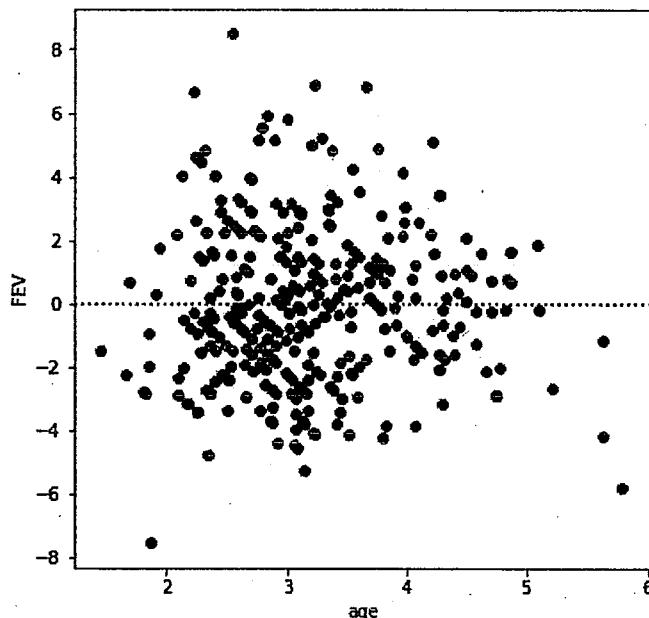


7/25/2019

Ex1



```
In [41]: # Residual plot  
plt.figure(figsize=(6,6))  
sns.residplot(FEV_upper_10['age'], FEV_upper_10['FEV'])  
plt.show()
```



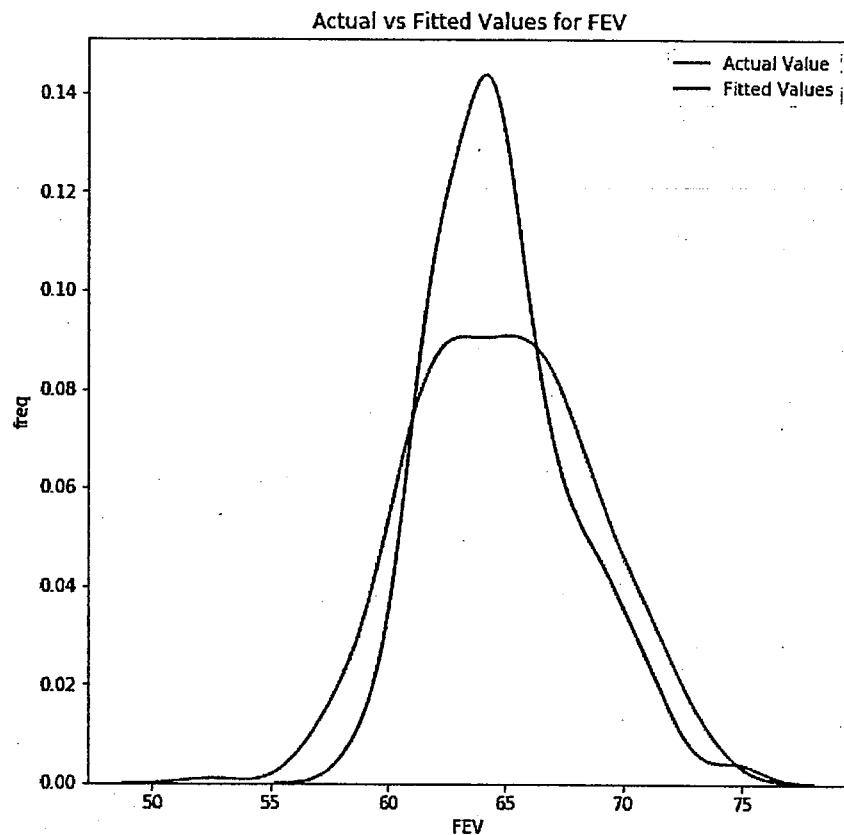
```
In [42]: Y_u_pred = lm.predict(X_u)
```



```
In [43]: # Distribution plot
plt.figure(figsize=(8,8))
ax1 = sns.distplot(Y_u, hist=False, color="r", label="Actual Value")
sns.distplot(Y_u_pred, hist=False, color="b", label="Fitted Values", ax=ax1)

plt.title('Actual vs Fitted Values for FEV')
plt.xlabel('FEV')
plt.ylabel('freq')

plt.show()
plt.close()
```



## Polinomial Regression



```
In [44]: def PlotPolly(model, independent_variable, dependent_variable, Name):
    x_new = np.linspace(1, 6, 300)
    y_new = model(x_new)

    plt.plot(independent_variable, dependent_variable, '.', x_new, y_new, '-')
    plt.title('Polynomial Fit with Matplotlib for FEV ~ Age')
    ax = plt.gca()
    ax.set_facecolor((0.898, 0.898, 0.898))
    fig = plt.gcf()
    plt.xlabel(Name)
    plt.ylabel('FEV')

    plt.show()
    plt.close()
```

```
In [45]: import numpy as np
```

```
In [46]: X_u_p = FEV_upper_10['age']
Y_u_p = FEV_upper_10['FEV']
```

```
In [47]: # X_u, Y_u
# Here we use a polynomial of the 3rd order (cubic)
f = np.polyfit(X_u_p, Y_u_p, 3)
p = np.poly1d(f)
p
```

```
Out[47]: poly1d([-0.14017129,  1.02159835,  1.95386068, 53.11846748])
```

```
In [48]: f
```

```
Out[48]: array([-0.14017129,  1.02159835,  1.95386068, 53.11846748])
```

```
In [49]: # Find R^2
from sklearn.metrics import r2_score
r_squared = r2_score(Y_u_p, p(X_u_p))
print('The R-square value of Polinomial Regression model is: ', r_squared)
```

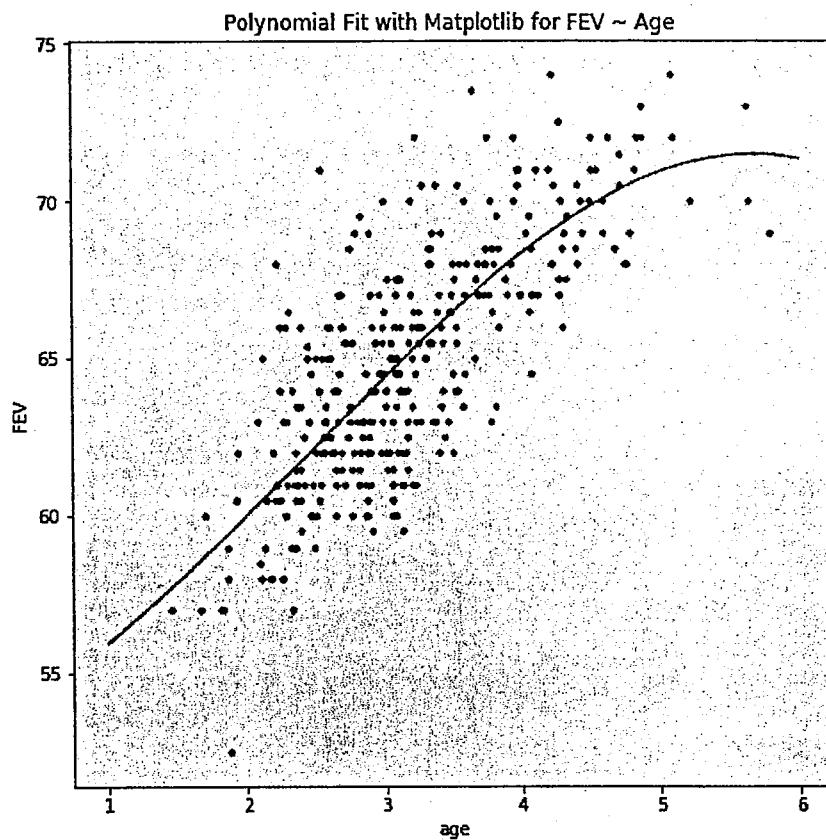
The R-square value of Polinomial Regression model is: 0.5897528827371681

```
In [50]: # Find the MSE
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y_u_p, p(X_u_p))
print('The mean square error of FEV and predicted value is: ', mse)
```

The mean square error of FEV and predicted value is: 5.730302574585822



```
In [51]: plt.figure(figsize=(8,8))
PlotPolly(p, X_u_p, Y_u_p, 'age')
```



```
In [52]: # predict new data
X_new_p = np.array([2, 3, 4, 5])
Y_new_p = p(X_new_p)
Y_new_p
```

```
Out[52]: array([59.99121194, 64.38980989, 68.30852135, 70.90631857])
```

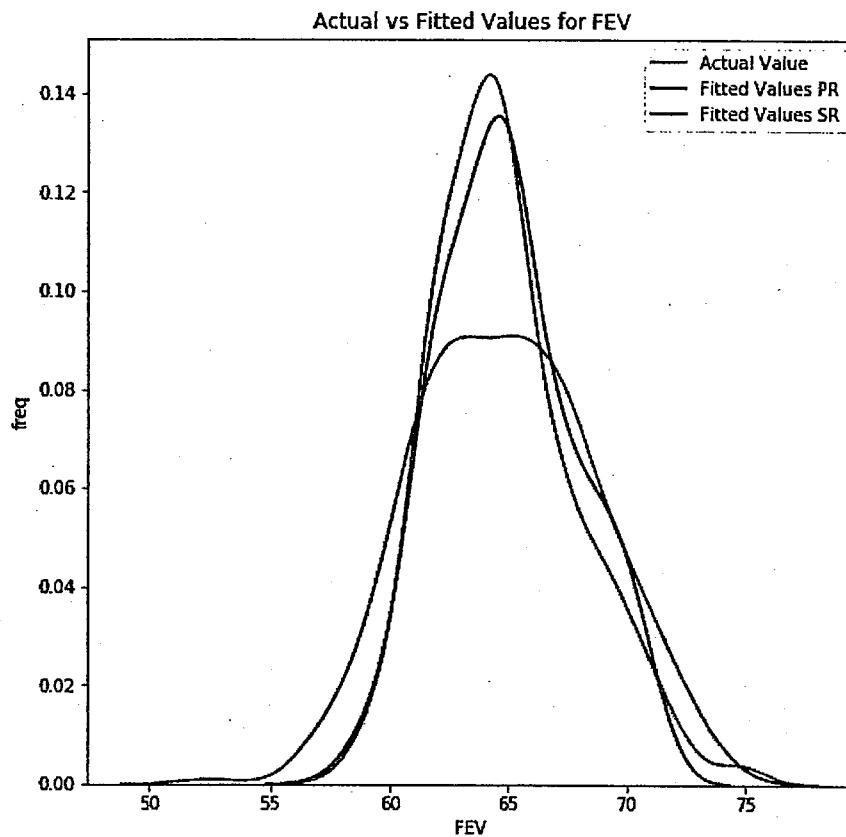
```
In [53]: # trực quan hóa dữ liệu
Y_u_p_pred = p(X_u_p)
```



```
In [54]: # Distribution plot
plt.figure(figsize=(8,8))
ax1 = sns.distplot(Y_u_p, hist=False, color="r", label="Actual Value")
sns.distplot(Y_u_p_pred, hist=False, color="b", label="Fitted Values PR" , ax=ax1)
sns.distplot(Y_u_pred, hist=False, color="g", label="Fitted Values SR" , ax=ax1)

plt.title('Actual vs Fitted Values for FEV')
plt.xlabel('FEV')
plt.ylabel('freq')

plt.show()
plt.close()
```



```
In [55]: # PR tốt hơn so với SR => có thể chọn PR
```

```
In [56]: ### Phần 2: FEV_Lower_10, dành cho dữ liệu có age, ht
### Dự đoán FEV từ age, ht
```

## Multiple Linear Regression

```
In [57]: # Với X_l có age, ht => y_l
```



```
In [58]: X_l = FEV_lower_10[['age', 'ht']]
Y_l = FEV_lower_10['FEV']
```

```
In [59]: X_train_l, X_test_l, y_train_l, y_test_l = train_test_split(X_l, Y_l, test_size=0.2)
```

```
In [60]: lm1 = LinearRegression()
lm1
```

```
Out[60]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                           normalize=False)
```

```
In [61]: # Train model
lm1.fit(X_train_l,y_train_l)
```

```
Out[61]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                           normalize=False)
```

```
In [62]: b = lm1.intercept_
b
```

```
Out[62]: -3.3451304885308764
```

```
In [63]: m1, m2 = lm1.coef_[0], lm1.coef_[1]
m1, m2
```

```
Out[63]: (0.016615263396740344, 0.0922009278932709)
```

```
In [64]: # Predict trên X_train_l
yHat_train_l = lm1.predict(X_train_l)
```

```
In [65]: # Predict trên X_test_u
yHat_test_l = lm1.predict(X_test_l)
```

```
In [66]: # Find the MSE
mse1 = mean_squared_error(Y_l, lm1.predict(X_l))
print('The mean square error of price and predicted value is: ', mse1)
```

The mean square error of price and predicted value is: 0.08537795222524265

```
In [67]: # Find the R^2
print('The R-square is: ', lm1.score(X_l, Y_l))
```

The R-square is: 0.6726855966146938

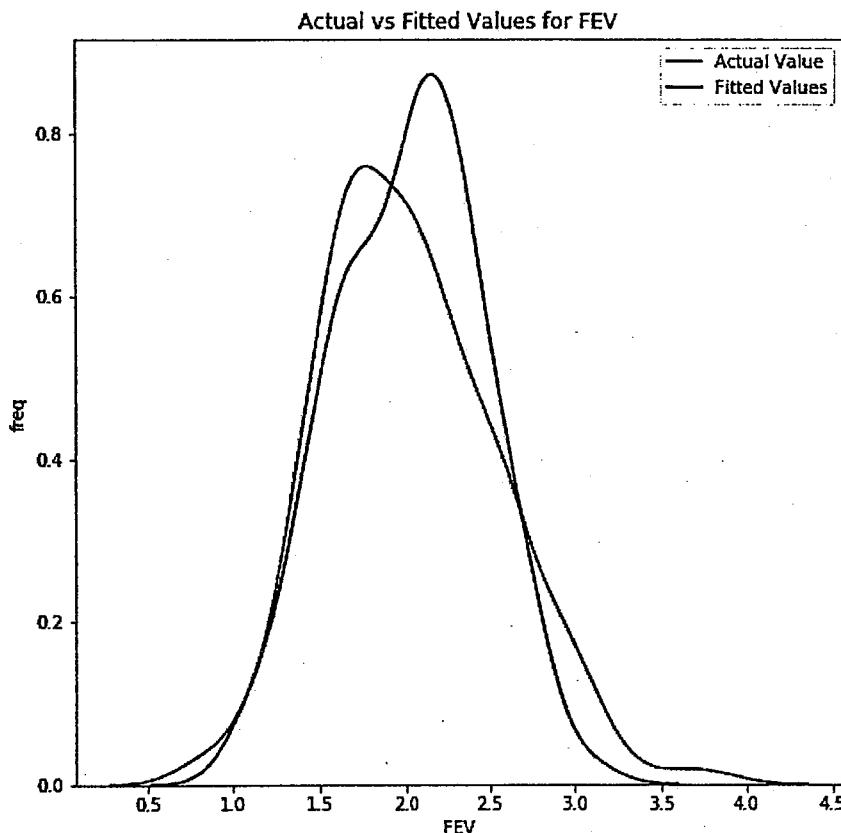
```
In [68]: # New prediction
age = [5, 6, 7, 8, 9]
ht = [49.5, 55, 57, 60, 62]
X_l_new = pd.DataFrame({'age': age, 'ht': ht})
yHat_l_new = lm1.predict(X_l_new)
yHat_l_new
```

```
Out[68]: array([1.30189176, 1.82561213, 2.02662925, 2.31984729, 2.52086441])
```



