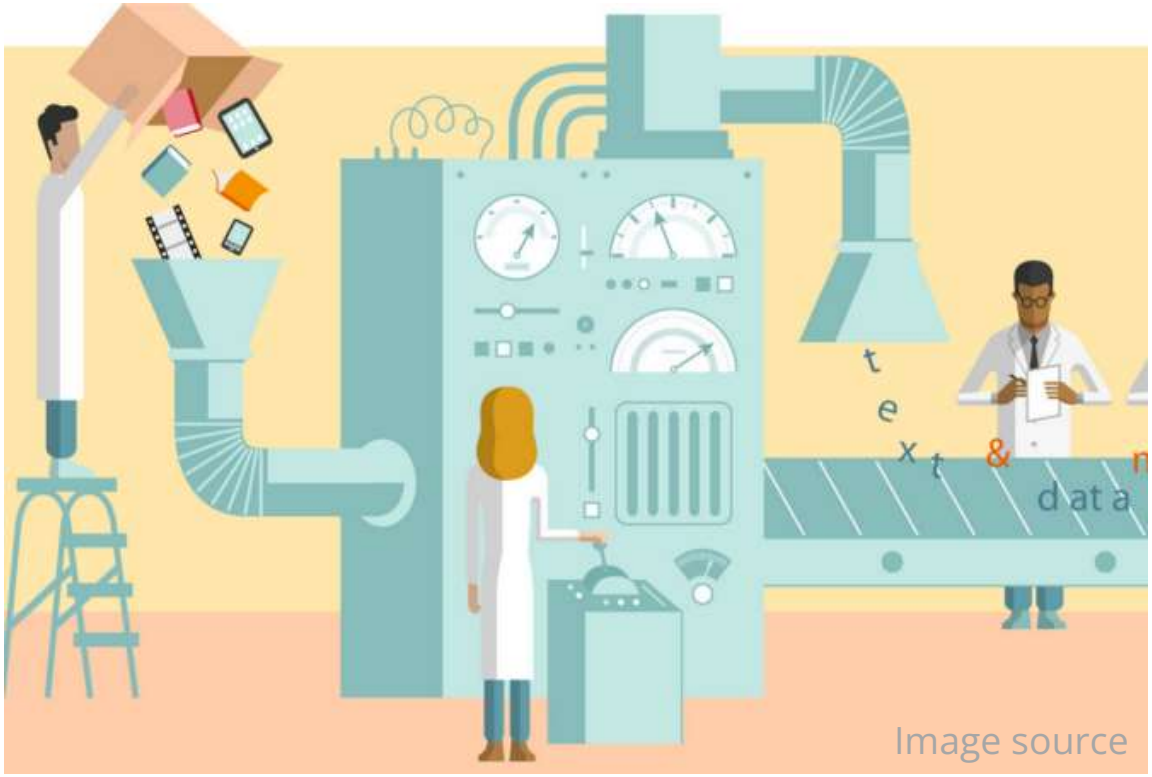


Chapter 5 Data preparation

CSS 341 Introduction to Data Science

Chukiat Worasucheep

Important Notice

การเรียนการสอนหัวข้อนี้ ผ่านทางสื่อออนไลน์ (Online meeting)

และมีการบันทึกภาพและเสียงเพื่อประโยชน์ทางการศึกษาต่อไปในอนาคต.

หากท่านไม่ยินยอมให้มีการเผยแพร่การบันทึกดังกล่าว ขอให้แจ้งให้ผู้สอนทราบภายใน 36 ชั่วโมง.

Learning objectives

- เข้าใจความสำคัญของการเตรียมข้อมูล (data preparation)
- เข้าใจภาพรวมกระบวนการในการเตรียมข้อมูล
- หัดใช้ Python ในการสำรวจข้อมูลเบื้องต้น
- หัดใช้ Python ในการกระทำแนวคอลัมน์บนข้อมูล dataframe
- หัดใช้ Python ในการกระทำแนวแถว (rows) บนข้อมูล dataframe
- หัดใช้ Python ในการรวมและแทนที่ค่าใน dataframe
- หัดใช้ Python ในการแบ่งข้อมูลเป็นช่วง
- หัดใช้ Python ในการสร้างตัวแปรดัมมี (dummy variable) และการเข้ารหัสข้อมูลแบบง่าย

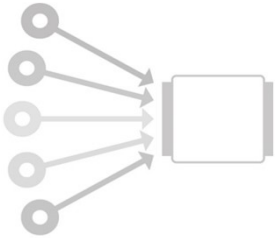
Outline

- What is data preparation process?
- Data inspection
- Data manipulation: column-wise
- Data manipulation: row-wise
- Data combination and data replacement
- Discretization (variable bucketing)
- Data encoding
- Data normalization and standardization

Data science process

Data Preparation 04

06 Data modeling



Data acquisition 02

07 Deployment and maintenance

05 Data visualization

03 Data Exploration

01 Understand business and problems

Why we need data preparation?

- Data in the real world is often not perfect e.g.
  - Inconsistent: กรุงเทพมหานคร or กรุงเทพฯ or กทม.
  - Incomplete: missing values or incorrect
  - Noisy: outlier removal
  - Different formats: dd/mm/yyyy, d-mmm-yy, etc.
  - Dispersed: data integration (merge data)
  - Not meaningful as it should be. Data transformation or generation
- Data should be formatted for further modeling tools (more *efficient* or *faster*)
  - Normalize to [0, 1] or standardize (Z-score) with its mean and standard deviation
  - Change nominal data (e.g. yes/no) to {0, 1} value

Common steps of data preparation in data science

1. Import data and tools
   - Import libraries
   - Read from csv
   - Connect to a database
2. Data inspection
   - Observe data characteristics
   - Deal with missing values
   - Handle (find and remove) duplicate data
   - Handle outliers
3. Data manipulation
   - Column operations (select, rename, drop, sort by column, create new variables)
   - Row operations (slice, filter)
   - Combine dataframes
     - merge, concat, append, join
   - Replace values in a dataframe with condition
   - Discretization (bucket variables)
   - Data encoding
   - Normalization and standardization

1. Essential and useful libraries

1. Pandas
   - Nice syntax and high-level abstraction with data structures
   - Rich functionalities of data manipulation tools (extract, clean, etc.)
2. Numpy
   - Powerful and fast N-dimensional arrays (implemented in C)
   - Array-oriented computing
   - Used by Pandas and many other libraries
3. Matplotlib
   - Dozens of chart types and platform independent
   - a MATLAB replacement, with the advantage of being free and open source
   - Low memory consumption and better runtime behavior
4. Scikit-learn
   - Comprehensive machine learning tools for Python
5. Scipy
   - High-level commands for data manipulation and visualization
   - Multidimensional image processing and optimization
6. Keras
   - High-level tools for deep learning and neural network modules
7. BeautifulSoup and Scrapy
   - Web scraping and web crawling

Install program jupyter

d:\>pip install jupyter

Command Prompt for Notebooks

```
D:\notebooks>pip install jupyter
Defaulting to user installation because normal site-packages is not writeable
Collecting jupyter
  Using cached jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Requirement already satisfied: qtconsole in c:\users\chuki\appdata\roaming\python\python38\site-packages (from jupyter) (5.1.1)
Requirement already satisfied: notebook in c:\users\chuki\appdata\roaming\python\python38\site-packages (from jupyter) (6.4.10)
Requirement already satisfied: ipykernel in c:\users\chuki\appdata\roaming\python\python38\site-packages (from jupyter) (6.2.0)
Requirement already satisfied: nbconvert in c:\users\chuki\appdata\roaming\python\python38\site-packages (from jupyter) (6.4.4)
Requirement already satisfied: jupyter-console in c:\users\chuki\appdata\roaming\python\python38\site-packages (from jupyter) (6.4.0)
Requirement already satisfied: ipywidgets in c:\users\chuki\appdata\roaming\python\python38\site-packages (from jupyter) (7.6.3)
Requirement already satisfied: jupyter-client<8.0 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipykernel->jupyter) (7.0.1)
Requirement already satisfied: matplotlib-inline<0.2.0,>=0.1.0 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipykernel->jupyter) (0.1.2)
Requirement already satisfied: tornado<7.0,>=4.2 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipykernel->jupyter) (6.1)
Requirement already satisfied: traitlets<6.0,>=4.1.0 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipykernel->jupyter) (5.0.5)
Requirement already satisfied: debugpy<2.0,>=1.0.0 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipykernel->jupyter) (1.4.1)
Requirement already satisfied: ipython<8.0,>=7.23.1 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipykernel->jupyter) (7.26.0)
Requirement already satisfied: widgetsnbextension~=3.5.0 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipywidgets->jupyter) (3.5.1)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipywidgets->jupyter) (1.0.0)
Requirement already satisfied: nbformat>=4.2.0 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from ipywidgets->jupyter) (5.1.3)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from packaging->bleach->nbconvert->jupyter) (2.4.7)

Installing collected packages: jupyter
Successfully installed jupyter-1.0.0
WARNING: You are using pip version 22.0.3; however, version 22.2.2 is available.
You should consider upgrading via the 'c:\program files\python38\python.exe -m pip install --upgrade pip' command.

D:\notebooks>
```

Install program Jupyter lab

d:\>pip install jupyterlab

Administrator: Command Prompt

```
C:\Windows\system32>pip install jupyterlab
Requirement already satisfied: jupyterlab in c:\users\chuki\appdata\roaming\python\python38\s
Requirement already satisfied: packaging in c:\users\chuki\appdata\roaming\python\python38\s
Requirement already satisfied: jupyterlab-server~=2.3 in c:\users\chuki\appdata\roaming\pytho
) (2.7.1)
Requirement already satisfied: tornado>=6.1.0 in c:\users\chuki\appdata\roaming\python\pytho
Requirement already satisfied: jinja2>=2.1 in c:\users\chuki\appdata\roaming\python\python38
Requirement already satisfied: nbclassic~=0.2 in c:\users\chuki\appdata\roaming\python\pytho
)
Requirement already satisfied: jupyter-core in c:\users\chuki\appdata\roaming\python\python3
Requirement already satisfied: jupyter-server~=1.4 in c:\users\chuki\appdata\roaming\python\
1.10.2)
Requirement already satisfied: ipython in c:\users\chuki\appdata\roaming\python\python38\site
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\chuki\appdata\roaming\python\pytho
terlab) (2.0.1)
Requirement already satisfied: ipython-genutils in c:\users\chuki\appdata\roaming\python\pyth
1.4->jupyterlab) (0.2.0)
```

