



Exploratory Data Analysis (EDA) with Python

Parisut Jitpakdee, Ph.D
Senior Data scientist
Big Data Institute (BDI)

Overview

- **Learning Outcome**
 - Understand the basic concepts of exploratory data analysis
 - Understand the role of statistics in data exploration
 - Choose appropriate data analysis techniques to explore and analyze data
- **Agenda**
 - Basic concepts of exploratory data analysis (EDA)
 - EDA techniques with Python

ข้อมูล (Data) - 1

- Attribute หมายถึง คุณลักษณะของข้อมูล ยกตัวอย่างเช่น
 - เพศ
 - ที่อยู่
 - ความสูง
- ในงานวิทยาศาสตร์ข้อมูล เรียก Attribute ว่า Feature
- วัตถุข้อมูล (Data Object) หมายถึงข้อมูลซึ่งอยู่ภายใต้ Feature และมีคุณลักษณะสอดคล้องกับ Attribute ที่กำหนดไว้
 - Feature ที่ชื่อว่า เพศ ประกอบไปด้วย Data Object ชาย หญิง และเพศอื่น ๆ

ข้อมูล - 2

- ตัวอย่าง Feature และ Data Object

Row No.	ES1	SB1	SB2	SB3	ZB1	ZB2	ZB3	PT1	PT2	PT3
1	3	3	3	5	3	3	4	3	4	3
2	4	3	3	5	3	3	5	3	2	4
3	3	3	4	2	4	3	4	4	2	4
4	4	3	4	2	3	3	4	4	2	3
5	3	4	2	4	3	4	3	3	2	4
6	3	3	4	2	4	3	3	4	3	4
7	4	3	4	2	3	3	4	3	2	4
8	3	4	3	4	2	3	2	3	4	2
9	3	3	2	4	3	2	4	3	4	2
10	4	3	2	4	3	4	2	2	3	4
11	4	3	2	4	3	3	2	3	2	4
12	3	3	4	3	5	2	3	3	2	4
13	3	4	3	4	2	4	3	4	2	3
14	4	3	4	3	2	2	4	2	3	4
15	4	4	3	4	2	4	3	4	2	3
16	3	3	2	4	3	2	4	4	2	3
17	4	4	3	4	2	3	3	3	4	2
18	3	3	2	4	3	2	3	4	2	3

Feature

Data Object

คุณลักษณะของชุดข้อมูล (Data Set) - 1

- มิติของข้อมูล (Data Dimensionality) คือ จำนวนของ Feature ภายใน Data Set
 - Data Set ที่มีจำนวน Feature น้อย ง่ายต่อการนำเอาข้อมูลมาทำการวิเคราะห์
 - Data Set ส่วนใหญ่ มักมี Feature เป็นจำนวนมาก
 - Data Scientist ต้องทำการวิเคราะห์รูปแบบของข้อมูล เพื่อตรวจสอบว่าสอดคล้องกับวัตถุประสงค์ในการวิเคราะห์มูลเพียงใด
- การกระจายตัวของข้อมูล
 - ข้อมูลที่มีลักษณะโน้มเอียงไปในทางใดทางหนึ่ง ไม่เหมาะสมสำหรับวิเคราะห์ด้วย Data Mining Technique
 - นักวิทยาศาสตร์ข้อมูลต้องทำการตรวจสอบว่าข้อมูลเบี้ยง หรือเบี้ยวหรือไม่ ก่อนจะนำมาวิเคราะห์ โดยการใช้ Data Visualization
 - กระบวนการ Data Pre-Processing ยังทำการปรับแต่งให้ข้อมูลมีการกระจายตัวที่ดีก่อนนำมาวิเคราะห์

คุณลักษณะของชุดข้อมูล (Data Set) - 2

- หน่วยวัดของข้อมูล (Resolution) คือ การที่ Data Object สามารถถูกสรุปได้ในหลากหลายหน่วยวัด ยกตัวอย่างเช่น ความสูง สามารถอธิบายได้ทั้งในหน่วยเซนติเมตร และนิวตัน
- Data Scientist ต้องทำการกำหนด Resolution ให้เหมาะสมต่อการวิเคราะห์ข้อมูล ยกตัวอย่างเช่น
 - การกำหนดหน่วยวัดของความกดอากาศเป็นรายเดือน การกำหนดแบบนี้ไม่สามารถนำไปวิเคราะห์การเกิดขึ้นของพายุได้อย่างทันท่วงที
 - ควรทำการกำหนดหน่วยวัดเป็นรายชั่วโมง เพื่อประโยชน์ในการนำไปพยากรณ์การเกิดพายุ
- ในการนี้ที่ Data Scientist ทำการรวบรวม Data Object มาจากหลายแหล่งข้อมูล เพื่อนำมารวมลงใน Feature เดียว จำเป็นต้องทำการสำรวจ Resolution และปรับแต่งให้อยู่ใน Resolution เดียวกันทั้งหมด

ลักษณะของข้อมูลเชิงปริมาณ -1

- Nominal เป็นข้อมูลที่มีลักษณะของการจัดกลุ่ม
 - เพศ ประกอบไปด้วย หญิง ชาย และเพศอื่น ๆ
 - จังหวัด ประกอบไปด้วย กรุงเทพฯ นครสวรรค์ ฉะเชิงเทรา ฯลฯ
 - Data scientist สามารถทำการเปรียบเทียบข้อมูลที่เป็น Nominal ได้ เช่น มีจำนวนเพศชายไม่เท่ากับเพศหญิง
- ทำการพرรณข้อมูลด้วย Mode คือการที่มีจำนวนความถี่ซ้ำกันมากที่สุด หรือค่าที่มีผู้ตอบเลือกมากที่สุด
 - การคิดค่า Mode สามารถทำการแจกแจงความถี่ด้วย จำนวนความถี่ที่ซ้ำกันมากที่สุดในแต่ละช่วงข้อมูล ยกตัวอย่าง
 - 0 – 5 3
 - 6 – 10 7
 - ช่วงของข้อมูล 0 – 5 มีค่า Mode = 3 แปลว่ามีคนเลือกเลข 3 มากที่สุด
 - ช่วงของข้อมูล 6 – 10 มีค่า Mode = 7 แปลว่ามีคนเลือกเลข 7 มากที่สุด
- สำหรับข้อมูลที่เป็น Nominal สามารถพรรณข้อมูลได้คือ ผู้ตอบแบบสอบถามจากภาคเหนือมีจำนวนผู้ตอบแบบสอบถามจากภาคเชียงใหม่มากที่สุด

ลักษณะของข้อมูลเชิงปริมาณ -2

- Data scientist พึงไม่นำข้อมูลที่เป็น Nominal Scale มาทำการคำนวณด้วยการบวก ลบ คูณ หาร หรือค่าเฉลี่ย อาทิ มีจำนวนเพศชาย 7.5 คน เช่นนี้ไม่สมเหตุสมผลในการวิเคราะห์ข้อมูล
- การเปลี่ยนตัวหนังสือให้กลายเป็นตัวเลข ไม่สามารถทำให้ข้อมูลตัวเลขเหล่านั้น มีค่าที่แท้จริงแล้วนำไปคำนวณได้ยกตัวอย่างเช่น
 - เขตบางรัก เป็น 1
 - เขตสาธร เป็น 2
 - การนำเอาเลข $1 + 2 = 3$ ไม่มีนัยสำคัญใด ๆ ในการคำนวณและการแปลผลข้อมูล

ลักษณะของข้อมูลเชิงปริมาณ -3

- Ordinal Scale คือข้อมูลที่มีลักษณะเป็นลำดับ อาทิ ทัศนคติ พฤติกรรมการซื้อ ซึ่งถูกรวบรวมข้อมูลด้วย Likert Scale ดังตัวอย่าง

	เห็นด้วยมาก ที่สุด	เห็นด้วยมาก	เห็นด้วยปาน กลาง	ไม่เห็นด้วย	ไม่เห็นด้วย อย่างยิ่ง
1. วิทยาลัย นวัตกรรมจัดการ เรียนการสอนที่ สะท้อนถึงการบูร ณาการ วัฒนธรรม สื่อ สร้างสรรค์ เทคโนโลยี ดิจิทัล และการ บริการเข้าด้วยกัน	<input type="radio"/>				

ลักษณะของข้อมูลเชิงปริมาณ -4

- ข้อมูลที่เป็น Ordinal Scale ยังครอบคลุมต่อข้อมูลอื่น ๆ ที่มีลักษณะเรียงลำดับ อาทิ เลขที่ถนน ผลการเรียน
- การนำข้อมูลที่เป็น Ordinal Scale มาทำพารณาสามารถทำได้โดยการใช้ Median หรือค่ากึ่งกลางของข้อมูล อาทิ 5, 6, 9, 10, 11, 13, 14, 16, 17 ค่า Median ของข้อมูลชุดดังกล่าวคือ 11
- Data Scientist สามารถวิเคราะห์ความสัมพันธ์ของ 2 Features ที่มีลักษณะเป็น Ordinal Scale ได้โดยใช้ correlation test ยกตัวอย่างเช่น
 - การใช้งานง่ายของ Application
 - ความตั้งใจในการซื้อสินค้า
 - หากทั้ง 2 features มีความสัมพันธ์ต่อกันในทางบวก Data Scientist สามารถแปลผลได้ว่า ยิ่ง Application ใช้งานง่าย ผู้ซื้อ ก็จะมีความตั้งใจในการซื้อสินค้ามากขึ้น

ลักษณะของข้อมูลเชิงปริมาณ - 5

- Data scientist พึงไม่นำข้อมูลที่เป็น Ordinal Scale มาทำการคำนวณด้วยการบวก ลบ คูณ หาร หรือค่าเฉลี่ย
 - Ordinal Scale ที่ถูกรวบรวมข้อมูลผ่าน Likert Scale ถูกแทนให้เป็นข้อมูลเชิงปริมาณตั้งแต่ 1 – 5
 - 5 หมายถึง ตั้งใจซื่อมากที่สุด
 - 4 หมายถึง ตั้งใจซื่อมาก
 - การนำ $(5 + 4)/2 = 4.5$ ไม่อาจนำมาแปลผลออกมาเป็นลักษณะใดลักษณะหนึ่ง ว่าผู้ซื้อตั้งใจซื่อสินค้าผ่าน Application ในระดับใดกันแน่
 - Ordinal Scale ที่มีลักษณะอื่น ๆ อาทิ เลขที่ถนน
 - ถนนเลขที่ 10
 - ถนนเลขที่ 11
 - $10 + 11 = 21$ เช่นนี้ไม่มีนัยในการแปลผลใด ๆ

ลักษณะของข้อมูลเชิงปริมาณ - 6

- Interval Scale คือ ช่วงของข้อมูลที่แตกต่างกันมีนัยสำคัญในการแปลผล อาทิ อุณหภูมิ หรือ คะแนนสอบ
- สามารถทำการประมาณข้อมูลได้ โดยการใช้ Mean
 - คะแนนเฉลี่ยของนักเรียนในวิชา Data Exploration
- สามารถทำการทดสอบความสัมพันธ์ (Correlation Test) ระหว่าง 2 Features ได้
 - ความสัมพันธ์ระหว่าง Feature คะแนนสอบวิชา Data Exploration กับ Feature คะแนนสอบวิชา Data-Preprocessing
 - สามารถแปลผลได้คือ ยิ่งนักศึกษาทำคะแนนสอบวิชา Data Exploration ได้มากเที่ยงได้ นักศึกษา ก็จะยิ่งทำคะแนนสอบวิชา Data-Preprocessing ได้มากเท่านั้น ในกรณีที่เป็นความสัมพันธ์เชิงบวก
- T-Test
 - นำค่าเฉลี่ยของ Feature มาทดสอบกับ ค่าที่กำหนดไว้แล้ว อาทิ คะแนนเฉลี่ยของนักเรียนทั้งห้องมากกว่าหรือเท่ากับ 75
 - นำค่าเฉลี่ยของ 2 Features มาทดสอบว่าแตกต่างกันเพียงใด อาทิ คะแนนเฉลี่ยการสอบ Pre-Test และ Post-Test ของนักเรียนทั้งห้อง

ลักษณะของข้อมูลเชิงปริมาณ - 7

- Ratio Scale หมายถึงตัวเลข อาทิ ปริมาณเงิน ระยะทาง
- Data Scientist สามารถคำนวณทางคณิตศาสตร์กับข้อมูลที่เป็น Ratio ได้ อาทิ การบวก ลบ คูณ หาร
- Regression อาทิ การพยากรณ์ความสูงจากน้ำหนัก
- Multiple regression อาทิ การคำนวณราคาที่พึ่งจาก ระยะทางและระยะเวลาที่เปิดให้บริการ

ลักษณะของข้อมูลเชิงปริมาณ - 7

- Discrete Data คือ ข้อมูลที่สามารถนับเป็นจำนวนเต็มได้ อาทิ เพศชาย 50 คน ลูกค้า 320 คน
 - Nominal Scale
 - Ordinal Scale
- Continuous Data คือข้อมูลที่เป็นเลขจำนวนจริง อาทิ ระยะทาง ความสูง ซึ่งต้องใช้มาตรวัดที่มีความละเอียดในการวัด
 - Interval Scale
 - Ratio Scale

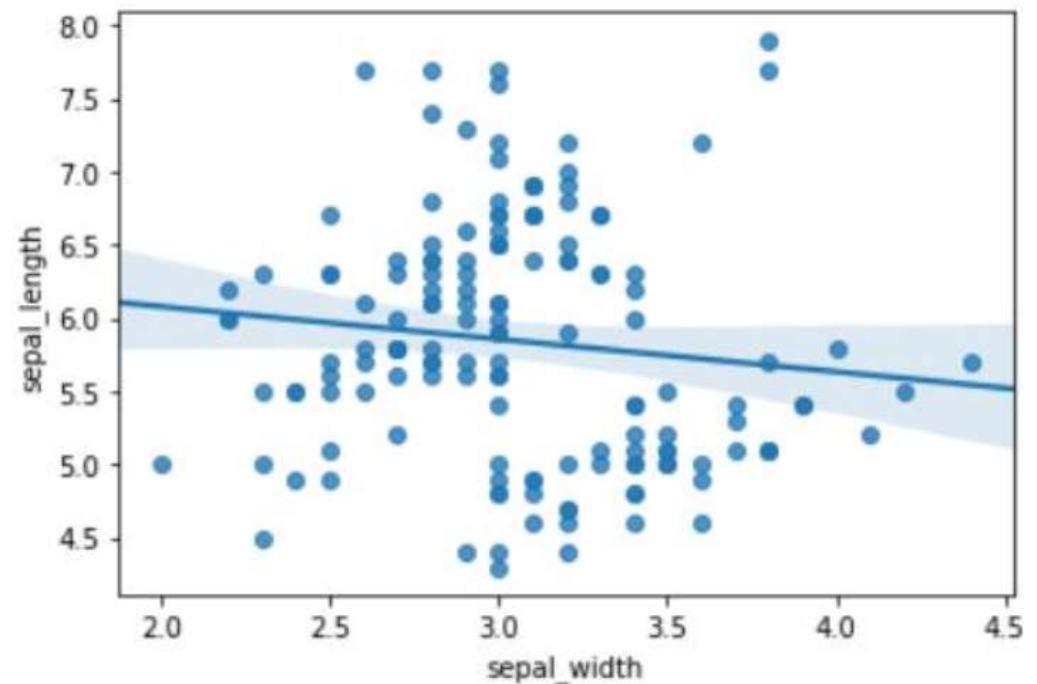
คุณภาพข้อมูล

- Bias

- ในการนี้ที่นักวิทยาศาสตร์ข้อมูลต้องการรวบรวมข้อมูลเกี่ยวกับระดับการยอมรับทางเทคโนโลยีของคนในประเทศไทย
 - การเลือกชุดข้อมูลเฉพาะประชากรภายในกรุงเทพฯ และปริมณฑล ย่อมไม่สามารถใช้เป็นตัวแทนของคนไทยทั้งประเทศได้
 - ชุดข้อมูลที่ถูกเลือกมาใช้เป็นตัวแทนของแต่ละจังหวัดสัมพันธ์กับสัดส่วนประชากรของแต่ละจังหวัดหรือไม่ อาทิ เชียงใหม่มีประชากร 5 ล้าน คน จังหวัดลำพูน มีประชากร 2 ล้าน หากสุ่มตัวอย่างด้วยจำนวนเท่ากัน ย่อมไม่สามารถนำมารวมกันแล้วใช้เป็นตัวแทนประชากรทั้งหมดได้

คุณภาพข้อมูล

- Outlier
 - Data object ที่มีความผิดปกติ
 - เมื่อเกิด Data object ที่มี Outlier ไม่ว่าจะทางลบหรือทางบวก ย่อมทำให้ค่าเฉลี่ยของ feature ผิดเพี้ยนตามไปด้วย
 - กระบวนการ Data Exploration จะทำให้เห็นค่าเหล่านี้
 - Scatter Plot
 - อย่างไรก็ต้อง Data Scientist ควรทำการพิจารณา ก่อนว่าค่าที่พบ เป็น Outlier จริงหรือไม่



คุณภาพข้อมูล

- Missing Value

- เกิดขึ้นได้จากการที่ User ไม่กรอกข้อมูล
- เกิดขึ้นได้จากการที่ เครื่องมือรวบรวมข้อมูลไม่ครอบคลุมในสิ่งที่ User จำเป็นต้องกรอก
 - ความถี่ในการเกิดประจำเดือน – หาก User เป็นเพศชายจะต้องข้ามในการตอบข้อนี้ไป
 - จำนวนเงินเดือน – หาก User ไม่มีงานทำต้องข้ามในการตอบข้อนี้ไป

- การจัดการกับ Missing Value

- ขจัด Data Object ที่เป็น Missing Value
- ทำการคำนวณค่าใหม่ อาทิ ค่าเฉลี่ย หรือค่า Mode และเติมเต็มลงไปใน Missing Value
- ทำการพยากรณ์ค่าที่เป็นไปได้ด้วยเทคนิคในการพยากรณ์ และเติมเต็มค่าใหม่ลงไปใน Missing Value