In [69]: `yHat_1 = lm1.predict(X_1)`

In [70]: `# Đánh giá mô hình  
# Distribution plot  
plt.figure(figsize=(8,8))  
ax1 = sns.distplot(Y_1, hist=False, color="r", label="Actual Value")  
sns.distplot(yHat_1, hist=False, color="b", label="Fitted Values" , ax=ax1)  
  
plt.title('Actual vs Fitted Values for FEV')  
plt.xlabel('FEV')  
plt.ylabel('freq')  
  
plt.show()  
plt.close()`



## Multiple Polynominal

In [71]: `from sklearn.preprocessing import PolynomialFeatures`

In [72]: `pr=PolynomialFeatures(degree=2)  
pr`

Out[72]: `PolynomialFeatures(degree=2, include_bias=True, interaction_only=False)`



```
In [73]: X_1_pr=pr.fit_transform(X_1)
```

```
In [74]: X_1.shape, X_1_pr.shape
```

```
Out[74]: ((309, 2), (309, 6))
```

```
In [75]: lm1_t = LinearRegression()
```

```
In [76]: lm1_t.fit(X_1_pr, Y_1)
```

```
Out[76]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

```
In [77]: Yhat_t = lm1_t.predict(X_1_pr)
```

```
In [78]: b1_t = lm1_t.intercept_
```

```
In [79]: m1_t = lm1_t.coef_
```

```
In [80]: b1_t, m1_t
```

```
Out[80]: (-0.42182399116130265,
array([ 0.          , -0.58556875,  0.06971195, -0.01718083,  0.01546802,
       -0.00085841]))
```

```
In [81]: X_1_new = pd.DataFrame({'age': age, 'ht':ht})
X_1_pr_new=pr.fit_transform(X_1_new)
yHat_1_t_new = lm1_t.predict(X_1_pr_new)
yHat_1_t_new
```

```
Out[81]: array([1.39657488, 1.78817454, 1.99368881, 2.31115186, 2.56998682])
```

```
In [82]: # Find R^2
from sklearn.metrics import r2_score
r_squared = r2_score(Y_1, Yhat_t)
print('The R-square value is: ', r_squared)
```

```
The R-square value is:  0.6823609472684569
```

```
In [83]: # Find MSE
print('The mean square error of price and predicted value using Polinomial Fit: '
      'mean_squared_error(Y_1, Yhat_t))
```

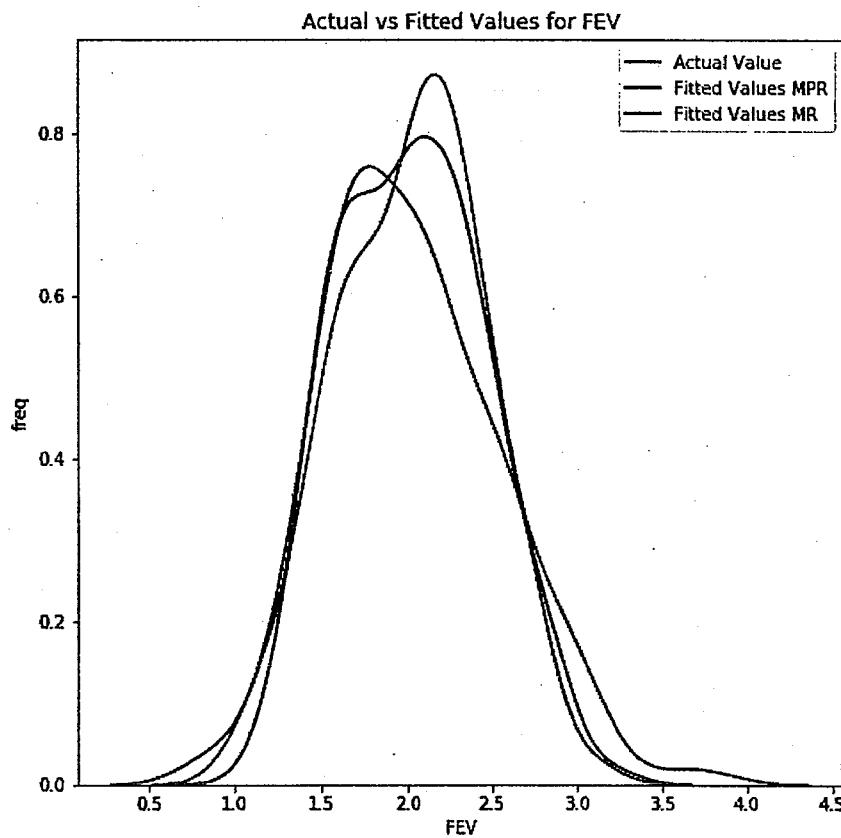
```
The mean square error of price and predicted value using Polinomial Fit:  0.0
8285419641940045
```



```
► In [84]: # Đánh giá mô hình
# Distribution plot
plt.figure(figsize=(8,8))
ax1 = sns.distplot(Y_1, hist=False, color="r", label="Actual Value")
sns.distplot(Yhat_t, hist=False, color="b", label="Fitted Values MPR" , ax=ax1)
sns.distplot(yHat_1, hist=False, color="g", label="Fitted Values MR" , ax=ax1)

plt.title('Actual vs Fitted Values for FEV')
plt.xlabel('FEV')
plt.ylabel('freq')

plt.show()
plt.close()
```



```
In [85]: # MPL cho kết quả tốt hơn so với MR => Có thể chọn MPL
```



## Ex2: Economy

- Cho dữ liệu economy.xlsx. Bộ dữ liệu chứa thông tin về các tỷ lệ Interest\_Rate, Unemployment\_Rate và Stock\_Index\_Price theo thời gian (năm, tháng). Người ta dự đoán Stock\_Index\_Price dựa trên Interest\_Rate, hoặc Unemployment\_Rate hoặc cả 2.

### Yêu cầu:

- Đọc dữ liệu, tiền xử lý dữ liệu, tổng quan ban đầu về dữ liệu.

#### 1. Single variable

- Thực hiện Simple Linear Regression để dự đoán Stock\_Index\_Price từ Interest\_Rate, với các đánh giá mô hình: Training vs Testing 80:20, Cross Validation.
- Nhận xét kết quả. Trực quan hóa kết quả.
- Kiểm chứng overfitting, underfitting của model vừa chọn
- Đánh giá mô hình vừa xây dựng: có cần phải cải tiến gì không? Nếu cần thì thay đổi mô hình.
- Nhận xét kết quả. Trực quan hóa kết quả với mô hình mới thay đổi. So sánh với mô hình ban đầu. Mô hình sau có tốt hơn không? Quyết định chọn mô hình nào? Lý do?

#### 2. Multiple variables

- Thực hiện Multiple Linear Resgression để dự đoán Stock\_Index\_Price từ Interest\_Rate, Unemployment\_Rate, với các đánh giá mô hình: Training vs Testing 80:20, Cross Validation.
- Nhận xét kết quả. Trực quan hóa kết quả.
- Kiểm chứng overfitting, underfitting của model vừa chọn
- Đánh giá mô hình vừa xây dựng: có cần phải cải tiến gì không? Nếu cần thì thay đổi mô hình.
- Nhận xét kết quả. Trực quan hóa kết quả với mô hình mới thay đổi. So sánh với mô hình ban đầu. Mô hình sau có tốt hơn không? Quyết định chọn mô hình nào? Lý do?

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: def DistributionPlot(RedFunction, BlueFunction, RedName, BlueName, Title):
    ax1 = sns.distplot(RedFunction, hist=False, color="r", label=RedName)
    ax2 = sns.distplot(BlueFunction, hist=False, color="b", label=BlueName, ax=ax1)

    plt.title>Title
    plt.xlabel('Stock Index Price')
    plt.ylabel('Proportion')
```



```
In [3]: def PollyPlot(xtrain, xtest, y_train, y_test, lr,poly_transform):
    width = 12
    height = 10
    plt.figure(figsize=(width, height))

    #training data
    #testing data
    #lr: Linear regression object
    #poly_transform: polynomial transformation object

    xmax=max([xtrain.values.max(), xtest.values.max()])
    xmin=min([xtrain.values.min(), xtest.values.min()])
    x=np.arange(xmin, xmax, 0.1)

    plt.plot(xtrain, y_train, 'ro', label='Training Data')
    plt.plot(xtest, y_test, 'go', label='Test Data')
    plt.plot(x, lr.predict(poly_transform.fit_transform(x.reshape(-1, 1))), label=
    plt.ylim([0, 2000])# nho chinh lai ylim cho phu hop voi tung bai toan
    plt.ylabel('Stock Index Price')
    plt.legend()
```

```
In [4]: # Đọc dữ liệu, tiền xử lý dữ liệu, tổng quan ban đầu về dữ liệu
data = pd.read_excel("economy.xlsx")
```

```
In [5]: data.head()
```

Out[5]:

|   | Year | Month | Interest_Rate | Unemployment_Rate | Stock_Index_Price |
|---|------|-------|---------------|-------------------|-------------------|
| 0 | 2017 | 12    | 2.75          | 5.3               | 1464              |
| 1 | 2017 | 11    | 2.50          | 5.3               | 1394              |
| 2 | 2017 | 10    | 2.50          | 5.3               | 1357              |
| 3 | 2017 | 9     | 2.50          | 5.3               | 1293              |
| 4 | 2017 | 8     | 2.50          | 5.4               | 1256              |

```
In [6]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 5 columns):
Year          24 non-null int64
Month         24 non-null int64
Interest_Rate 24 non-null float64
Unemployment_Rate 24 non-null float64
Stock_Index_Price 24 non-null int64
dtypes: float64(2), int64(3)
memory usage: 1.0 KB
```



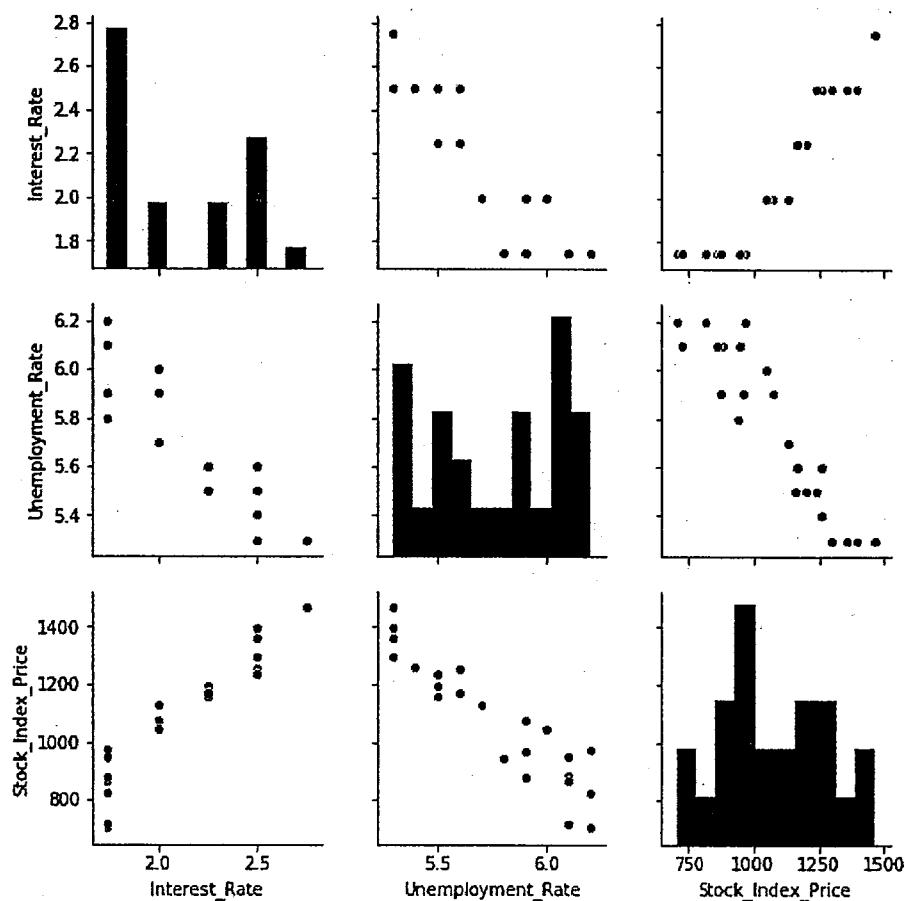
In [7]: `data.describe()`

Out[7]:

|       | Year        | Month     | Interest_Rate | Unemployment_Rate | Stock_Index_Price |
|-------|-------------|-----------|---------------|-------------------|-------------------|
| count | 24.000000   | 24.000000 | 24.000000     | 24.000000         | 24.000000         |
| mean  | 2016.500000 | 6.500000  | 2.072917      | 5.775000          | 1070.083333       |
| std   | 0.510754    | 3.526299  | 0.349527      | 0.33002           | 210.735341        |
| min   | 2016.000000 | 1.000000  | 1.750000      | 5.30000           | 704.000000        |
| 25%   | 2016.000000 | 3.750000  | 1.750000      | 5.50000           | 928.250000        |
| 50%   | 2016.500000 | 6.500000  | 2.000000      | 5.85000           | 1061.000000       |
| 75%   | 2017.000000 | 9.250000  | 2.500000      | 6.10000           | 1239.000000       |
| max   | 2017.000000 | 12.000000 | 2.750000      | 6.20000           | 1464.000000       |

In [8]: `# không có dữ liệu null`

In [9]: `sns.pairplot(data[["Interest_Rate", "Unemployment_Rate", "Stock_Index_Price"]])  
plt.show()`





In [10]: `# Nhận xét: Stock Index Price gần như tương quan thuận với Interest Rate  
# và gần như tương quan nghịch với Unemployment Rate`

## Single Variable

### Simple Linear Regression

#### Training & Testing

In [11]: `from sklearn.model_selection import train_test_split  
from sklearn.linear_model import LinearRegression`

In [12]: `y = data.Stock_Index_Price  
X = data[["Interest_Rate"]]`

In [13]: `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_`

In [14]: `print("number of test samples :", X_test.shape[0])  
print("number of training samples:", X_train.shape[0])`

number of test samples : 5  
number of training samples: 19

In [15]: `lre = LinearRegression()`

In [16]: `lre.fit(X_train, y_train)`

Out[16]: `LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)`

In [17]: `# R^2, with training data  
lre.score(X_train, y_train)`

Out[17]: `0.9047208848737547`

In [18]: `# R^2, with testing data  
lre.score(X_test, y_test)`

Out[18]: `0.557883455637328`

In [19]: `# R^2, with dataset  
lre.score(X, y)`

Out[19]: `0.8751503552613233`



In [20]: # Có sự chênh lệch score ~0.1 giữa training dataset và testing dataset  
# Với cả dataset: có thể nói rằng ~ 87.52% variation của Stock Index Price  
# được giải thích bằng simple Linear model này.