Install commonly used packages

d:\>pip install numpy pandas matplotlib sklearn scipy

```
Administrator: Command Prompt
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>pip install numpy pandas matplotlib sklearn scipy
Requirement already satisfied: numpy in c:\users\chuki\appdata\roaming\python\python38\site-packages (1.19.5)
Requirement already satisfied: pandas in c:\users\chuki\appdata\roaming\python\python38\site-packages (1.3.1)
Requirement already satisfied: matplotlib in c:\users\chuki\appdata\roaming\python\python38\site-packages (3.4.2)
Requirement already satisfied: sklearn in c:\users\chuki\appdata\roaming\python\python38\site-packages (0.0)
Requirement already satisfied: scipy in c:\users\chuki\appdata\roaming\python\python38\site-packages (1.5.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\chuki\appdata\roaming\python\python38\site-packa
Requirement already satisfied: pytz>=2017.3 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\chuki\appdata\roaming\python\python38\site-packages (
Requirement already satisfied: pillow>=6.2.0 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from
Requirement already satisfied: cycler>=0.10 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\chuki\appdata\roaming\python\python38\site-packages (f
Requirement already satisfied: scikit-learn in c:\users\chuki\appdata\roaming\python\python38\site-packages (from
Requirement already satisfied: six in c:\program files\python38\lib\site-packages (from cycler>=0.10->matplotlib)
Requirement already satisfied: joblib>=0.11 in c:\users\chuki\appdata\roaming\python\python38\site-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\chuki\appdata\roaming\python\python38\site-package
WARNING: You are using pip version 22.0.3; however, version 22.0.4 is available.
You should consider upgrading via the 'c:\program files\python38\python.exe -m pip install --upgrade pip' command.

C:\Windows\system32>
```

Common startup code to import libraries

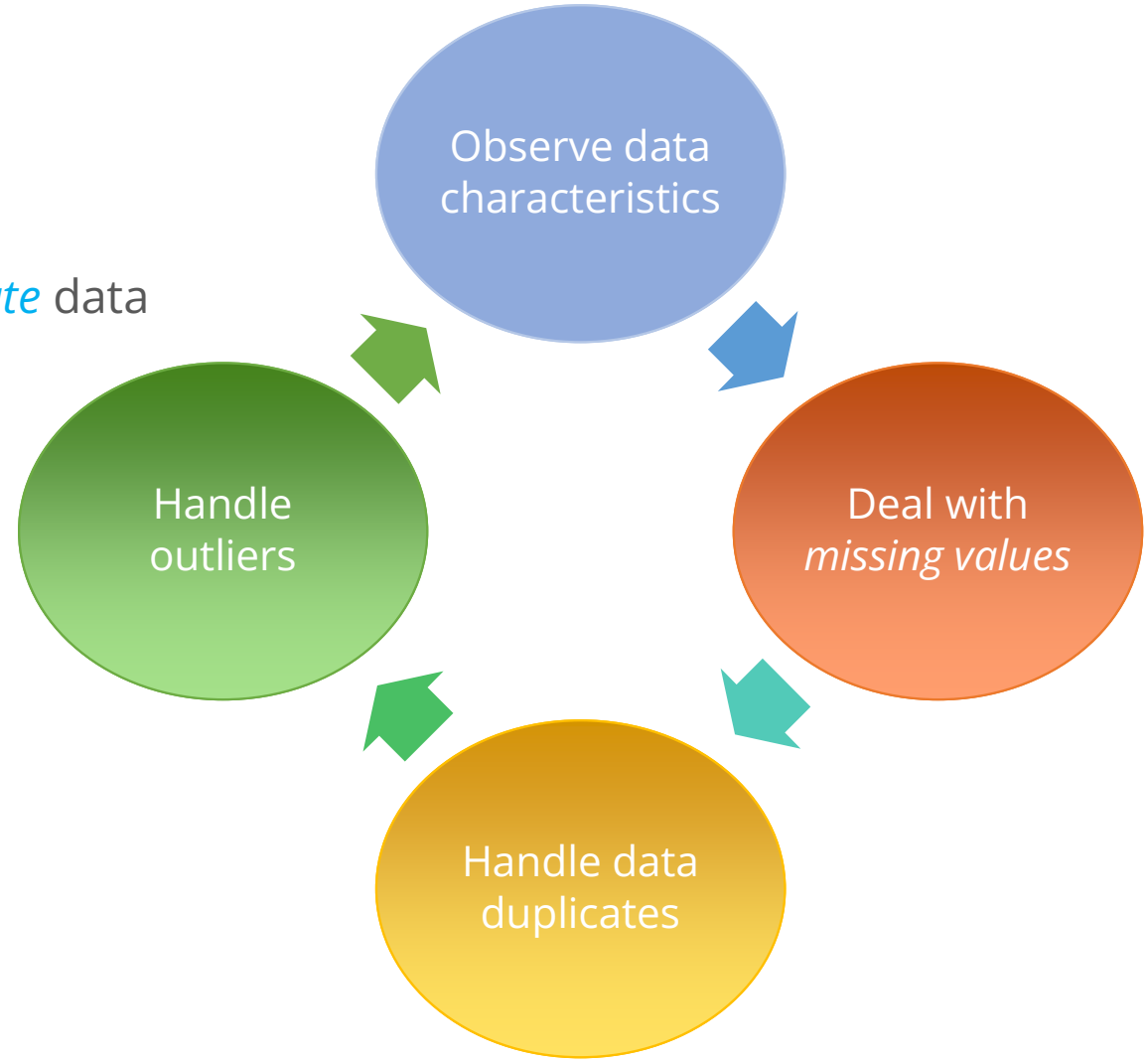
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
#import os


#warnings.filterwarnings('ignore')
%matplotlib inline     # the plots will be displayed directly below the cell
```

Outline

- What is data preparation process?
- Data inspection
- Data manipulation: column-wise
- Data manipulation: row-wise
- Data combination and data replacement
- Discretization (variable bucketing)
- Data encoding
- Data normalization and standardization

2. Data inspection

- Goals
  1. Observe data characteristics
  2. Deal with *missing values*
  3. Handle (find and remove) *duplicate* data
  4. Handle *outliers*



Let's see. Anything wrong with these (bank.csv) data?

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	id	age	job	marital	education	default	balance	housing	loan	contact	duration	campaign	previous	y
2	1	60	technician	married	secondary	no	358	no	no	cellular	95	1	failure	no
3	2	30	admin.	married	secondary	no	265	yes	no	cellular	342	1		no
4	3	47	managem	single	tertiary	no		no	no	telephone	220	3		yes
5	4	51	management		tertiary	no		yes	no	cellular	60	1	failure	no
6	5	30	admin.		secondary	no	873	no	no	cellular	301	1	failure	no
7	6	41	services	married	secondary	no	1141	yes	no	cellular	75	1	failure	no
8	7	58	retired	married	primary	no	565	no	no	telephone	153	1	success	yes
9	8	38	managem	married	tertiary	no	569	yes	no	cellular	105	2	failure	no
10	10	42	technician	single	secondary	no	-15	no	no	cellular	89	1	failure	no
11	8	38	managem	married	tertiary	no	569	yes	no	cellular	105	2	failure	no
12	11	555	blue-collar	married	primary	no	5131	yes	no	cellular	348	1		no
13	12	40	managem	married	tertiary	no	3352	yes	no	cellular	9999	2		yes
14	13	33	services	single	secondary	no	100	yes	no	cellular	11	8		no
15	14	555	technician	divorced	secondary	no	1738	no	no	cellular	79	1		no
16	15	51	admin.	divorced	tertiary	no	66	yes	no	cellular	8888	1		no
17	16	51	housemaic	married	primary	no	605	no	no	cellular	91	2		no

Data inspection: Observe data characteristics

```
# description of index, entries, columns, data types, memory info
df.info()

#  view some basic statistical details like percentile, mean, std etc.
df.describe()

# number of columns & rows
df.shape

# column names
df.columns

# check out first few rows
df.head()

# number of unique values of a column
df['col'].nunique()

# show unique values of a column 'col'
df['col'].unique()

# value counts
df['col'].value_counts()
```

Data inspection: Deal with missing values

- *Missing values* ค่าที่หายไป

- Find missing values
  - Numpy's np.nan ('Not a Number')
  - Panda's None

- Remove the row (record).
- Fill in the missing value manually.
  - ดี แต่ ยากหากมีจำนวนมาก
- Use a global constant to fill in the missing value.
  - เช่น ค่า 'n/a' หรือ 0
- Use the attribute mean (average) to fill in the missing value.
- Use the attribute mean (average) for all samples belonging to the same class as the given tuple.
  - เช่น ค่าเฉลี่ยยอดขายสินค้านั้น ๆ ในเมือง/สาขาเดียวกัน
- Use the most probable value to fill in the missing value.
  - เช่น ใช้ค่าที่ได้จากการประมาณค่าในอดีต (เช่น regression)

Data inspection: Deal with missing values

```
# show null/NA values per column
df.isnull().sum()

# show NA as % of total observations per column
df.isnull().sum() / len(df) * 100

# drop all rows containing null
df.dropna()

# drop all columns containing null
df.dropna(axis=1)

# ...only with rows with all values are NA
df2 = df1.dropna(how='all')

# replace all missing values with a value
df.fillna(-9999)

# fill missing values with strings
df.fillna('data missing')

# fill missing values with NaN
df.fillna(np.NaN)
```

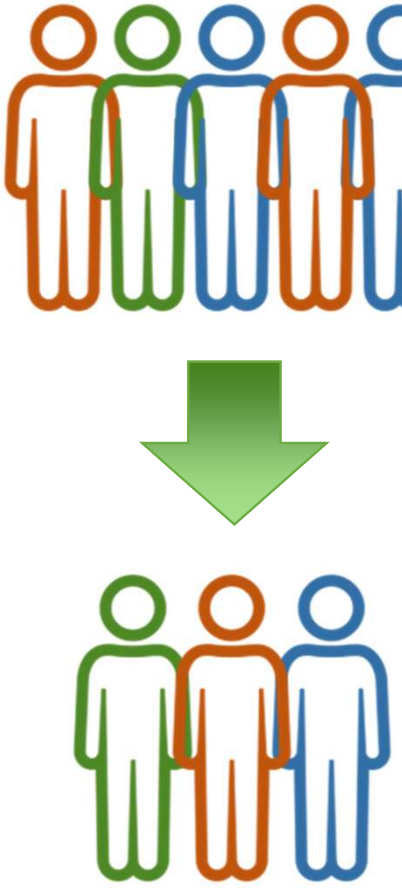
Data inspection: Deal with missing values

