



# Data Preparation

## Data preprocessing

Connect the data, Duplicates, Missing values....

Jimgo Hui-Chun Hung

Graduate Institute of Network Learning  
Technology

National Central University

[hch@cl.ncu.edu.tw](mailto:hch@cl.ncu.edu.tw)

# Data Preparation

- ◎ Data Preparation is a key step for data analysis.
- ◎ Data Preparation is vital to good machine learning performance.
- ◎ Garbage in garbage out
  - → data need to be well-prepared, well cleaned up,

# Data Preparation Steps

- ◎ Exploration to understand data problems
  - Remove duplicates.
  - Treat missing values.
  - Treat errors and outliers,
  - Scale the features
  - Split the dataset,
  - Visualization to check results



# Duplicates

- ◎ Duplicate case are overweighed and bias.
- ◎ Identity duplicate cases
  - By unique ID
  - By Value – with caution!
- ◎ Removal strategies
  - Keep most recent (or oldest)
  - Keep first
  - Keep last



# Missing Values

- ◎ Use exploration to detect
- ◎ How are missing values coded?
  - NULL
  - 9999, 0 , NA, ?.....
- ◎ Treatment strategies
  - Remove column with mostly missing values
  - If few rows remove
  - Forward or backward fill
  - Impute: mean, median...



# Errors and Outliers

- ◎ Use exploration to detect
  - Erroneous values ?
  - Important data ?
- ◎ Identify cases with domain knowledge
  - Statistics
  - Visualization
- ◎ Treatment strategies
  - Limit to min-max range
  - Same as missing values





File reading and writing

Missing values

Duplicate values

Data corresponding\Replace\bin

Among two Data tables

A decorative background featuring a network diagram. It consists of numerous nodes, represented by circles of varying sizes and shades of gray, connected by thin, light gray lines. Some nodes are highlighted with a solid blue dot, and others are enclosed in a blue circular outline. The network is more densely packed on the left and right sides of the slide, with the central area being mostly white space containing the title.

# File reading and writing



# File reading and writing

◎ File reading and writing



open()



pandas



# File reading and writing

◎ 檔案開啟 **Open()** the file

◎ Using open()

- `f = open('name', 'model')`

◎ model

- `r` read 讀取(檔案需存在)

- `w` write 新建檔案寫入(檔案可不存在，若存在則清空)

- `a` append 資料附加到舊檔案後面(游標指在EOF)



# File reading and writing

## ◎ read()

- f.read()

- f.read(size)

  - ◎ indicate the size

- f.readline()

  - ◎ Read one line and end with \n

- f.readlines()

  - ◎ Read each line into an item in a list



Hui-Chun Hung

Python程式語言起步走~  
使用 Python 來做機器學習初探

# File reading and writing

◎ 檔案寫入 **writing a file**

◎ `open()...write()`

- `f = open('name', 'a')`
- `f.write("write something....")`
- `f.close()`



# File reading and writing

## ◎ Via pandas and read as a Dataframe

- import pandas as pd
- df = pd.read\_csv("file name.csv")
- df.to\_csv("file name.csv")

◎ In Chinese, the coding error...

◎ df = pd.read\_csv("檔名.csv", encoding='big5')



# File reading and writing

Format Type	Data Description	Reader	Writer
text	<a href="#">CSV</a>	<a href="#">read_csv</a>	<a href="#">to_csv</a>
text	<a href="#">JSON</a>	<a href="#">read_json</a>	<a href="#">to_json</a>
text	<a href="#">HTML</a>	<a href="#">read_html</a>	<a href="#">to_html</a>
text	Local clipboard	<a href="#">read_clipboard</a>	<a href="#">to_clipboard</a>
binary	<a href="#">MS Excel</a>	<a href="#">read_excel</a>	<a href="#">to_excel</a>
binary	<a href="#">HDF5 Format</a>	<a href="#">read_hdf</a>	<a href="#">to_hdf</a>
binary	<a href="#">Feather Format</a>	<a href="#">read_feather</a>	<a href="#">to_feather</a>
binary	<a href="#">Msgpack</a>	<a href="#">read_msgpack</a>	<a href="#">to_msgpack</a>
binary	<a href="#">Stata</a>	<a href="#">read_stata</a>	<a href="#">to_stata</a>
binary	<a href="#">SAS</a>	<a href="#">read_sas</a>	
binary	<a href="#">Python Pickle Format</a>	<a href="#">read_pickle</a>	<a href="#">to_pickle</a>
SQL	<a href="#">SQL</a>	<a href="#">read_sql</a>	<a href="#">to_sql</a>
SQL	<a href="#">Google Big Query</a>	<a href="#">read_gbq</a>	<a href="#">to_gbq</a>

<http://pandas.pydata.org/pandas-docs/stable/io.html>



Hui-Chun Hung  
Python 程式語言起步走~  
使用 Python 來做機器學習初探

A decorative background featuring a network diagram. It consists of numerous nodes, represented by circles of varying sizes and shades of gray, connected by thin, light gray lines. Some nodes are highlighted with a solid blue dot, and others are enclosed in a blue outline. The network is distributed across the slide, with a denser cluster on the left side and more sparse connections towards the right and bottom.

# Deal with Missing values

# Deal with Missing values

- ◎ First determine if there is missing data NaN
- ◎ and the distribution of missing values
  - `df1.isnull()`
  - `df1.isnull().sum()`





# Deal with Missing values

## ◎ Method 1:

- 是將 NaN 的值用其他值代替, 比如代替成 0

- ◎ `df2 = df1.fillna(value=0)`

- ◎ `df2`



# Deal with Missing values

## ◎ Method 2:

- 將 NaN 值刪除

- ◎ df3 = df1.dropna()

- ◎ df3

## ◎ #其他用法

- ◎ df1.dropna(subset=['欄位名稱']) #針對特定欄位名稱

- ◎ df1.dropna(axis=0) # 0: 對資料列進行操作; 1: 對該欄位進行操作

- ◎ df1.dropna(how='any') #'any': 存在 NaN 就 drop 掉; '  
all': 全是 NaN 才 drop

- ◎ df1.dropna(thresh=3) #至少要有thresh個非NaN值



# Deal with Missing values

## ◎ Method 3:

### ○ 差補法 (使用sklearn)

◎ `from sklearn.preprocessing import Imputer`

◎ `imr = Imputer(missing_values='NaN', strategy=_____, axis=0)`

◎ `imr = imr.fit(df1)`

◎ `imputed_data = imr.transform(df1.values)`

◎ `imputed_data`

◎ `# strategy='mean', 'median', 'most_frequent'`



A decorative background graphic consisting of a network of nodes and edges. The nodes are represented by circles of varying sizes and colors, including light gray, dark gray, and blue. Some nodes are highlighted with a blue outline. The edges are thin gray lines connecting the nodes, forming a complex, interconnected web. The overall style is clean and modern, typical of a technical or data-related presentation.

# Remove the duplicates

# Remove the duplicates

## ◎ Check first

- `df.duplicated()`

## ◎ Remove the duplicates

- `df.drop_duplicates()`

- ◎ 針對特定欄位：可加上欄位名稱與保留順序

- ◎ #例如：`df.drop_duplicates(['col1'], keep='last')`



A decorative background featuring a network diagram with nodes and edges. The nodes are represented by circles of varying sizes and colors (blue, grey, white), connected by thin lines. Some nodes are highlighted with blue outlines. The network is distributed across the top-left and bottom-right corners of the slide.

# Data map/replace/ bin

# map

- ◎ 新建一個字典
  - {key01 : value01, key02 : value02,.....}
  - 用字典裡的value取代原本的key(原本資料表的值)
- ◎ 在特定欄位用map()執行
  - `df["欄位"].map({dict})`
- ◎ 例如：將性別欄位(女、男)變成(0、1)
  - `gender_to_boolean = {"female":0,"male":1} #字典`
  - `df['gender'].map(sex_to_boolean)`



# replace

- ◎ 取代請愛用replace()
- ◎ df.replace(原本的值, 新的值)
  - `df['col2'].replace("-",0)`
- ◎ df.replace(也可以用字典方式)
  - `df.replace({"NULL":0, "-":-1})`





# bin

## ◎ Two list

- bins

- ◎ 分箱相的間隔點list

- labels

- ◎ 各區間對應的labels

- `bins = [0, 60, 70, 80, 90, 100]`

- `labels = ['E','D','C','B','A']`

- `pd.cut(df['score'],bins, right=False, labels=labels)`



A decorative background graphic consisting of a network of nodes and edges. The nodes are represented by circles of varying sizes and colors (gray, blue, and white with blue outlines). The edges are thin gray lines connecting the nodes. The network is more dense on the left and bottom edges of the slide, with some nodes highlighted in blue or circled in blue.

# Merge two dataframe

# Concatenating dataframes

- ◎ *#concat*
- ◎ import pandas as pd
- ◎ import numpy as np
  
- ◎ *#three dfs*
- ◎ df1 = pd.DataFrame(np.ones((3,4))\*0, columns=['a','b','c','d'])
- ◎ df2 = pd.DataFrame(np.ones((3,4))\*1, columns=['a','b','c','d'])
- ◎ df3 = pd.DataFrame(np.ones((3,4))\*2, columns=['a','b','c','d'])
  
- ◎ *#concatenating*
- ◎ res = pd.concat([df1, df2, df3], axis=0, ignore\_index=True)



# Merging dataframes

- ◎ *# merge*
- ◎ import pandas as pd
- ◎ left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],  
                          'A': ['A0', 'A1', 'A2', 'A3'],  
                          'B': ['B0', 'B1', 'B2', 'B3']})
- ◎ right = pd.DataFrame({'key': ['K1', 'K2', 'K3', 'K4'],  
                          'C': ['C0', 'C1', 'C2', 'C3'],  
                          'D': ['D0', 'D1', 'D2', 'D3']})
- ◎ res = pd.merge(left, right, on='key')



# Merging dataframes

- ◎ *# merge*
- ◎ import pandas as pd
- ◎ left = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],  
                          'key2': ['K0', 'K1', 'K0', 'K1'],  
                          'A': ['A0', 'A1', 'A2', 'A3'], 'B': ['B0', 'B1', 'B2', 'B3']})
- ◎ right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],  
                          'key2': ['K0', 'K0', 'K0', 'K0'],  
                          'C': ['C0', 'C1', 'C2', 'C3'], 'D': ['D0', 'D1', 'D2', 'D3']})
- ◎ *# join by key with different name with ['left', 'right', 'outer', 'inner']*
- ◎ res = pd.merge(left, right, on=['key1', 'key2'], how='inner')





# Q & A

[hch@cl.ncu.edu.tw](mailto:hch@cl.ncu.edu.tw)