In [21]: `yhat_LR = lre.predict(X)`

In [22]: `df_compare_LR = pd.DataFrame({"Actual": y.values, "Predicted": yhat_LR})  
df_compare_LR.head()`

Out[22]:

|   | Actual | Predicted   |
|---|--------|-------------|
| 0 | 1464   | 1442.990489 |
| 1 | 1394   | 1305.483696 |
| 2 | 1357   | 1305.483696 |
| 3 | 1293   | 1305.483696 |
| 4 | 1256   | 1305.483696 |

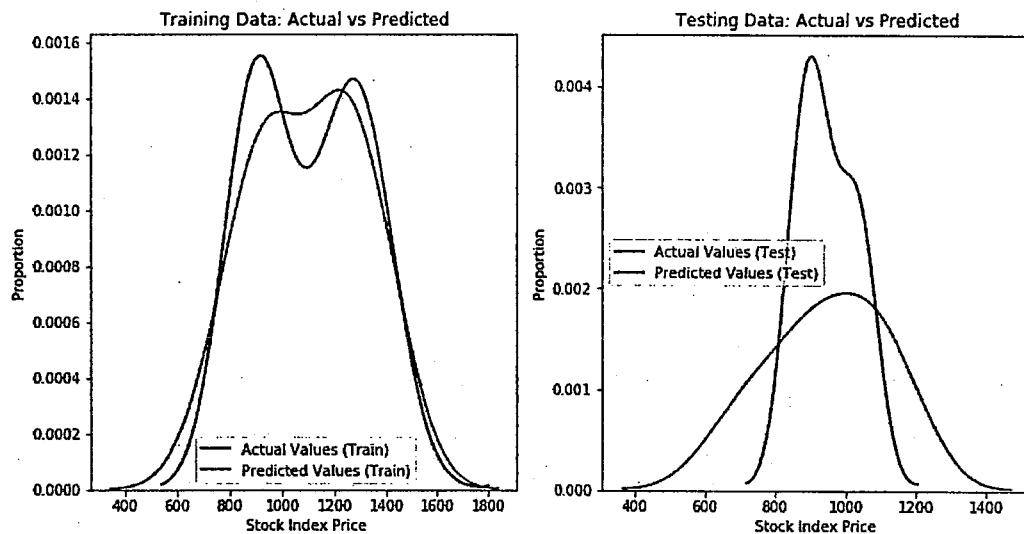
## Overfitting, Underfitting & Model Selection

In [23]: `yhat_train = lre.predict(X_train)`

In [24]: `yhat_test = lre.predict(X_test)`



```
In [25]: plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
Title = 'Training Data: Actual vs Predicted'
DistributionPlot(y_train, yhat_train, "Actual Values (Train)", "Predicted Values (Train)", Title)
plt.subplot(1,2,2)
Title = 'Testing Data: Actual vs Predicted'
DistributionPlot(y_test, yhat_test, "Actual Values (Test)", "Predicted Values (Test)", Title)
plt.show()
```



```
In [26]: y_train_mean = y_train.mean()
yhat_train_mean = yhat_train.mean()
display(y_train_mean, yhat_train_mean)
```

1102.842105263158

1102.8421052631577

```
In [27]: y_train_std = y_train.std()
yhat_train_std = yhat_train.std()
display(y_train_std, yhat_train_std)
```

212.0765750482842

196.3403049033732

```
In [28]: y_test_mean = y_test.mean()
yhat_test_mean = yhat_test.mean()
display(y_test_mean, yhat_test_mean)
```

945.6

947.9660326086954



```
In [29]: y_test_std = y_test.std()
yhat_test_std = yhat_test.std()
display(y_test_std, yhat_test_std)
```

168.7966231889726

67.36429603760097

Với kết quả trên ta thấy:

- Training Data: thì trung bình dự đoán bằng thực tế, độ lệch chuẩn của dự đoán thấp hơn thực tế, do đó phân phối ở dự đoán có hình vẽ cao hơn thực tế
- Testing Data: thì trung bình dự đoán lớn hơn thực tế, độ lệch chuẩn của dự đoán thấp hơn thực tế, , do đó phân phối ở dự đoán có hình vẽ cao hơn thực tế
- So sánh Hình 1 và Hình 2, chúng ta thấy là sự phân phối dữ liệu thử nghiệm ở trong Hình 1 tốt hơn trong việc fit dữ liệu.
- Sự khác biệt này trong Hình 2 rất rõ ràng để nhận biết.

## Cross validation

```
In [30]: from sklearn.model_selection import cross_val_score
```

```
In [31]: Rcross = cross_val_score(lre, X, y, cv=4)
```

```
In [32]: Rcross
```

```
Out[32]: array([ 0.54003747, -0.41018072, -9.10622679, -3.82769559])
```

```
In [33]: print("The mean of the folds are", Rcross.mean(), "and the standard deviation is"
The mean of the folds are -3.201016409479328 and the standard deviation is 3.7
76522357133681
```

```
In [34]: -1 * cross_val_score(lre,X, y, cv=4, scoring='neg_mean_squared_error')
```

```
Out[34]: array([ 2695.48263889, 3496.30806753, 12196.25064078, 26237.58682948])
```

```
In [35]: from sklearn.model_selection import cross_val_predict
```

```
In [36]: yhat = cross_val_predict(lre,X, y, cv=4)
yhat[0:5]
```

```
Out[36]: array([1443.83333333, 1305.08333333, 1305.08333333, 1305.08333333,
1305.08333333])
```



```
In [37]: df_compare = pd.DataFrame({"Actual": y.values, "Predicted": yhat})
df_compare.head()
```

Out[37]:

|   | Actual | Predicted   |
|---|--------|-------------|
| 0 | 1464   | 1443.833333 |
| 1 | 1394   | 1305.083333 |
| 2 | 1357   | 1305.083333 |
| 3 | 1293   | 1305.083333 |
| 4 | 1256   | 1305.083333 |

## Áp dụng Polinomial Regression

```
In [38]: from sklearn.preprocessing import PolynomialFeatures
```

```
In [39]: # Sử dụng 45% dataset làm testing data
```

```
In [40]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.45, random_s
```

```
In [41]: pr = PolynomialFeatures(degree=3)
X_train_pr = pr.fit_transform(X_train)
X_test_pr = pr.fit_transform(X_test)
pr
```

nếu dùng Vong lặp.

```
Out[41]: PolynomialFeatures(degree=3, include_bias=True, interaction_only=False)
```

```
In [42]: X_train.shape, X_train_pr.shape
```

```
Out[42]: ((13, 1), (13, 4))
```

```
In [43]: poly = LinearRegression()
poly.fit(X_train_pr, y_train)
```

```
Out[43]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

```
In [44]: yhat = poly.predict(X_test_pr)
yhat[0:5]
```

```
Out[44]: array([1056.97959184, 1056.97959184, 895.30102041, 895.30102041,
895.30102041])
```



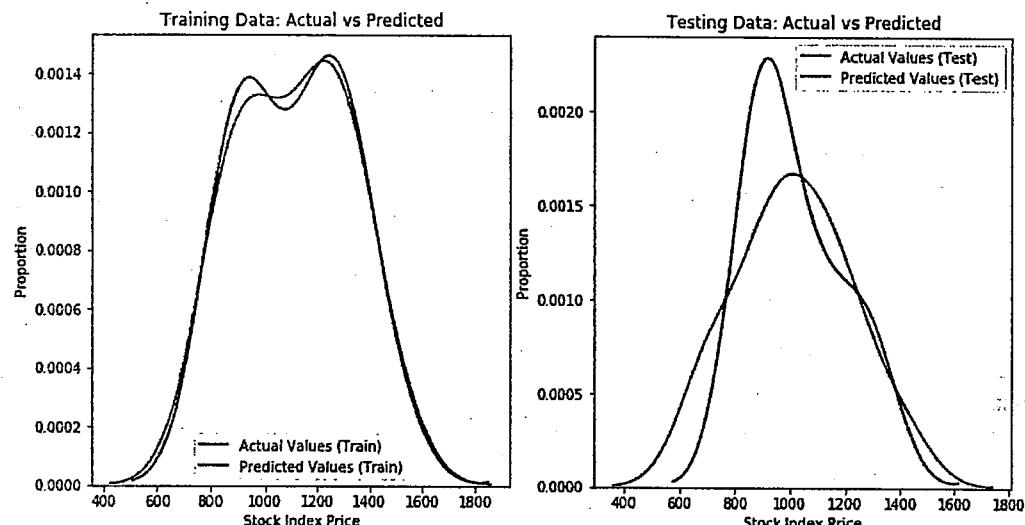
```
In [45]: df_compare_pr= pd.DataFrame({"Actual": y_test.values, "Predicted": yhat})
df_compare_pr.head()
```

Out[45]:

|   | Actual | Predicted   |
|---|--------|-------------|
| 0 | 1075   | 1056.979592 |
| 1 | 1130   | 1056.979592 |
| 2 | 704    | 895.301020  |
| 3 | 943    | 895.301020  |
| 4 | 876    | 895.301020  |

```
In [46]: yhat_test_pr= poly.predict(X_test_pr)
yhat_train_pr = poly.predict(X_train_pr)
```

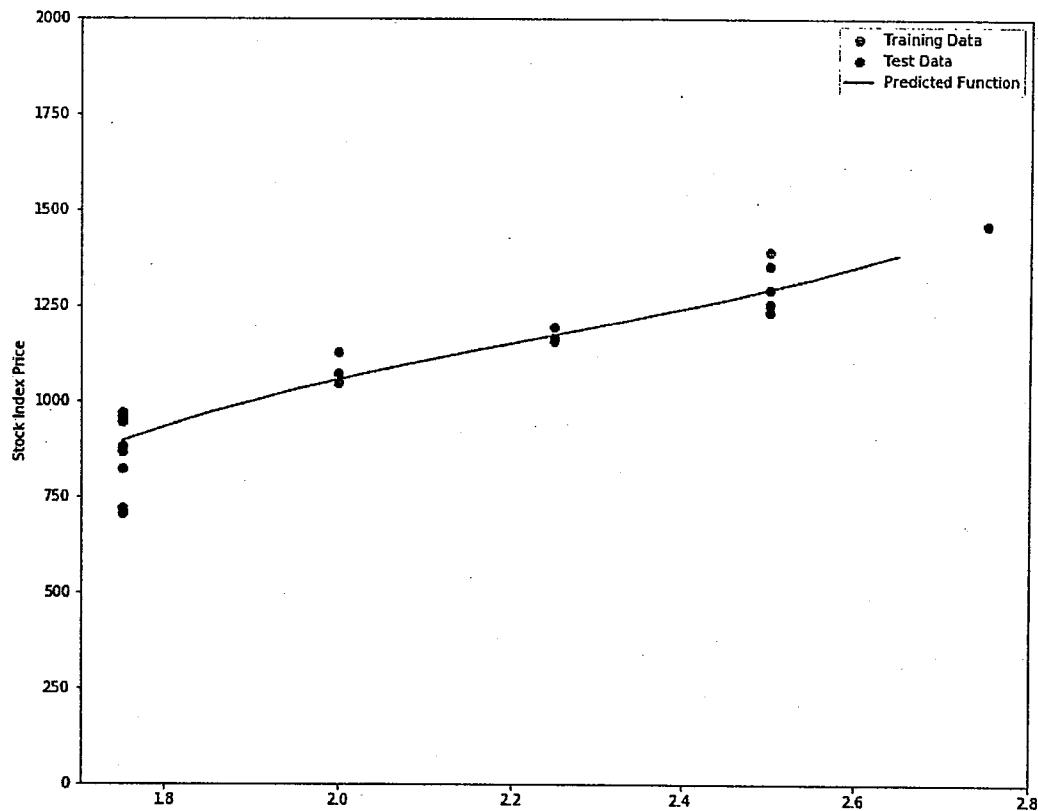
```
In [47]: plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
Title = 'Training Data: Actual vs Predicted'
DistributionPlot(y_train, yhat_train_pr, "Actual Values (Train)", "Predicted Value"
plt.subplot(1,2,2)
Title = 'Testing Data: Actual vs Predicted'
DistributionPlot(y_test, yhat_test_pr, "Actual Values (Test)", "Predicted Values ("
plt.show()
```



```
In [48]: # Quan sát hình ta thấy training dataset fit hơn testing dataset.
# Tuy nhiên, kết quả với Polynominal ổn hơn ở testing dataset so với Linear Regre
```



In [49]: `PollyPlot(X_train, X_test, y_train, y_test, poly, pr)`



In [50]: `#  $R^2$  of the training data:  
poly.score(X_train_pr, y_train)`

Out[50]: `0.9599539129896563`

In [51]: `#  $R^2$  of the test data:  
poly.score(X_test_pr, y_test)`

Out[51]: `0.7728352484103491`

In [52]: `#  $R^2$  of the dataset:  
X_pr = pr.fit_transform(X)  
poly.score(X_pr, y)`

Out[52]: `0.8812772617796455`

In [53]: `# Với cả dataset: có thể nói rằng ~ 88.12% variation của Stock Index Price  
# được giải thích bằng polinomial linear model này.`

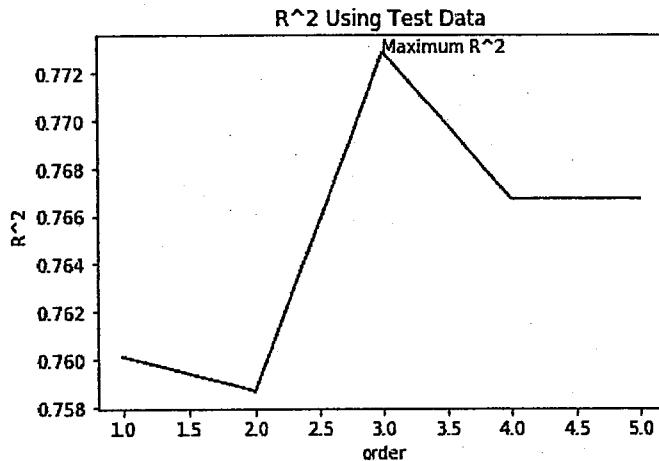