```
# fill missing value ด้วยค่าที่กำหนดพร้อมกัน >1 cols
df.fillna({'col1': 'val1', 'col2': 'val2'})

# replace NA of a column with forward filling
df['col'] = df['col'].fillna(method='ffill')

# replace NA of a column with backward filling
df['col'] = df['col'].fillna(method='bfill')

# fill missing values with mean column values
df.fillna(df.mean())
```

```
# replace NA of specific column with its mean value
df['col'] = df['col'].fillna(df['col'].mean())

# interpolation of missing values
# useful in time-series
df['col'].interpolate(method='linear')
df['col'].interpolate(method='quadratic')
```

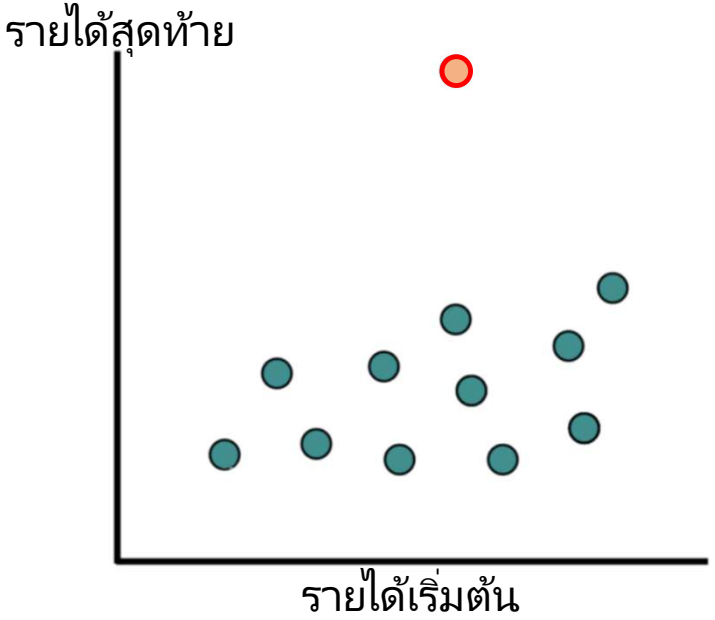
Data inspection: Handle duplicate data

```
# check if rows are duplicated
df.duplicated()

# drop or remove duplicate data
df.drop_duplicated(inplace=True)

# drop or remove duplicate data only
considering certain column(s)
df.drop_duplicates(['column’])
df.drop_duplicates(['marital', 'education'])
```



Data inspection: Handle outliers

- *Outliers* เป็นค่าที่เก็บได้ที่แตกต่างจากค่าอื่นอย่างมาก ซึ่งต้องได้รับการพิจารณาเป็นพิเศษ ว่าค่านั้นถูกต้องจริง ๆ ตามนั้น หรือ เกิดจากการใส่ข้อมูลผิดพลาด (เช่น ป้อนผิด)
- e.g. [20, 24, 32, 19, 29, 18, 222, 30, 18]



- Causes:
  - Data entry errors (human errors)
  - Measurement errors (instrument errors)
  - Sampling errors (extracting or mixing data from wrong or various sources)
  - Not an error (natural)

- *Common methods* to handle outliers:
  1. Use constant [upper, lower] boundaries.
  2. Use statistical Z-score.
  3. Use statistical interquartile range (IQR).

1. Use constants to detect outliers

[32]:
```
# ต้องการหาดู rows ที่ duration มากกว่า 5000
df1[df1['duration'] > 5000]
```

[32]:

	id	age	job	marital	education	default	balance	housing	loan	contact	duration	campaign	previous	y
11	12	40	management	married	tertiary	no	3352.0	yes	no	cellular	9999	2	NaN	yes
14	15	51	admin.	divorced	tertiary	no	66.0	yes	no	cellular	8888	1	NaN	no

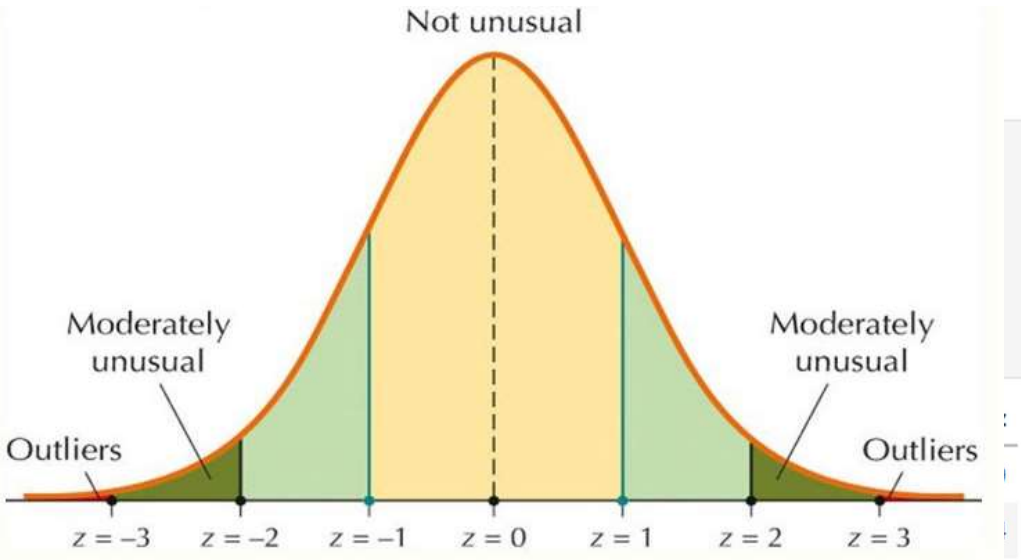
```
# ลบ rows ตามเงื่อนไขที่ต้องการ
df2 = df1.drop(df1[df1.duration > 5000].index)
df2.head(14)
```

After remove all rows with duration > 5000

[35]:

	id	age	job	marital	education	default	balance	housing	loan	contact	duration	campaign	previous	y
0	1	60	technician	married	secondary	no	358.0	no	no	cellular	95	1	failure	no
1	2	30	admin.	married	secondary	no	265.0	yes	no	cellular	342	1	NaN	no
2	3	47	management	single	tertiary	no	NaN	no	no	telephone	220	3	NaN	yes
3	4	51	management	NaN	tertiary	no	NaN	yes	no	cellular	60	1	failure	no
4	5	30	admin.	NaN	secondary	no	873.0	no	no	cellular	301	1	failure	no
5	6	41	services	married	secondary	no	1141.0	yes	no	cellular	75	1	failure	no
6	7	58	retired	married	primary	no	565.0	no	no	telephone	153	1	success	yes
7	8	38	management	married	tertiary	no	569.0	yes	no	cellular	105	2	failure	no
8	10	42	technician	single	secondary	no	-15.0	no	no	cellular	89	1	failure	no
10	11	555	blue-collar	married	primary	no	5131.0	yes	no	cellular	348	1	NaN	no
12	13	33	services	single	secondary	no	100.0	yes	no	cellular	11	8	NaN	no
13	14	555	technician	divorced	secondary	no	1738.0	no	no	cellular	79	1	NaN	no
15	16	51	housemaid	married	primary	no	605.0	no	no	cellular	91	2	NaN	no
16	17	40	housemaid	married	secondary	no	4136.0	yes	no	cellular	508	3	NaN	no

2. Use z-score to detect outliers



std	663.972968	11.579713	1522.905986	2946.568665	0.660225	7.870670
min	12.000000	27.000000	0.000000	1416.000000	1.000000	3.064278
25%	405.000000	37.000000	775.000000	1472.000000	1.000000	3.213862
50%	884.000000	40.000000	1819.000000	1628.000000	1.000000	3.630558
75%	1308.000000	51.000000	2580.000000	1925.000000	2.000000	4.423884
max	1966.000000	63.000000	5112.000000	9999.000000	3.000000	25.990593

- Z score tell you how the data is, compared to the mean and s.d. of the whole dataset.
- | Z-score| > 3 is usually treated to be outliners.

$$Z = \frac{X - \mu}{\sigma}$$

Where

X is the raw score
μ is the mean
σ is the standard deviation

Use z-score to detect outliers: results

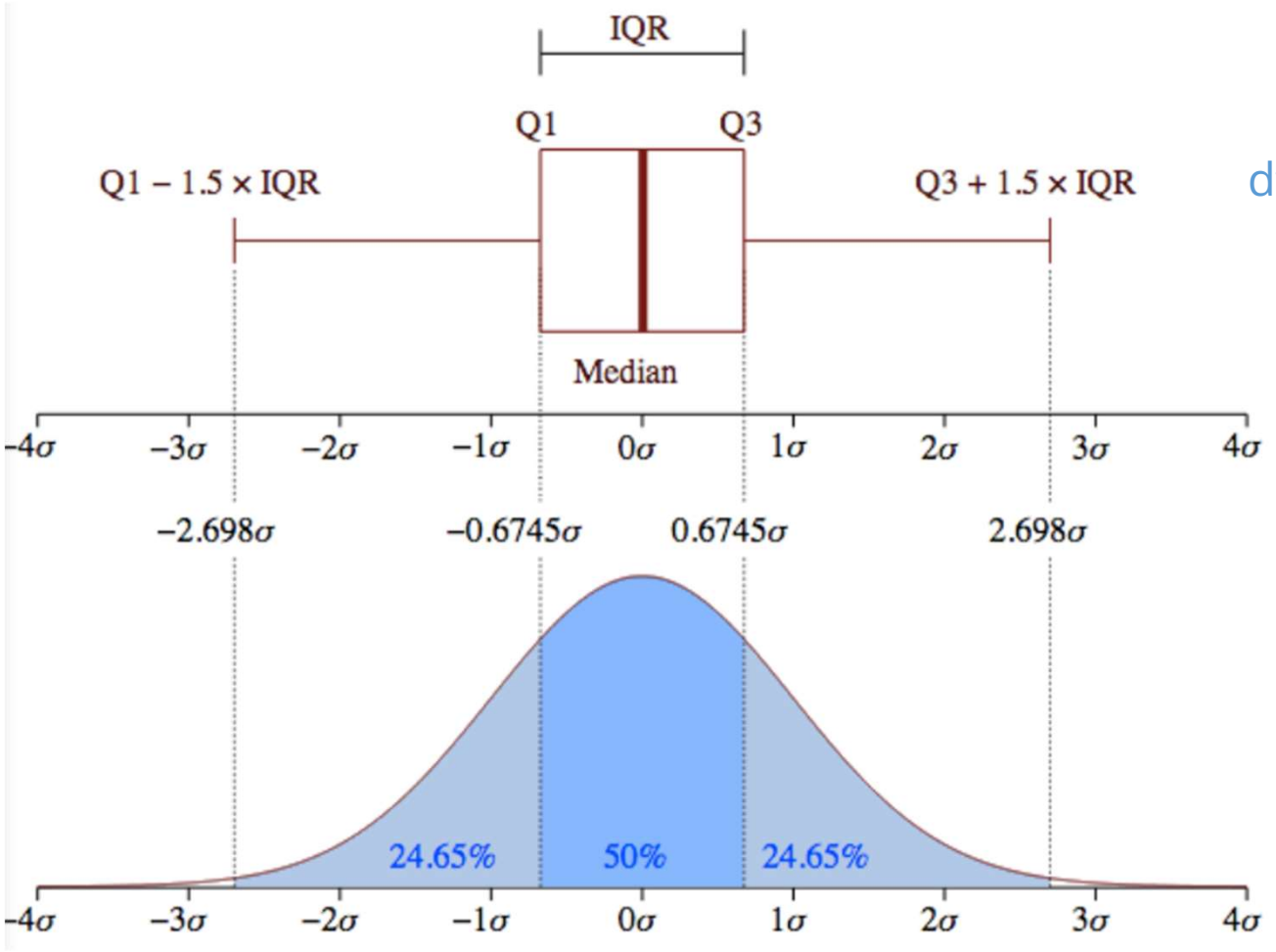
```
# ลบ rows ตามเงื่อนไขที่ต้องการ
df2 = df1.drop(df1[condition].index)
df2.describe()
```

[37]:

	id	age	balance	duration	campaign	z
count	1986.000000	1986.000000	1984.000000	1986.000000	1986.000000	1986.000000
mean	1001.320242	41.490433	1289.106351	252.046324	2.001511	-0.045004
std	576.780131	20.095995	2379.217043	212.190309	1.512500	0.566673
min	1.000000	18.000000	-910.000000	5.000000	1.000000	-0.704763
25%	502.250000	33.000000	152.500000	112.250000	1.000000	-0.418343
50%	1001.500000	38.000000	544.500000	191.000000	1.000000	-0.208034
75%	1500.750000	48.000000	1533.250000	317.000000	2.000000	0.128461
max	2000.000000	555.000000	37378.000000	1370.000000	16.000000	2.940593

3. Use interquartile range to handle outliers

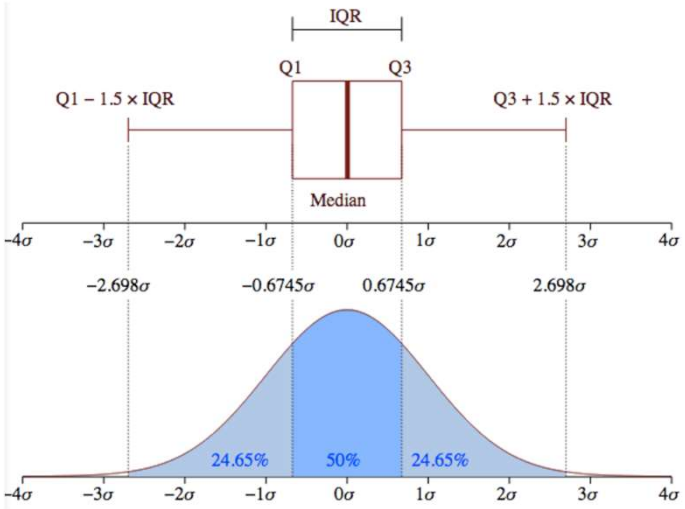
df.quantile(0.25)



df.quantile(0.75)

Use interquartile range to handle outliers