คุณภาพข้อมูล

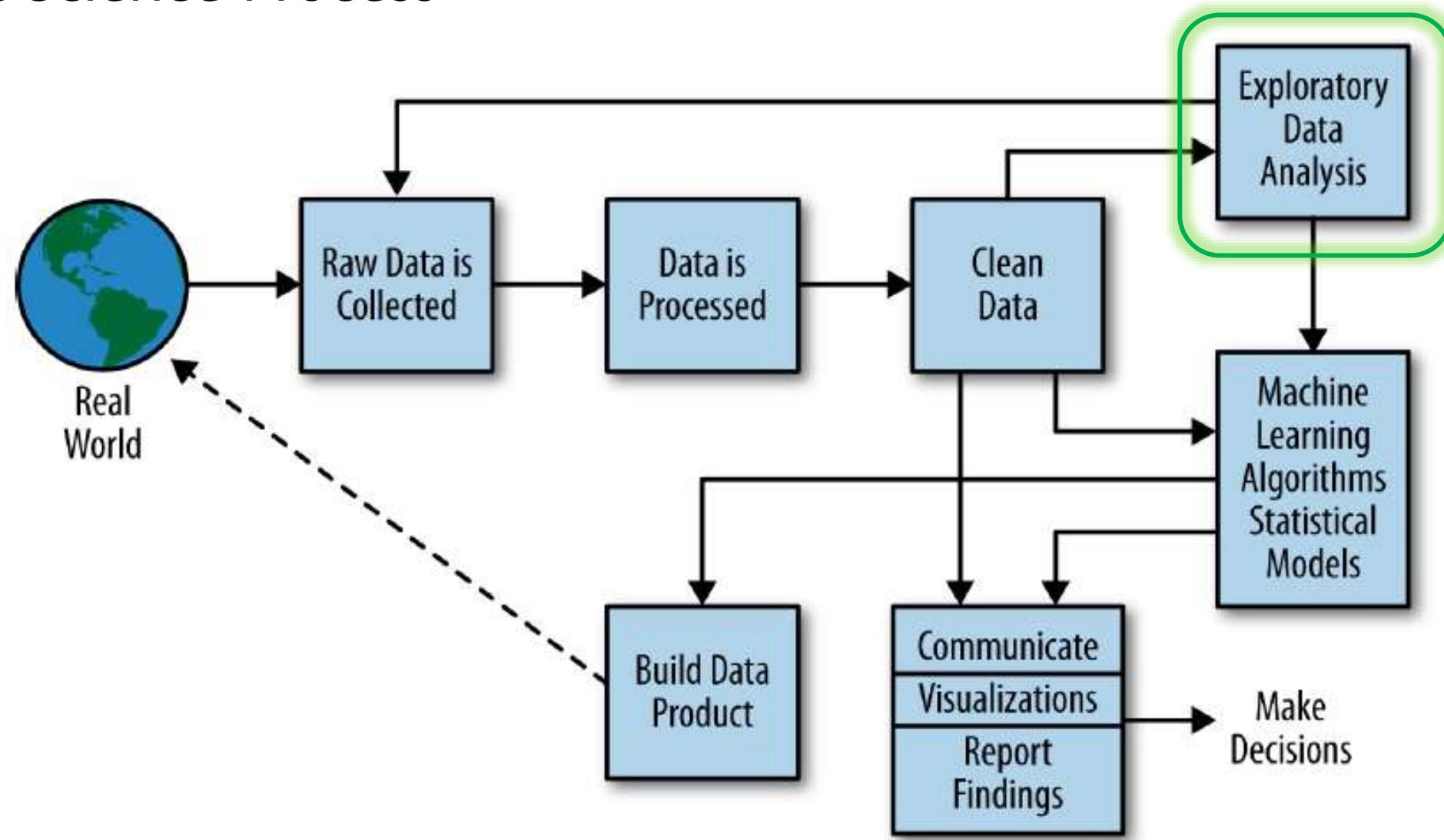
- Duplicate Data

- สามารถเกิดขึ้นได้ เมื่อ Data Scientist ทำการรวบรวมข้อมูลจาก Data Set ที่มาจากการหลาย Sources
 - อาทิ 1 คน มีหลายอีเมล์ที่ใช้ในการติดต่อ
- เราใช้กระบวนการ Data Pre-Processing ในการจัดการกับ Duplicate Data

- Inconsistency Data

- สาเหตุหลัก มักเกิดจากการที่ User ทำ Data Entry ผิดตั้งแต่แรก
 - ความสูงของคนติดลบไม่ได้
 - การพิมพ์ชื่อคนผิด

Data Science Process



Exploratory Data Analysis (EDA)

- Exploratory data analysis or “**EDA**” is a critical step in analyzing the data
- The main reasons are
 - detection of mistakes, outliers or abnormalities
 - checking of assumptions
 - preliminary selection of appropriate models
 - determining relationships among the explanatory variables
 - assessing the direction and rough size of relationships between explanatory and outcome variables

Types of EDA

- **Graphical or Non-graphical**
 - **Non-graphical** methods usually involve with calculation of summary statistics
 - **Graphical** methods obviously summarize the data in a diagrammatic or pictorial way
- **Univariate or Multivariate**
 - **Univariate** methods look at one variable (column) at a time while
 - **Multivariate** methods look at two or more variables at a time

Summary EDA Types

	UNIVARIATE	MUTIVARIATE
Graphical	<p>Quantitative Variable</p> <ul style="list-style-type: none"> • Histogram • Boxplots • Normal plot <p>Categorical Variable</p> <ul style="list-style-type: none"> • Bar Charts • Line Plot 	<p>One Categorical and One Quantitative Variable</p> <ul style="list-style-type: none"> • Side-by-side Boxplots <p>Two or More Categorical Variables</p> <ul style="list-style-type: none"> • Grouped Bar Chart <p>Two or More Quantitative Variables</p> <ul style="list-style-type: none"> • Scatterplot • Correlation Heatmap
Non-Graphical	<p>Categorical Variable</p> <ul style="list-style-type: none"> • Tabular <p>Quantitative Variable</p> <ul style="list-style-type: none"> • Location (mean, median) • Spread (IQR, std dev, range) • Modality (mode) • Shape (skewness, kurtosis) • Outliers 	<p>One Categorical and One Quantitative Variable</p> <ul style="list-style-type: none"> • Mean • Median • Range and Spread measures <p>Two or More Quantitative Variables</p> <ul style="list-style-type: none"> • Correlation • Covariance • Descriptive stat per

Univariate non-graphical EDA

Univariate non-graphical EDA

Categorical data

- The characteristics of interest for a categorical variable are simply the range of values and the frequency (or relative frequency) of occurrence for each value.
- Therefore, the only useful univariate non-graphical techniques for categorical variables is some form of tabulation of the frequencies, usually along with calculation of the fraction (or percent) of data that falls in each category.



Univariate non-graphical EDA

Quantitative data

#ดูค่าทางสถิติพื้นฐาน

df.describe()

	Account length	Area code	Number vmail messages
count	667.000000	667.000000	667.000000
mean	102.841079	436.157421	8.407796
std	40.819480	41.783305	13.994480
min	1.000000	408.000000	0.000000
25%	76.000000	408.000000	0.000000
50%	102.000000	415.000000	0.000000
75%	128.000000	415.000000	20.000000
max	232.000000	510.000000	51.000000

Univariate non-graphical EDA

Categorical data

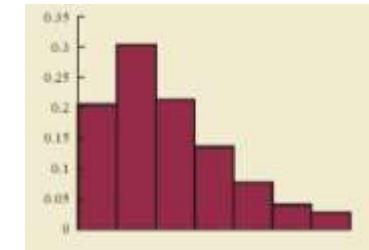
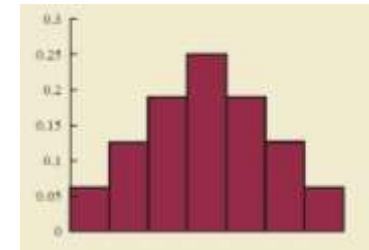
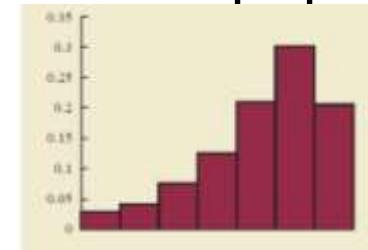
```
#ดูค่าทางสถิติพื้นฐานสำหรับข้อมูลที่เป็น object และ bool  
df.describe(include=['object',bool])
```

	State	International plan	Voice mail plan	Churn
count	667	667	667	667
unique	51	2	2	2
top	AZ	No	No	False
freq	19	614	478	572

Univariate non-graphical EDA

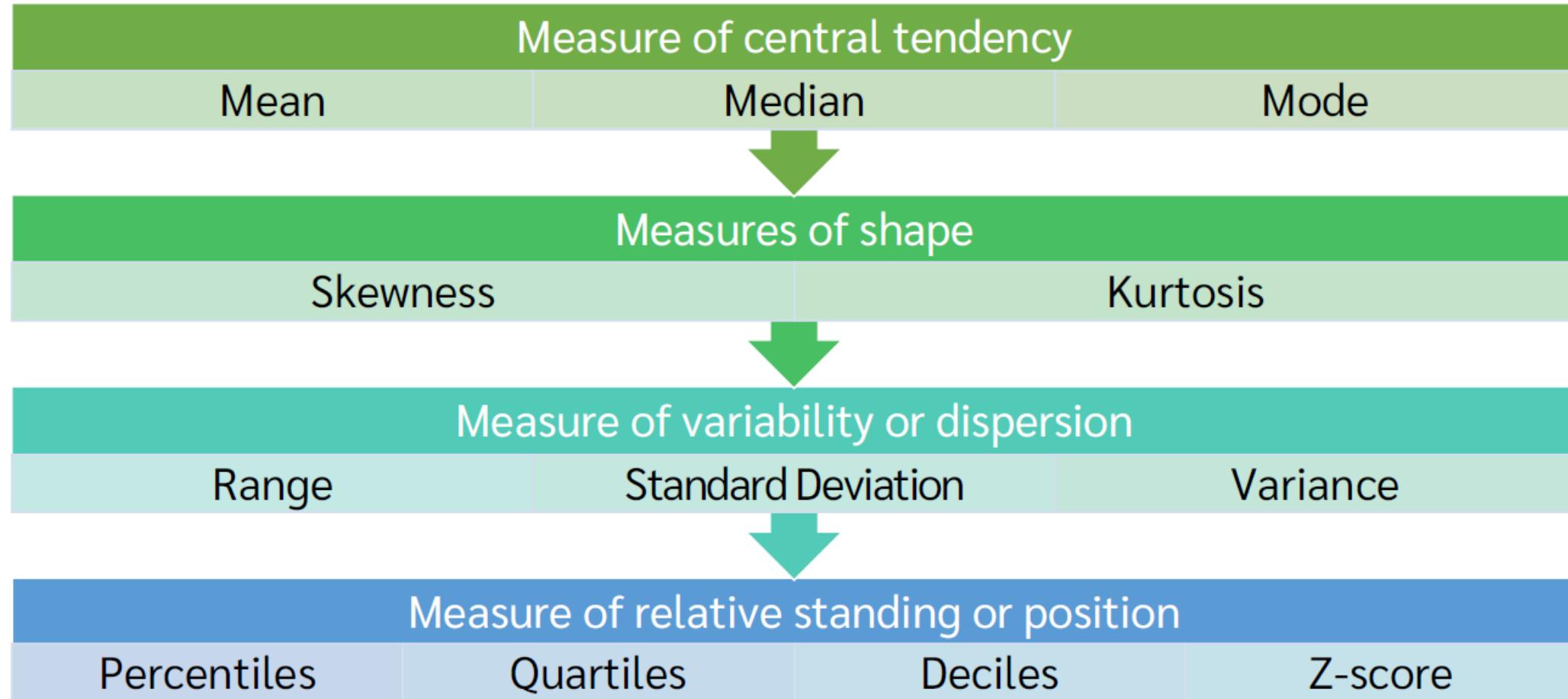
Quantitative data

- Univariate EDA for a quantitative variable is a way to make preliminary assessments about the **population distribution** of the variable using the data of the observed sample.
- The **characteristics of the population distribution** of a quantitative variable are its
 - Center
 - Spread
 - Shape
 - Outliers



Univariate non-graphical EDA

Quantitative data



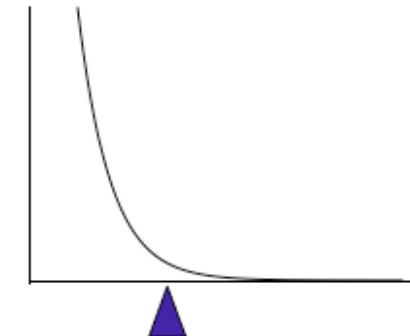
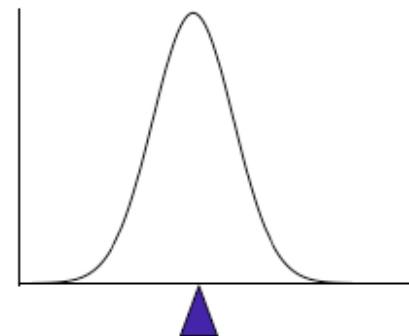
Central tendency (Middle Value)

Mean

I. The Mean

To calculate the average \bar{x} of a set of observations, add their value and divide by the number of observations:

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \dots + x_n}{n} = \frac{1}{n} \sum_{i=1}^n x_i$$



Central tendency

Median

- **Median** – the exact middle value
- **Calculation:**
 - If there are an odd number of observations, find the middle value
 - If there are an even number of observations, find the middle two values and average them
- **Example**

Some data:

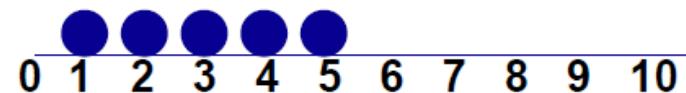
Age of participants: 17 19 21 22 23 23 23 38

$$\text{Median} = (22+23)/2 = 22.5$$

Central tendency

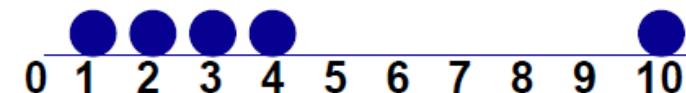
Which location measure is the best?

- Mean is best for symmetric distributions without outliers
- Median is useful for skewed distributions or data with outliers



Mean = 3

Median = 3



Mean = 4

Median = 3

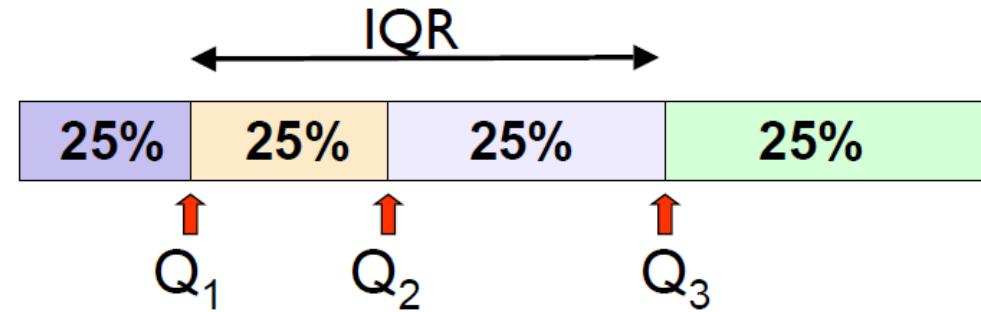
Scale: Standard deviation

$$\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$

For Normally distributed data, approximately 95% of the values lie within 2 SD of the mean.

1. Score (in the units that are meaningful)
2. Mean
3. Each score's deviation from the mean
4. Square that deviation
5. Sum all the squared deviations (Sum of Squares)
6. Divide by $n-1$
7. Square root – now the value is in the units we started with!!!

Scale: Quartiles and IQR



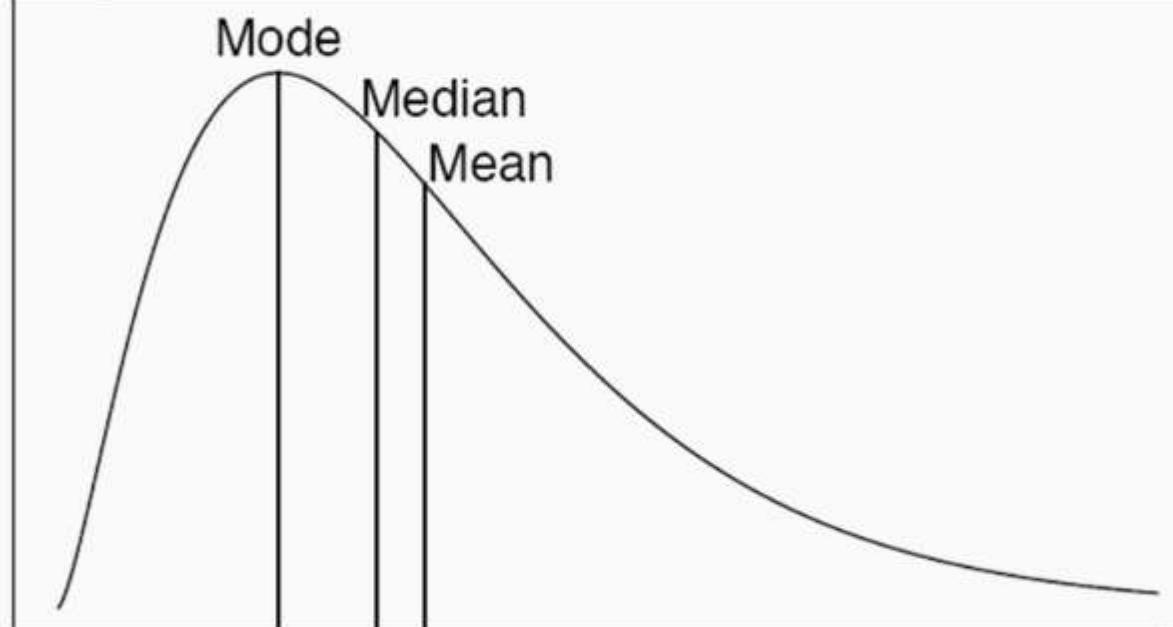
- The first quartile, Q_1 , is the value for which 25% of the observations are smaller and 75% are larger
- Q_2 is the same as the median (50% are smaller, 50% are larger)
- Only 25% of the observations are greater than the third quartile

Skewness

Positively skewed

- Longer tail in the high value
- Mean > Median > Mode

Positively skewed or skewed to the right

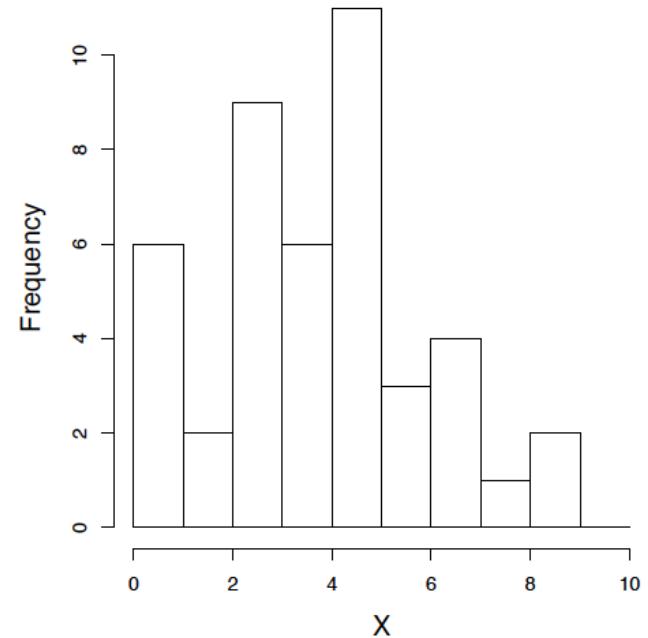


Univariate Graphical EDA

Univariate Graphical EDA

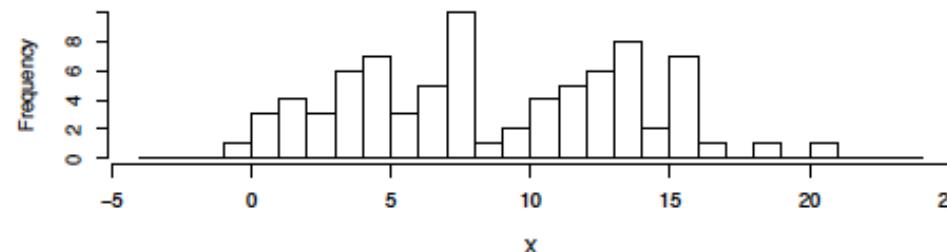
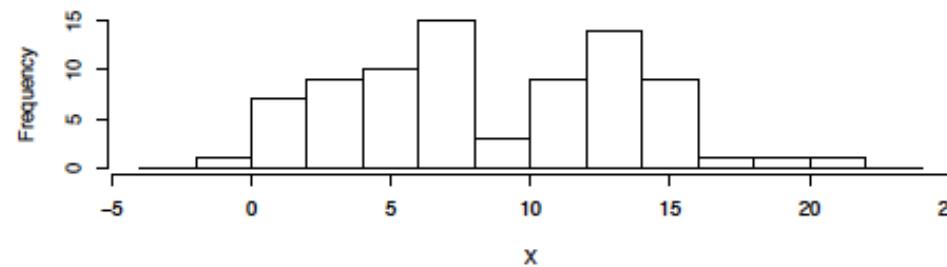
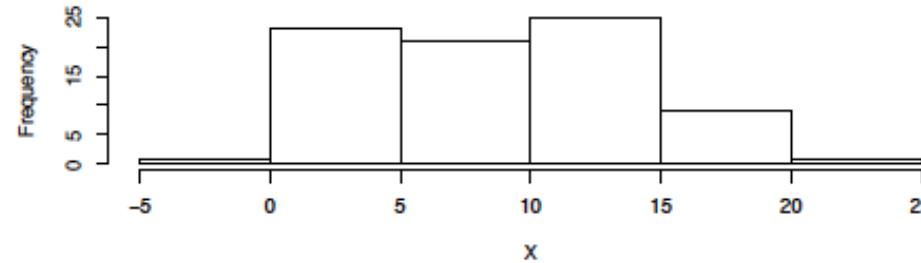
Histogram

- **Histogram** is a graphical representation of the **distribution of numerical data**
- It provides a view of data **density** and the **shape** of data distribution
- To construct a histogram, the first step is to
 - bin the range of values
 - count how many values fall into each interval
- The **bins** are usually specified as consecutive, non-overlapping intervals of a variable.
- The bins (intervals) must be adjacent and are usually equal size.