```
In [54]: Rsqu_test = []
order = [1, 2, 3, 4, 5]
lst = [0,0]
for n in order:
    pr = PolynomialFeatures(degree=n)
    x_train_pr = pr.fit_transform(X_train)
    x_test_pr = pr.fit_transform(X_test)
    lre.fit(x_train_pr, y_train)
    score = lre.score(x_test_pr, y_test)
    Rsqu_test.append(score)
    if score > lst[1]:
        lst = [n,score]
print(n, lre.score(x_test_pr, y_test))

plt.plot(order, Rsqu_test)
plt.xlabel('order')
plt.ylabel('R^2')
plt.title('R^2 Using Test Data')
plt.text(lst[0], lst[1], 'Maximum R^2 ')
```

1 0.7601053409693143  
 2 0.7586611419106483  
 3 0.7728352484103491  
 4 0.7667065886474286  
 5 0.7667065886470983

Out[54]: Text(3,0.772835,'Maximum R^2 ')



In [55]: # Như vậy với order = 3 thì R^2 Lớn nhất => chọn mô hình Linear Regression với pol

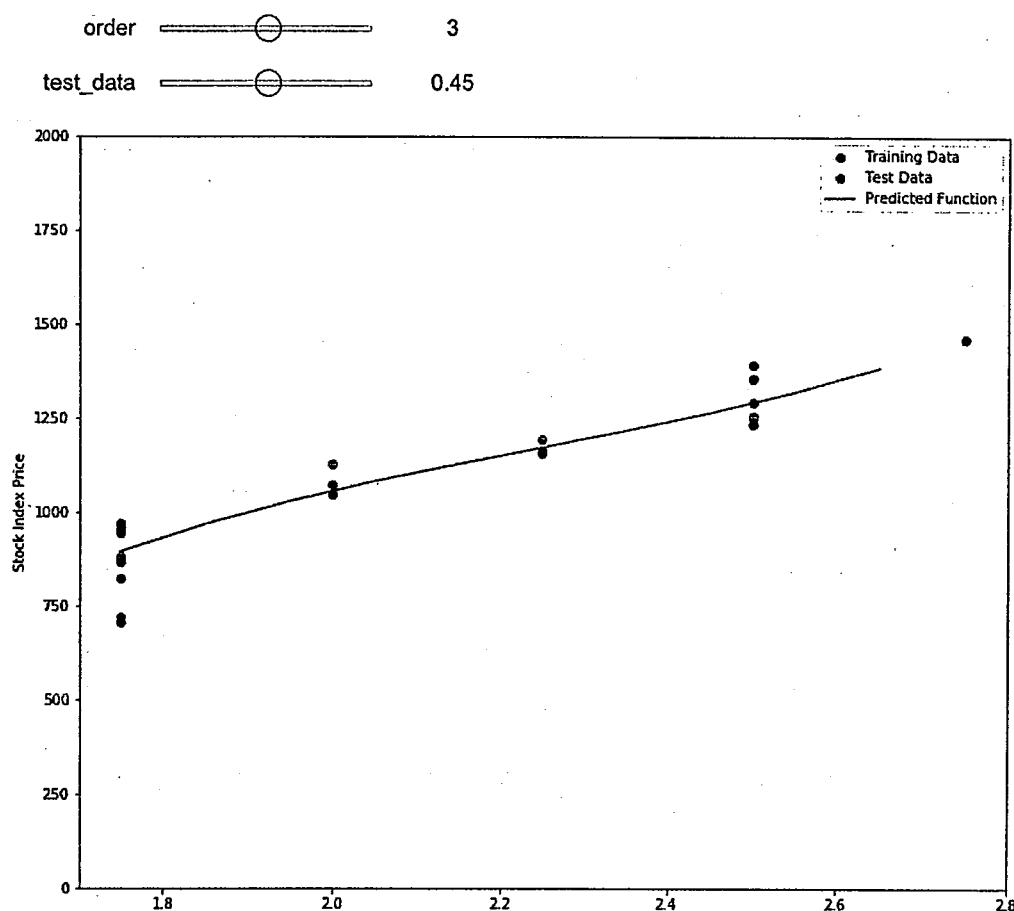
In [56]: # # Kiểm tra kết quả với các tham số khác nhau



```
In [57]: def f(order, test_data):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_data,
                                                       random_state=0)
    pr = PolynomialFeatures(degree=order)
    X_train_pr = pr.fit_transform(X_train)
    X_test_pr = pr.fit_transform(X_test)
    poly = LinearRegression()
    poly.fit(X_train_pr, y_train)
    PollyPlot(X_train, X_test, y_train, y_test, poly, pr)
```

```
In [58]: from ipywidgets import interact, interactive, fixed, interact_manual
```

```
In [59]: interact(f, order=(0, 6, 1), test_data=(0.05, 0.95, 0.1))
```



```
Out[59]: <function __main__.f(order, test_data)>
```

```
In [60]: # Nhận xét: Lựa chọn mô hình
```

## Multiple Variables

Phần này các bạn suy nghĩ và thực hiện nhé.





Trường ĐH Khoa Học Tự Nhiên TP: Hồ Chí Minh  
TRUNG TÂM TIN HỌC

# DATA PRE-PROCESSING AND ANALYSIS

## Bài 9: *Logistic Regression & Fraud Detection*

Phòng LT & Mạng

[https://csc.edu.vn/lap-trinh-va-cSDL/Data-Pre-processing-and-Analysis\\_196](https://csc.edu.vn/lap-trinh-va-cSDL/Data-Pre-processing-and-Analysis_196)

2019



## Nội dung

### 1. Logistic Regression

### 2. Fraud detection





## Logistic Regression

### □ Giới thiệu

- Logistic Regression là một thuật toán thuộc nhóm Supervised Learning sử dụng rất hiệu quả cho classification.
- Logistic Regression (còn gọi là Logit Regression) thường được sử dụng để ước tính xác suất mà một mẫu thuộc về một lớp cụ thể (ví dụ: xác suất một email có phải là spam hay không?, xác suất một giao dịch có phải là gian lận hay không?...)



## Logistic Regression

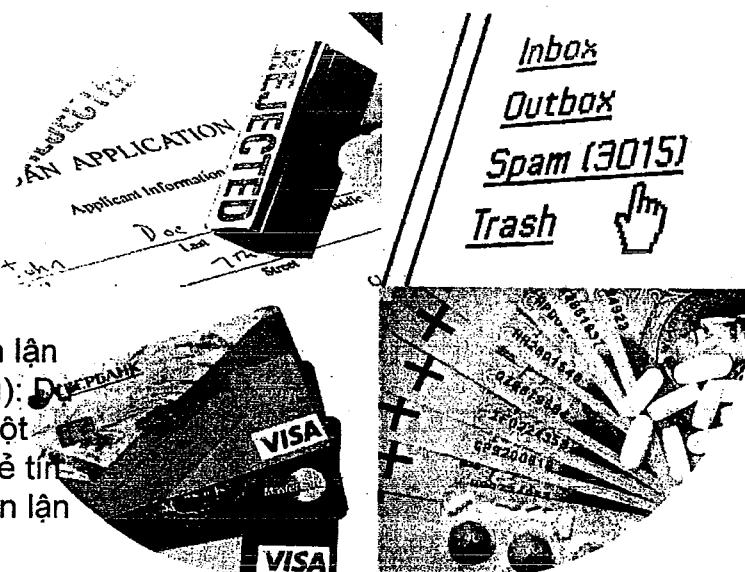
- Nếu xác suất ước tính lớn hơn 50%, thì mô hình dự đoán mẫu thuộc về lớp đó (được gọi là lớp positive, có nhãn "1"), hoặc ngược lại dự đoán rằng nó không thuộc về lớp đó (được gọi là lớp negative, có nhãn "0") => tạo ra một phân loại nhị phân, Binary Classifier.
- Binary Classifier (phân loại nhị phân): outcome chỉ có 1 và 0 hay đúng và sai dù trong tên có Regression, có thể gọi là Conditional Class probabilities.



## Logistic Regression



Credit Card Fraud (Gian lận thẻ tín dụng): Dự đoán liệu một giao dịch thẻ tín dụng có gian lận hay không?



Spam Detection :  
dự đoán 1 email có là Spam hay không?

Data Pre-processing & Analysis

5

## Logistic Regression



### ❑ Các ứng dụng

- Health (Y tế): Dự đoán một khái mô lành tính hoặc ác tính?
- Marketing (Tiếp thị): Dự đoán liệu một người dùng cụ thể có mua sản phẩm bảo hiểm hay không?
- Banking (Ngân hàng): Dự đoán liệu khách hàng sẽ mặc định cho một khoản vay không?



Data Pre-processing & Analysis

6



## Logistic Regression

### ❑ Lý do không áp dụng Linear Regression với bài toán chỉ có hai outcome

- Khi response variable chỉ có hai giá trị cụ thể, ta mong muốn có một mô hình dự đoán giá trị là 0 hoặc 1 hoặc là một điểm xác suất nằm trong khoảng từ 0 đến 1. Linear Regression không có khả năng này. Do đó, nếu ta sử dụng nó để mô hình binary response variable, mô hình kết quả có thể không hạn chế các giá trị Y được dự đoán trong 0 và 1

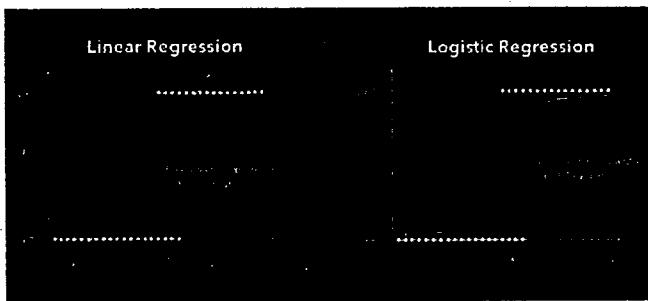


Data Pre-processing & Analysis

7



## Logistic Regression



Y được dự đoán có thể  
vượt quá khoảng 0..1

Y được dự đoán nằm  
trong khoảng 0..1

Trong logistic regression, ta nhận được một probability score, điểm xác suất, phản ánh xác suất xảy ra sự kiện.



Data Pre-processing & Analysis

8

## Logistic Regression



### □ Thuật toán

- Là một mô hình hồi quy, trong đó response variable (biến phụ thuộc) có các giá trị phân loại như TRUE/FALSE hoặc 1/0. Nó đo lường xác suất của một phản ứng nhị phân như giá trị của response variable dựa trên phương trình toán học liên quan tới response variable và các biến predictor variable.



## Logistic Regression



- Phương trình toán học (Sigmoid)

$$S(z) = \frac{1}{1 + e^{-z}}$$

Với  $z = \log(p/(1-p)) = (c + w_1*x_1 + w_2*x_2 + w_3*x_3 + \dots)$

$$\Rightarrow S(z) = p = 1/(1 + e^{-(c + w_1*x_1 + w_2*x_2 + w_3*x_3 + \dots)})$$

- Trong đó:

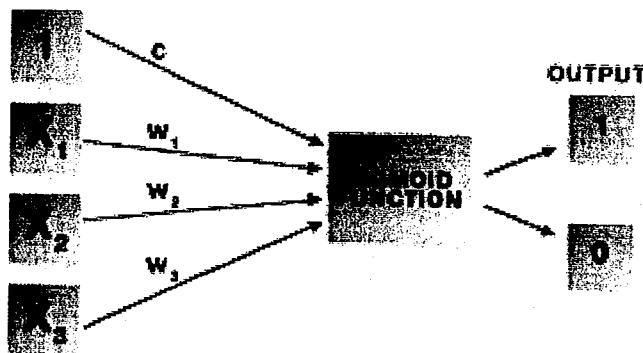
- $S(z) = p = y$ : response variable  $\Rightarrow$  xác suất dự đoán, với  $p \geq 0.5$  thì class = 1, còn  $p < 0.5$  thì class = 0
- $x_1, x_2, \dots, x_n$ : các predictor variable
- $c, w_1, w_2, w_3, \dots, w_n$  là những hằng số được gọi là hệ số (coefficient).



# Logistic Regression



INPUT FEATURES



$$y = \text{logistic}(c + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + \dots + x_n * w_n)$$

$$y = 1 / 1 + e^{[-(c + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + \dots + x_n * w_n)]}$$



Data Pre-processing & Analysis

11

# Logistic Regression



## Ưu điểm

- Không đưa ra giả định về phân phối các lớp trong không gian đặc trưng
- Dễ dàng mở rộng đến nhiều lớp (đa thức hồi quy).
- Huấn luyện nhanh
- Rất nhanh khi phân loại các bản ghi không xác định
- Độ chính xác cao cho nhiều tập dữ liệu đơn giản
- Khả năng phản ứng với overfitting (do bằng mean square error, độ lỗi).
- Có thể giải thích các hệ số mô hình như các chỉ số về tầm quan trọng của tính năng



Data Pre-processing & Analysis



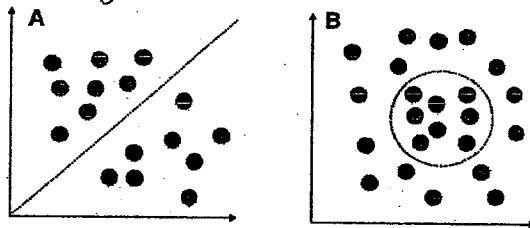
# Logistic Regression



## Khuyết điểm

- Ranh giới quyết định tuyến tính: Mô hình này chỉ phù hợp với loại dữ liệu mà hai class là gần với linearly separable. Một kiểu dữ liệu mà Logistic Regression không làm việc được là khi dữ liệu mà một class chứa các điểm nằm trong 1 vòng tròn, class kia chứa các điểm bên ngoài đường tròn đó.

*Học ở môn machine learning* linear vs. nonlinear problems



13

# Logistic Regression



## Khuyết điểm

- Một hạn chế nữa của Logistic Regression là nó yêu cầu các điểm dữ liệu được tạo ra một cách độc lập với nhau. Trên thực tế, các điểm dữ liệu có thể bị ảnh hưởng bởi nhau.



## Logistic Regression



### ❑ Xây dựng Logistic Regression sử dụng sklearn

- Dùng sklearn.linear\_model.LogisticRegression

- Đây là một mô hình phân lớp hoạt động hiệu quả, dễ triển khai.



## Logistic Regression



### • Các bước thực hiện

- Chọn model sẽ sử dụng là: LogisticRegression
- Tạo một tập dữ liệu feature và một tập target chứa các nhãn cho các thực thể (chỉ có hai loại nhãn)
- Áp dụng mô hình
- Hoàn chỉnh model cho training data
- Sử dụng model hoàn chỉnh (fitted model) cho dữ liệu chưa biết (unseen data)
- Đánh giá độ chính xác



## Logistic Regression



- Ví dụ: áp dụng với bài toán Xác suất vượt qua kỳ thi so với số giờ học

| hours | pass |   | X = data[['hours']] |    | Y = data[['pass']] |   |
|-------|------|---|---------------------|----|--------------------|---|
| 0     | 0.50 | 0 |                     |    |                    |   |
| 1     | 0.75 | 0 |                     |    |                    |   |
| 2     | 1.00 | 0 |                     |    |                    |   |
| 3     | 1.25 | 0 |                     |    |                    |   |
| 4     | 1.50 | 0 | →                   |    |                    |   |
| 5     | 1.75 | 0 |                     |    |                    |   |
| 6     | 1.75 | 1 |                     |    |                    |   |
| 7     | 2.00 | 0 |                     |    |                    |   |
| 8     | 2.25 | 1 |                     |    |                    |   |
| 9     | 2.50 | 0 |                     |    |                    |   |
| 10    | 2.75 | 1 |                     |    |                    |   |
| 11    | 3.00 | 0 |                     |    |                    |   |
| 12    | 3.25 | 1 |                     |    |                    |   |
| 13    | 3.50 | 0 |                     |    |                    |   |
| 14    | 4.00 | 1 |                     |    |                    |   |
| 15    | 4.25 | 1 |                     |    |                    |   |
| 16    | 4.50 | 1 |                     |    |                    |   |
| 17    | 4.75 | 1 |                     |    |                    |   |
| 18    | 5.00 | 1 |                     |    |                    |   |
| 19    | 5.50 | 1 | Data Pre-pr.        | 19 | 5.50               | 1 |

17

## Logistic Regression