```
def filter_with_iqr(df, column, whisker_width=1.5):
    # Calculate Q1, Q2 and IQR
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1
    # Apply filter with respect to IQR, including optional whiskers
    filter = (df[column] >= q1 - whisker_width*iqr) & (df[column] <= q3 + whisker_width*iqr)
    return df.loc[filter]
```



Test using interquartile range to handle outliers

```
# copy of df1 to df2, so as leave df1 untouched and...
# then change df2's duration with some very high values
df2 = df1.copy(deep=True).drop(['campaign', 'z'], axis=1)
df2.loc[0:11:2, 'duration'] = 5555
df2.loc[11:20:2, 'duration'] = -5555
df2.describe()
```

	id	age	balance	duration
count	1999.000000	1999.000000	1997.000000	1999.000000
mean	1000.995998	41.504252	1292.196795	263.286643
std	577.212823	20.051332	2374.701519	497.562458
min	1.000000	18.000000	-910.000000	-5555.000000
25%	501.500000	33.000000	154.000000	113.000000
50%	1001.000000	38.000000	551.000000	192.000000
75%	1500.500000	48.000000	1536.000000	323.000000
max	2000.000000	555.000000	37378.000000	8888.000000

```
# Example for whiskers = 1.5 (default)
df2filtered = filter_with_iqr(df2, 'duration',
whisker_width=1.5)
df2filtered.describe()
```

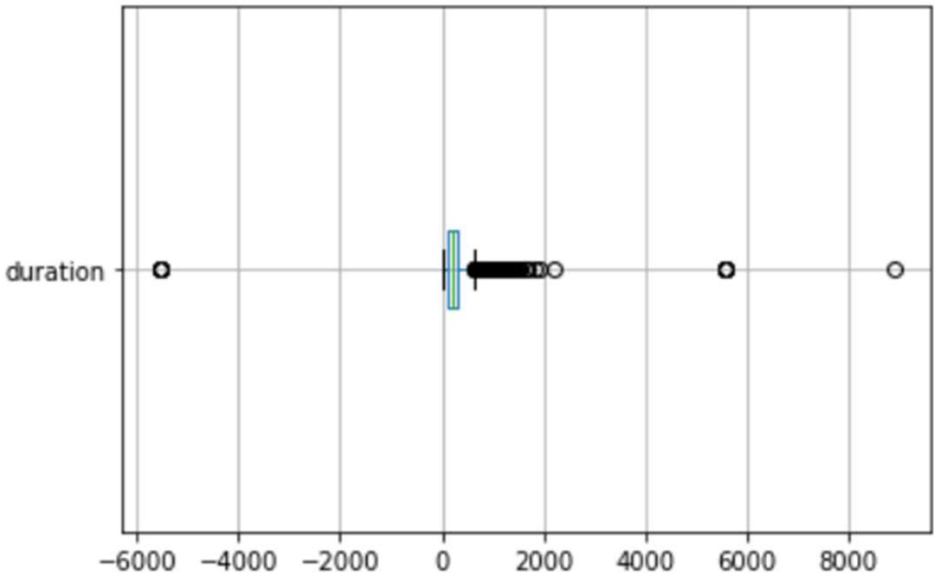
	id	age	balance	duration
count	1865.000000	1865.000000	1864.000000	1865.000000
mean	1006.210724	41.098660	1258.243562	214.022520
std	577.294653	16.597544	2337.888988	138.989708
min	2.000000	18.000000	-910.000000	5.000000
25%	501.000000	33.000000	144.750000	109.000000
50%	1009.000000	38.000000	525.000000	182.000000
75%	1504.000000	47.000000	1497.750000	288.000000
max	2000.000000	555.000000	37378.000000	637.000000

Comparison before and after with boxplot

Before

```
df2.boxplot(column=['duration'], vert=False)
```

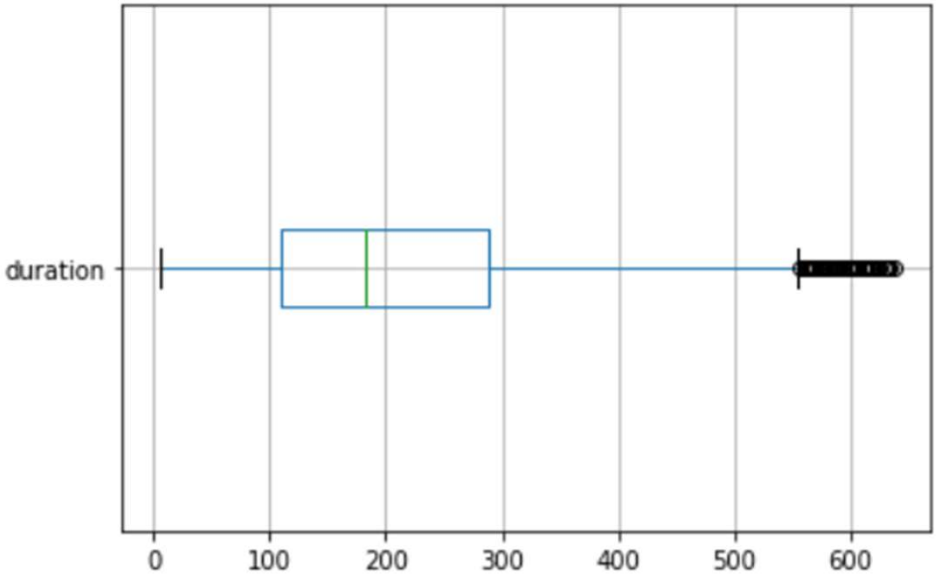
```
<AxesSubplot:>
```



After

```
df2filtered.boxplot(column=['duration'], vert=False)
```

```
<AxesSubplot:>
```



Outline

- What is data preparation process?
- Data inspection
- Data manipulation: column-wise
- Data manipulation: row-wise
- Data combination and data replacement
- Discretization (variable bucketing)
- Data encoding
- Data normalization and standardization

Data manipulation: column-wise