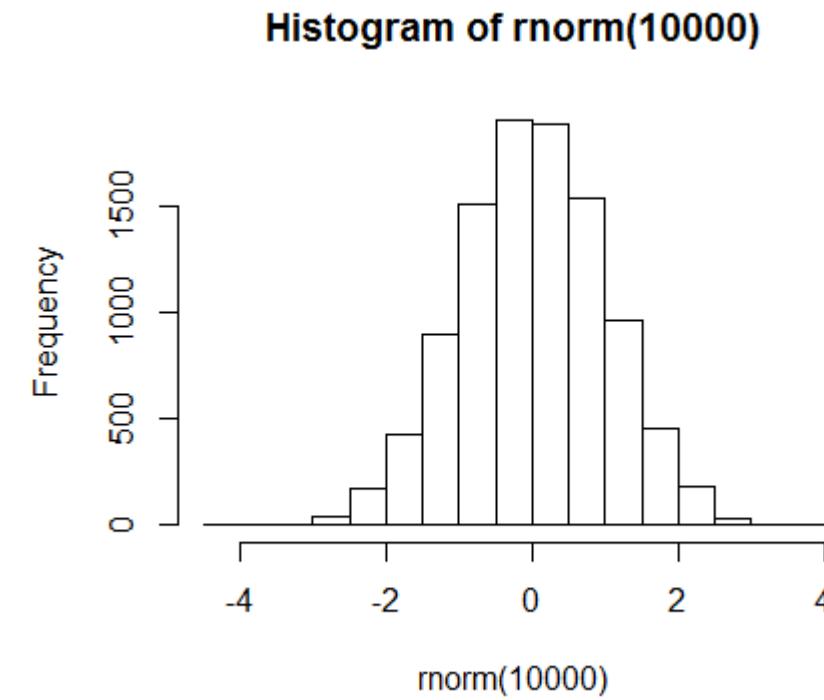
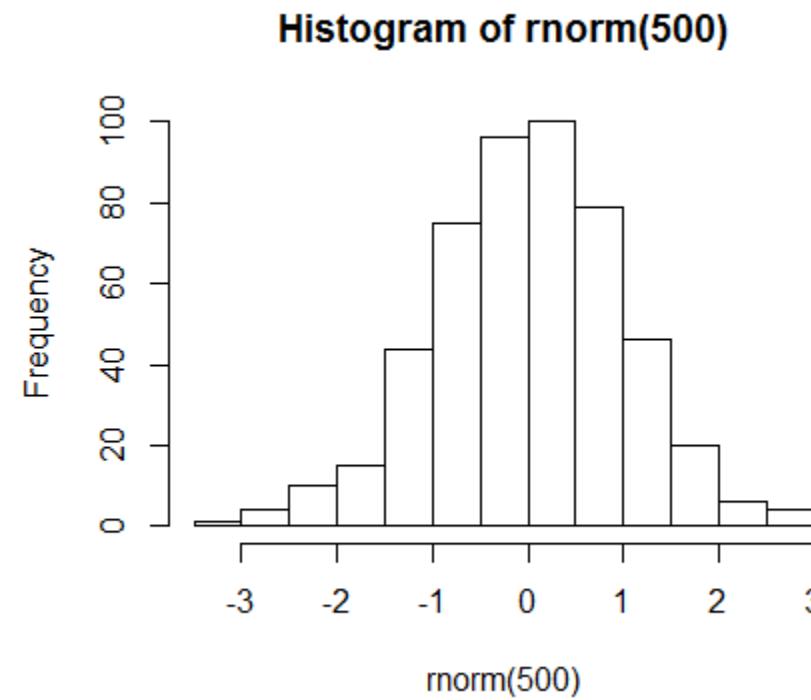
Univariate Graphical EDA

Effects of Histogram Bin



Univariate Graphical EDA

Effects of Number of Samples



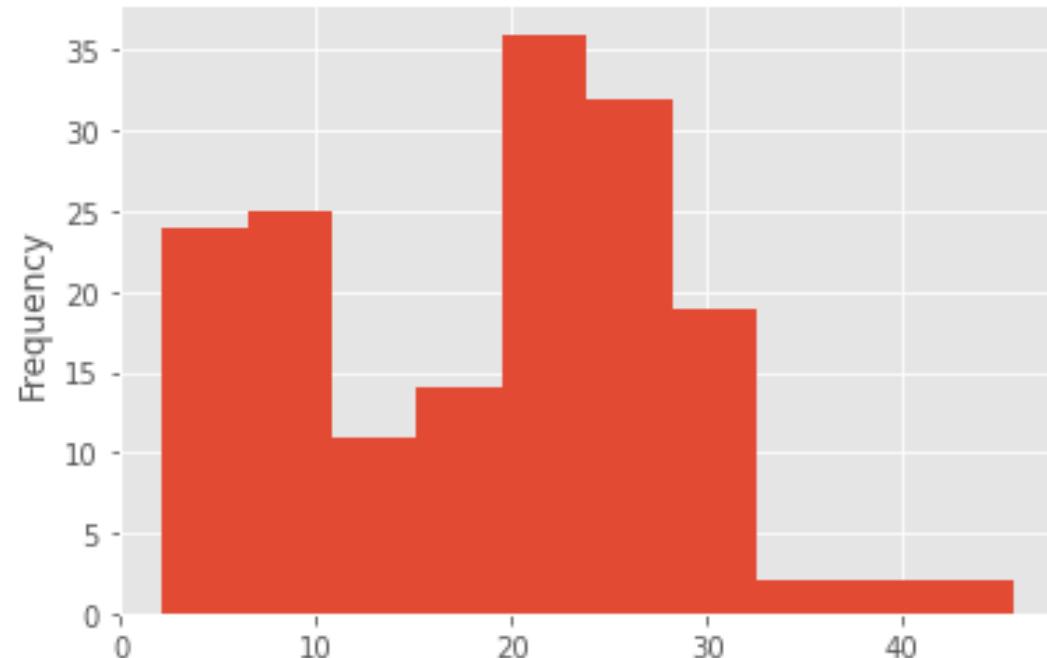
Univariate Graphical EDA

Histogram in Python

- Histogram
 - `plot()` method

```
Country
Afghanistan      4.5
Albania          22.3
Algeria          26.6
Angola            6.8
Antigua and Barbuda 19.1
Argentina         28.5
Armenia           20.9
Australia         30.4
Austria           21.9
Azerbaijan        19.9
Name: Obesity, dtype: float64
```

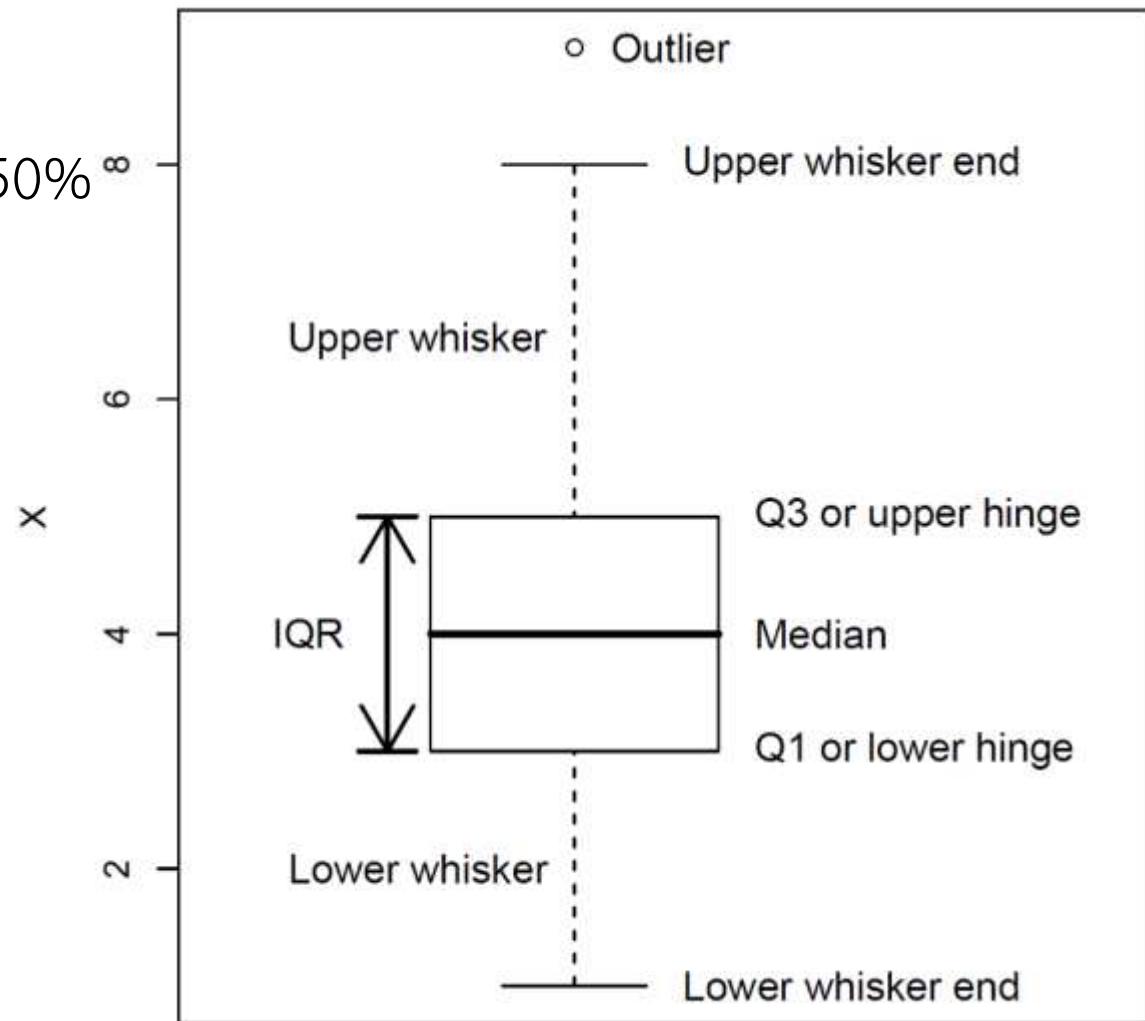
```
# An easy way to create the histogram.
df_fat['Obesity'].plot.hist()
<AxesSubplot:ylabel='Frequency'>
```

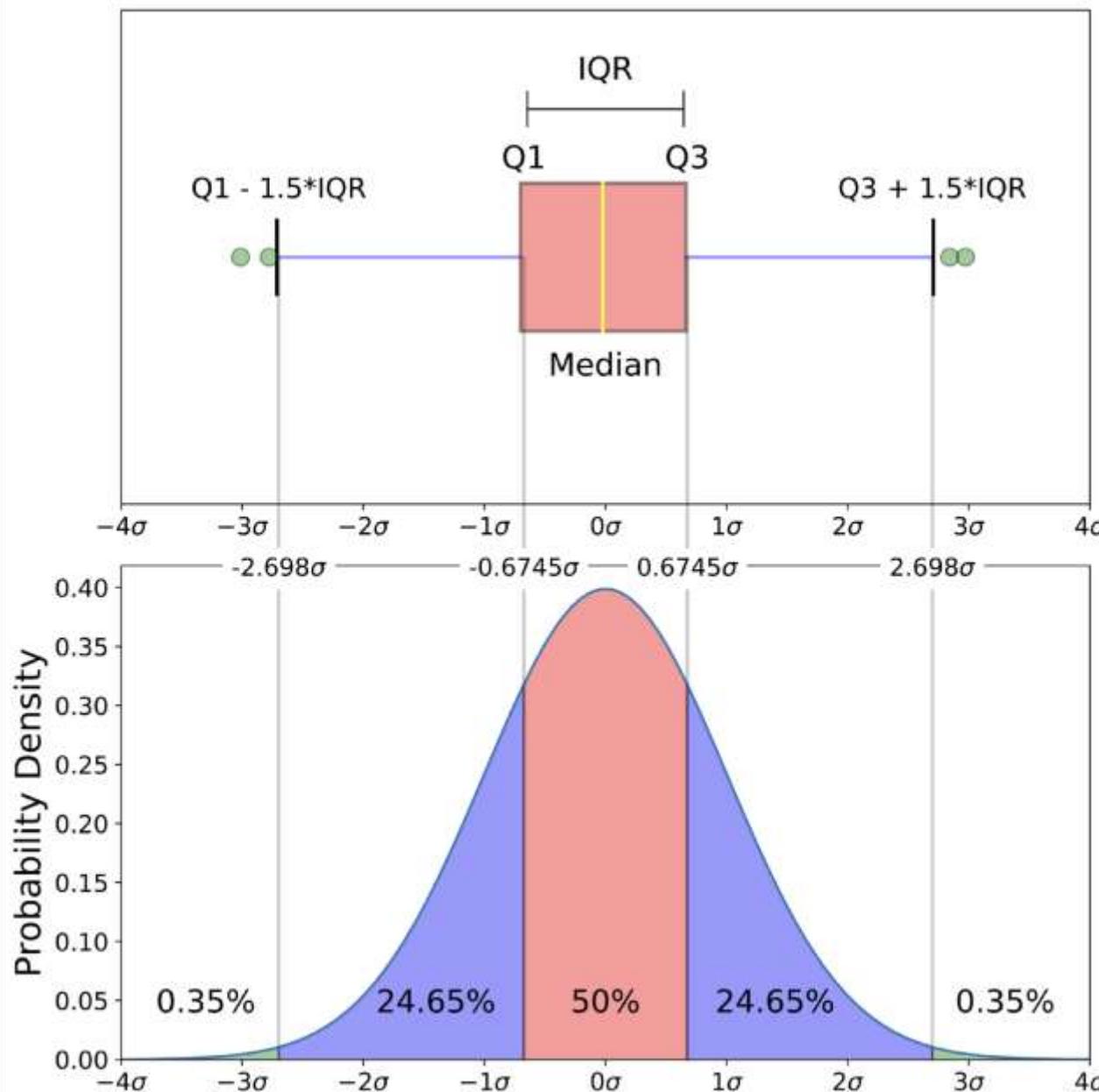


Univariate Graphical EDA

Boxplot

- The box in boxplot represents the middle 50% of the data
- The middle line indicates median
- Whiskers can be designated as either
 - Max/Min
 - Outlier boundaries
 - Upper = $Q3 + 1.5 * IQR$
 - Lower = $Q1 - 1.5 * IQR$



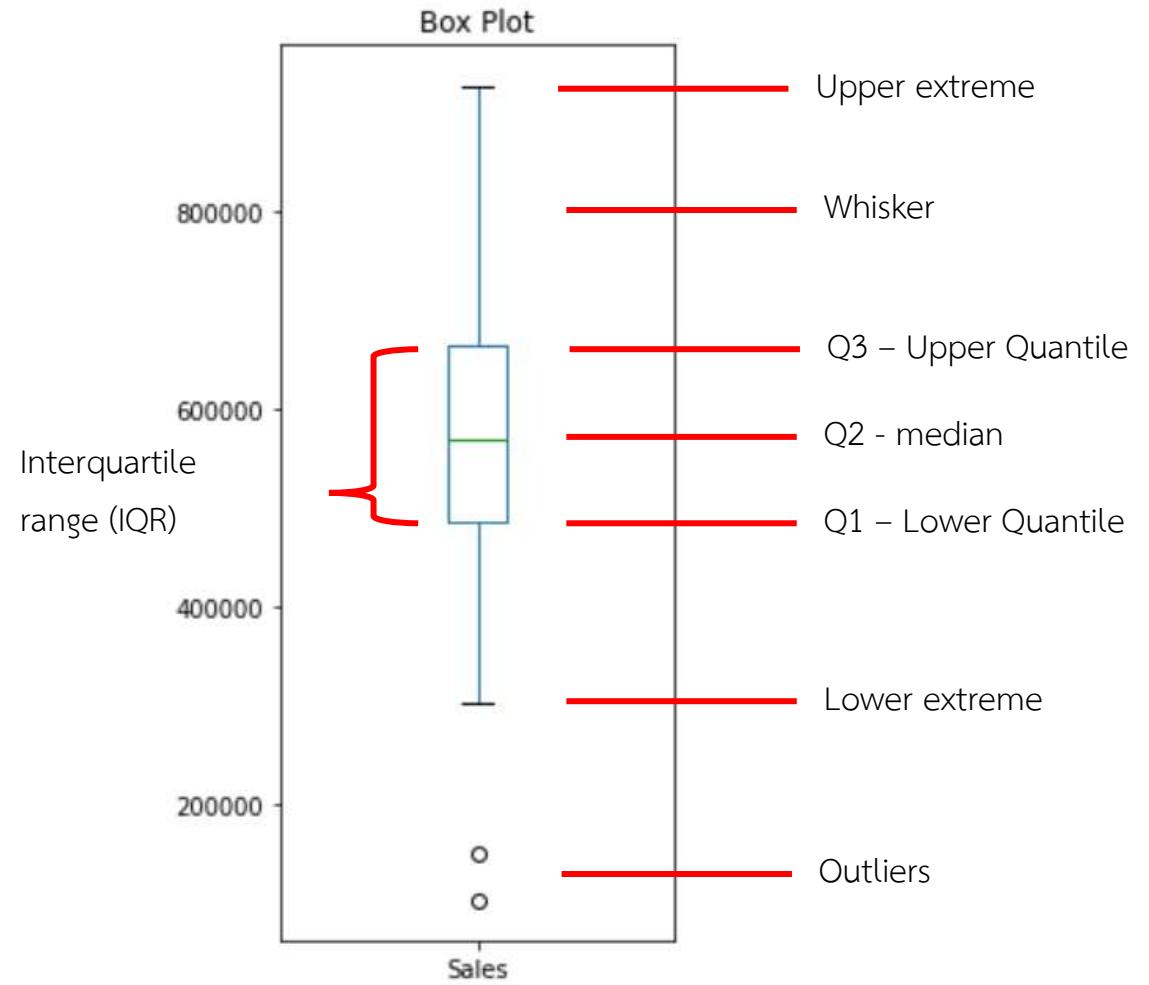


Univariate Graphical EDA

Boxplot in Python

- Box Plots with matplotlib library

```
#We only take the variable sales.  
df_oper_sales = df_operations.loc[:, 'Sales']  
  
df_oper_sales.plot(kind='box', figsize=(3,7))  
plt.title('Box Plot')  
plt.show()
```



Multivariate Graphical EDA

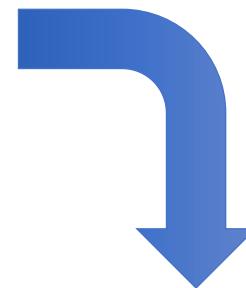
Multivariate Non-Graphical EDA

- Multivariate non-graphical EDA techniques show the **relationship between two or more variables** in the form of either [cross-tabular](#) or [statistics](#)