```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)

from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()

clf.fit(X_train,Y_train)

print('score Scikit learn: ', clf.score(X_test,Y_test))

score Scikit learn:  0.6666666666666666

Yhat_train = clf.predict(X_train)

Yhat_test = clf.predict(X_test)
Yhat_test
array([1, 1, 1, 1, 1, 1], dtype=int64)

from sklearn.metrics import accuracy_score
print("Train Accuracy is ", accuracy_score(Y_train,Yhat_train)*100,"%")

Train Accuracy is 78.57142857142857 % gồm bốn dữ liệu 1 (4 mẫu test) → cần thu hẹp

print("Test Accuracy is ", accuracy_score(Y_test,Yhat_test)*100,"%") thêm Data. Tuy nhiên
hiện tại chưa bị overfitting.

Test Accuracy is 66.66666666666666 %
```

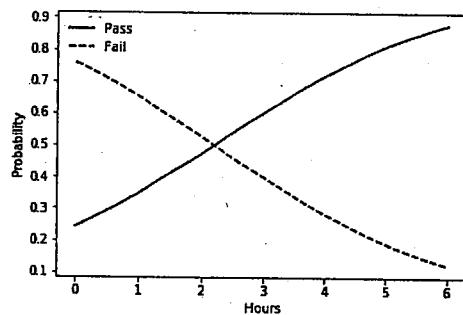
# Logistic Regression



```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

X_new = np.linspace(0, 6, 1000).reshape(-1, 1) # hours
y_proba = clf.predict_proba(X_new)
y_proba.shape Xác suất
(1000, 2)

plt.plot(X_new, y_proba[:, 1], "g-", label="Pass")
plt.plot(X_new, y_proba[:, 0], "b--", label="Fail")
plt.legend()
plt.xlabel("Hours")
plt.ylabel("Probability")
# + more Matplotlib code to make the image look pretty
Text(0,0.5,'Probability')
```



Fail Pass  
[0.6 | 0.4] → fail  
1

sis

19

## Nội dung



### 1. Logistic Regression

### 2. Fraud detection *Nhận diện gian lận.*



## Fraud detection



### ❑ Gian lận – Fraud là gì?

- Gian lận là một vấn đề lớn đối với nhiều doanh nghiệp và có thể thuộc nhiều loại khác nhau: gian lận bảo hiểm, gian lận thẻ tín dụng, xác định hành vi trộm cắp, rửa tiền, trốn thuế, gian lận y tế, giao dịch gian lận cả ngoại tuyến và trực tuyến... Những vấn đề này ảnh hưởng đến các công ty thuộc mọi quy mô trong nhiều ngành công nghiệp. Nạn nhân có thể là các tổ chức phát hành thẻ tín dụng, công ty bảo hiểm, thương nhân bán lẻ, nhà sản xuất, nhà cung cấp từ doanh nghiệp đến doanh nghiệp, nhà cung cấp dịch vụ...
- Một mô hình dự đoán có thể giúp loại bỏ các tiêu cực và giảm thiểu rủi ro gian lận trong kinh doanh.



## Fraud detection



### ● Gian lận

- Không phổ biến
- Được che giấu
- Thay đổi theo thời gian
- Có tổ chức





## Fraud detection

### ❑ Phát hiện gian lận – Fraud detection

- Là một thách thức

888  
888  
888  
888  
888  
888  
888  
888  
888

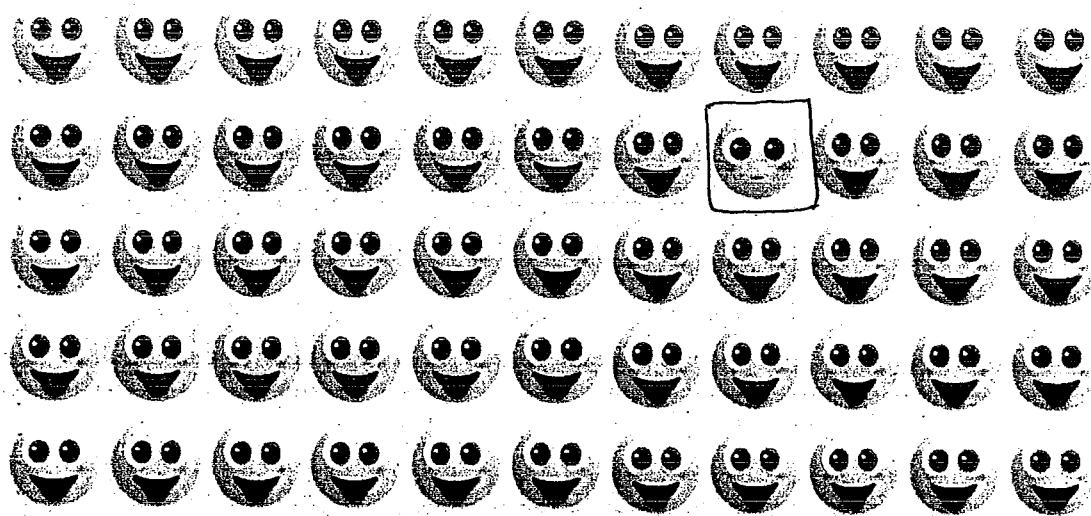


## Fraud detection

555  
555  
555  
555  
555  
555  
555  
555



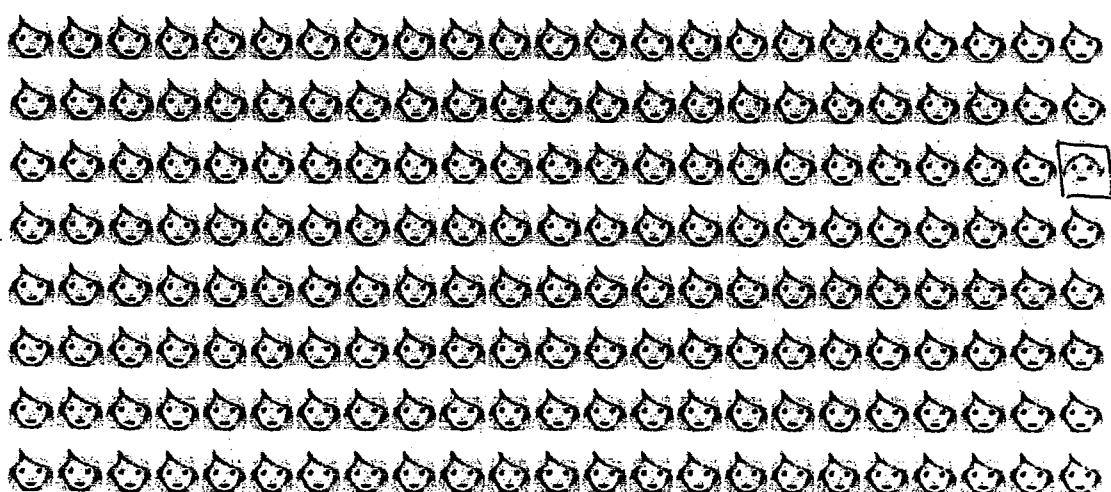
## Fraud detection



Data Pre-processing & Analysis

25

## Fraud detection



Data Pre-processing & Analysis

26

## Fraud detection



### ❑ Làm thế nào để công ty đối phó với gian lận?

Nhóm phân tích gian lận:

- Thường sử dụng các hệ thống dựa trên quy tắc, dựa trên ngưỡng được đặt thủ công và dựa trên kinh nghiệm
- Kiểm tra tin tức
- Nhận danh sách bên ngoài có chứa các tài khoản và tên của kẻ lừa đảo
- Đôi khi sử dụng thuật toán machine learning để phát hiện hành vi gian lận hoặc đáng ngờ



Data Pre-processing & Analysis

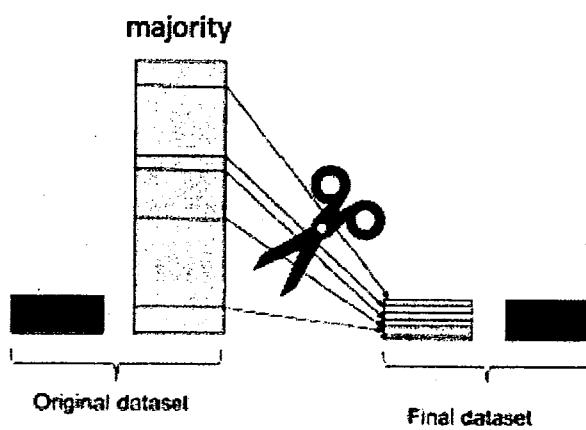
27.

## Fraud detection algorithm



### ❑ Tăng khả năng phát hiện thành công bằng cách sử dụng thay đổi dữ liệu

- Udersampling



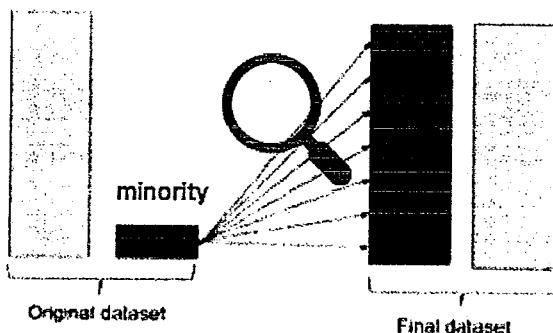
Data Pre-processing & Analysis

28.

## Fraud detection algorithm



### • Oversampling



## Fraud detection algorithm



### □ Sử dụng resampling method nào?

- Random Under Sampling (RUS): bỏ bớt dữ liệu, tính toán hiệu quả
- Random Over Sampling (ROS): đơn giản nhưng huấn luyện mô hình trên dữ liệu có nhiều bản sao giống nhau
- Synthetic Minority Oversampling Technique (SMOTE): bộ dữ liệu thực tế và tinh vi hơn, nhưng ta đang huấn luyện trên "fake" data



# Fraud detection algorithm



## ☐ Khi nào sử dụng resampling method?

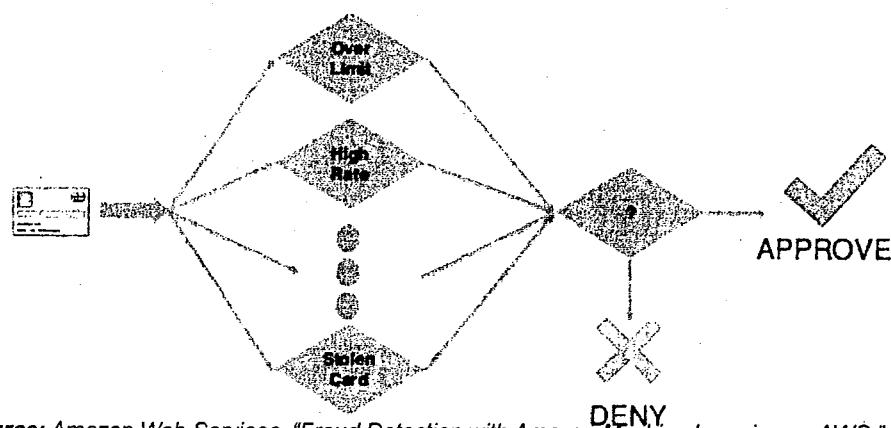
- Chỉ sử dụng trên training set, không sử dụng trên test set



## Fraud detection algorithm



# □ Phát hiện gian lận truyền thông với các hệ thống dựa trên quy tắc Rule-Based Fraud Detection



三

**DENY**  
Source: Amazon Web Services. "Fraud Detection with Amazon Machine Learning on AWS." 22 Sept. 2017. Reading.



## Fraud detection algorithm

### ❑ Hạn chế của việc sử dụng các hệ thống dựa trên quy tắc

- Các ngưỡng thresholds được sửa cho mỗi quy tắc để xác định gian lận
- Giới hạn outcomes yes/no
- Không thể tương tác giữa các tính năng



## Fraud detection algorithm

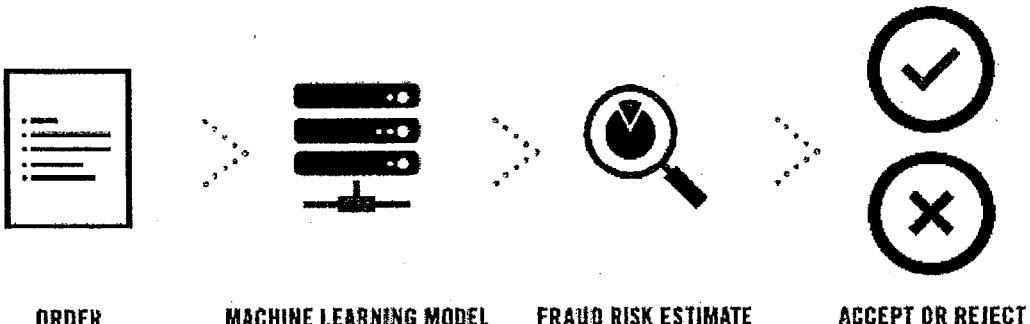


### ❑ Tại sao sử dụng machine learning để xác hiện gian lận

- Machine learning model thích ứng với dữ liệu do đó có thể thay đổi theo thời gian
- Sử dụng tất cả dữ liệu được kết hợp thay vì ngưỡng cho mỗi tính năng
- Có thể cho biết score, thay vì chỉ có yes/no
- Thông thường sẽ có hiệu suất tốt hơn và có thể được kết hợp với các quy tắc



## Fraud detection algorithm



Data Pre-processing & Analysis

35

## Fraud detection sử dụng dữ liệu gán nhãn



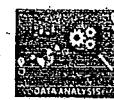
### Classification method trong phát hiện gian lận

- Mục tiêu của phân loại (classification): Sử dụng các trường hợp gian lận đã biết để đào tạo một mô hình giúp phát hiện ra các trường hợp gian lận mới.
- Ví dụ:
  - Email Spam/Not spam
  - Giao dịch trực tuyến gian lận? Yes/No
  - Khối u lành tính/ác tính?



Data Pre-processing & Analysis

36



## Fraud detection sử dụng dữ liệu gán nhãn

- Biên phái dự đoán:  $y \in \{0, 1\}$ 
  - 0: Negative class ("majority" normal cases, trường hợp "đa số" bình thường)
  - 1: Positive class ("minority" fraud cases, trường hợp "thiểu số" gian lận )



## Fraud detection sử dụng dữ liệu gán nhãn

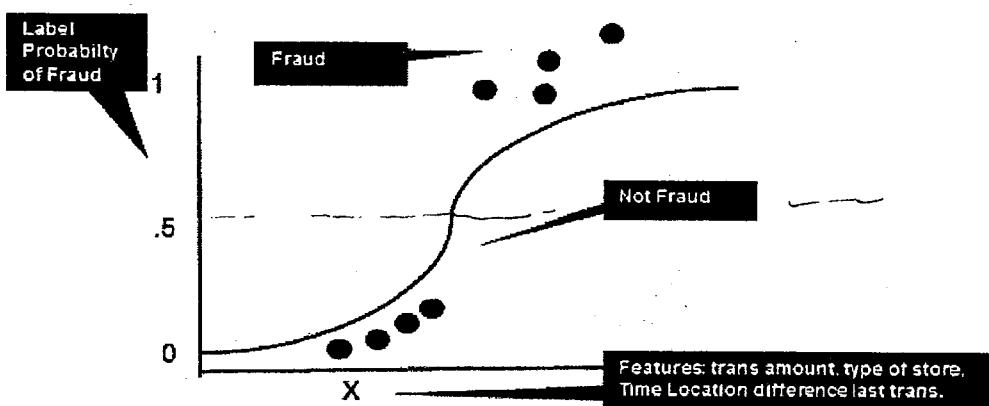
- Classification method thường được dùng để phát hiện gian lận
  - Logistic Regression (*full scale dữ liệu*)
  - Decision Trees và Random Forests (học ở môn Machine Learning) (*lõi cũn scale dữ liệu*)
  - Neuron network (học ở môn Deep Learning) (*full scale dữ liệu*).



## Fraud detection sử dụng dữ liệu gán nhãn



### ❑ Logistic Regression

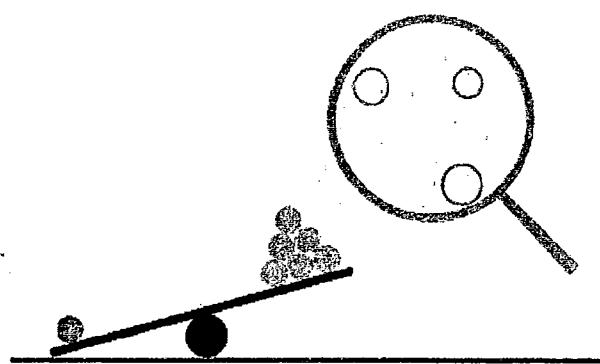


## Fraud detection sử dụng dữ liệu gán nhãn



### ❑ Đo lường hiệu suất phát hiện gian lận

- Độ chính xác (accuracy) không phải là tất cả, nhất là khi làm việc với các vấn đề phát hiện gian lận.



## Fraud detection sử dụng dữ liệu gán nhãn

- ☐ False positives, false negatives & gian lận thực tế bị phát hiện

|      |       | Reality                   |                          |
|------|-------|---------------------------|--------------------------|
|      |       | True                      | False                    |
| True | True  | Correct<br>😊              | Type I<br>False Positive |
|      | False | Type II<br>False Negative | Correct<br>😊             |

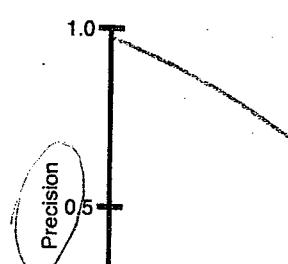


Data Pre-processing & Analysis

41

## Fraud detection sử dụng dữ liệu gán nhãn

- ☐ Precision Recall



$$\text{Precision} = \frac{\# \text{True Positives}}{\# \text{True Positives} + \# \text{False Positives}}$$

$$\text{Recall} = \frac{\# \text{True Positives}}{\# \text{True Positives} + \# \text{False Negatives}}$$

$$F\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$= \frac{2 \times TP}{2 \times TP + FP + FN}$$



Data Pre-processing & Analysis

42

## Fraud detection sử dụng dữ liệu gán nhãn



### ❑ Ví dụ: Phát hiện gian lận thẻ tín dụng

#### ● Bước 1: Đọc dữ liệu, kiểm tra sơ bộ dữ liệu