```
# select a column
df['age'] หรือ df.age

# select multiple columns and create a new dataframe X
df[['age', 'education', 'balance']]

# select with column indices
df.iloc[:, [1,3,4]]

# drop a column from dataframe
df.drop('outcome', axis=1)

# save all column names to a list
df.columns.tolist()

# rename columns
df.rename(columns={'oldcol': 'newcol1', 'oldcol2': 'newcol2'})

# sort values by column 'col' in ascending order
df.sort_values(by = 'col', ascending = True)

# Data generation

# add new calculated column
df['newcol'] = df['col'] * 2

# create a conditional calculated column
df['newcol'] = ['short' if i<3 else 'long' for i in df['col']]
```

Column manipulation – rename column(s)

```
# กลับมาใช้ df1 ใหม่ เปลี่ยนชื่อ column y เป็น outcome และลบ col id, housing, loan, campaign, z ทิ้ง
df2 = df1.rename(columns={"y": "outcome"})
df2.drop(['id', 'housing', 'loan', 'campaign', 'z'], axis=1, inplace=True) # ต้องระบุ axis ด้วย ไม่งั้น error
df2
```

	age	job	marital	education	default	balance	contact	duration	previous	outcome
0	60	technician	married	secondary	no	358.0	cellular	95	failure	no
1	30	admin.	married	secondary	no	265.0	cellular	342	NaN	no
2	47	management	single	tertiary	no	NaN	telephone	220	NaN	yes
3	51	management	NaN	tertiary	no	NaN	cellular	60	failure	no
4	30	admin.	NaN	secondary	no	873.0	cellular	301	failure	no
...	...	...	...	...	...	...	...	...	...	...
1995	33	management	single	tertiary	no	935.0	cellular	385	success	yes
1996	31	management	married	tertiary	no	1224.0	cellular	75	success	no
1997	60	technician	married	secondary	no	824.0	cellular	127	other	yes
1998	61	retired	married	primary	no	1720.0	cellular	150	failure	no
1999	42	blue-collar	married	secondary	no	2717.0	cellular	181	failure	no

1999 rows × 10 columns

Column manipulation – rename columns

```
# rename df2's age -> AGE, and education -> EDUCATION into df3
df3 = df2.rename(columns={'age': 'AGE', 'education': 'EDUCATION'})
df3.head(4)
```

	AGE	job	marital	EDUCATION	default	balance	contact	duration	previous	outcome
0	60	technician	married	secondary	no	358.0	cellular	95	failure	no
1	30	admin.	married	secondary	no	265.0	cellular	342	NaN	no
2	47	management	single	tertiary	no	NaN	telephone	220	NaN	yes
3	51	management	NaN	tertiary	no	NaN	cellular	60	failure	no
4	30	admin.	NaN	secondary	no	873.0	cellular	301	failure	no

Column manipulation: sorting

df2.sort_values(by='age', ascending=True)

```
[67]: df2.sort_values(by="age", ascending=True)
```

[67]:

	age	job	marital	education	default
1414	18	student	single	unknown	no
1544	19	student	single	primary	no
108	19	student	single	secondary	no
539	19	student	single	unknown	no
1404	20	student	married	unknown	no
...	...	...	...	...	...
436	86	retired	married	primary	no
271	88	retired	married	secondary	no
238	93	retired	married	unknown	no
13	555	technician	divorced	secondary	no
10	555	blue-collar	married	primary	no

เรียงขึ้นตาม age และลงตาม balance

df2.sort_values(by=['age', 'balance'], ascending=[True, False])

```
[69]: # เรียงขึ้นตาม age และลงตาท balance
      df2.sort_values(by=['age', 'balance'], ascending=[True, False])
```

[69]:

	age	job	marital	education	default	balance	contact	duration	previous	outcome
1414	18	student	single	unknown	no	108.0	cellular	92	success	yes
1544	19	student	single	primary	no	608.0	cellular	236	success	yes
108	19	student	single	secondary	no	424.0	cellular	121	other	no
539	19	student	single	unknown	no	108.0	cellular	168	success	yes
1404	20	student	married	unknown	no	292.0	cellular	385	failure	yes
...	...	...	...	...	...	...	...	...	...	...
436	86	retired	married	primary	no	1255.0	cellular	192	success	no
271	88	retired	married	secondary	no	433.0	telephone	161	failure	no
238	93	retired	married	unknown	no	775.0	cellular	476	success	yes
10	555	blue-collar	married	primary	no	5131.0	cellular	348	NaN	no
13	555	technician	divorced	secondary	no	1738.0	cellular	79	NaN	no

แบบฝึกหัด

กลับมาถ่าย df1 เข้า df2 ใหม่ แล้วตัดให้เหลือ columns น้อยๆ เพื่อความสะดวก

```
df2 = df1.copy(deep=True).drop(['education', 'contact', 'previous'], axis=1)
df2
```

แก้ค่า age ใน df2 ให้มีค่าเท่ากับ age สูงสุดที่เหลือจากการตัด age สูงสุด 10 ลำดับ (ข้อมูล) ออกไปแล้ว

#

Hint: ขั้นตอน

1. ให้ df3 <-- df2 ที่ค่า age สูงสุด จำนวน 10 rows พอ

2. หาค่า age ที่ต่ำสุดจากผลลัพธ์ในข้อที่แล้ว

3. แก้ค่า age ใน df2 ตามเงื่อนไข (อะไร) ให้เท่ากับ age ที่กำหนดในข้อ 2.

[104]:

	age	job	marital	default	balance	duration
13	555	technician	divorced	no	1738.0	79
10	555	blue-collar	married	no	5131.0	348
238	93	retired	married	no	775.0	476
271	88	retired	married	no	433.0	161
436	86	retired	married	no	1255.0	192
579	84	retired	divorced	no	1680.0	113
1857	83	retired	married	no	6236.0	766
1736	83	retired	married	no	425.0	883
56	82	housemaid	divorced	no	1381.0	86
1741	82	retired	married	no	243.0	275

						duration
						86
						161
						113
						275
238	81	retired	married	no	775.0	476
1857	81	retired	married	no	6236.0	766
436	81	retired	married	no	1255.0	192
1736	81	retired	married	no	425.0	883
13	81	technician	divorced	no	1738.0	79
10	81	blue-collar	married	no	5131.0	348
1900	80	retired	married	no	668.0	250
1789	80	retired	married	no	155.0	290

เฉลย

```
# copy original data before processing
df2 = df1.copy(deep=True).drop(['education', 'contact', 'previous'], axis=1)
df2


# get the top-age rows
df3 = df2.sort_values(by='age', ascending=False).head(12)
df3


# get to max age
maxage = df3['age'].min()
maxage


# change the rows with age >= maxage
df2.loc[df2['age']>=maxage, 'age'] = maxage-1
df2
```

เฉลย

Data (or column) generations

```
# advance calculation of new column
# คำนวณ duration เป็นนาที, สร้าง col category short/long แบ่งด้วย duration ที่ 3 นาที, แล้วกำหนด ratio = balance / age
df2['duration'] = df1['duration'] / 60
df2['category'] = ['short' if i<3 else 'long' for i in df2['duration']]
df2['ratio'] = df2['balance'] / df2['age']
df2.head(10)
```

	id	age	job	marital	default	balance	housing	loan	duration	campaign	y	z	category	ratio
0	1	60	technician	married	no	358.0	no	no	1.583333	1	no	-0.464285	short	5.966667
1	2	30	admin.	married	no	265.0	yes	no	5.700000	1	no	0.195484	long	8.833333
2	3	47	management	single	no	NaN	no	no	3.666667	3	yes	-0.130394	long	NaN
3	4	51	management	NaN	no	NaN	yes	no	1.000000	1	no	-0.557775	short	NaN
4	5	30	admin.	NaN	no	873.0	no	no	5.016667	1	no	0.085968	long	29.100000
5	6	41	services	married	no	1141.0	yes	no	1.250000	1	no	-0.517708	short	27.829268
6	7	58	retired	married	no	565.0	no	no	2.550000	1	yes	-0.309360	short	9.741379

Outline

- What is data preparation process?
- Data inspection
- Data manipulation: column-wise
- Data manipulation: row-wise
- Data combination and data replacement
- Discretization (variable bucketing)
- Data encoding
- Data normalization and standardization

Data manipulation: row-wise

```
# select rows 3 to 7
df.iloc[3:8,]

# select rows 3 to 49 and columns 1 to 3
df.iloc[3:50, 1:4]

# randomly select 10 rows
df.sample(10)

# find rows with specific strings
df[df['job'].str.contains('tech')]

# หา rows ที่ job เป็น technician หรือ management
df2[df2['job'].isin(['technician', 'management'])]
```

```
# conditional row filtering
# หา rows ที่ age ไม่ต่ำกว่า 85
df2[df2.age >= 85]

# multi-conditional filtering
df2[ (df2.age >= 85) & (df2.housing != 'yes') ]

# ลบ rows โดยระบุ index
df3.drop(df3.index[2]) # 2 is row index to be deleted

# ลบ row โดยระบุเงื่อนไข ว่า duration < 5
df3.drop(df3[df3.duration < 5].index)
```

Select rows with indices

```
df2 = df1.drop(['default', 'campaign', 'previous', 'y', 'z'], axis=1)

# เลือก rows ที่ 3 ถึง 7
df2.iloc[3:8,]
```

	id	age	job	marital	education	balance	housing	loan	contact	duration
3	4	51	management	NaN	tertiary	NaN	yes	no	cellular	60
4	5	30	admin.	NaN	secondary	873.0	no	no	cellular	301
5	6	41	services	married	secondary	1141.0	yes	no	cellular	75
6	7	58	retired	married	primary	565.0	no	no	telephone	153
7	8	38	management	married	tertiary	569.0	yes	no	cellular	105

Find rows containing a specific string

หา rows ที่ job มีคำว่า tech อยู่

```
df2[df2['job'].str.contains('tech')]
```

	id	age	job	marital	education	balance	housing
0	1	60	technician	married	secondary	358.0	no
8	10	42	technician	single	secondary	-15.0	no
13	14	555	technician	divorced	secondary	1738.0	no
18	19	50	technician	single	secondary	-97.0	yes
29	30	45	technician	single	secondary	-55.0	no
...	...	...	...	...	...	...	...
1957	1958	41	technician	married	secondary	-312.0	yes
1973	1974	29	technician	married	tertiary	676.0	no
1991	1992	37	technician	married	primary	1653.0	yes
1993	1994	32	technician	married	tertiary	15.0	no
1997	1998	60	technician	married	secondary	824.0	yes

308 rows × 10 columns

หา rows ที่ job เป็น technician หรือ management