Multivariate Non-Graphical EDA

Cross-Tabulation

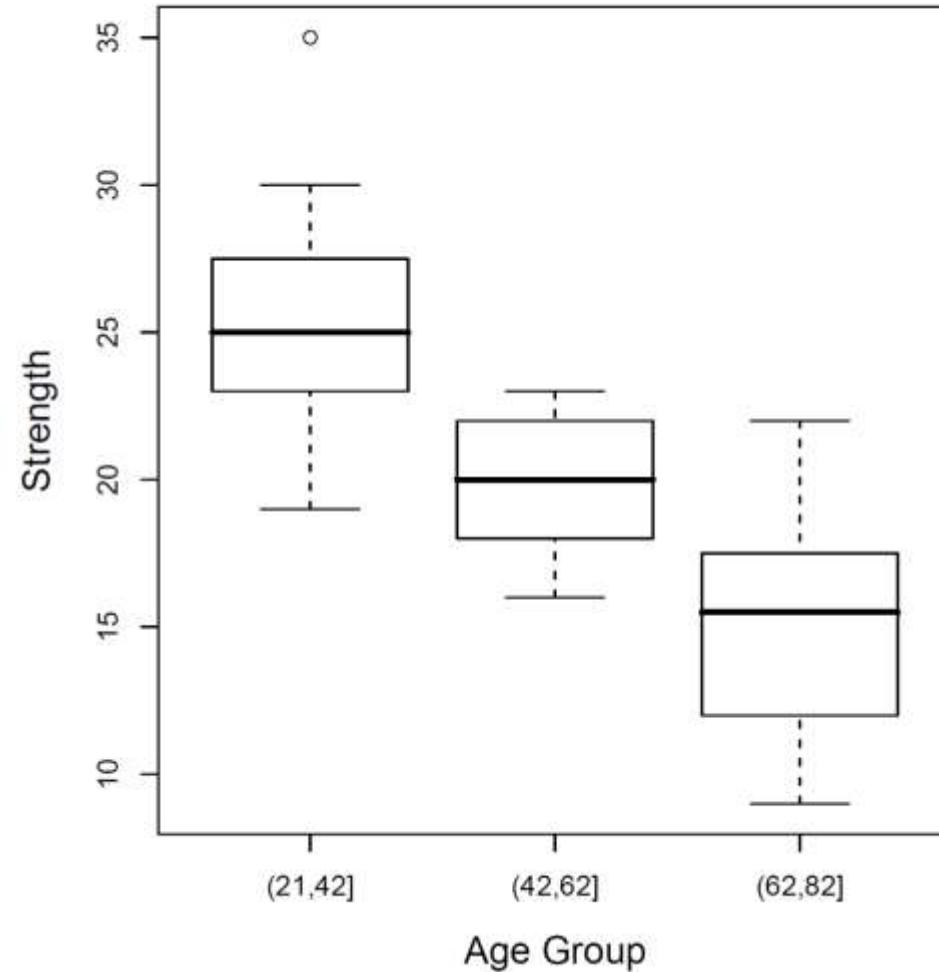
Subject ID	Age Group	Sex
GW	young	F
JA	middle	F
TJ	young	M
JMA	young	M
JMO	middle	F
JQA	old	F
AJ	old	F
MVB	young	M
WHH	old	F
JT	young	F
JKP	middle	M



Age Group / Sex	Female	Male	Total
young	2	3	5
middle	2	1	3
old	3	0	3
Total	7	4	11

Multivariate Graphical EDA

side-by-side boxplot



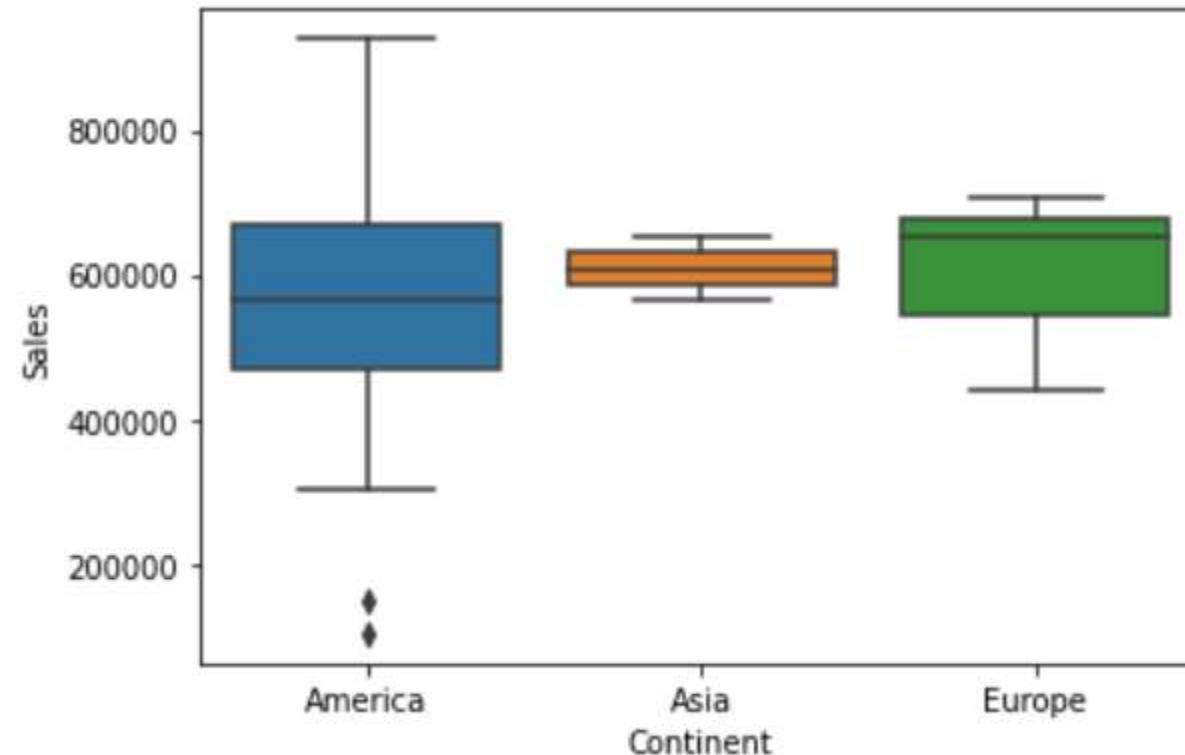
Side-by-side boxplots are good for examining the relationship between a categorical variable and a quantitative variable.

Multivariate Graphical EDA

side-by-side boxplot in Python

- Box Plots with seaborn library

```
sns.boxplot(x="Continent", y="Sales", data=df_operations)
```

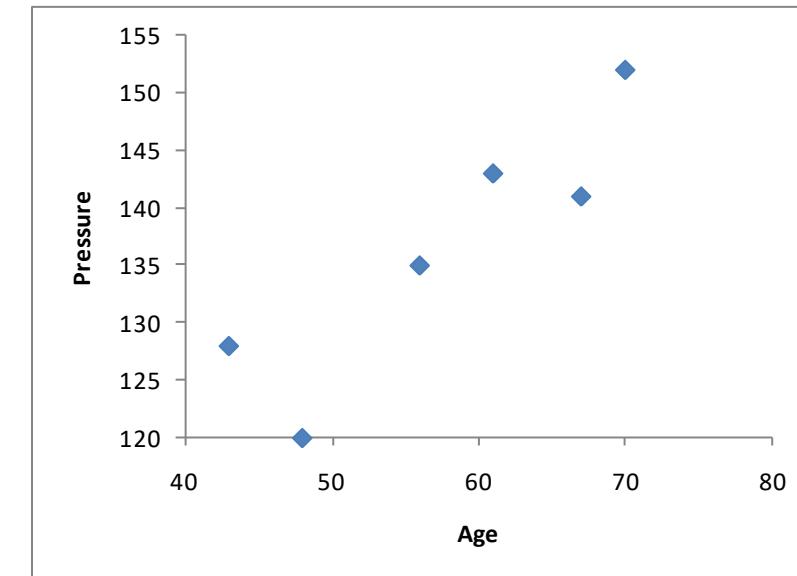


Multivariate Graphical EDA

Scatter plot

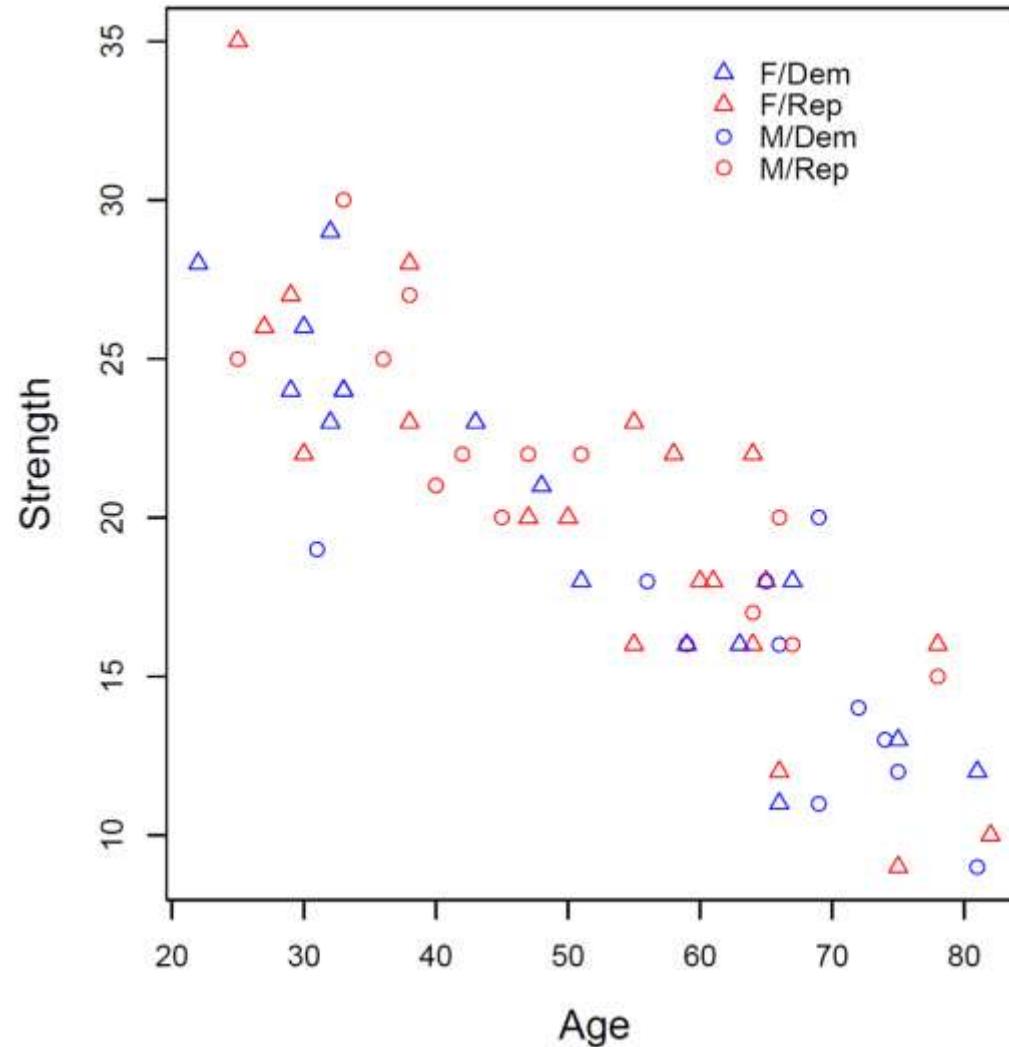
- A **scatter plot** is a graph of the ordered pairs (x,y) of numbers consisting of the **independent variable** x and the **dependent variable** y.

Subject	Age x	Pressure y
A	43	128
B	48	120
C	56	135
D	61	143
E	67	141
F	70	152



Multivariate Graphical EDA

Scatter plot with categorical data



- One or two additional **categorical variables** can be accommodated on the scatterplot
- Encoding the additional information in the **symbol type** and/or **color**.
- Here, colors code political party and gender

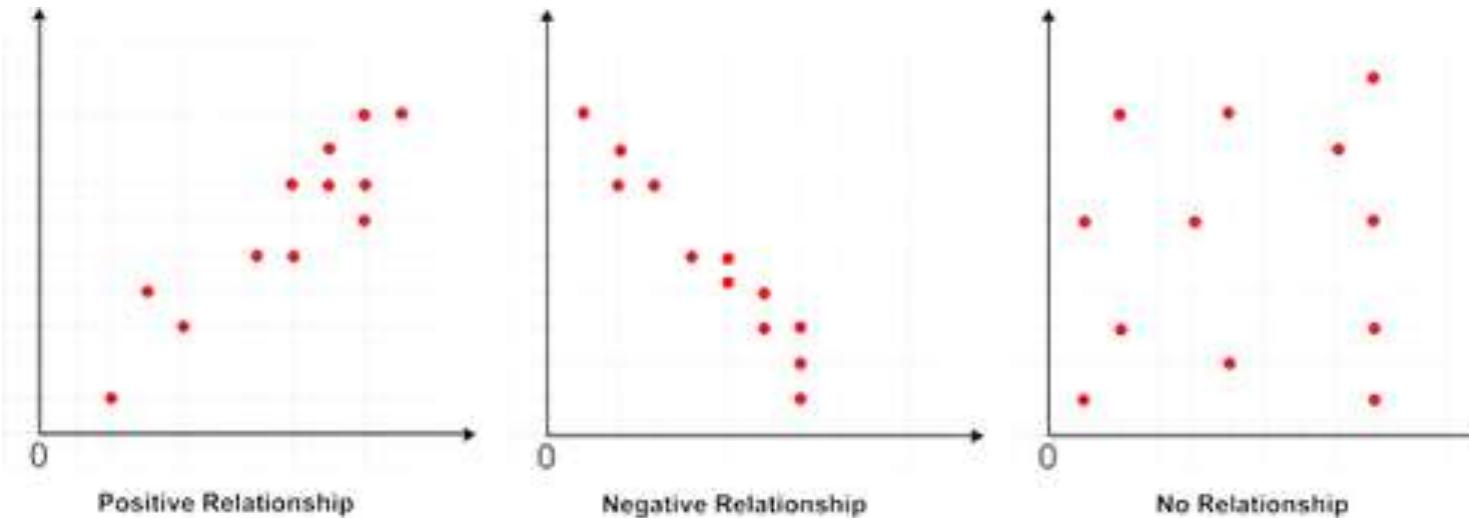
Relationship Between Variable

The study of the relationship between variable

Subject ID	Age	Strength	Age-50	Str-19	product
GW	38	20	-12	+1	-12
JA	62	15	+12	-4	-48
TJ	22	30	-28	+11	-308
JMA	38	21	-12	+2	-24
JMO	45	18	-5	-1	+5
JQA	69	12	+19	-7	-133
AJ	75	14	+25	-5	-125
MVB	38	28	-12	+9	-108
WHH	80	9	+30	-10	-300
JT	32	22	-18	+3	-54
JKP	51	20	+1	+1	+1
Total			0	0	-1106

Positive and negative relationship

- Simple relationships can also be positive or negative
- A **positive relationship** exists when both variables increase or decrease at the same time.
- In a **negative relationship**, as one variable increases, the other variable decreases, and vice versa.



Correlation Analysis

- Pearson's Correlation Coefficient.

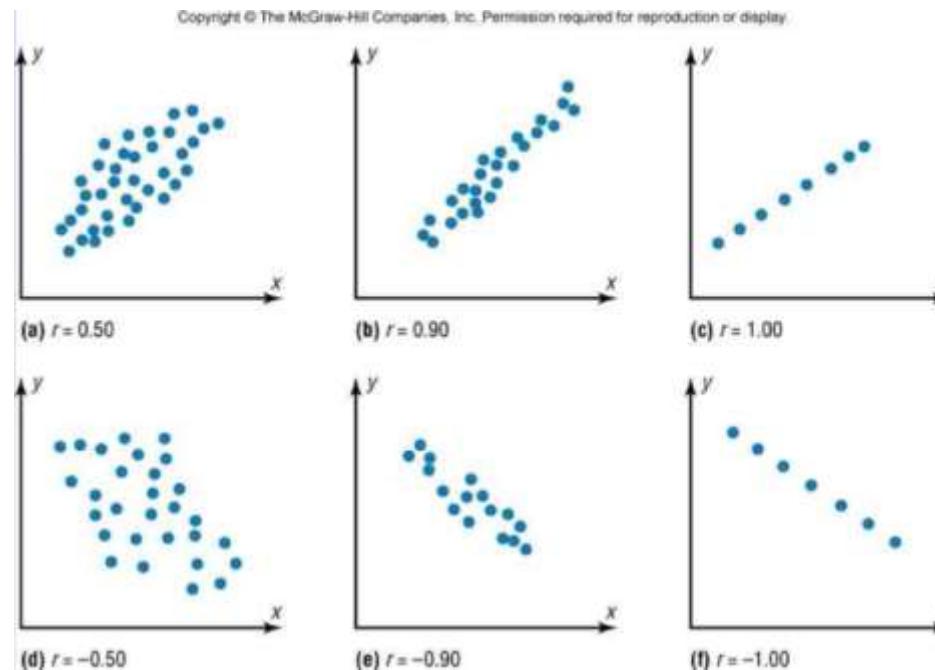
$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}}$$

$$r = \frac{(20 * 4213.468) - (190.3998 * 361.6)}{\sqrt{[(20 * 2501.446) - 190.3998^2][(20 * 8568.5) - 361.6^2]}}$$

$$r = \frac{559552262.4}{23654.85706} = 0.651908$$

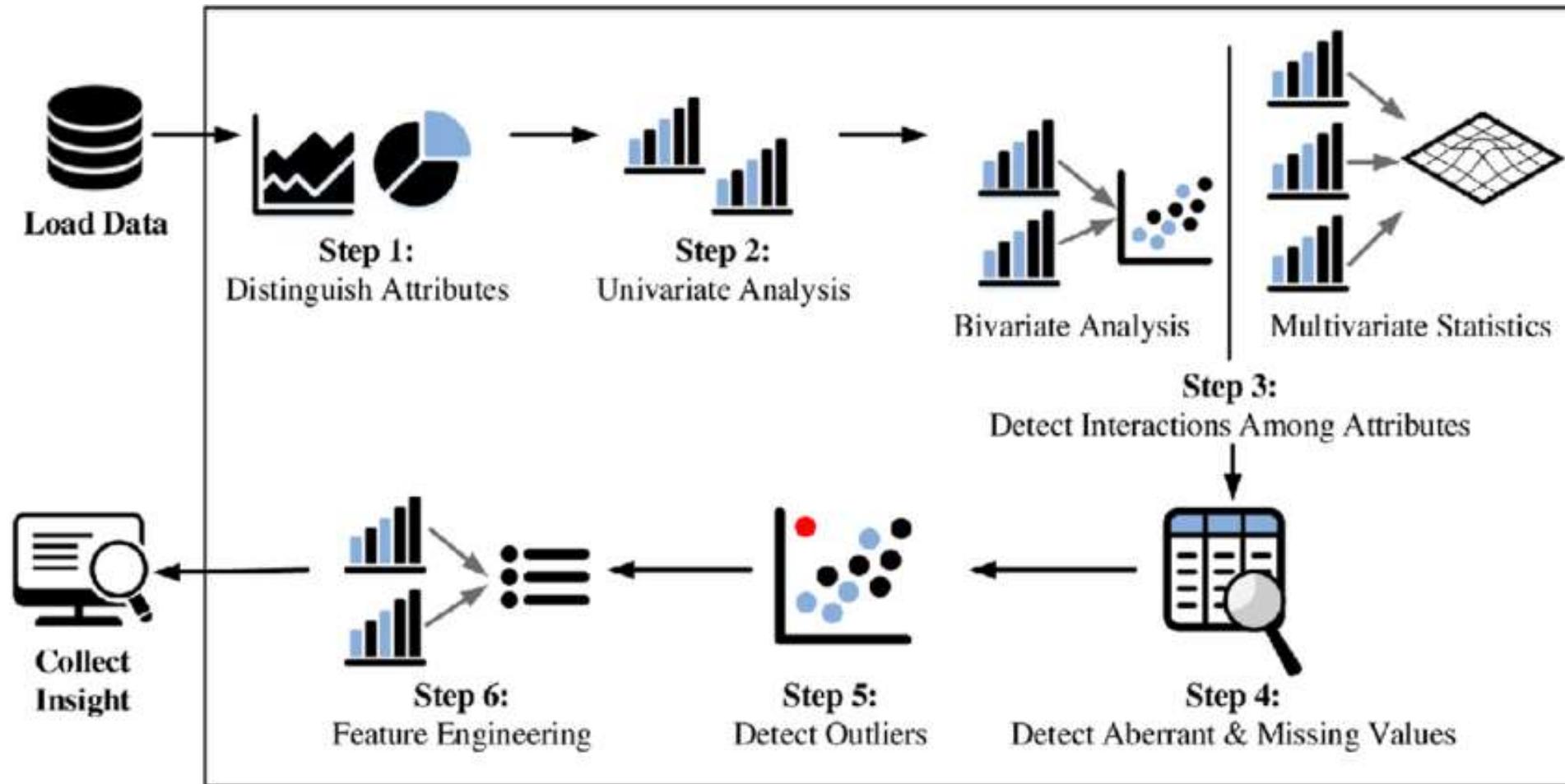
Meat	Obesity	x	y	xy	x^2	y^2
6.1244	4.5	27.5598	37.50828	20.25		
8.7428	22.3	194.9644	76.43655	497.29		
3.8961	26.6	103.6363	15.1796	707.56		
11.0268	6.8	74.98224	121.5903	46.24		
14.3202	19.1	273.5158	205.0681	364.81		
19.2693	28.5	549.1751	371.3059	812.25		
10.8165	20.9	226.0649	116.9967	436.81		
11.6002	30.4	352.6461	134.5646	924.16		
8.1099	21.9	177.6068	65.77048	479.61		
11.9993	19.9	238.7861	143.9832	396.01		
17.4941	32.1	561.5606	306.0435	1030.41		
1.8407	3.4	6.25838	3.388176	11.56		
13.1382	24.8	325.8274	172.6123	615.04		
11.5636	26.6	307.5918	133.7168	707.56		
5.6817	24.5	139.2017	32.28171	600.25		
10.3435	22.4	231.6944	106.988	501.76		
3.2849	8.2	26.93618	10.79057	67.24		
21.1476	18.7	395.4601	447.221	349.69		
190.3998	361.6	4213.468	2501.446	8568.5		