```
df = pd.read_csv("chapter_1/creditcard_sampledata_3.csv", index_col=0)
```

```
# Explore the features available in your dataframe  
df.head(3)
```

|        | V1       | V2        | V3        | V4        | V5        | V6        | V7        | V8        | V9        | V10       | ... |
|--------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----|
| 258647 | 1.725265 | -1.337256 | -1.012687 | -0.361656 | -1.431611 | -1.098681 | -0.842274 | -0.026594 | -0.032409 | 0.215113  | ... |
| 69263  | 0.683254 | -1.681875 | 0.533349  | -0.326064 | -1.455603 | 0.101832  | -0.520590 | 0.114036  | -0.601760 | 0.444011  | ... |
| 96552  | 1.067973 | -0.656667 | 1.029738  | 0.253899  | -1.172715 | 0.073232  | -0.745771 | 0.249803  | 1.383057  | -0.483771 | ... |

3 rows × 30 columns

```
1. Check the data types of each column.
```

```
df.columns
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11',  
       'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21',  
       'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount', 'Class'],  
      dtype='object')
```

còn lại là thuộc tính (x)  
có gian lận/không gian lận (y)

Data Pre-processing & Analysis

43

## Fraud detection sử dụng dữ liệu gán nhãn



```
| # Count the occurrences of fraud and no fraud and print them
```

```
| occ = df['Class'].value_counts()  
| print(occ)
```

```
0    5000  
1     50  
Name: Class, dtype: int64
```

```
# Print the ratio of fraud cases  
print(occ / len(df.index))
```

```
0    0.990099  
1    0.009901  
Name: Class, dtype: float64
```

```
# X: input, y: output  
X = df.drop(['Class'], axis=1)  
y = df['Class']
```

Data Pre-processing & Analysis

44



## Fraud detection sử dụng dữ liệu gán nhãn

### • Bước 2: Áp dụng model Logistic Regression với dữ liệu gốc, nhận xét kết quả

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=0)
model_o = LogisticRegression()
model_o.fit(X_train, y_train)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn',
tol=0.0001, verbose=0, warm_start=False)

# training score
model_o.score(X_train, y_train)
0.9987623762376238

# testing score
model_o.score(X_test, y_test)
0.997029702970297
```



## Fraud detection sử dụng dữ liệu gán nhãn

```
y_o_pred = model_o.predict(X_test)

accuracy_score(y_test, y_o_pred)
0.997029702970297

conf_mat0 = confusion_matrix(y_true=y_test, y_pred=y_o_pred)
print('Confusion matrix:\n', conf_mat0)

Confusion matrix:
[[1003  1]
 [ 2  4]] → 6 nhầm → trật đ minh phu la doi {mãnh chua cat

# Print classification report using predictions
print(classification_report(y_test, y_o_pred))

precision    recall   f1-score   support
          0       1.00      1.00      1.00      1004
          1       0.80      0.67      0.73       6

   micro avg       1.00      1.00      1.00      1010
   macro avg       0.90      0.83      0.86      1010
 weighted avg     1.00      1.00      1.00      1010
```





## Fraud detection sử dụng dữ liệu gán nhãn

- Bước 3: Áp dụng model Logistic Regression với dữ liệu resampling, nhận xét kết quả

```
# resample
from imblearn.over_sampling import SMOTE
method = SMOTE(kind='borderline1')

# Apply resampling to the training data only
X_resampled, y_resampled = method.fit_sample(X_train, y_train)

# Count the occurrences of fraud and no fraud and print them
occ_no = y_resampled[y_resampled==0].size
print(occ_no)
3996 dữ liệu mua hàng giả

occ_fraud = y_resampled[y_resampled==1].size
print(occ_fraud)
```



Data Pre-processing & Analysis

47



## Fraud detection sử dụng dữ liệu gán nhãn

```
model = LogisticRegression()
model.fit(X_resampled, y_resampled)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn',
tol=0.0001, verbose=0, warm_start=False)

# training score
model.score(X_resampled, y_resampled)
0.9981231231231231 OK

# testing score
model.score(X_test, y_test)
0.998019801980198 OK
```



Data Pre-processing & Analysis

48



## Fraud detection sử dụng dữ liệu gán nhãn

```
# Get your performance metrics
y_pred = model.predict(X_test)

accuracy_score(y_test, y_pred)
0.998019801980198

conf_mat = confusion_matrix(y_true=y_test, y_pred=y_pred)
print('Confusion matrix:\n', conf_mat)

Confusion matrix:
[[1002    2]
 [   0    6]] → mô hình kết luận bao bì là an (không mâu).

# Print classification report using predictions
print(classification_report(y_test, y_pred))

precision    recall    f1-score   support
          0       1.00      1.00      1.00      1004
          1       0.75      1.00      0.86        6

   micro avg       1.00      1.00      1.00      1010
   macro avg       0.88      1.00      0.93      1010
weighted avg       1.00      1.00      1.00      1010
```



Data Pre-processing & Analysis

49



## Fraud detection sử dụng dữ liệu gán nhãn

```
# Calculate average precision and the PR curve
average_precision = average_precision_score(y_test, y_pred)
average_precision
0.75

# Print ROC_AUC score using probabilities
print(roc_auc_score(y_test, probs[:, 1] ) )

0.9991699867197875
```



Data Pre-processing & Analysis

50



DATA ANALYSIS



## Data Pre-processing & Analysis

51



## Chapter 9

### Exercise 1: Admit to university?

Xem xét việc có được vào trường đại học hay không dựa trên bộ dữ liệu sinh viên 400 mẫu có tên là binary.csv

Yêu cầu: Hãy đọc dữ liệu từ tập tin này, áp dụng Logistic Regression để thực hiện việc xác định có được vào trường đại học hay không dựa vào các thông tin như: gre, gpa, rank.

1. Đọc dữ liệu, tiền xử lý dữ liệu nếu cần, trực quan hóa dữ liệu để thấy sự tương quan giữa các biến
2. Tạo X\_train, X\_test, y\_train, y\_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.2
3. Áp dụng thuật toán Logistic Regression
4. Kiểm tra độ chính xác. Đánh giá mô hình bằng kiểm tra underfitting và overfitting
5. Tìm kết quả Cho dữ liệu Test: X\_now = [[600, 4, 2],[400, 3, 3]]

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import math
```

```
In [2]: data = pd.read_csv("binary.csv")
```

```
In [3]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
admit    400 non-null int64
gre      400 non-null int64
gpa      400 non-null float64
rank     400 non-null int64
dtypes: float64(1), int64(3)
memory usage: 12.6 KB
```

7/25/2019

ex1\_university



In [4]: `data.describe()`

Out[4]:

|       | admit      | gre        | gpa        | rank       |
|-------|------------|------------|------------|------------|
| count | 400.000000 | 400.000000 | 400.000000 | 400.000000 |
| mean  | 0.317500   | 587.700000 | 3.389900   | 2.485000   |
| std   | 0.466087   | 115.516536 | 0.380567   | 0.94446    |
| min   | 0.000000   | 220.000000 | 2.260000   | 1.000000   |
| 25%   | 0.000000   | 520.000000 | 3.130000   | 2.000000   |
| 50%   | 0.000000   | 580.000000 | 3.395000   | 2.000000   |
| 75%   | 1.000000   | 660.000000 | 3.670000   | 3.000000   |
| max   | 1.000000   | 800.000000 | 4.000000   | 4.000000   |

In [5]: `data.head()`

Out[5]:

|   | admit | gre | gpa  | rank | chiasclae  |
|---|-------|-----|------|------|--|
| 0 | 0     | 380 | 3.61 | 3    | - Hiện tại chưa có dữ liệu, chia ra 3 lớp ngoài kia. |
| 1 | 1     | 660 | 3.67 | 3    |  |
| 2 | 1     | 800 | 4.00 | 1    |  |
| 3 | 1     | 640 | 3.19 | 4    |  |
| 4 | 0     | 520 | 2.93 | 4    |  |

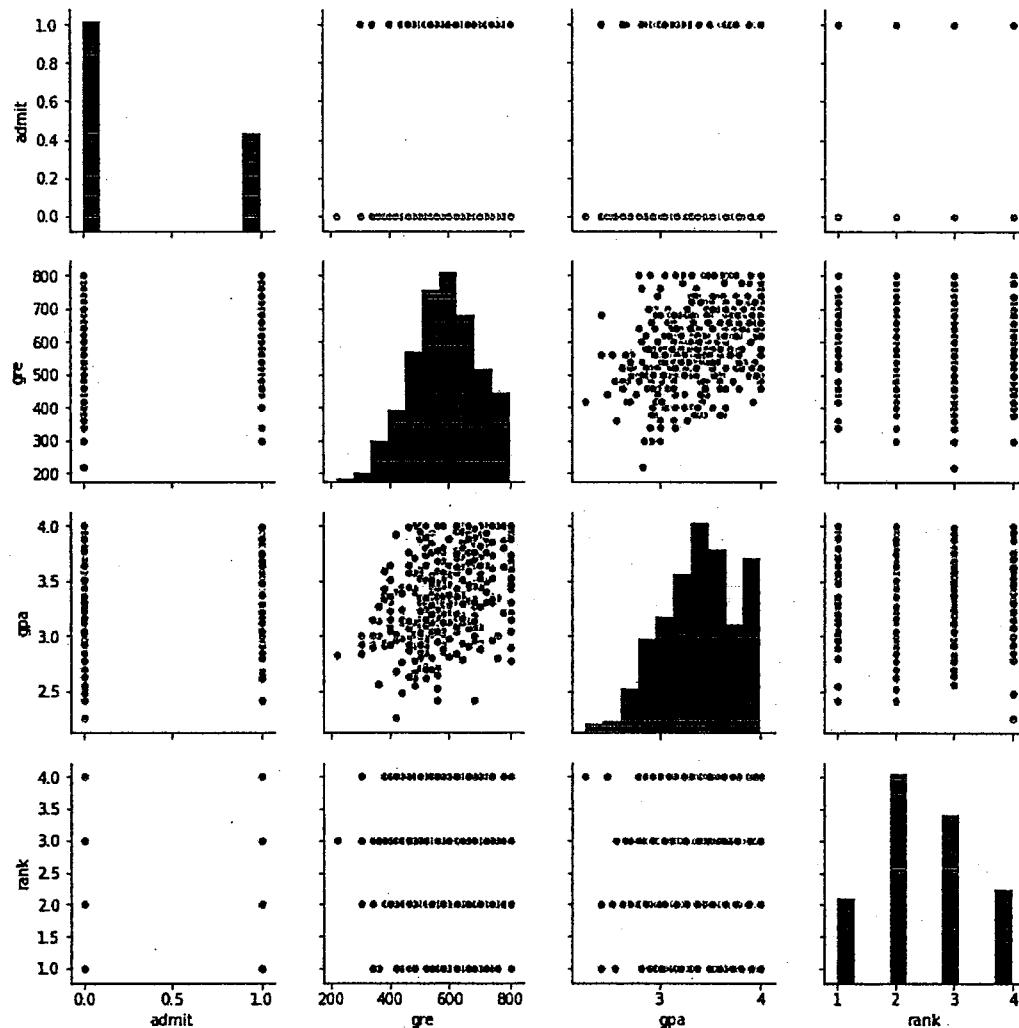
In [6]: `import seaborn as sns`

7/25/2019

ex1\_university



In [7]: `sns.pairplot(data)  
plt.show()`



In [8]: `X = data[['gre', 'gpa', 'rank']]  
X.head()`

Out[8]:

|   | gre | gpa  | rank |
|---|-----|------|------|
| 0 | 380 | 3.61 | 3    |
| 1 | 660 | 3.67 | 3    |
| 2 | 800 | 4.00 | 1    |
| 3 | 640 | 3.19 | 4    |
| 4 | 520 | 2.93 | 4    |