```
df2[df2['job'].isin(['technician', 'management'])]
```

	id	age	job	marital	education	balance	housing
0	1	60	technician	married	secondary	358.0	no
2	3	47	management	single	tertiary	NaN	no
3	4	51	management	NaN	tertiary	NaN	yes
7	8	38	management	married	tertiary	569.0	yes
8	10	42	technician	single	secondary	-15.0	no
...	...	...	...	...	...	...	...
1993	1994	32	technician	married	tertiary	15.0	no
1994	1995	39	management	married	tertiary	691.0	yes
1995	1996	33	management	single	tertiary	935.0	no
1996	1997	31	management	married	tertiary	1224.0	yes
1997	1998	60	technician	married	secondary	824.0	yes

736 rows × 10 columns

Outline

- What is data preparation process?
- Data inspection
- Data manipulation: column-wise
- Data manipulation: row-wise
- Data combination and data replacement
- Discretization (variable bucketing)
- Data encoding
- Data normalization and standardization

Merge dataframe – pd.merge คล้าย *join* operations ใน SQL

```
In [93]: df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'], 'data1': range(7)})
         df2 = pd.DataFrame({'key': ['a', 'b', 'd'], 'data2': range(3)})
```

```
In [94]: df1
```

Out[94]:

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

```
In [95]: df2
```

Out[95]:

	key	data2
0	a	0
1	b	1
2	d	2

```
In [96]: # ใช้ pd.merge() เพื่อ merge หรือ join datasets
         # หากไม่กำหนด column จะใช้ column ที่ชื่อตรงกัน
         pd.merge(df1, df2)
```

Out[96]:

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0
4	a	4	0
5	a	5	0

การ merge จะรวม record ที่ key มีค่าเท่ากัน

```
In [97]: # แต่แนะนำให้ระบุ column ทุกครั้งที่ merge ด้วย on=ชื่อ_key
         pd.merge(df1, df2, on='key')
```

Merge ด้วยมากกว่า 1 key (compound key) ก็ได้

[109]:
```
# ทดลอง merge ด้วย composite key (key ที่มากกว่า 1 col)
#
and_op = pd.DataFrame({'A':['0', '0', '1','1'],
                       'B':['0', '1', '0','1'],
                       'AND':[False, False, False, True]})
or_op = pd.DataFrame({'A':['0', '0', '1','1'],
                      'B':['0', '1', '0','1'],
                      'OR':[False, True, True, True]})
and_op
```

[109]:

	A	B	AND
0	0	0	False
1	0	1	False
2	1	0	False
3	1	1	True

[110]:
```
or_op
```

[110]:

	A	B	OR
0	0	0	False
1	0	1	True
2	1	0	True
3	1	1	True

[111]:
```
result = pd.merge(and_op, or_op, on=['A','B'])
result
```

[111]:

	A	B	AND	OR
0	0	0	False	False
1	0	1	False	True
2	1	0	False	True
3	1	1	True	True

เตรียมข้อมูลสำหรับ merge ด้วย key ต่างกัน

```
#ถ้า 2 dataframe จะ merge กันด้วย column ชื่อต่างกัน ก็ได้ โดยการระบุใช้ชัดเจน
df3 = pd.DataFrame({'left_key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'], 'data1': range(7)})
df4 = pd.DataFrame({'right_key': ['a', 'b', 'd'], 'data2': range(3)})
```

```
df3
```

	left_key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

```
df4
```

	right_key	data2
0	a	0
1	b	1
2	d	2

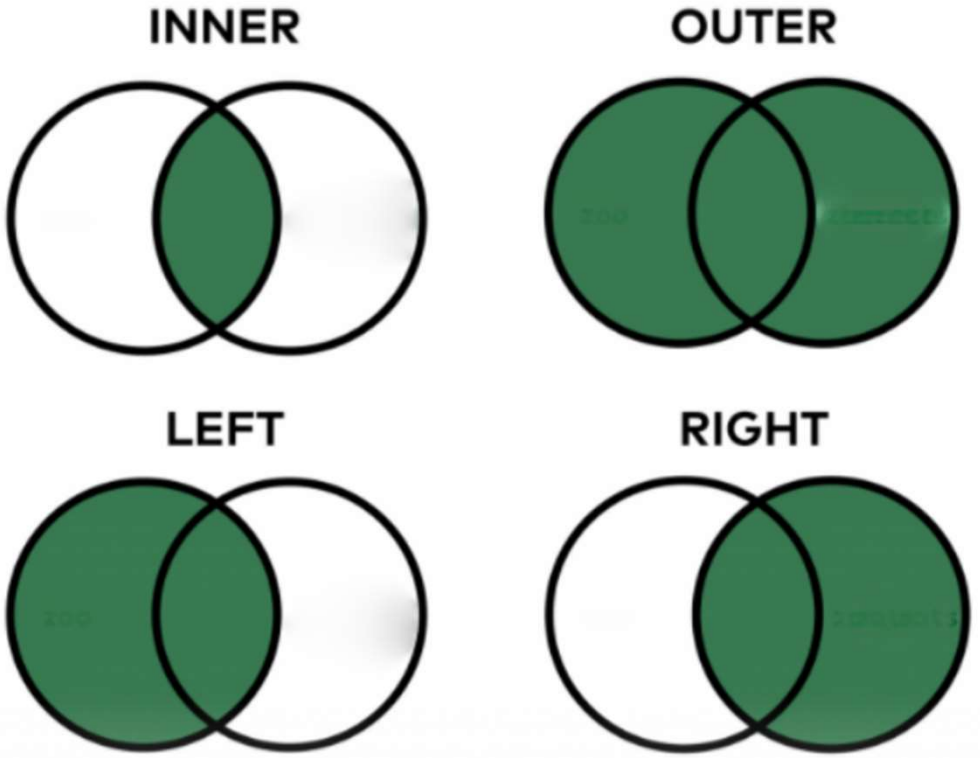
```
# merge ระบุชื่อ key จากสอง dataframeด้วย left_on='' กับ right_on=''
pd.merge(df3, df4, left_on='left_key', right_on='right_key')
```

	left_key	data1	right_key	data2
0	b	0	b	1
1	b	1	b	1
2	b	6	b	1
3	a	2	a	0
4	a	4	a	0
5	a	5	a	0

Four different ways of *merge*

- Merge มี 4 วิธี ที่ผ่านมาไม่ระบุ จะเรียกว่าเป็น inner join คือเอา record เฉพาะที่มี key ตรงกันในทั้ง dataset ซ้ายและขวา

(default)



กำหนดวิธี merge แบบอื่นด้วย how & left join เอา dataset ซ้ายเป็นหลัก

df1

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

df2

	key	data2
0	a	0
1	b	1
2	d	2

```
In [102]: # กำหนดวิธี join ด้วย how=[method]
          # left join เอา dataset ซ้ายเป็นหลัก
          pd.merge(df1, df2, how='left')
```

Out[102]:

	key	data1	data2
0	b	0	1.0
1	b	1	1.0
2	a	2	0.0
3	c	3	NaN
4	a	4	0.0
5	a	5	0.0
6	b	6	1.0

Merge – right เอา dataset ขวาเป็นหลัก

df1

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

df2

	key	data2
0	a	0
1	b	1
2	d	2

```
In [103]: # right join เอา dataset ขวาเป็นหลัก
          pd.merge(df1, df2, how='right')
```

Out[103]:

	key	data1	data2
0	b	0.0	1
1	b	1.0	1
2	b	6.0	1
3	a	2.0	0
4	a	4.0	0
5	a	5.0	0
6	d	NaN	2

Merge – outer คงทุก rows ของทั้งสอง datasets ไว้

df1

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

df2

	key	data2
0	a	0
1	b	1
2	d	2

```
In [104]: # Outer join คงทุก rows ของทั้งสอง dataset ไว้
          pd.merge(df1, df2, how='outer')
```

Out[104]:

	key	data1	data2
0	b	0.0	1.0
1	b	1.0	1.0
2	b	6.0	1.0
3	a	2.0	0.0
4	a	4.0	0.0
5	a	5.0	0.0
6	c	3.0	NaN
7	d	NaN	2.0

รวมข้อมูลของ 2 dataframes ที่โครงสร้างเหมือนกันตาม rows

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

pd.concat() *faster

df.append()

Add rows from one dataframe to another dataframe

[99]:
```
first = df1.iloc[:4,:6]
first
```

[99]:

	id	age	job	marital	education	default
0	1	60	technician	married	secondary	no
1	2	30	admin.	married	secondary	no
2	3	47	management	single	tertiary	no
3	4	51	management	NaN	tertiary	no

[100]:
```
second = df1.iloc[4:9,:6]
second
```

[100]:

	id	age	job	marital	education	default
4	5	30	admin.	NaN	secondary	no
5	6	41	services	married	secondary	no
6	7	58	retired	married	primary	no
7	8	38	management	married	tertiary	no
8	10	42	technician	single	secondary	no

pd.concat([df1, df2])

[101]:
```
df = pd.concat([first, second])
df
```

[101]:

	id	age	job	marital	education	default
0	1	60	technician	married	secondary	no
1	2	30	admin.	married	secondary	no
2	3	47	management	single	tertiary	no
3	4	51	management	NaN	tertiary	no
4	5	30	admin.	NaN	secondary	no
5	6	41	services	married	secondary	no
6	7	58	retired	married	primary	no
7	8	38	management	married	tertiary	no
8	10	42	technician	single	secondary	no

Add rows from one dataframe to another dataframe

```
[99]: first = df1.iloc[:4,:6]
      first
```

[99]:

	id	age	job	marital	education	default
0	1	60	technician	married	secondary	no
1	2	30	admin.	married	secondary	no
2	3	47	management	single	tertiary	no
3	4	51	management	NaN	tertiary	no

```
[100]: second = df1.iloc[4:9,:6]
       second
```

[100]:

	id	age	job	marital	education	default
4	5	30	admin.	NaN	secondary	no
5	6	41	services	married	secondary	no
6	7	58	retired	married	primary	no
7	8	38	management	married	tertiary	no
8	10	42	technician	single	secondary	no

df1.append(df2)