Correlation coefficient and scatter plot



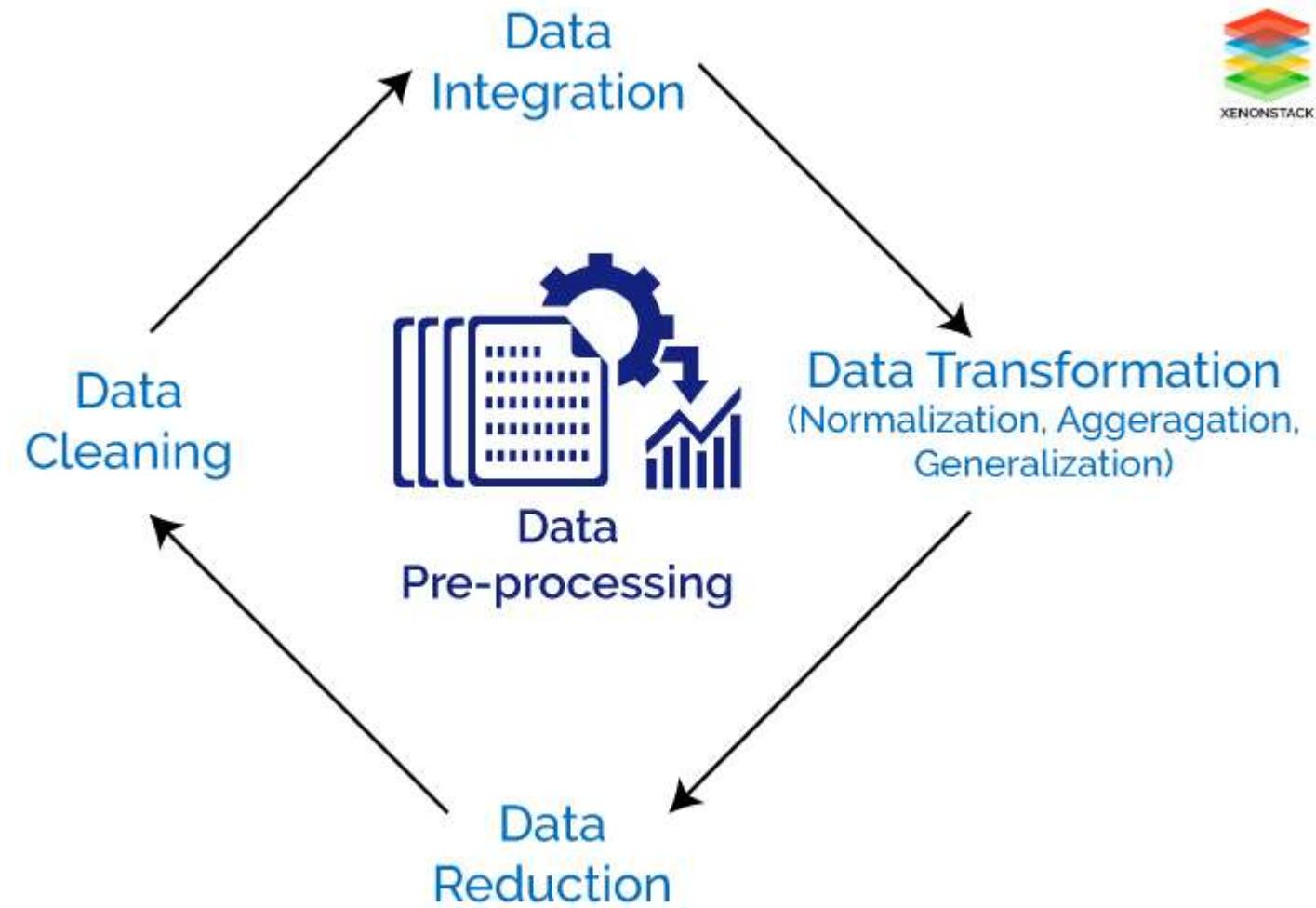
- The range of the correlation coefficient is from -1 to 1.
- If there is a strong positive linear relationship between the variables, the value of r will be close to 1.
- If there is a strong negative linear relationship, the value of r will be close to -1.
- When there is no linear relationship between the variables or only a weak one, the value of r will be close to 0

EDA Process Overview



Data Preparation

- Data Exploration
- Data Cleaning
 - missing data
 - noisy data, outliers
 - duplication
- Data Integration
- Data Transformation
 - Normalization
 - Aggregation
 - Generalization

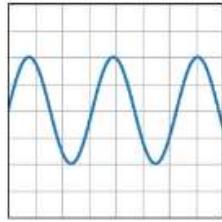


matplotlib

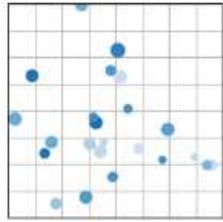
- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Basic

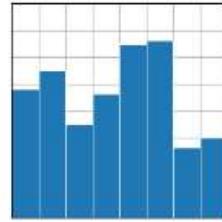
Basic plot types, usually y versus x.



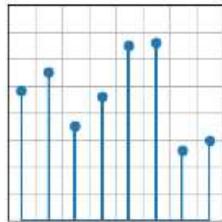
`plot(x, y)`



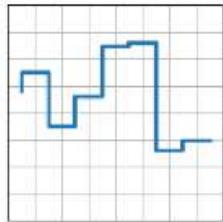
`scatter(x, y)`



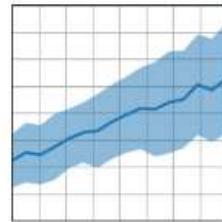
`bar(x, height)`



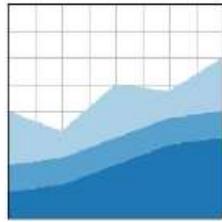
`stem(x, y)`



`step(x, y)`



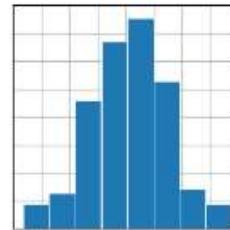
`fill_between(x, y1, y2)`



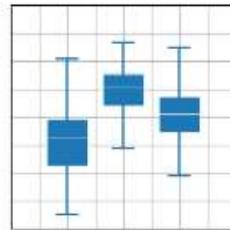
`stackplot(x, y)`

Statistics plots

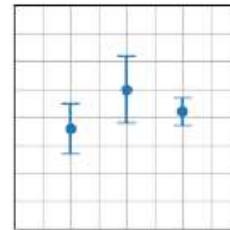
Plots for statistical analysis.



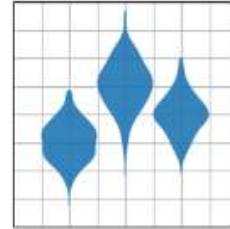
`hist(x)`



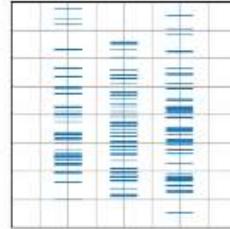
`boxplot(X)`



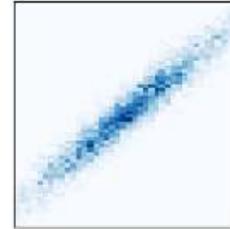
`errorbar(x, y, yerr, xerr)`



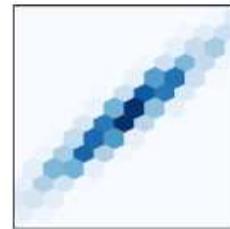
`violinplot(D)`



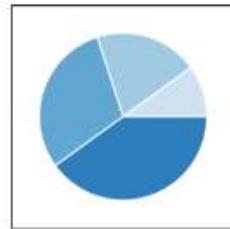
`eventplot(D)`



`hist2d(x, y)`



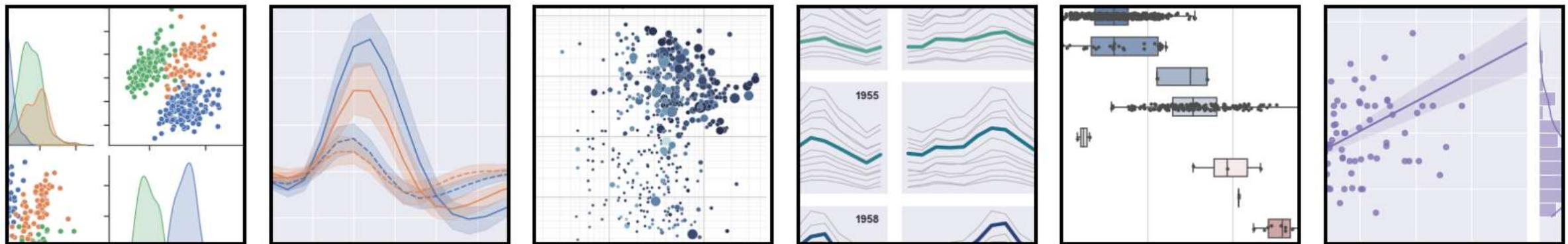
`hexbin(x, y, C)`



`pie(x)`



- Seaborn is a Python data visualization library based on matplotlib.
- It provides a high-level interface for drawing attractive and informative statistical graphics



Data Exploration

Load Data

- โหลดข้อมูลเข้าสู่โปรแกรม Python โดยใช้ Pandas ซึ่งเป็นไลบรารีที่ช่วยในการจัดการข้อมูลและการวิเคราะห์ข้อมูลในรูปแบบตาราง (DataFrame)

python

 Copy code

```
import pandas as pd

# โหลดข้อมูลจากไฟล์ CSV
data = pd.read_csv('data.csv')
```

Data Exploration

python

 Copy code

```
# แสดงตัวอย่างข้อมูลหัวตาราง  
data.head()
```

```
# แสดงข้อมูลเกี่ยวกับตาราง  
data.info()
```

```
# สรุปสถิติเบื้องต้นของข้อมูล  
data.describe()
```

Data type

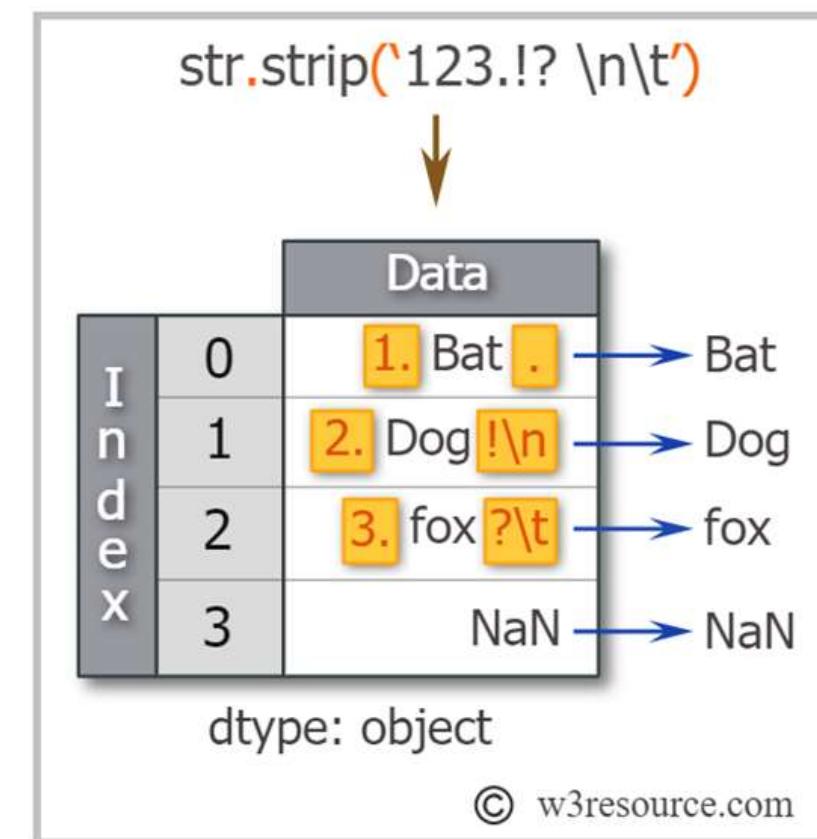
Datatype	Example	Python data type
Text data	First name, last name, address ...	str
Integers	# Subscribers, # products sold ...	int
Decimals	Temperature, \$ exchange rates ...	float
Binary	Is married, new customer, yes/no, ...	bool
Dates	Order dates, ship dates ...	datetime
Categories	Marriage status, gender ...	category

Strings to integers

```
SalesOrderID  Revenue
0            0  39545$
1            1  31465$
2            2  13426$
```

```
Sale_data.dtypes
```

```
SalesOrderID      int64
Revenue          object
dtype: object
```



Split

```
prefix      name      surname      name_full
0  mr.      xxx       yyy       mr. xxx yyy
1  ms.      abcd      efghijk    ms. abcd efghijk
2  miss     dfhjdj    fhdfhdhdf miss dfhjdj fhdfhdhdf
```

```
#Split
```

```
mydata[ "name_full" ].str.split(expand=True)
```

	0	1	2
0	mr.	xxx	yyy
1	ms.	abcd	efghijk
2	miss	dfhjdj	fhdfhdhdf

Replace

	prefix	name	surname
0	นส.	xxx	yyy
1	นางสาว	abcd	efghijk
2	นส.	dfhjдж	fhdfhdhdf

```
mydata['prefix']=mydata['prefix'].str.replace("นส.","นางสาว")
mydata['name']=mydata['name'].str.replace("f","_")
mydata['surname']=mydata['surname'].str.replace("f","_")
print(mydata)
```

	prefix	name	surname
0	นางสาว	xxx	yyy
1	นางสาว	abcd	e_ghijk
2	นางสาว	d_hjдж	_hd_hdhd_

Change type

```
  sex   name  surname
0    1     xxx      yyy
1    2    abcd     efghijk
2    1  dfhjdj  fhdfhdfhdf
```

```
mydata['sex'].describe()
```

```
count      3.000000
mean      1.333333
std       0.577350
min      1.000000
25%      1.000000
50%      1.000000
75%      1.500000
max      2.000000
Name: sex, dtype: float64
```

```
# Convert to categorical
mydata['sex'] = mydata['sex'].astype('category')
mydata['sex'].describe()
```

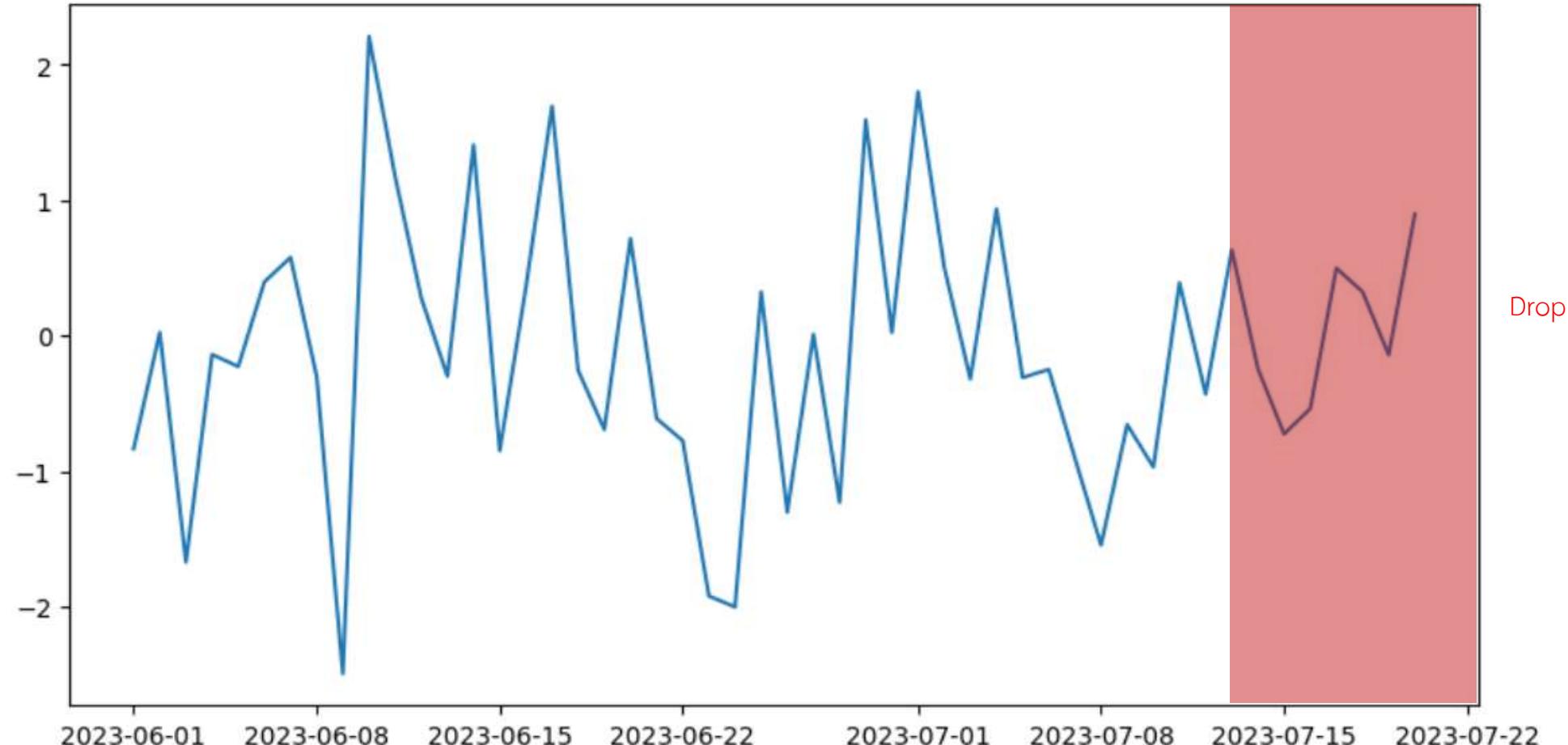
count	3
unique	2
top	1
freq	2
Name: sex, dtype: int64	

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype  
 ---  --       --           --      
 0   sex      3 non-null    category
 1   name     3 non-null    object  
 2   surname  3 non-null    object  
dtypes: category(1), object(2)
```

Date Range Problem

```
plt.figure(figsize=(10,5))
plt.plot(df['date'],df['x']);
```



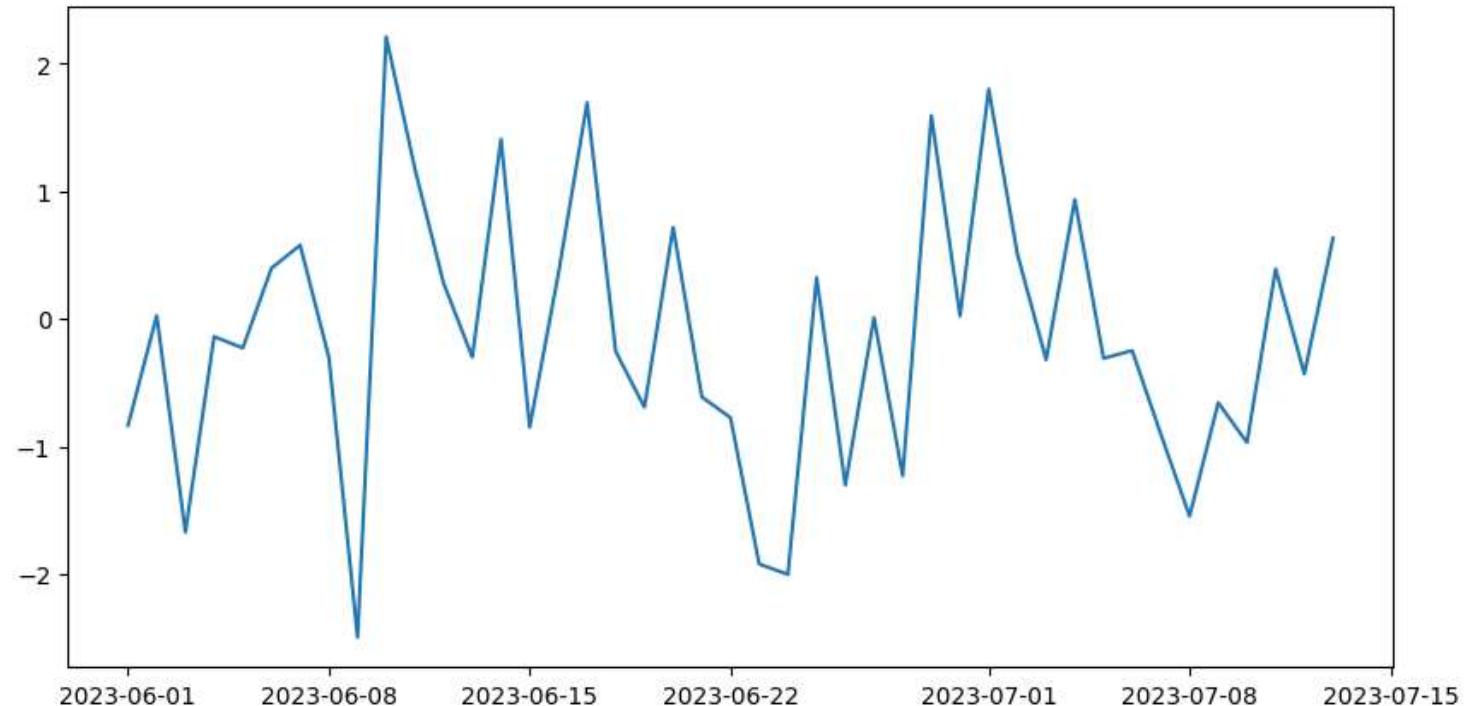
Date Range Problem

```
import datetime as dt
today_date = pd.Timestamp('today')
print(today_date)
```

2023-07-13 02:28:30.460829

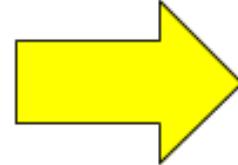
```
# Drop values using filtering
df2 = df[df['date'] < today_date]
```

```
plt.figure(figsize=(10,5))
plt.plot(df2['date'],df2['x']);
```



Categorical Data

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Data Cleaning

Data Cleaning

python

 Copy code

```
# ตรวจสอบข้อมูลที่หายไป
data.isnull().sum()

# ลบข้อมูลที่หายไป
data = data.dropna()

# ตรวจสอบและลบข้อมูลที่ซ้ำกัน
data.duplicated().sum()
data = data.drop_duplicates()
```

Missing data



Occurs when no data value is stored for a variable in an observation

Can be represented as NA , nan , 0 ,

Technical error

Human error

Breast Cancer Wisconsin Data Set

- University of Wisconsin Hospitals Madison, Wisconsin, USA
- [https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+\(original\)](https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original))
- Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

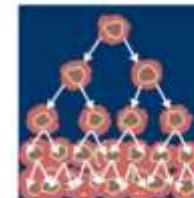


Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Breast Cancer Wisconsin (Original) Data Set

[Download Data Folder](#) [Data Set Description](#)

Abstract: Original Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	Number of Instances:	699	Area:	Life
Attribute Characteristics:	Integer	Number of Attributes:	10	Date Donated:	1992-07-15
Associated Tasks:	Classification	Missing Values?	Yes	Number of Web Hits:	502764

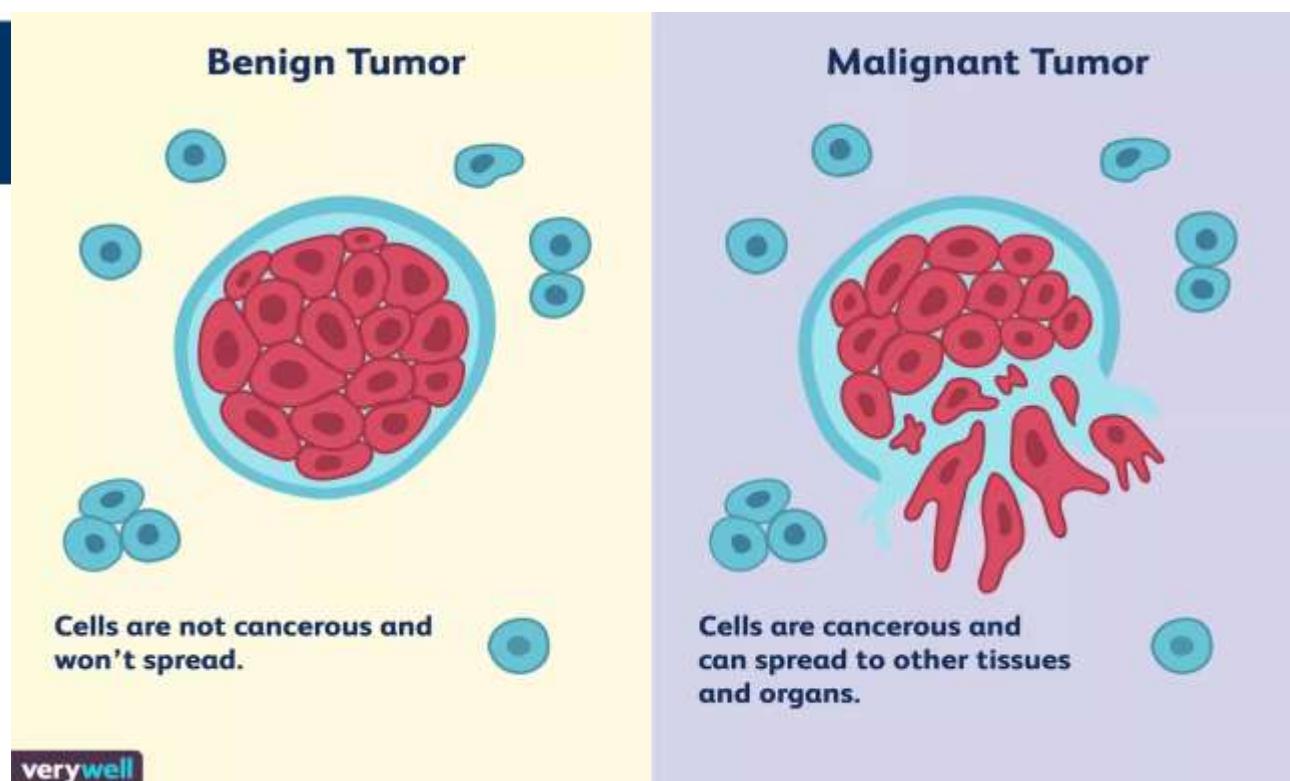
Source:

Creator:

Dr. William H. Wolberg (physician)
University of Wisconsin Hospitals
Madison, Wisconsin, USA

Donor:

University of Wisconsin Hospitals



Breast Cancer Wisconsin Data Example

```
#load Data "breast_cancer_data.csv"
mydata = pd.read_csv('breast_cancer_data.csv')
print(mydata)
```

```
  patient_id  clump_thickness  ...  class  doctor_name
0      1000025            5.0  ...  benign  Dr. Doe
1      1002945            5.0  ...  benign  Dr. Smith
2      1015425            3.0  ...  benign  Dr. Lee
3      1016277            6.0  ...  benign  Dr. Smith
4      1017023            4.0  ...  benign  Dr. Wong
..        ...
694     776715            3.0  ...  benign  Dr. Lee
695     841769            2.0  ...  benign  Dr. Smith
696     888820            5.0  ...  malignant  Dr. Lee
697     897471            4.0  ...  malignant  Dr. Lee
698     897471            4.0  ...  malignant  Dr. Wong
[699 rows x 12 columns]
```

```
mydata.info()
```

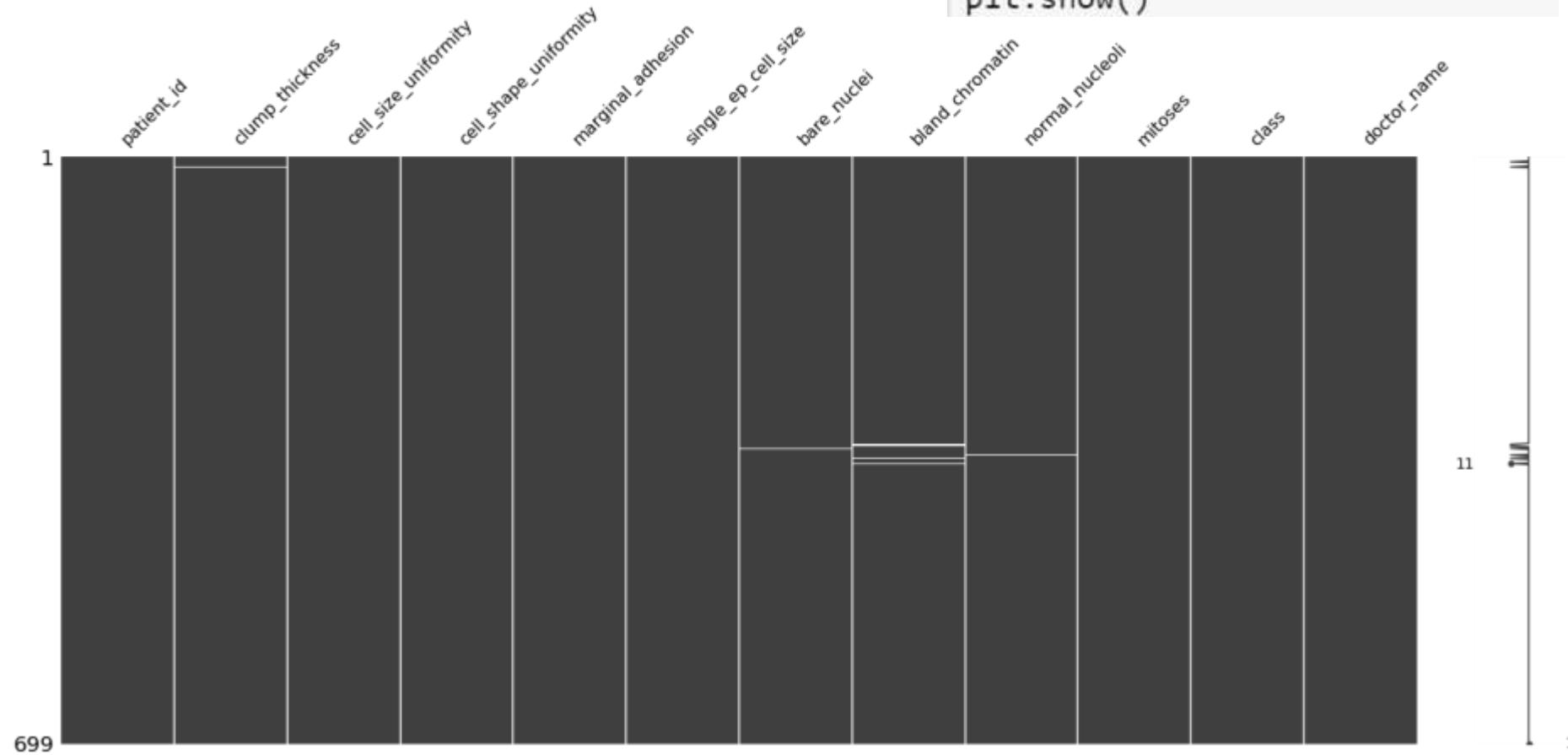
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   patient_id      699 non-null    int64  
 1   clump_thickness  698 non-null    float64 
 2   cell_size_uniformity  698 non-null  float64 
 3   cell_shape_uniformity  699 non-null  int64  
 4   marginal_adhesion  699 non-null  int64  
 5   single_ep_cell_size  699 non-null  int64  
 6   bare_nuclei       697 non-null    object  
 7   bland_chromatin   695 non-null    float64 
 8   normal_nucleoli  698 non-null    float64 
 9   mitoses          699 non-null    int64  
 10  class            699 non-null    object  
 11  doctor_name      699 non-null    object  
dtypes: float64(4), int64(5), object(3)
```

Breast Cancer Wisconsin Data Example

```
# Get summary of missingness
mydata.isna().sum()
```

patient_id	0
clump_thickness	1
cell_size_uniformity	1
cell_shape_uniformity	0
marginal_adhesion	0
single_ep_cell_size	0
bare_nuclei	2
bland_chromatin	4
normal_nucleoli	1
mitoses	0
class	0
doctor_name	0
dtype: int64	

```
import missingno as msno
import matplotlib.pyplot as plt
# Visualize missingness
msno.matrix(mydata)
plt.show()
```



How to deal with missing data?

Simple approaches:

1. Drop missing data
2. Impute with statistical measures (mean, median, mode..)

More complex approaches:

1. Imputing using an algorithmic approach
2. Impute with machine learning models

Drop missing values

```
# Drop missing values
mydata_dropped = mydata.dropna(subset = ['bland_chromatin'])
mydata_dropped.shape
```

```
(695, 12)
```

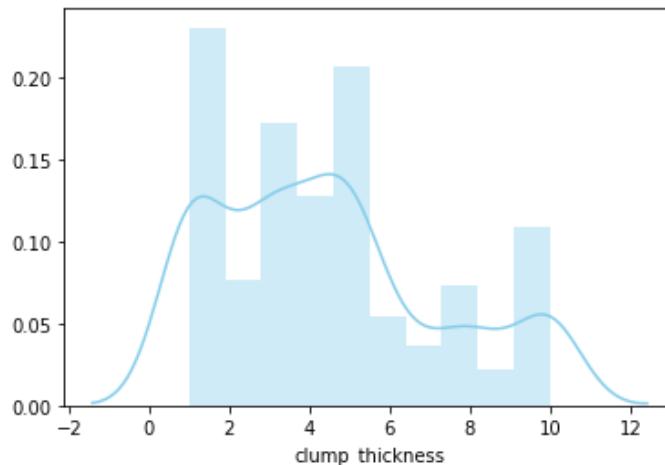
```
mydata.shape
```

```
(699, 12)
```

```
mydata_dropped.isna().sum()
```

patient_id	0
clump_thickness	1
cell_size_uniformity	1
cell_shape_uniformity	0
marginal_adhesion	0
single_ep_cell_size	0
bare_nuclei	2
bland_chromatin	0
normal_nucleoli	1
mitoses	0
class	0
doctor_name	0
dtype: int64	

Replacing with statistical measures



```
#Replacing with statistical measures
clump_thickness_mene = mydata_dropped['clump_thickness'].mean()
mydata_imputed = mydata_dropped.fillna({'clump_thickness': clump_thickness_mene})
mydata_imputed.isna().sum()
```

```
patient_id          0
clump_thickness    0
cell_size_uniformity 1
cell_shape_uniformity 0
marginal_adhesion 0
single_ep_cell_size 0
bare_nuclei        2
bland_chromatin    0
normal_nucleoli    1
mitoses             0
class               0
doctor_name         0
dtype: int64
```

Fill gaps forward

In [944]: df

Out[944]:

	one	two	three
a	0.059117	1.138469	-2.400634
b	NaN	NaN	NaN
c	-0.280853	0.025653	-1.386071
d	NaN	NaN	NaN
e	0.863937	0.252462	1.500571
f	1.053202	-2.338595	-0.374279
g	NaN	NaN	NaN
h	-2.359958	-1.157886	-0.551865

In [945]: df.fillna(method='pad')

Out[945]:

	one	two	three
a	0.059117	1.138469	-2.400634
b	0.059117	1.138469	-2.400634
c	-0.280853	0.025653	-1.386071
d	-0.280853	0.025653	-1.386071
e	0.863937	0.252462	1.500571
f	1.053202	-2.338595	-0.374279
g	1.053202	-2.338595	-0.374279
h	-2.359958	-1.157886	-0.551865

Duplicate Problem

All columns have the same values

first_name	last_name	address	height	weight
Justin	Saddlemyer	Boulevard du Jardin Botanique 3, Bruxelles	193 cm	87 kg
Justin	Saddlemyer	Boulevard du Jardin Botanique 3, Bruxelles	193 cm	87 kg

What are duplicate values?

Most columns have the same values

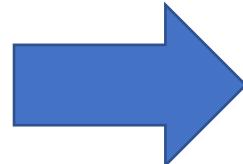
first_name	last_name	address	height	weight
Justin	Saddlemeyer	Boulevard du Jardin Botanique 3, Bruxelles	193 cm	87 kg
Justin	Saddlemeyer	Boulevard du Jardin Botanique 3, Bruxelles	194 cm	87 kg

How to find duplicate values?

#แปลงให้เป็นตัวพิมพ์เล็กให้หมด

```
mydata['name']=mydata['name'].str.lower()  
mydata['surname']=mydata['surname'].str.lower()  
mydata.sort_values(by = 'name')
```

	name	surname	weight
4	Xxx	Xxxx	49
5	ZZZ	ZZZZ	70
3	aaa	aaaa	55
0	xxx	xxxx	50
1	yyy	xxxx	60
2	zzz	zzzz	70



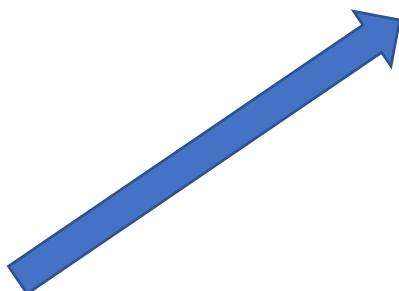
	name	surname	weight
3	aaa	aaaa	55
0	xxx	xxxx	50
4	xxx	xxxx	49
1	yyy	xxxx	60
2	zzz	zzzz	70
5	zzz	zzzz	70

How to find duplicate values?

	name	surname	weight
3	aaa	aaaa	55
0	xxx	xxxx	50
4	xxx	xxxx	49
1	yyy	xxxx	60
2	zzz	zzzz	70
5	zzz	zzzz	70

```
# Get duplicate rows
duplicates = mydata.duplicated()
mydata[duplicates].sort_values(by = 'name')
```

	name	surname	weight
5	zzz	zzzz	70



How to find duplicate values?

```
#subset : List of column names to check for duplication.  
#keep : Whether to keep first ( 'first' ), last ( 'last' ) or all ( False )  
column_names = ['name', 'surname']  
duplicates = mydata.duplicated(subset = column_names, keep = False)  
mydata[duplicates].sort_values(by = 'name')
```

		name	surname	weight
	name	surname	weight	
3	aaa	aaaa	55	
0	xxx	xxxx	50	
4	xxx	xxxx	49	
1	yyy	xxxx	60	
2	zzz	zzzz	70	
5	zzz	zzzz	70	

A red box highlights the second and fourth rows of the original data table, which correspond to the rows with indices 0 and 4 in the sorted duplicates table. A blue arrow points from the red box to the second row of the sorted duplicates table.

How to treat duplicate values?

	name	surname	weight
3	aaa	aaaa	55
0	xxx	xxxx	50
4	xxx	xxxx	49
1	yyy	xxxx	60
2	zzz	zzzz	70
5	zzz	zzzz	70

```
# Drop duplicates
#subset : List of column names to check for duplication
#keep : Whether to keep first ( 'first' ), last ( 'last' )
#inplace : Drop duplicated rows directly inside DataFrame
mydata2=mydata.drop_duplicates()
mydata2.sort_values(by = 'name')
```

	name	surname	weight
3	aaa	aaaa	55
0	xxx	xxxx	50
4	xxx	xxxx	49
1	yyy	xxxx	60
2	zzz	zzzz	70

How to treat duplicate values?

```
column_names = ['name', 'surname']
mydata2=mydata.drop_duplicates(subset = column_names, keep = 'first')
mydata2.sort_values(by = 'name')
```

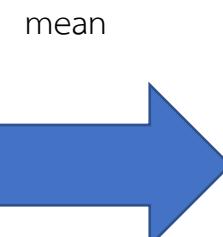
	name	surname	weight
3	aaa	aaaa	55
0	xxx	xxxx	50
4	xxx	xxxx	49
1	yyy	xxxx	60
2	zzz	zzzz	70
5	zzz	zzzz	70

	name	surname	weight
3	aaa	aaaa	55
0	xxx	xxxx	50
1	yyy	xxxx	60
2	zzz	zzzz	70

Group by column names and produce statistical summaries

```
# Group by column names and produce statistical summaries
column_names = ['name', 'surname']
summaries = {'weight': 'mean'}
mydata2 = mydata.groupby(by = column_names).agg(summaries).reset_index()
mydata2.sort_values(by = 'name')
```

	name	surname	weight
3	aaa	aaaa	55
0	xxx	xxxx	50
4	xxx	xxxx	49
1	yyy	xxxx	60
2	zzz	zzzz	70
5	zzz	zzzz	70

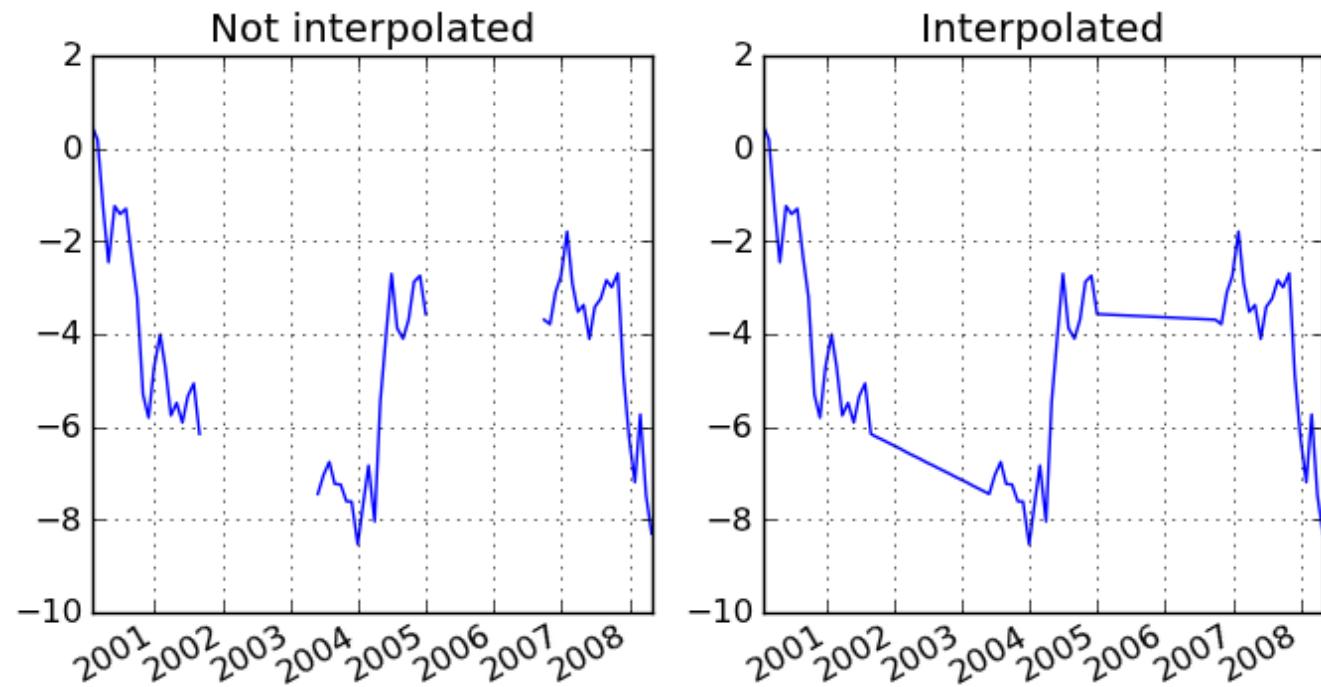


	name	surname	weight
0	aaa	aaaa	55.0
1	xxx	xxxx	49.5
2	yyy	xxxx	60.0
3	zzz	zzzz	70.0

Interpolate

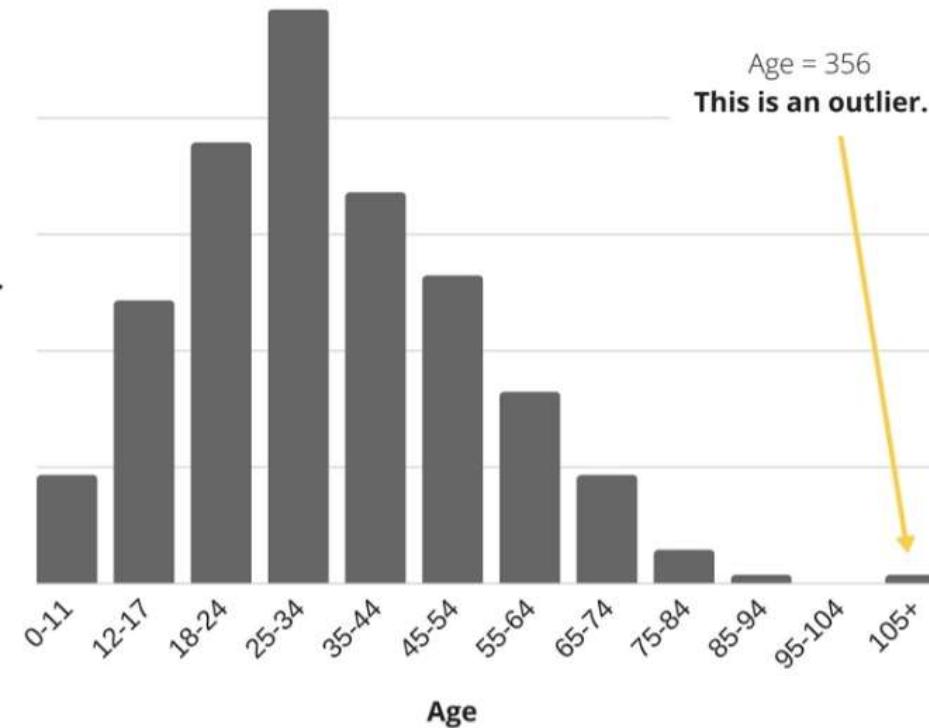
Filling in `NaN` in a `Series` via linear interpolation.

```
>>> s = pd.Series([0, 1, np.nan, 3])
>>> s
0    0.0
1    1.0
2    NaN
3    3.0
dtype: float64
>>> s.interpolate()
0    0.0
1    1.0
2    2.0
3    3.0
dtype: float64
```



Dealing with outlier

- **Outlier** is an observation in a given dataset that lies far from the rest of the observations.

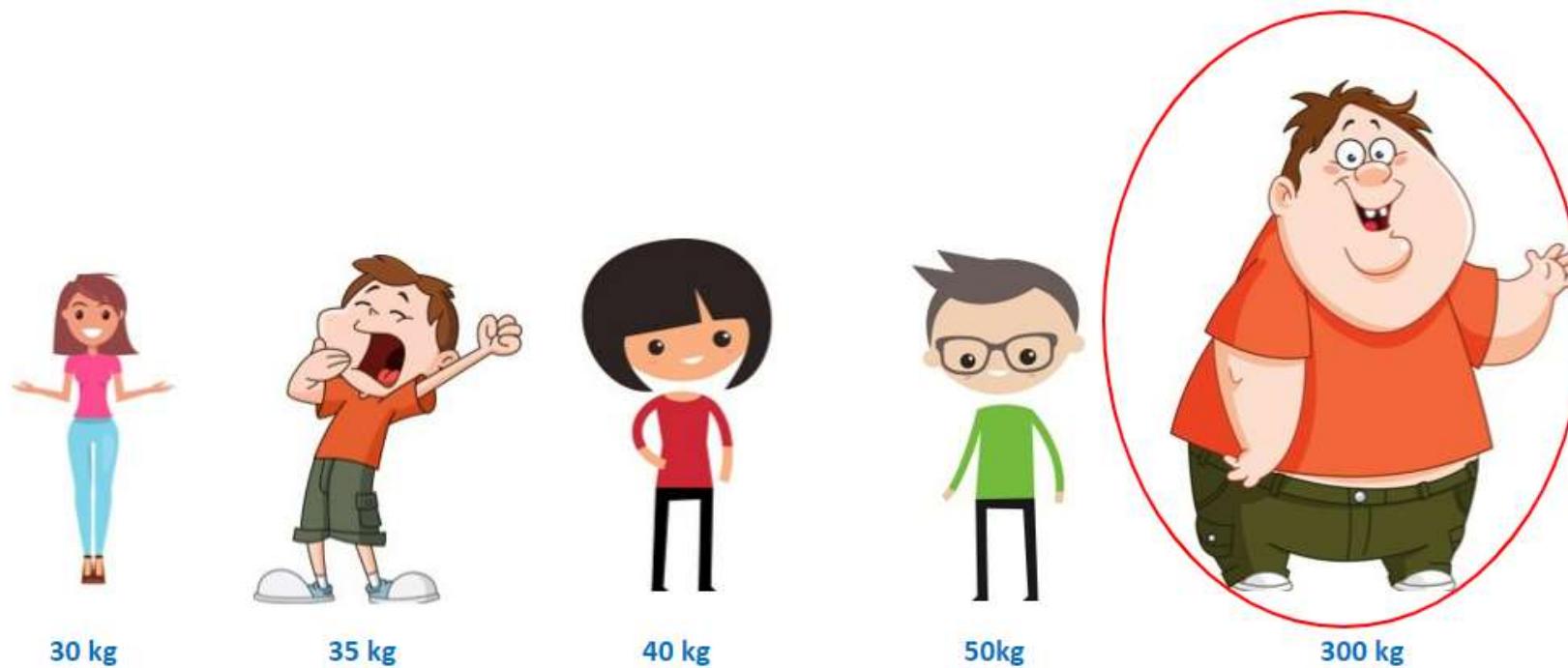


Dealing with outlier

- Affect of outlier data

Average weight of first 4 kids = $(30 + 35 + 40 + 50)/4 = 38.75$ kg

Average weight of all kids = $(30 + 35 + 40 + 50 + 300)/5 = 91$ kg



Data Analysis

python

 Copy code

```
import matplotlib.pyplot as plt
import seaborn as sns

# แสดงการกระจายข้อมูล
sns.scatterplot(x='column1', y='column2', data=data)

# แสดงการแจกแจงความถี่ของข้อมูล
sns.histplot(data['column3'])

# แสดงแผนภูมิแท่ง
sns.barplot(x='category', y='count', data=data)

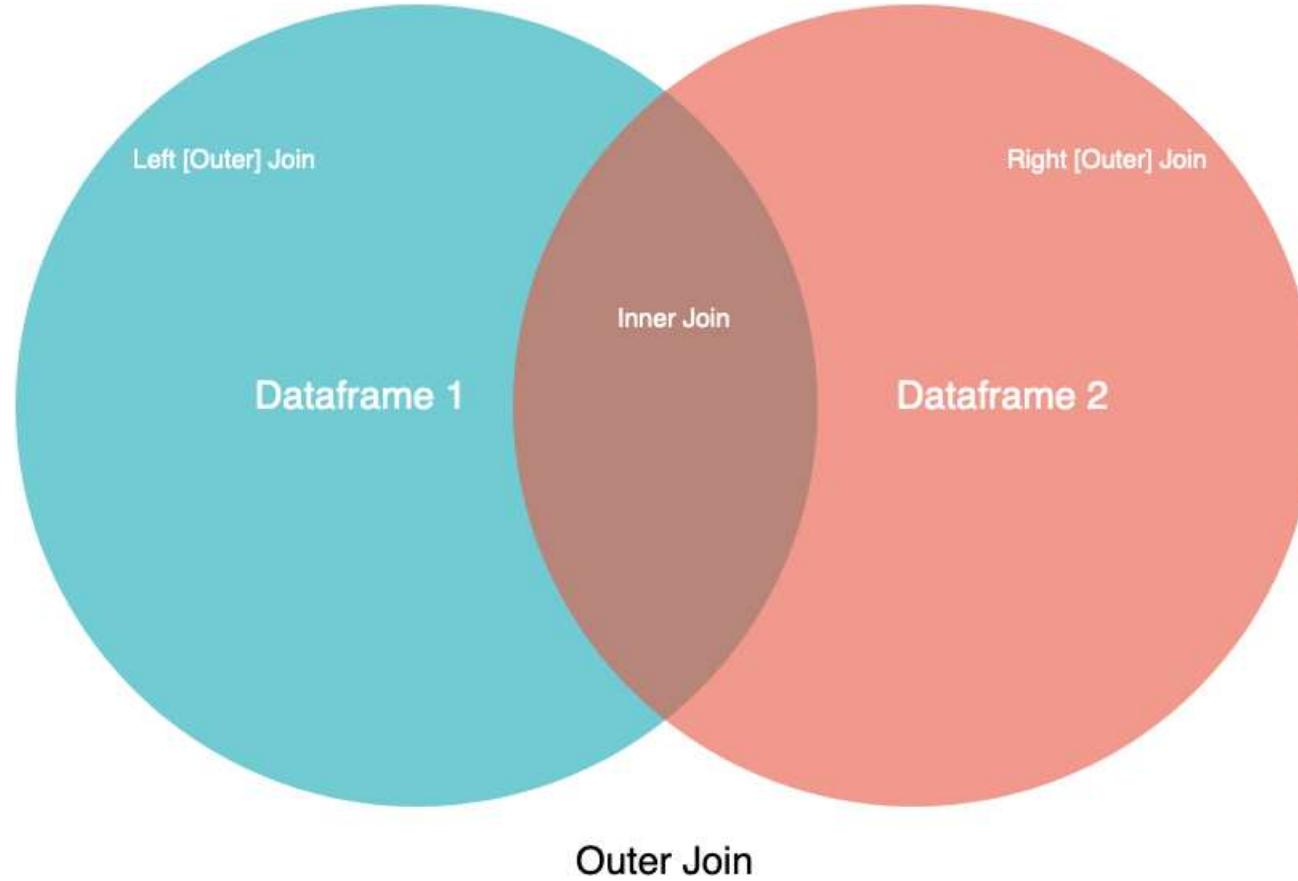
# แสดงแผนภูมิกลุ่ม (box plot)
sns.boxplot(x='category', y='value', data=data)
```

Data Integration

Combine Data with Merge and Join

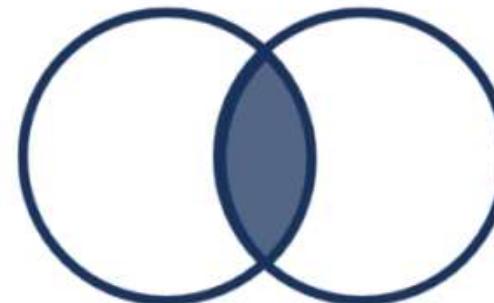
- Both join and merge can be used to **combines** two dataframes but the
- **Concat** method combines two dataframes **on the basis of their indexes**
- **Merge** method is more versatile and **allows us to specify columns** beside the index to join on for both dataframes.

Combine Data

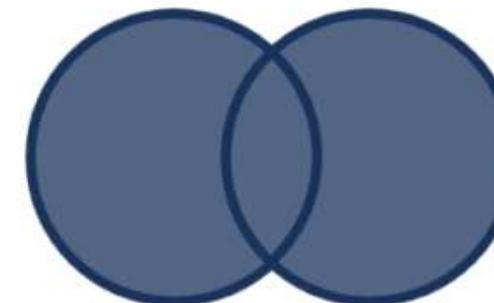


Join Methods

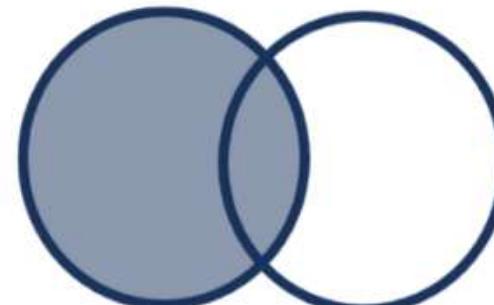
INNER JOIN



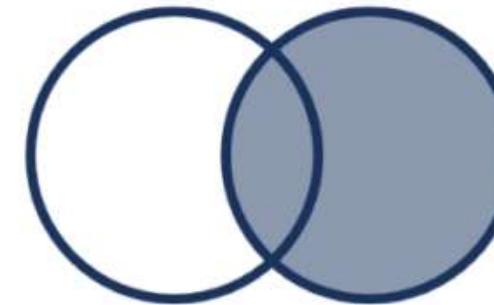
OUTER JOIN



LEFT JOIN



RIGHT JOIN



Concat

	A	B
001	1	-3
002	2	-2
003	3	-1

+

	A	B
004	-6	0

`pd.concat([x, y], axis=0)`



	A	B
001	1	-3
002	2	-2
003	3	-1
004	-6	0

	A	B
001	1	-3
002	2	-2
003	3	-1

+

	C
001	-1.0
002	0.0
003	1.0

`pd.concat([x, w], axis=1)`

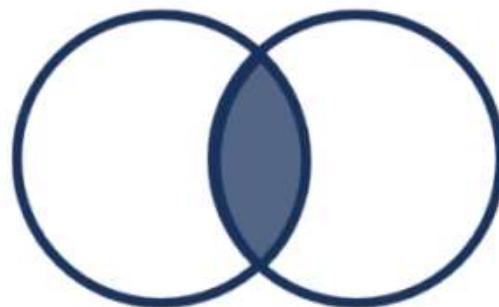


	A	B	C
001	1	-3	-1.0
002	2	-2	0.0
003	3	-1	1.0

Merge

	ID	Value A		ID	Value B
0	PY101	-35	+	0	PY132 List
1	PY243	23		1	PY155 Tuple
2	PY132	36		2	PY101 Array

INNER JOIN

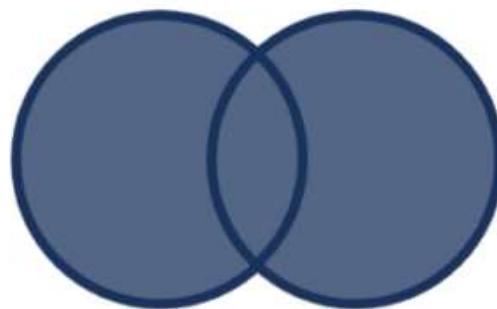


	ID	Value A	Value B
0	PY101	-35	Array
1	PY132	36	List

Merge

	ID	Value A		ID	Value B
0	PY101	-35	+	0	PY132 List
1	PY243	23		1	PY155 Tuple
2	PY132	36		2	PY101 Array

OUTER JOIN

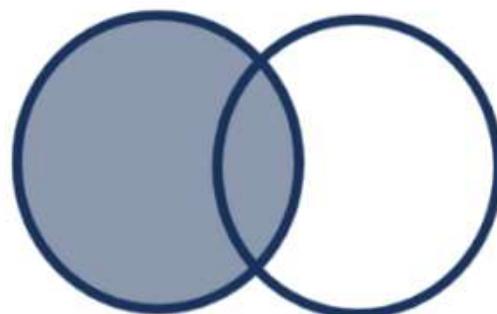


	ID	Value A	Value B
0	PY101	-35.0	Array
1	PY243	23.0	NaN
2	PY132	36.0	List
3	PY155	NaN	Tuple

Merge

	ID	Value A		ID	Value B
0	PY101	-35	+	0	PY132 List
1	PY243	23		1	PY155 Tuple
2	PY132	36		2	PY101 Array

LEFT JOIN

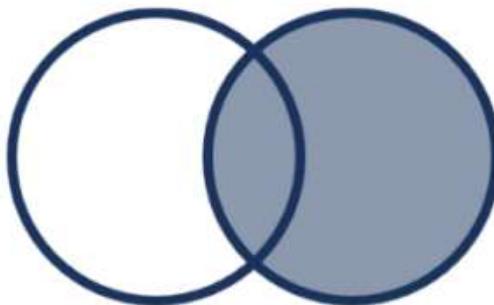


	ID	Value A	Value B
0	PY101	-35	Array
1	PY243	23	NaN
2	PY132	36	List

Merge

	ID	Value A		ID	Value B
0	PY101	-35	+	0	PY132 List
1	PY243	23		1	PY155 Tuple
2	PY132	36		2	PY101 Array

RIGHT JOIN

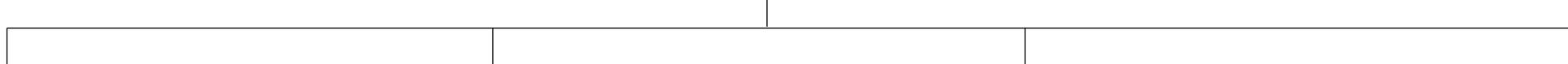


	ID	Value A	Value B
0	PY132	36.0	List
1	PY155	NaN	Tuple
2	PY101	-35.0	Array

Merge

	ID	Value A		ID	Value B
0	PY101	-35	+	0	PY132
1	PY243	23		1	PY155
2	PY132	36		2	PY101

```
x.merge(y, on='ID', how=' ')
```



	ID	Value A	Value B		ID	Value A	Value B		ID	Value A	Value B
0	PY101	-35	Array	0	PY101	-35.0	Array	0	PY101	-35	Array
1	PY132	36	List	1	PY243	23.0	NaN	1	PY243	23	NaN
				2	PY132	36.0	List	2	PY132	36	List
				3	PY155	NaN	Tuple	2	PY101	-35.0	Array

inner

outer

left

right

LAB 1

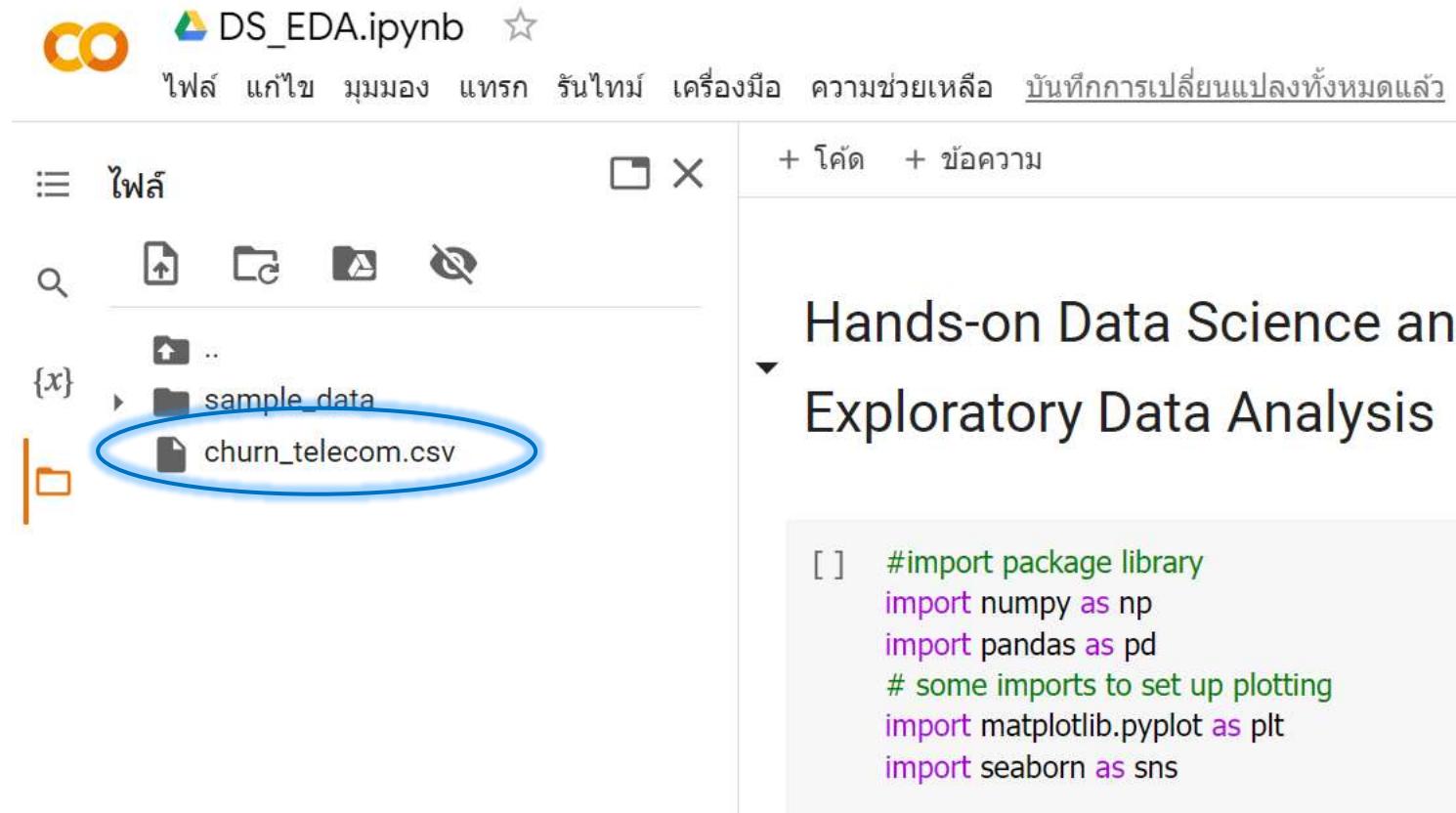
Example: Churn_Telecom

```
# import package library
import numpy as np
import pandas as pd

# some imports to set up plotting
import matplotlib.pyplot as plt
import seaborn as sns
```

[Link Colab](#)

Example: Churn_Telecom



The screenshot shows a Jupyter Notebook interface with the following details:

- File:** DS_EDA.ipynb
- Toolbar:** ไฟล์ แก้ไข นุมมอง แทรก รันไทม์ เครื่องมือ ความช่วยเหลือ บันทึกการเปลี่ยนแปลงทั้งหมดแล้ว
- File Explorer:** ไฟล์
- File List:** sample_data, churn_telecom.csv (The 'churn_telecom.csv' file is highlighted with a blue oval)
- Code Cell:** + โคด + ข้อความ
- Section Title:** Hands-on Data Science and
Exploratory Data Analysis
- Code:**

```
[ ] #import package library
import numpy as np
import pandas as pd
# some imports to set up plotting
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('churn_telecom.csv')
```

Example: Churn_Telecom

```
#ดูตัวอย่างข้อมูล 5 แถวแรก
df.head()
```

State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	
0	LA	117	408	No	No	0	184.5	97	31.37
1	IN	65	415	No	No	0	129.1	137	21.95
2	NY	161	415	No	No	0	332.9	67	56.59
3	SC	111	415	No	No	0	110.4	103	18.77
4	HI	49	510	No	No	0	119.3	117	20.28

Example: Churn_Telecom

#ສ່ວນຂໍ້ອມມາ 3 ແລ້ວ

df.sample(3)

State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	
95	IN	78	415	No	No	0	208.9	119	35.51	252.4	132	21.45
530	HI	171	510	No	No	0	189.8	122	32.27	173.7	85	14.76
531	DE	90	415	No	No	0	198.5	124	33.75	266.6	100	22.66

Example: Churn_Telecom

```
#ดูรายละเอียดข้อมูล  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 667 entries, 0 to 666  
Data columns (total 20 columns):  
 #  Column           Non-Null Count Dtype  
---  
 0  State            667 non-null  object  
 1  Account length  667 non-null  int64  
 2  Area code        667 non-null  int64  
 3  International plan 667 non-null  object  
 4  Voice mail plan 667 non-null  object  
 5  Number vmail messages 667 non-null  int64  
 6  Total day minutes 667 non-null  float64  
 7  Total day calls  667 non-null  int64  
 8  Total day charge 667 non-null  float64  
 9  Total eve minutes 667 non-null  float64  
 10 Total eve calls  667 non-null  int64  
 11 Total eve charge 667 non-null  float64  
 12 Total night minutes 667 non-null  float64  
 13 Total night calls 667 non-null  int64
```

Data Transformation

Data Scaling and Normalization

- Scaling vs. Normalization: What's the difference?
 - in **scaling**, you're changing the *range* of your data, while
 - in **normalization**, you're changing the *shape of the distribution* of your data

Data Scaling

- Data Scaling Methods.

- Simple Feature Scaling

$$x_{new} = \frac{x_{current}}{x_{maximum}}$$

- Min-Max Scaling

$$x_{new} = \frac{x_{current} - x_{minimum}}{x_{maximum} - x_{minimum}}$$

$$0 \leq x_{new} \leq 1$$

- Standard or Z-score Scaling

$$x_{new} = \frac{x_{current} - Mean}{Standard Deviation}$$

$$-3 \leq x_{new} \leq 3$$

Data Scaling

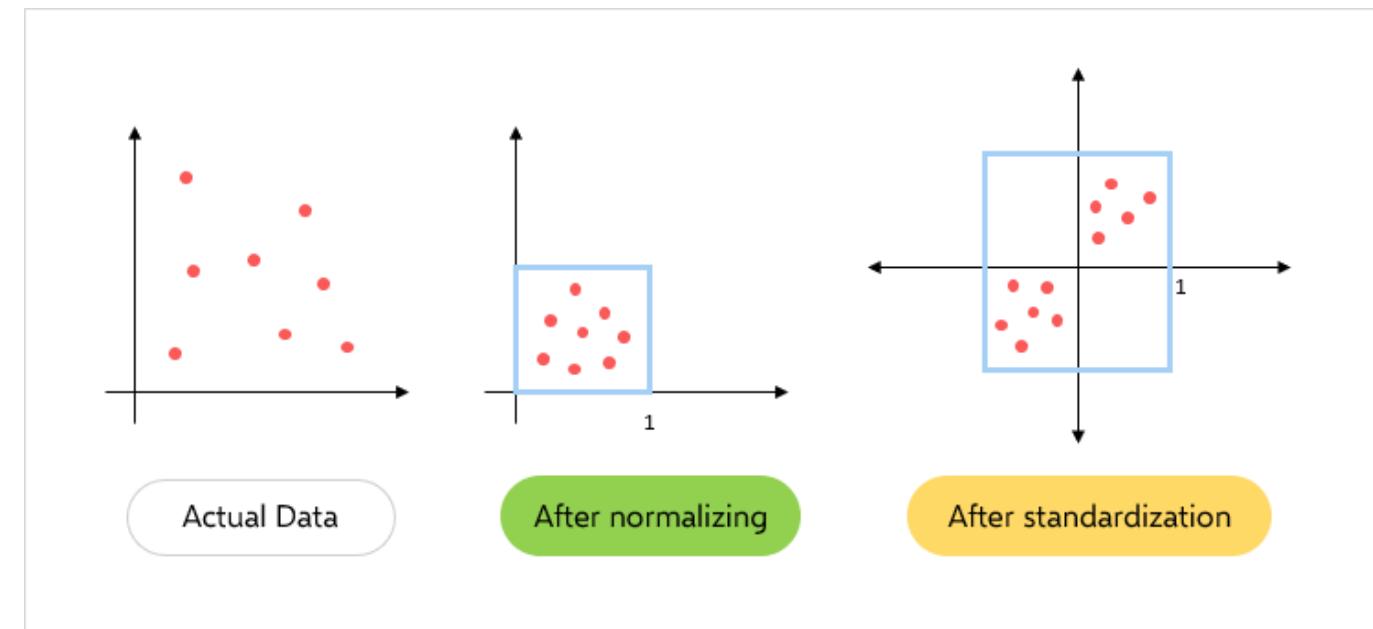
- It consists of putting the data in a **similar range** to be able to compare them.

No	Employee ID	First Name	Last Name	Age	Worked years	Salary	Status	Grade
0	1	1000001	John	Denver	23	1	500	Single Elementary
1	2	1000002	Peter	Hank	30	3	900	Married High School
2	3	1000003	Jack	Sullivan	27	2	900	Married High School
3	4	1000004	Marco	Aurelio	40	8	1500	Married Master Degree
4	5	1000005	Claudia	Perez	35	5	1300	Single Master Degree
5	6	1000006	Sally	Royal	19	1	1400	Single Graduate
6	7	1000007	Peter	Miller	33	4	600	Married Graduate
7	8	1000008	Susan	Gordon	35	10	2000	Married Master Degree

Data Scaling

Standardisation		
	Age	Salary
0	0.758874	7.494733e-01
1	-1.711504	-1.438178e+00
2	-1.275555	-8.912655e-01
3	-0.113024	-2.532004e-01
4	0.177609	6.632192e-16
5	-0.548973	-5.266569e-01
6	0.000000	-1.073570e+00
7	1.340140	1.387538e+00
8	1.630773	1.752147e+00
9	-0.258340	2.937125e-01

Max-Min Normalization		
	Age	Salary
0	0.739130	0.685714
1	0.000000	0.000000
2	0.130435	0.171429
3	0.478261	0.371429
4	0.565217	0.450794
5	0.347826	0.285714
6	0.512077	0.114286
7	0.913043	0.885714
8	1.000000	1.000000
9	0.434783	0.542857



Data Scaling

- Applying Simple Feature Scaling method in Python.

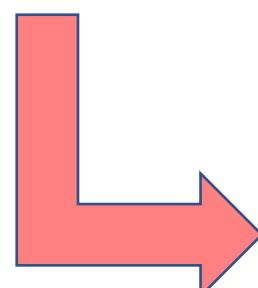
```
df_employees[['Age', 'Worked years', 'Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300

```
df_norm1 = df_employees

df_norm1["Age"] = df_norm1["Age"] / df_norm1["Age"].max()
df_norm1["Worked years"] = df_norm1["Worked years"] / df_norm1["Worked years"].max()
df_norm1["Salary"] = df_norm1["Salary"] / df_norm1["Salary"].max()

df_norm1[['Age', 'Worked years', 'Salary']].head()
```



	Age	Worked years	Salary
0	0.575	0.1	0.25
1	0.750	0.3	0.45
2	0.675	0.2	0.45
3	1.000	0.8	0.75
4	0.875	0.5	0.65

Data Scaling

- Applying Min-Max method in Python.

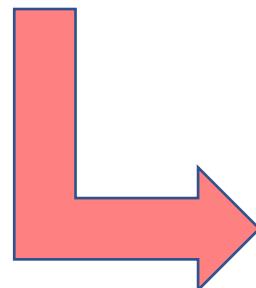
```
df_employees[['Age', 'Worked years', 'Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300

```
df_norm2 = df_employees

df_norm2["Age"] = (df_norm2["Age"] - df_norm2["Age"].min()) / (df_norm2["Age"].max() - df_norm2["Age"].min())
df_norm2["Worked years"] = (df_norm2["Worked years"] - df_norm2["Worked years"].min()) / (df_norm2["Worked years"].max() - df_norm2["Worked years"].min())
df_norm2["Salary"] = (df_norm2["Salary"] - df_norm2["Salary"].min()) / (df_norm2["Salary"].max() - df_norm2["Salary"].min())

df_norm2[['Age', 'Worked years', 'Salary']].head()
```



	Age	Worked years	Salary
0	0.190476	0.000000	0.000000
1	0.523810	0.222222	0.266667
2	0.380952	0.111111	0.266667
3	1.000000	0.777778	0.666667
4	0.761905	0.444444	0.533333

Data Scaling

- Applying Z-score method in Python.

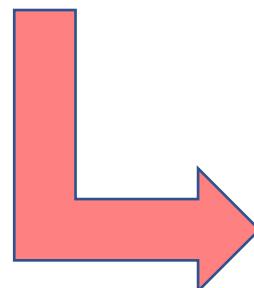
```
df_employees[['Age', 'Worked years', 'Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300

```
df_norm3 = df_employees

df_norm3["Age"] = (df_norm3["Age"] - df_norm3["Age"].mean()) / df_norm3["Age"].std()
df_norm3["Worked years"] = (df_norm3["Worked years"] - df_norm3["Worked years"].mean()) / df_norm3["Worked years"].std()
df_norm3["Salary"] = (df_norm3["Salary"] - df_norm3["Salary"].mean()) / df_norm3["Salary"].std()

df_norm3[['Age', 'Worked years', 'Salary']].head()
```

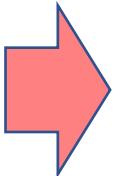


	Age	Worked years	Salary
0	-1.044119	-0.989598	-1.264654
1	-0.036004	-0.380615	-0.471146
2	-0.468053	-0.685106	-0.471146
3	1.404160	1.141844	0.719117
4	0.684078	0.228369	0.322363

Grouping Numerical Data into Classes

- Grouping the data into classes.
- Sales have a range that goes from 100,000 to a little more than 900,000

Sales
150000
651750
563750
706250
640375
519375
870375
926250
676250
103750
567925



Classes	Classes Range
Low	0 - 299,999
Medium	300,000 - 599,999
High	600,000 - 999,999

Grouping into Classes

- Grouping with Python.

```
my_class = np.linspace(min(df_test["Sales"]), max(df_test["Sales"]), 4)
group_names = ["Low", "Medium", "High"]

df_test["Sales Category"] = pd.cut(df_test["Sales"], my_class, labels = group_names, include_lowest = True)

df_test[['Sales', 'Sales Category']].head()
```

	Sales	Sales Category
0	150000.0	Low
2	563750.0	Medium
3	706250.0	High
4	553509.0	Medium
5	519375.0	Medium

Converting Categorical Variables to Numeric Variables

- Converting categorical variables to numeric variables.
- get_dummies()

```
df_dummies = pd.get_dummies(df_test['Payment Type'])
df_test2 = pd.concat([df_test, df_dummies], axis = 1, sort = False)

df_test2.head()
```

Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent	Purchases in thousands	Sales in thousands	Refunds in thousands	Sales Category	Cash	Credit Card	Transfer
10000	Person	Cash	1200000.0	1500000.0	240	Canada	America	120.0	150.000	0.240	Low	1	0	0
10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America	451.0	563.750	0.902	Medium	0	1	0
10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe	565.0	706.250	1.130	High	0	0	1
10004	Person	Transfer	512300.0	553509.0	1024	Argentina	America	512.3	553.509	1.024	Medium	0	0	1
10005	Person	Transfer	415500.0	519375.0	0	Canada	America	415.5	519.375	0.000	Medium	0	0	1

LAB 2

[Link Colab](#)

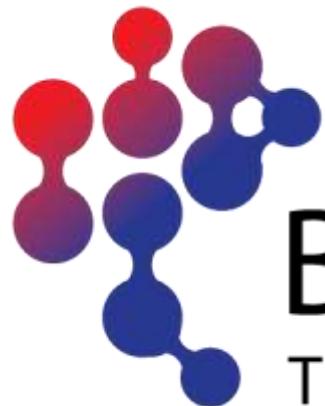
Workshop

car_data



BIG DATA
THAILAND

Follow us on



BIG DATA
THAILAND

Website



Facebook



Blockdit



Twitter