```
In [9]: Y = data[['admit']]
Y.head()
```

Out[9]:

|   | admit |
|---|-------|
| 0 | 0     |
| 1 | 1     |
| 2 | 1     |
| 3 | 1     |
| 4 | 0     |

```
In [10]: type(X)
```

Out[10]: pandas.core.frame.DataFrame

```
In [11]: X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)
```

```
In [12]: from sklearn.linear_model import LogisticRegression
```

```
In [13]: clf = LogisticRegression()
```

```
In [14]: from sklearn.utils.validation import column_or_1d
```

```
In [15]: clf.fit(X_train, column_or_1d(Y_train))
```

```
c:\program files\python36\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
FutureWarning)
```

```
Out[15]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                           intercept_scaling=1, max_iter=100, multi_class='warn',
                           n_jobs=None, penalty='l2', random_state=None, solver='warn',
                           tol=0.0001, verbose=0, warm_start=False)
```

```
In [16]: clf.intercept_
```

Out[16]: array([-1.58141065])

```
In [17]: clf.coef_
```

Out[17]: array([[ 0.00261024, 0.21658372, -0.60483839]])

```
In [18]: print('score Scikit learn: ', clf.score(X_test,Y_test))
```

score Scikit learn: 0.675

```
In [19]: # Kiểm tra overfitting và underfitting
```



```
In [20]: Yhat_train = clf.predict(X_train)
```

```
In [21]: Yhat_test = clf.predict(X_test)  
Yhat_test
```

```
In [22]: from sklearn.metrics import accuracy_score  
print("Train Accuracy is ", accuracy_score(Y_train,Yhat_train)*100,"%")
```

Train Accuracy is 71.25 %

```
In [23]: print("Test Accuracy is ", accuracy_score(Y_test,Yhat_test)*100,"%")
```

Test Accuracy is 67.5 %

In [24]: # Mô hình fit với training data set hơn testing dataset. Độ chính xác của train và  
# Đô chính xác chung của mô hình cao hay không cao?  
# Có giải pháp nào không?

```
In [25]: X_now = [[600, 4, 2],[400, 3, 3]]  
Y_now = clf.predict(X_now)  
Y now
```

```
Out[25]: array([0, 0], dtype=int64)
```



## Exercise 2: Titanic

Xem xét việc một hành khách có sống sót hay không dựa trên bộ dữ liệu titanic (train.csv có 891 mẫu và test.csv có 418 mẫu )

Yêu cầu: Hãy đọc dữ liệu từ các tập tin này, áp dụng Logistic Regression để thực hiện việc xác định một hành khách có sống sót hay không dựa trên những thông tin được cung cấp.

1. Đọc dữ liệu train.csv, tiền xử lý dữ liệu nếu cần
2. Tạo X\_train, X\_test, y\_train, y\_test từ dữ liệu ở câu 1 với tỷ lệ dữ liệu test là 0.2
3. Áp dụng thuật toán Logistic Regression: fit model, tìm độ chính xác, đánh giá mô hình bằng kiểm tra underfitting và overfitting?
4. Đọc dữ liệu test.csv. Tiền xử lý dữ liệu như train.csv. Tìm kết quả cho dữ liệu test.
5. Ghi kết quả vào file test\_pred.csv

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import math
```

In [2]:

```
data = pd.read_csv("titanic/train.csv")
```

In [3]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass          891 non-null int64
Name            891 non-null object
Sex             891 non-null object
Age             714 non-null float64 → Thiếu hơn 100 .
SibSp           891 non-null int64
Parch           891 non-null int64
Ticket          891 non-null object
Fare            891 non-null float64
Cabin           204 non-null object → thiếu n → bỏ
Embarked        889 non-null object → Xem xét việc lên tàu cáng nào có thể đến việc survis
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

In [4]:

```
# Nhận xét: theo như thông tin trên, dữ liệu Age bị thiếu => tiến hành cập nhật cá
# Thông tin Cabin thiếu nhiều thông tin => drop bỏ cột này
# Thông tin Embarked bị thiếu 2 ô => xóa 2 dòng thiếu này
```



In [5]: `data.mean()`

```
Out[5]: PassengerId    446.000000
         Survived      0.383838
         Pclass        2.308642
         Age           29.699118
         SibSp         0.523008
         Parch         0.381594
         Fare          32.204208
         dtype: float64
```

In [6]: `# thay nan bằng mean  
data = data.fillna(data.mean())`

In [7]: `del data['Cabin']`

In [8]: `data = data.dropna()`

In [9]: `data.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 11 columns):
PassengerId    889 non-null int64
Survived       889 non-null int64
Pclass          889 non-null int64
Name            889 non-null object
Sex             889 non-null object
Age             889 non-null float64
SibSp           889 non-null int64
Parch           889 non-null int64
Ticket          889 non-null object
Fare            889 non-null float64
Embarked        889 non-null object
dtypes: float64(2), int64(5), object(4)
memory usage: 83.3+ KB
```

In [10]: `data.describe()`

Out[10]:

|       | PassengerId | Survived   | Pclass     | Age        | SibSp      | Parch      | Fare       |
|-------|-------------|------------|------------|------------|------------|------------|------------|
| count | 889.000000  | 889.000000 | 889.000000 | 889.000000 | 889.000000 | 889.000000 | 889.000000 |
| mean  | 446.000000  | 0.382452   | 2.311586   | 29.653446  | 0.524184   | 0.382452   | 32.096681  |
| std   | 256.998173  | 0.486260   | 0.834700   | 12.968366  | 1.103705   | 0.806761   | 49.697504  |
| min   | 1.000000    | 0.000000   | 1.000000   | 0.420000   | 0.000000   | 0.000000   | 0.000000   |
| 25%   | 224.000000  | 0.000000   | 2.000000   | 22.000000  | 0.000000   | 0.000000   | 7.895800   |
| 50%   | 446.000000  | 0.000000   | 3.000000   | 29.699118  | 0.000000   | 0.000000   | 14.454200  |
| 75%   | 668.000000  | 1.000000   | 3.000000   | 35.000000  | 1.000000   | 0.000000   | 31.000000  |
| max   | 891.000000  | 1.000000   | 3.000000   | 80.000000  | 8.000000   | 6.000000   | 512.329200 |

7/25/2019

ex2\_titanic

In [11]: `data.head()`

Out[11]:

|   | PassengerId | Survived | Pclass | Name  | Sex    | Age  | SibSp | Parch | Ticket              | Fare    | Emb |
|---|-------------|----------|--------|---|--------|------|-------|-------|---------------------|---------|-----|
| 0 | 1           | 0        | 3      | Braund,<br>Mr. Owen<br>Harris                                     | male   | 22.0 | 1     | 0     | A/5 21171           | 7.2500  |     |
| 1 | 2           | 1        | 1      | Cumings,<br>Mrs. John<br>Bradley<br>(Florence<br>Briggs<br>Th...) | female | 38.0 | 1     | 0     | PC 17599            | 71.2833 |     |
| 2 | 3           | 1        | 3      | Heikkinen,<br>Miss.<br>Laina                                      | female | 26.0 | 0     | 0     | STON/O2.<br>3101282 | 7.9250  |     |
| 3 | 4           | 1        | 1      | Futrelle,<br>Mrs.<br>Jacques<br>Heath<br>(Lily May<br>Peel)       | female | 35.0 | 1     | 0     | 113803              | 53.1000 |     |
| 4 | 5           | 0        | 3      | Allen, Mr.<br>William<br>Henry                                    | male   | 35.0 | 0     | 0     | 373450              | 8.0500  |     |

In [12]: `df = data[['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]`In [13]: `df.head()`

Out[13]:

|   | Survived | Pclass | Sex    | Age  | SibSp | Parch | Fare    | Embarked |
|---|----------|--------|--------|------|-------|-------|---------|----------|
| 0 | 0        | 3      | male   | 22.0 | 1     | 0     | 7.2500  | S        |
| 1 | 1        | 1      | female | 38.0 | 1     | 0     | 71.2833 | C        |
| 2 | 1        | 3      | female | 26.0 | 0     | 0     | 7.9250  | S        |
| 3 | 1        | 1      | female | 35.0 | 1     | 0     | 53.1000 | S        |
| 4 | 0        | 3      | male   | 35.0 | 0     | 0     | 8.0500  | S        |

```
# Categorical boolean mask
categorical_feature_mask = df.dtypes==object
# filter categorical columns using mask and turn it into a list
categorical_cols = df.columns[categorical_feature_mask].tolist()
categorical_cols
```

Out[14]: `['Sex', 'Embarked']`In [15]: `df_now = pd.get_dummies(data=df, columns=categorical_cols, drop_first=True)`

7/25/2019

ex2\_titanic



In [16]: df\_now.head()

Out[16]:

|   | Survived | Pclass | Age  | SibSp | Parch | Fare    | Sex_male | Embarked_Q | Embarked_S |
|---|----------|--------|------|-------|-------|---------|----------|------------|------------|
| 0 | 0        | 3      | 22.0 | 1     | 0     | 7.2500  | 1        | 0          | 1          |
| 1 | 1        | 1      | 38.0 | 1     | 0     | 71.2833 | 0        | 0          | 0          |
| 2 | 1        | 3      | 26.0 | 0     | 0     | 7.9250  | 0        | 0          | 1          |
| 3 | 1        | 1      | 35.0 | 1     | 0     | 53.1000 | 0        | 0          | 1          |
| 4 | 0        | 3      | 35.0 | 0     | 0     | 8.0500  | 1        | 0          | 1          |

In [17]: df\_now.tail()

Out[17]:

|     | Survived | Pclass | Age       | SibSp | Parch | Fare  | Sex_male | Embarked_Q | Embarked_S |
|-----|----------|--------|-----------|-------|-------|-------|----------|------------|------------|
| 886 | 0        | 2      | 27.000000 | 0     | 0     | 13.00 | 1        | 0          | 1          |
| 887 | 1        | 1      | 19.000000 | 0     | 0     | 30.00 | 0        | 0          | 1          |
| 888 | 0        | 3      | 29.699118 | 1     | 2     | 23.45 | 0        | 0          | 1          |
| 889 | 1        | 1      | 26.000000 | 0     | 0     | 30.00 | 1        | 0          | 0          |
| 890 | 0        | 3      | 32.000000 | 0     | 0     | 7.75  | 1        | 1          | 0          |

In [18]: df\_now.isnull().any()

Out[18]:

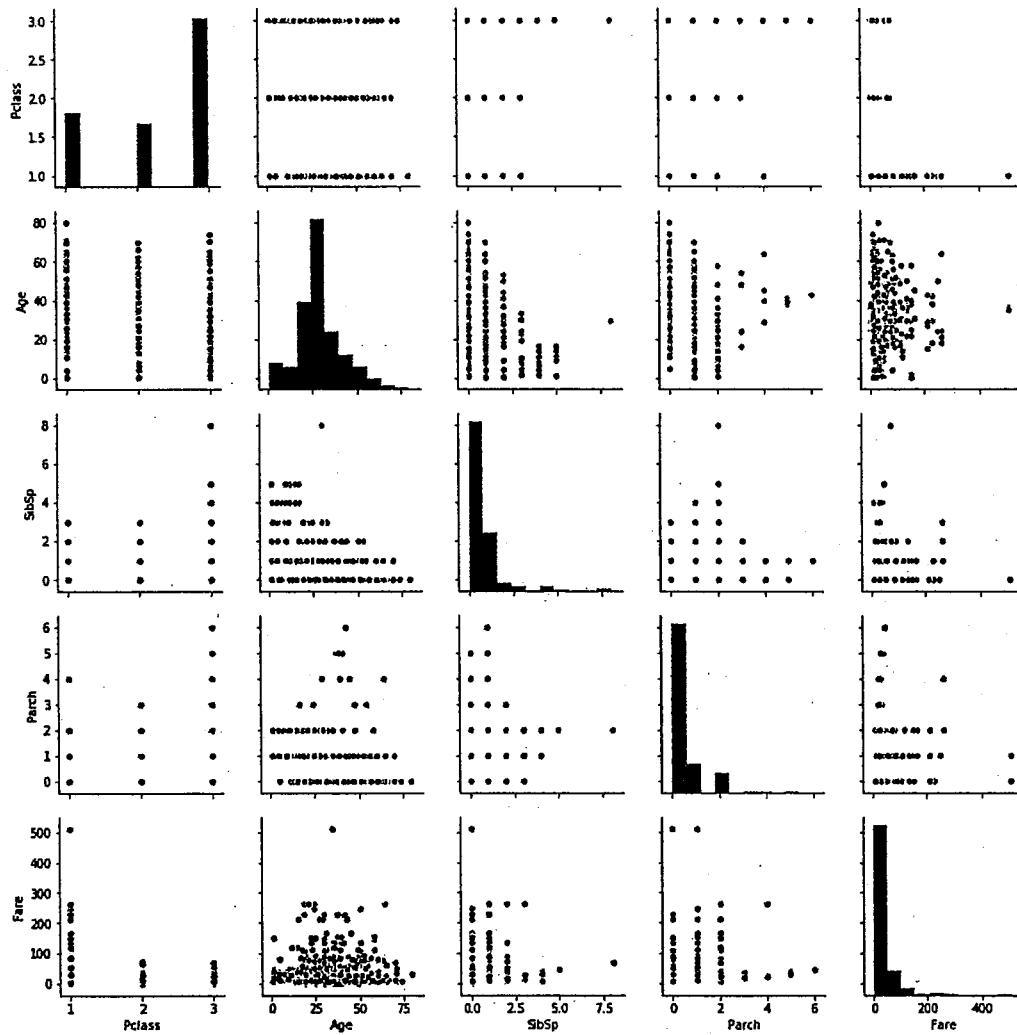
|             |       |
|-------------|-------|
| Survived    | False |
| Pclass      | False |
| Age         | False |
| SibSp       | False |
| Parch       | False |
| Fare        | False |
| Sex_male    | False |
| Embarked_Q  | False |
| Embarked_S  | False |
| dtype: bool |       |

In [19]: import seaborn as sns

7/25/2019

ex2\_titanic