```
[102]: df = first.append(second)
       df
```

[102]:

	id	age	job	marital	education	default
0	1	60	technician	married	secondary	no
1	2	30	admin.	married	secondary	no
2	3	47	management	single	tertiary	no
3	4	51	management	NaN	tertiary	no
4	5	30	admin.	NaN	secondary	no
5	6	41	services	married	secondary	no
6	7	58	retired	married	primary	no
7	8	38	management	married	tertiary	no
8	10	42	technician	single	secondary	no

More about merge(), join(), and concat()

Advanced topic

- merge คล้ายกับ join มาก. แต่ by default แล้ว join จะรวมด้วย index ของ dataframe (แต่กำหนด column เองก็ได้) ส่วน merge ไม่จำเป็นต้องมี index แต่จะเลือก column ที่ชื่อตรงกันให้ หรือจะเลือก column เองก็ได้เช่นกัน
- merge ยังสามารถตรวจสอบ key คล้ายๆ กับ database ได้ด้วย parameter → validate (default is None)
  - “one_to_one” or “1:1”: check if merge keys are unique in both left and right datasets.
  - “one_to_many” or “1:m”: check if merge keys are unique in left dataset.
  - “many_to_one” or “m:1”: check if merge keys are unique in right dataset.
  - “many_to_many” or “m:m”: allowed, but does not result in checks.
- concat สามารถรวม dataframe ตามแนว column ก็ได้ ด้วย axis = 1

Replace values with condition

Original data

	id	age	job	marital	education	default
0	1	60	technician	married	secondary	no
1	2	30	admin.	married	secondary	no
2	3	47	management	single	tertiary	no
3	4	51	management	NaN	tertiary	no
4	5	30	admin.	NaN	secondary	no
5	6	41	services	married	secondary	no
6	7	58	retired	married	primary	no
7	8	38	management	married	tertiary	no
8	10	42	technician	single	secondary	no

1

เงื่อนไขเพื่อเลือก row

col ที่จะแก้ไข

```
[104]: df.loc[df['age'] > 45, 'job'] = 'retired'
       df
```

[104]:

	id	age	job	marital	education	de
0	1	60	retired	married	secondary	
1	2	30	admin.	married	secondary	
2	3	47	retired	single	tertiary	
3	4	51	retired	NaN	tertiary	
4	5	30	admin.	NaN	secondary	
5	6	41	services	married	secondary	
6	7	58	retired	married	primary	
7	8	38	management	married	tertiary	
8	10	42	technician	single	secondary	

Replace value

Original data

	id	age	job	marital	education	default
0	1	60	technician	married	secondary	no
1	2	30	admin.	married	secondary	no
2	3	47	management	single	tertiary	no
3	4	51	management	NaN	tertiary	no
4	5	30	admin.	NaN	secondary	no
5	6	41	services	married	secondary	no
6	7	58	retired	married	primary	no
7	8	38	management	married	tertiary	no
8	10	42	technician	single	secondary	no

2

```
[105]: df.age = df.age.replace(60, 61)
       df
```

[105]:

	id	age	job	marital	education	default
0	1	61	retired	married	secondary	no
1	2	30	admin.	married	secondary	no
2	3	47	retired	single	tertiary	no
3	4	51	retired	NaN	tertiary	no
4	5	30	admin.	NaN	secondary	no
5	6	41	services	married	secondary	no
6	7	58	retired	married	primary	no
7	8	38	management	married	tertiary	no
8	10	42	technician	single	secondary	no

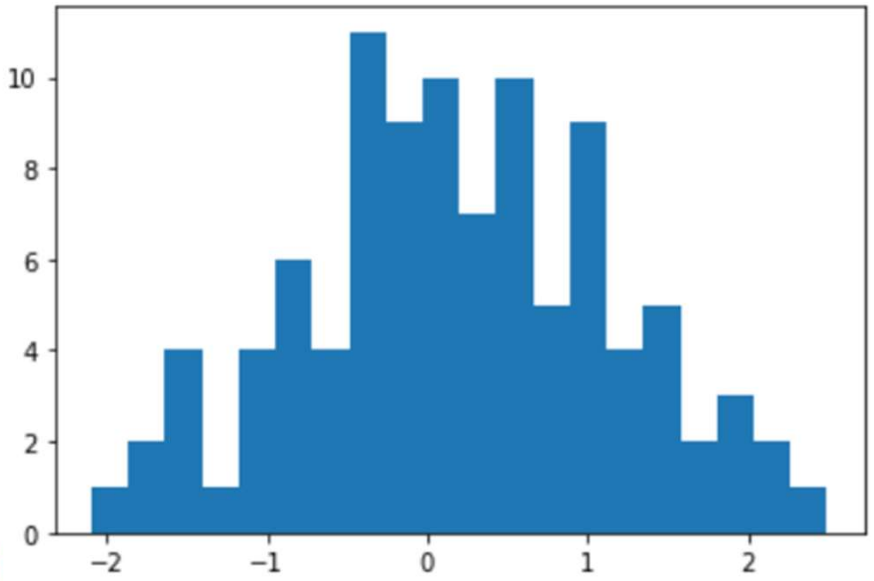
Outline

- What is data preparation process?
- Data inspection
- Data manipulation: column-wise
- Data manipulation: row-wise
- Data combination and data replacement
- Discretization (variable bucketing)
- Data encoding
- Data normalization and standardization

Discretization (variable bucketing)

- To go from a continuous variable to a categorical variable. For example, cut could convert ages to groups of age ranges.

```
import matplotlib.pyplot as plt
np.random.seed(999)
data = np.random.randn(100)    # Normally distributed
plt.hist(data, bins=20)
plt.show()
```




```
[93]: # pd.cut(dataframe, n) แบ่งข้อมูลออกเป็น n ช่วงๆ ตามค่าของข้อมูล
      #
      c = pd.cut(data, 4)
      c.value_counts()
```

```
[93]: (-2.096, -0.946]    12
      (-0.946, 0.2]       40
      (0.2, 1.346]        35
      (1.346, 2.491]      13
      dtype: int64
```

```
[94]: # pd.qcut(dataframe, n) แบ่งข้อมูลออกเป็น n ช่วงๆ ละจำนวนเท่าๆ กัน
      #
      cats = pd.qcut(data, 4)
      cats.value_counts()
```

```
[94]: (-2.092, -0.455]    25
      (-0.455, 0.14]      25
      (0.14, 0.918]       25
      (0.918, 2.491]      25
      dtype: int64
```

Outline

- What is data preparation process?
- Data inspection
- Data manipulation: column-wise
- Data manipulation: row-wise
- Data combination and data replacement
- Discretization (variable bucketing)
- Data encoding
- Data normalization and standardization

Data encoding

- Many machine learning techniques like ANN, KNN, etc. (except tree-based like DT) require (or prefer) numerical data.
- But data we have may be categorical e.g., gender [male, female], education [secondary, bachelor, graduate], etc.
- This requires data encoding techniques
- Commonly used data encoding are:
  - ✓ Label encoder
  - ✓ Dummy variables

Data encoding: 1) Label encoder

Education	LE_edu
Secondary	0
Bachelor	1
Master	2
Doctor	3

Ordinal

```
secondary    5
tertiary     3
primary      2
unknown      1
Name: education, dtype: int64
```

```
# reload df1 dataframe
df1 = pd.read_csv(filename)
df2 = df1[['age', 'job', 'education', 'balance']][66:77].reset_index()

# ลองดูข้อมูล education ก่อนทำ dummy variable
df2.education.value_counts()

from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()  # create instance
df2['new'] = labelencoder.fit_transform(df2['education'])
df2
```

	index	age	job	education	balance	new
0	66	35	blue-collar	primary	455.0	0
1	67	42	technician	unknown	254.0	3
2	68	37	blue-collar	secondary	-100.0	1
3	69	34	admin.	secondary	2374.0	1
4	70	31	management	tertiary	775.0	2
5	71	50	technician	secondary	-70.0	1
6	72	36	management	tertiary	203.0	2
7	73	60	technician	primary	4243.0	0
8	74	38	blue-collar	secondary	105.0	1
9	75	32	management	tertiary	2213.0	2
10	76	49	admin.	secondary	132.0	1

Data encoding for *nominal* data

Education	LE_edu
Secondary	0
Bachelor	1
Master	2
Doctor	3

Ordinal

Color	LE_color
Orange	0
Mango	1
Pineapple	2
Strawberry	3

Nominal?

Data encoding: 2) Dummy variables

- ใช้ pd.get_dummies() เพื่อเปลี่ยนค่า k ค่าของ column หนึ่งให้กลายเป็น k columns ใหม่ที่มีค่า 0 หรือ 1 สอดคล้องกับข้อมูลเดิม โดยจะมี 1 ได้เพียง 1 (จาก k) column เท่านั้น
- ใช้เพิ่มประสิทธิภาพในการจำแนกประเภทข้อมูล (classification) และอื่นๆ

```
# เรียก pd.get_dummies() เพื่อสร้างตัวแปร dummy variables
# ขึ้นมาจาก col ที่เลือก และใส่ prefix หน้าชื่อแต่ละ column ด้วย
#
dummies = pd.get_dummies(df2['education'], prefix='edu', dtype=np.int)
dummies
```

	edu_primary	edu_secondary	edu_tertiary	edu_unknown
0	1	0	0	0
1	0	0	0	1
2	0	1	0	0
3	0	1	0	0
4	0	0	1	0
5	0	1	0	0
6	0	0	1	0
7	1	0	0	0
8	0	1	0	0
9	0	0	1	0
10	0	1	0	0

get_dummies(drop_first=True) เพื่อลดจำนวน column ใหม่ลง

```
# option drop_first=True removes the first level to get k-1 dummies out of k categorical levels
#
dum2 = pd.get_dummies(df2['education'], prefix='edu', drop_first=True)
dum2
```

[99]:

	edu_secondary	edu_tertiary	edu_unknown
0	0	0	0
1	0	0	1
2	1	0	0
3	1	0	0
4	0	1	0
5	1	0	0
6	0	1	0
7	0	0	0
8	1	0	0
9	0	1	0
10	1	0	0

```
dataset = pd.get_dummies(dataset, columns = ['education', 'color'])
```

แล้วเรียก df2.join(dummies) เพื่อรวมเข้า df1 เดิม (ดูหน้าถัดไป ...)

เรียกใช้ df.join(dummies) เพื่อรวมเข้าใน dataframe เดิม

df = df2.join(dum2)

df

```
[103]: df = df2.join(dum2)
       df
```

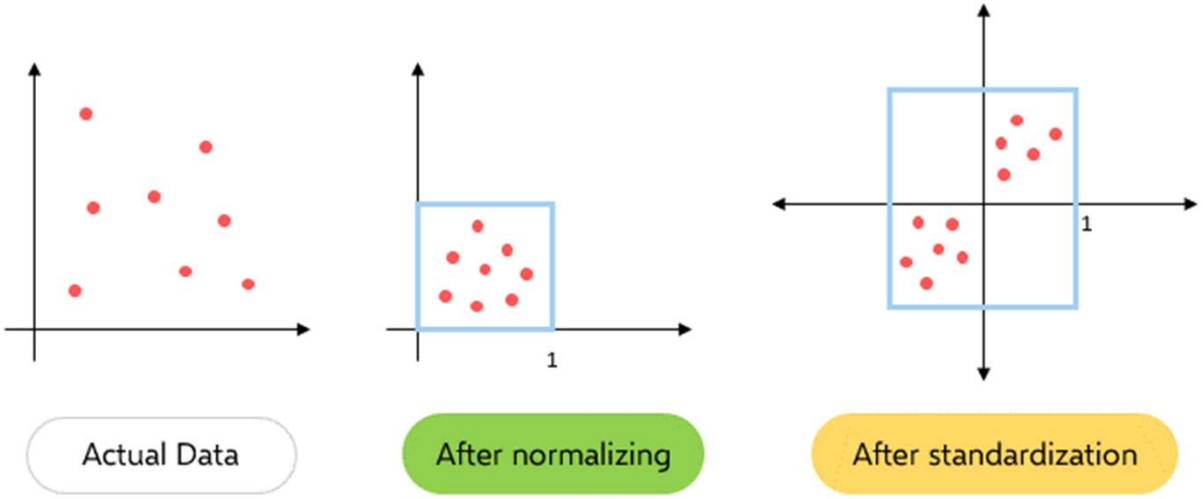
[103]:

	index	age	job	education	balance	new	edu_secondary	edu_tertiary	edu_unknown
0	66	35	blue-collar	primary	455.0	0	0	0	0
1	67	42	technician	unknown	254.0	3	0	0	1
2	68	37	blue-collar	secondary	-100.0	1	1	0	0
3	69	34	admin.	secondary	2374.0	1	1	0	0
4	70	31	management	tertiary	775.0	2	0	1	0
5	71	50	technician	secondary	-70.0	1	1	0	0
6	72	36	management	tertiary	203.0	2	0	1	0
7	73	60	technician	primary	4243.0	0	0	0	0
8	74	38	blue-collar	secondary	105.0	1	1	0	0
9	75	32	management	tertiary	2213.0	2	0	1	0
10	76	49	admin.	secondary	132.0	1	1	0	0

Outline

- What is data preparation process?
- Data inspection
- Data manipulation: column-wise
- Data manipulation: row-wise
- Data combination and data replacement
- Discretization (variable bucketing)
- Data encoding
- Data normalization and standardization

Data normalization and standardization



Standardization

Standardization:

$$z = \frac{x - \mu}{\sigma}$$

with mean:

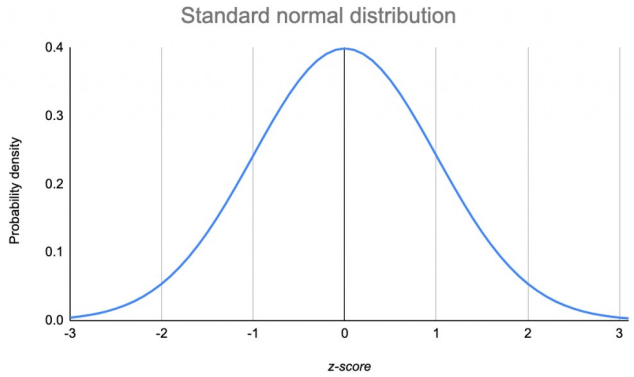
$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

and standard deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Normalization เช่น ไปที่ [0,1]

$$z = \frac{x \ - \min(x)}{\max(x) \ - \min(x)}$$



Sklearn's normalization (left) and standardization (right)

```
from sklearn.preprocessing import MinMaxScaler

# fit scaler on training data
norm = MinMaxScaler().fit(X_train['col'])

# transform training data column
X_train_norm = norm.transform(X_train['col'])

# transform testing data column
X_test_norm = norm.transform(X_test['col’])
```

```
from sklearn.preprocessing import StandardScaler

# fit on training data column
scale = StandardScaler().fit(X_train['col'])

# transform the training data column
X_train_stand[i] = scale.transform(X_train['col'])

# transform the testing data column
X_test_stand[i] = scale.transform(X_test['col'])
```

นำ df1 มาเตรียมทำ normalization and standardization

นำ df1(age, balance) มา 100 รายการเพื่อทดลองทำ normalization ในช่วง [0, 1]

[100]:
```
df2 = df1[['age', 'balance']][:100].reset_index()
df2['balance'] = df1['balance'].fillna(df1['balance'].mean())
print(df2.describe())
df2.head(10)
```

```
            index         age      balance
count  100.000000  100.000000   100.000000
mean    49.500000   50.660000  1219.786697
std     29.011492   73.230813  1515.584999
min      0.000000   25.000000  -470.000000
25%     24.750000   32.000000   156.750000
50%     49.500000   38.000000   576.000000
75%     74.250000   48.250000  1816.000000
max     99.000000  555.000000  6574.000000
```

	index	age	balance
0	0	60	358.000000
1	1	30	265.000000
2	2	47	1291.834835
3	3	51	1291.834835
4	4	30	873.000000
5	5	41	1141.000000
6	6	58	565.000000
7	7	38	569.000000
8	8	42	-15.000000
9	9	38	569.000000

Normalization with MinMaxScaler(default คือ [0, 1])

- นำ df2['balance'] มา normalize ในช่วง [0, 1]

```
[7]: from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     scaled_balance = scaler.fit_transform(df2['balance'].values.reshape(-1,1))
     scaled_balance[:10]
```

```
[7]: array([[0.11754685],
            [0.10434412],
            [0.25011852],
            [0.25011852],
            [0.19065872],
            [0.22870528],
            [0.14693356],
            [0.14750142],
            [0.06459398],
            [0.14750142]])
```

แปลงให้เป็น array (11,)

แปลงให้เป็น array 2D (11,1)

```
[97]: df2['balance'].values.shape

[97]: (11,)

[99]: df2['balance'].values.reshape(-1,1).shape

[99]: (11, 1)
```

รวม normalized balance เข้าไปใน df2 เพื่อใช้งานต่อไป

[8]:
```
# put the normalized data back into the dataframe
df2['scaled_balance'] = scaled_balance
df2.describe()
```

[8]:

	index	age	balance	scaled_balance
count	100.000000	100.000000	100.000000	100.000000
mean	49.500000	50.660000	1219.786697	0.239890
std	29.011492	73.230813	1515.584999	0.215160
min	0.000000	25.000000	-470.000000	0.000000
25%	24.750000	32.000000	156.750000	0.088976
50%	49.500000	38.000000	576.000000	0.148495
75%	74.250000	48.250000	1816.000000	0.324532
max	99.000000	555.000000	6574.000000	1.000000

Standardize with StandardScaler()

[9]:
```
from sklearn.preprocessing import StandardScaler

# fit on training data column
scale = StandardScaler().fit(df2['balance'].values.reshape(-1,1))
```

[10]:
```
# transform the training data column
stand_balance = scale.transform(df2['balance'].values.reshape(-1,1))
stand_balance[:10]
```

[10]:
```
array([[-0.57148112],
       [-0.6331527 ],
       [ 0.04777766],
       [ 0.04777766],
       [-0.22996648],
       [-0.05224623],
       [-0.43421213],
       [-0.43155959],
       [-0.81883057],
       [-0.43155959]])
```

รวม standardized balance เข้าไปใน df2 เพื่อใช้งานต่อไป

[11]:
```
df2['stand_balance'] = stand_balance
df2.describe()
```

[11]:

	index	age	balance	scaled_balance	stand_balance
count	100.000000	100.000000	100.000000	100.000000	1.000000e+02
mean	49.500000	50.660000	1219.786697	0.239890	-1.110223e-17
std	29.011492	73.230813	1515.584999	0.215160	1.005038e+00
min	0.000000	25.000000	-470.000000	0.000000	-1.120557e+00
25%	24.750000	32.000000	156.750000	0.088976	-7.049371e-01
50%	49.500000	38.000000	576.000000	0.148495	-4.269176e-01
75%	74.250000	48.250000	1816.000000	0.324532	3.953700e-01
max	99.000000	555.000000	6574.000000	1.000000	3.550568e+00

Normalization vs Standardization

- *Normalization* is good to use when you don't know that the distribution of your data follow a normal distribution.
- Outliers can adversely affect.
- Useful in algorithms that do not assume any distribution of the data like Artificial Neural Networks and K-Nearest Neighbors.

- *Standardization* can be helpful where the data follows a normal distribution.
- Standardization does not have a bounding range. So, it is tolerable to outliers.

Useful references

- Wes McKinney, “Python for Data Analysis : Data Wrangling with Pandas, NumPy, and Ipython”, 2nd edition, O’Reilly, 2017.
- https://medium.com/@pichitchai.pim/%E0%B8%A3%E0%B8%A7%E0%B8%A1-code-data-preparation-%E0%B8%94%E0%B9%89%E0%B8%A7%E0%B8%A2-python-57812ac01750
- https://towardsdatascience.com/essential-commands-for-data-preparation-with-pandas-ed01579cf214
- https://www.geeksforgeeks.org/indexing-and-selecting-data-with-pandas
- https://www.geeksforgeeks.org/python-pandas-merging-joining-and-concatenating

Common steps of data preparation in data science

1. Import data and tools
   - Import libraries
   - Read from csv
   - Connect to a database
2. Data inspection
   - Observe data characteristics
   - Deal with missing values
   - Handle (find and remove) duplicate data
   - Handle outliers
3. Data manipulation
   - Column operations (select, rename, drop, sort by column, create new variables)
   - Row operations (slice, filter)
   - Combine dataframes
   - Replace values in a dataframe with condition
   - Discretization (bucket variables)
   - Data encoding
   - Normalization and standardization