```
In [20]: sns.pairplot(data[["Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked"]])  
plt.show()
```



```
In [21]: X = df_now.drop('Survived', 1)  
X.head()
```

Out[21]:

|   | Pclass | Age  | SibSp | Parch | Fare    | Sex_male | Embarked_Q | Embarked_S |
|---|--------|------|-------|-------|---------|----------|------------|------------|
| 0 | 3      | 22.0 | 1     | 0     | 7.2500  | 1        | 0          | 1          |
| 1 | 1      | 38.0 | 1     | 0     | 71.2833 | 0        | 0          | 0          |
| 2 | 3      | 26.0 | 0     | 0     | 7.9250  | 0        | 0          | 1          |
| 3 | 1      | 35.0 | 1     | 0     | 53.1000 | 0        | 0          | 1          |
| 4 | 3      | 35.0 | 0     | 0     | 8.0500  | 1        | 0          | 1          |



```
In [22]: Y = df_now[['Survived']]
Y.head()
```

Out[22]:

| Survived |   |
|----------|---|
| 0        | 0 |
| 1        | 1 |
| 2        | 1 |
| 3        | 1 |
| 4        | 0 |

## Build & Test model

```
In [23]: X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)
```

```
In [24]: from sklearn.linear_model import LogisticRegression
```

```
In [25]: clf = LogisticRegression()
```

```
In [26]: from sklearn.utils.validation import column_or_1d
```

```
In [27]: clf.fit(X_train, column_or_1d(Y_train))
```

```
c:\program files\python36\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
FutureWarning)
```

```
Out[27]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                           intercept_scaling=1, max_iter=100, multi_class='warn',
                           n_jobs=None, penalty='l2', random_state=None, solver='warn',
                           tol=0.0001, verbose=0, warm_start=False)
```

```
In [28]: clf.intercept_
```

```
Out[28]: array([3.83284351])
```

```
In [29]: clf.coef_
```

```
Out[29]: array([[-0.77660966, -0.02552679, -0.30612226, -0.08730947,  0.00423455,
                 -2.48432325, -0.09490418, -0.2961227 ]])
```

```
In [30]: print('score Scikit learn: ', clf.score(X_test,Y_test))
```

```
score Scikit learn:  0.8258426966292135
```

```
In [31]: Yhat_train = clf.predict(X_train)
```



```
In [32]: Yhat_test = clf.predict(X_test)

In [33]: from sklearn.metrics import accuracy_score

In [34]: print("Train Accuracy is ", accuracy_score(Y_train,Yhat_train)*100,"%")
Train Accuracy is 79.88748241912799 %

In [35]: print("Test Accuracy is ", accuracy_score(Y_test,Yhat_test)*100,"%")
Test Accuracy is 82.58426966292134 %

In [36]: # Mô hình trên có độ chính xác của train và test gần nhau và khoảng 80%: không
```

## Make prediction on Test data

```
In [37]: data_test = pd.read_csv("titanic/test.csv")

In [38]: data_test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass          418 non-null int64
Name            418 non-null object
Sex             418 non-null object
Age             332 non-null float64
SibSp           418 non-null int64
Parch           418 non-null int64
Ticket          418 non-null object
Fare             417 non-null float64
Cabin           91 non-null object
Embarked         418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

```
In [39]: # Nhận xét: theo như thông tin trên, dữ liệu Age bị thiếu => tiến hành cập nhật cá
# Thông tin Cabin thiếu nhiều thông tin => drop bỏ cột này
# Thông tin Fare bị thiếu 1 ô => xóa 1 dòng thiếu này
```

```
In [40]: data_test.mean()
```

```
Out[40]: PassengerId    1100.500000
Pclass          2.265550
Age             30.272590
SibSp           0.447368
Parch           0.392344
Fare            35.627188
dtype: float64
```



```
In [41]: # thay nan bằng mean
data_test = data_test.fillna(data_test.mean())
```

```
In [42]: del data_test['Cabin']
```

```
In [43]: data_test = data_test.dropna()
```

```
In [44]: data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 418 entries, 0 to 417
Data columns (total 10 columns):
PassengerId    418 non-null int64
Pclass          418 non-null int64
Name            418 non-null object
Sex             418 non-null object
Age             418 non-null float64
SibSp           418 non-null int64
Parch           418 non-null int64
Ticket          418 non-null object
Fare            418 non-null float64
Embarked        418 non-null object
dtypes: float64(2), int64(4), object(4)
memory usage: 35.9+ KB
```

```
In [45]: data_test.describe()
```

Out[45]:

|       | PassengerId | Pclass     | Age        | SibSp      | Parch      | Fare       |
|-------|-------------|------------|------------|------------|------------|------------|
| count | 418.000000  | 418.000000 | 418.000000 | 418.000000 | 418.000000 | 418.000000 |
| mean  | 1100.500000 | 2.265550   | 30.272590  | 0.447368   | 0.392344   | 35.627188  |
| std   | 120.810458  | 0.841838   | 12.634534  | 0.896760   | 0.981429   | 55.840500  |
| min   | 892.000000  | 1.000000   | 0.170000   | 0.000000   | 0.000000   | 0.000000   |
| 25%   | 996.250000  | 1.000000   | 23.000000  | 0.000000   | 0.000000   | 7.895800   |
| 50%   | 1100.500000 | 3.000000   | 30.272590  | 0.000000   | 0.000000   | 14.454200  |
| 75%   | 1204.750000 | 3.000000   | 35.750000  | 1.000000   | 0.000000   | 31.500000  |
| max   | 1309.000000 | 3.000000   | 76.000000  | 8.000000   | 9.000000   | 512.329200 |

7/25/2019

ex2\_titanic



In [46]: `data_test.head()`

Out[46]:

|   | PassengerId | Pclass | Name   | Sex    | Age  | SibSp | Parch | Ticket  | Fare    | Embarked |
|---|-------------|--------|--|--------|------|-------|-------|---------|---------|----------|
| 0 | 892         | 3      | Kelly, Mr. James                             | male   | 34.5 | 0     | 0     | 330911  | 7.8292  | Q        |
| 1 | 893         | 3      | Wilkes, Mrs. James (Ellen Needs)             | female | 47.0 | 1     | 0     | 363272  | 7.0000  | S        |
| 2 | 894         | 2      | Myles, Mr. Thomas Francis                    | male   | 62.0 | 0     | 0     | 240276  | 9.6875  | Q        |
| 3 | 895         | 3      | Wirz, Mr. Albert                             | male   | 27.0 | 0     | 0     | 315154  | 8.6625  | S        |
| 4 | 896         | 3      | Hirvonen, Mrs. Alexander (Helga E Lindqvist) | female | 22.0 | 1     | 1     | 3101298 | 12.2875 | S        |

In [47]: `df_test = data_test[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]`

In [48]: `df_test.head()`

Out[48]:

|   | Pclass | Sex    | Age  | SibSp | Parch | Fare    | Embarked |
|---|--------|--------|------|-------|-------|---------|----------|
| 0 | 3      | male   | 34.5 | 0     | 0     | 7.8292  | Q        |
| 1 | 3      | female | 47.0 | 1     | 0     | 7.0000  | S        |
| 2 | 2      | male   | 62.0 | 0     | 0     | 9.6875  | Q        |
| 3 | 3      | male   | 27.0 | 0     | 0     | 8.6625  | S        |
| 4 | 3      | female | 22.0 | 1     | 1     | 12.2875 | S        |

In [49]: `# Categorical boolean mask  
categorical_feature_mask = df_test.dtypes==object  
# filter categorical columns using mask and turn it into a list  
categorical_cols = df_test.columns[categorical_feature_mask].tolist()  
categorical_cols`

Out[49]: `['Sex', 'Embarked']`

In [50]: `df_test_now = pd.get_dummies(data=df_test, columns=categorical_cols, drop_first=True)`

7/25/2019

ex2\_titanic



In [51]: df\_test\_now.head()

Out[51]:

|   | Pclass | Age  | SibSp | Parch | Fare    | Sex_male | Embarked_Q | Embarked_S |
|---|--------|------|-------|-------|---------|----------|------------|------------|
| 0 | 3      | 34.5 | 0     | 0     | 7.8292  | 1        | 1          | 0          |
| 1 | 3      | 47.0 | 1     | 0     | 7.0000  | 0        | 0          | 1          |
| 2 | 2      | 62.0 | 0     | 0     | 9.6875  | 1        | 1          | 0          |
| 3 | 3      | 27.0 | 0     | 0     | 8.6625  | 1        | 0          | 1          |
| 4 | 3      | 22.0 | 1     | 1     | 12.2875 | 0        | 0          | 1          |

In [52]: df\_test\_now.tail()

Out[52]:

|     | Pclass | Age      | SibSp | Parch | Fare     | Sex_male | Embarked_Q | Embarked_S |
|-----|--------|----------|-------|-------|----------|----------|------------|------------|
| 413 | 3      | 30.27259 | 0     | 0     | 8.0500   | 1        | 0          | 1          |
| 414 | 1      | 39.00000 | 0     | 0     | 108.9000 | 0        | 0          | 0          |
| 415 | 3      | 38.50000 | 0     | 0     | 7.2500   | 1        | 0          | 1          |
| 416 | 3      | 30.27259 | 0     | 0     | 8.0500   | 1        | 0          | 1          |
| 417 | 3      | 30.27259 | 1     | 1     | 22.3583  | 1        | 0          | 0          |

In [53]: df\_test\_now.isnull().any()

Out[53]:

```
Pclass      False
Age        False
SibSp      False
Parch      False
Fare       False
Sex_male   False
Embarked_Q False
Embarked_S False
dtype: bool
```

In [54]: X\_test\_now = df\_test\_now  
X\_test\_now.head()

Out[54]:

|   | Pclass | Age  | SibSp | Parch | Fare    | Sex_male | Embarked_Q | Embarked_S |
|---|--------|------|-------|-------|---------|----------|------------|------------|
| 0 | 3      | 34.5 | 0     | 0     | 7.8292  | 1        | 1          | 0          |
| 1 | 3      | 47.0 | 1     | 0     | 7.0000  | 0        | 0          | 1          |
| 2 | 2      | 62.0 | 0     | 0     | 9.6875  | 1        | 1          | 0          |
| 3 | 3      | 27.0 | 0     | 0     | 8.6625  | 1        | 0          | 1          |
| 4 | 3      | 22.0 | 1     | 1     | 12.2875 | 0        | 0          | 1          |

In [55]: Yhat\_test\_now = clf.predict(X\_test\_now)

In [56]: df\_test\_now['Survived'] = Yhat\_test\_now

7/25/2019

ex2\_titanic



In [57]: df\_test\_now.head()

Out[57]:

|   | Pclass | Age  | SibSp | Parch | Fare    | Sex_male | Embarked_Q | Embarked_S | Survived |
|---|--------|------|-------|-------|---------|----------|------------|------------|----------|
| 0 | 3      | 34.5 | 0     | 0     | 7.8292  | 1        | 1          | 0          | 0        |
| 1 | 3      | 47.0 | 1     | 0     | 7.0000  | 0        | 0          | 1          | 0        |
| 2 | 2      | 62.0 | 0     | 0     | 9.6875  | 1        | 1          | 0          | 0        |
| 3 | 3      | 27.0 | 0     | 0     | 8.6625  | 1        | 0          | 1          | 0        |
| 4 | 3      | 22.0 | 1     | 1     | 12.2875 | 0        | 0          | 1          | 1        |

In [58]: data\_test['Survived'] = Yhat\_test\_now

In [59]: data\_test.head()

Out[59]:

|   | PassengerId | Pclass | Name   | Sex    | Age  | SibSp | Parch | Ticket  | Fare    | Embarked | Survived |
|---|-------------|--------|--|--------|------|-------|-------|---------|---------|----------|----------|
| 0 | 892         | 3      | Kelly, Mr. James                             | male   | 34.5 | 0     | 0     | 330911  | 7.8292  | Q        |          |
| 1 | 893         | 3      | Wilkes, Mrs. James (Ellen Needs)             | female | 47.0 | 1     | 0     | 363272  | 7.0000  | S        |          |
| 2 | 894         | 2      | Myles, Mr. Thomas Francis                    | male   | 62.0 | 0     | 0     | 240276  | 9.6875  | Q        |          |
| 3 | 895         | 3      | Wirz, Mr. Albert                             | male   | 27.0 | 0     | 0     | 315154  | 8.6625  | S        |          |
| 4 | 896         | 3      | Hirvonen, Mrs. Alexander (Helga E Lindqvist) | female | 22.0 | 1     | 1     | 3101298 | 12.2875 | S        |          |

In [60]: data\_test.to\_csv('titanic/test\_pred.csv')

Thi: Đặt tên folder theo tên họ và tên  
Nội file: → gửi mail theo deadline  
Subject: (Reply mail cũ có gửi). hoặc ghi LDS5 - Họ tên

Nội dung để thi: