

*Introduction to Kaggle*

*Analytical and Critical  
Thinking*

*Intermediate Dataframe*

*Learning Progress Review Week 8*

By:  
**MARVEL TEAM**

Fikrie | Natalia | Satria

# 1. *Introduction to Kaggle*

## Kenapa kita menggunakan Kaggle sebagai *Data Scientist*?



- ❖ Komunitas *Data Science* terbesar di dunia berkumpul di satu platform bernama Kaggle.
- ❖ Banyak nya *Dataset* yang dapat digunakan untuk berlatih sebagai seorang *Data Scientist*.

## Kenapa kita menggunakan Kaggle sebagai *Data Scientist*?

- ❖ Tempatnya kompetisi *Data Science* mulai dari *Beginner* hingga *Expert*.
- ❖ Dapat digunakan sebagai platform diskusi antar Kagglers dari berbagai industri.
- ❖ Dapat mengikuti course yang telah disediakan oleh Kaggle.

## Sejarah Kaggle

- ❖ Kaggle dibangun pada Bulan April 2010 dan Pendiri nya adalah Anthony Goldbloom dan Ben Hammer.
- ❖ Pada awalnya Kaggle hanya sebagai penyedia platform penyedia kompetisi Machine Learning.
- ❖ Pada 8 Maret 2017 Google resmi mengakusisi Kaggle dan sekarang Kaggle adalah anak Perusahaan dari Google.

## Apa itu Kaggle ?

- ❖ Kaggle adalah sebuah komunitas online yang di bentuk oleh Anthony Goldbloom sebagai CEO dan Ben Hammer sebagai CTO di tahun 2010.
- ❖ Komunitas online ini menampung para pegiat Data Science yang ingin belajar lebih dalam tentang Machine Learning dan ilmu-ilmu terkait lainnya.
- ❖ Di dalam Kaggle terdapat berbagai banyak kegiatan yang dilakukan, salah satunya adalah kompetisi Machine Learning.

## *Kaggle Service*

- ❖ Data : Ada banyak dataset yang bisa digunakan di Kaggle. Dataset ini dapat memudahkan kita untuk membuat modelling yang ingin kita kembangkan.
- ❖ Code : Ada banyak sekali jumlah kode yang tersedia di Kaggle. Kode ini adalah hasil kontribusi dari para pengguna yang tergabung di Kaggle
- ❖ Komunitas : Di Kaggle, tergabung banyak Data Analyst, Data Scientist, dan Machine Learning Engineer yang siap berbagi ilmu.

## *Kaggle Service*

- ❖ Kompetisi : Kompetisi di Kaggle adalah salah satu yang paling di tunggu-tunggu oleh para kagglers.
- ❖ Course : Kaggle juga menyediakan materi yang bisa dipelajari bagi para data scientist pemula



## Bagaimana mengoptimalkan Kaggle?

- ❖ Belajar *course Data Science* yang telah di sediakan Kaggle.
- ❖ Meng-*explore dataset* yang di sediakan Kaggle.
- ❖ Dapat berdiskusi sesama Kagglers lainnya.
- ❖ Ikut berpartisipasi dalam kompetisi yang di sediakan Kaggle.
- ❖ Kita juga dapat Upload dataset dan Notebook yang kita miliki agar dapat dilihat oleh Kagglers lainnya.

## **2.** *Analytical and Critical Thinking*

*Design Thinking for Data Science*

# *Apa itu Design Thinking?*

## **Pengertian :**

Sebuah cara berpikir yang mengedepankan cara pandang pengguna dalam menyelesaikan masalah terkait bisnis yang tengah dijalani.

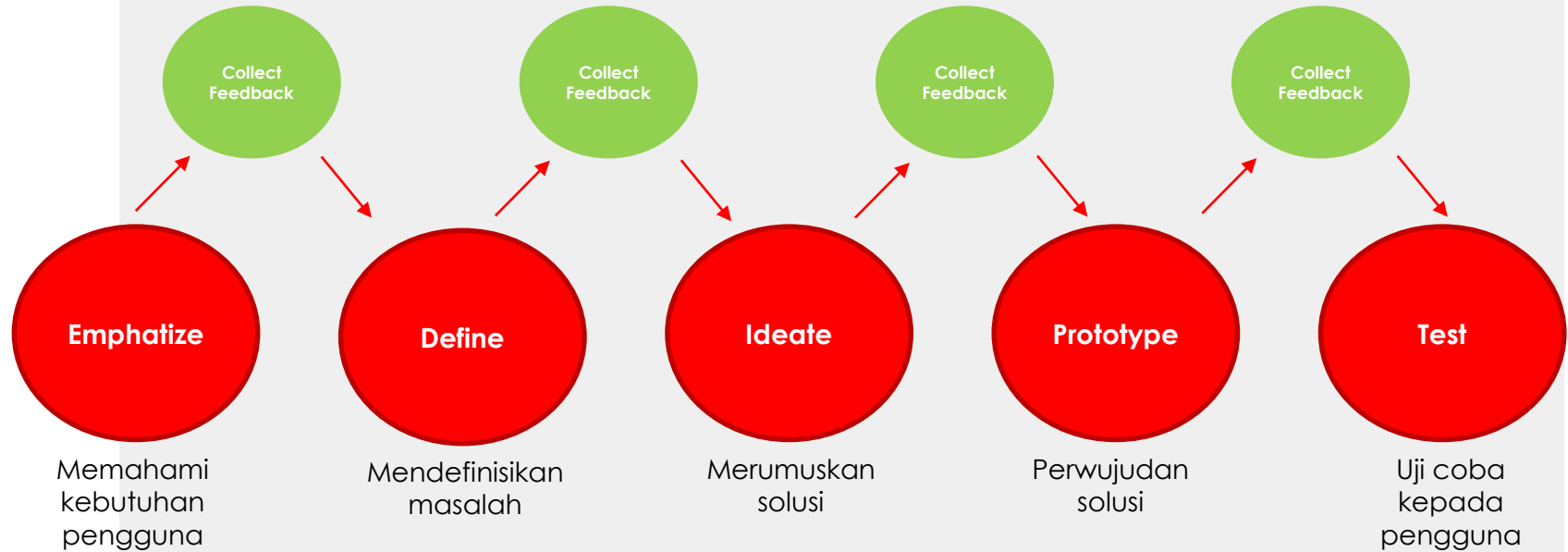
## **Bukan tentang :**

- Kita sudah memiliki teknologi, apa yang bisa kita lakukan dengannya?
- Lawan bisnis kita sudah mengeluarkan produk baru, bagaimana cara kita menyainginya?
- Kita mempunyai masalah, ayo kita rapat dan menyelesaikan masalahnya.

## **Akan tetapi tentang :**

- Apa yang pengguna butuhkan?
- Untuk siapa solusi ini kita rumuskan?
- Bagaimana kita bisa menyelesaikan masalah yang dialami pengguna?
- Bagaimana kita bisa berinovasi berdasarkan masalah yang dialami pengguna?

## *Siklus Design Thinking*



# *Emphasize*

**Pengertian :**

Memahami dan merasakan yang dialami pengguna sehingga merasa tergerak untuk melakukan sesuatu (solusi yang tepat).

**Memulai dengan :**

- Untuk siapa saya membuat solusi ini?
- Apa sebenarnya masalah pengguna?
- Apa yang sebenarnya dilakukan pengguna selama ini?

**Aktivitas kunci :**

- Mendengarkan
- Menyatu dengan pengguna
- Mengamati

# *The Ways to Emphatize*

<b>Asumsi Pengguna Produk Pemula</b>	<b>Pertanyaan Mendasar</b>	<b>Peta Empati</b>
<ul style="list-style-type: none"> <li>• Lupakan asumsi dan pengetahuan personal.</li> <li>• Gunakan perspektif yang baru.</li> <li>• Tanyakan segala hal.</li> <li>• Jangan menjustifikasi atau mensimplifikasi sesuatu terlebih dahulu.</li> </ul>	<ul style="list-style-type: none"> <li>• Mulai mencerna hubungan sebab – akibat.</li> <li>• Fokus pada sistemnya, bukan personalnya.</li> <li>• Abaikan istilah 'human error', 'kesalahan personal' dll.</li> <li>• Selalu gunakan kacamata pengguna.</li> </ul>	<ul style="list-style-type: none"> <li>• Say : segala sesuatu yang diucapkan oleh pengguna.</li> <li>• Think : kira-kira seperti apa yang pengguna pikirkan.</li> <li>• Do : apa yang dilakukan pengguna ketika kita berikan eksperimen.</li> <li>• Feel : apa yang sebenarnya pengguna rasakan.</li> </ul>

# Define

**Pengertian :**

Mendefinisikan masalah sebenarnya dengan mengklarifikasi melalui pertanyaan,

- Apa sebenarnya yang dibutuhkan pengguna?
- Apa sebenarnya masalah yang dialami pengguna?
- Apa sebenarnya tantangan yang dialami pengguna?
- Masukan apa yang bisa saya gunakan untuk memecahkan masalah?

**Aktivitas kunci :**

- Menyatukan aktivitas *Emphatize* dan *Define* untuk mendapatkan masukan-masukan penting dan mulai menginisiasi solusi.

# *Ideate*

**Pengertian :**

Mengelaborasi asumsi kemudian mengkreasikan ide.

**Aktivitas kunci :**

- Mencari kesinambungan masukan – masukan yang diperoleh dari tahap sebelumnya.
- Menciptakan solusi dan ide sebanyak-banyaknya.
- *Brainstorming* mengenai solusi dan ide yang akan dilakukan dengan tim.



# Prototype

**Pengertian :**

Mulai membentuk wujud nyata solusi dari *brainstorming* yang telah didiskusikan.

**Aktivitas kunci :**

- Membentuk produk yang visibilitasnya minimum agar mudah dilaksanakan.
- Berbiaya murah.
- Mudah diterjemahkan untuk pengembangan selanjutnya.

# Test

**Pengertian :**

Produk diujicobakan kepada pengguna sebenarnya.

**Aktivitas kunci :**

- Produk diujicoba.
- Memperhatikan setiap saran & kritik yang muncul.
- *Iterative*, memungkinkan adanya pengulangan siklus bagi *design thinking*.

## Quotes

*As a Data Scientist, you need to understand your audience even better than they understand themselves. But, the only way you'll get there is to develop a deep empathy for their habits, beliefs, quirks, workarounds, etc.*

### 3. *Intermediate Dataframe*

## Sorting in Dataframe

# Syntax untuk mengurutkan Dataframe

```
DataFrame.sort_values(by, axis=0, ascending=True, inplace=False,  
kind='quicksort', na_position='Last', ignore_index=False, key=None)
```

Parameter:

- *by* : List (index/baris atau kolom yang ingin diurutkan).
- *axis* : axis yang akan diurutkan, default 0. Pilihannya adalah 0 (index), 1 (kolom).
- *ascending* : ingin urutan dari kecil ke besar, default True. Jika ingin kebalikannya, dibuat *ascending=False*.
- *inplace* : untuk me-replace value yang lama ke hasil urutan yang baru, default False.
- *na\_position* : meletakkan NaN di awal atau akhir, default 'last'.

## Sorting in Dataframe

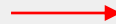
# Mengurutkan berdasarkan 1 kolom

Sebelum diurutkan

df

	col1	col2	col3	col4
0	A	2	0	a
1	A	1	1	B
2	B	9	9	c
3	NaN	8	4	D
4	D	7	2	e
5	C	4	3	F



Setelah diurutkan

df.sort\_values(by='col1')

	col1	col2	col3	col4
0	A	2	0	a
1	A	1	1	B
2	B	9	9	c
5	C	4	3	F
4	D	7	2	e
3	NaN	8	4	D

Diurutkan dari kecil ke besar. Jika ingin kebalikannya, syntax menjadi:

```
df.sort_values(by='col1', ascending=False)
```

NaN = 0, karena default na\_position 'last', jadi NaN diletakkan terakhir

## Sorting in Dataframe

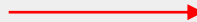
# Mengurutkan berdasarkan lebih dari 1 kolom

Sebelum diurutkan

df

	col1	col2	col3	col4
0	A	2	0	a
1	A	1	1	B
2	B	9	9	c
3	NaN	8	4	D
4	D	7	2	e
5	C	4	3	F



Setelah diurutkan

df.sort\_values(by=['col1', 'col2'])

	col1	col2	col3	col4
1	A	1	1	B
0	A	2	0	a
2	B	9	9	c
5	C	4	3	F
4	D	7	2	e
3	NaN	8	4	D

# Filtering Dataframe

## Memilih beberapa kolom

Cara Pertama

```
df[ ['sepal_length', 'sepal_width'] ].head(5)
```



	sepal_length	sepal_width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6



*head(n)* : untuk  
mendapatkan baris  
paling atas

*tail(n)* : untuk  
mendapatkan baris  
paling bawah

n = jumlah baris  
yang diinginkan



## Filtering Dataframe

# Memilih beberapa kolom

Cara Kedua

`df.filter(items = ['sepal_length', 'sepal_width'])` → Memakai atribut *filter*

↗

	sepal_length	sepal_width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6

## *Filtering Dataframe*

### **iloc dan loc**

**loc** : mengambil data berdasarkan nama kolom

**iloc** : mengambil data berdasarkan urutan index/baris


iloc = integer location

## Filtering Dataframe

# Filter menggunakan loc berdasarkan 1 kondisi

Kondisi yang diinginkan

```
df.loc[df.flower_class == 'Iris-setosa'].head()
```



	sepal_length	sepal_width	petal_length	petal_width	flower_class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

## Filtering Dataframe

# Filter menggunakan loc berdasarkan beberapa kondisi

Kondisi pertama

Kondisi kedua

```
df.loc[ (df.flower_class == 'Iris-setosa') & (df.sepal_length == 5.1) ]
```

↳

	sepal_length	sepal_width	petal_length	petal_width	flower_class
0	5.1	3.5	1.4	0.2	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
39	5.1	3.4	1.5	0.2	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa
46	5.1	3.8	1.6	0.2	Iris-setosa

## Operator

& : dan berarti harus memenuhi seluruh kondisi

or : atau berarti memenuhi salah satu kondisi

## Filtering Dataframe

**Filter** berdasarkan beberapa kondisi dan mengambil kolom yang diinginkan

Kondisi

Kolom yang diinginkan

```
df.loc[ (df.flower_class == 'Iris-setosa') & (df.sepal_length == 5.1) , ['flower_class', 'sepal_length']]
```



	flower_class	sepal_length
0	Iris-setosa	5.1
17	Iris-setosa	5.1
19	Iris-setosa	5.1
21	Iris-setosa	5.1
23	Iris-setosa	5.1
39	Iris-setosa	5.1
44	Iris-setosa	5.1
46	Iris-setosa	5.1

## Filtering Dataframe

### Update value di 1 kolom

Kondisi                      Kolom yang value-nya ingin diubah                      Isi value yang baru

```
df.loc[(df.flower_class == 'Iris-setosa'), ['flower_class']] = 'Flower Class V2'
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	flower_class
0	5.1	3.5	1.4	0.2	Flower Class V2
1	4.9	3.0	1.4	0.2	Flower Class V2
2	4.7	3.2	1.3	0.2	Flower Class V2
3	4.6	3.1	1.5	0.2	Flower Class V2
4	5.0	3.6	1.4	0.2	Flower Class V2

## Filtering Dataframe

## Update value di lebih dari 1 kolom

Kondisi                      Kolom yang value-nya ingin diubah                      Isi value yang baru

```
df.loc[(df.flower_class == 'Flower Class V2'), ['petal_length', 'sepal_length']] = [100, -99]
df.head()
```

↳

	sepal_length	sepal_width	petal_length	petal_width	flower_class
0	-99.0	3.5	100.0	0.0	Flower Class V2
1	-99.0	3.0	100.0	0.0	Flower Class V2
2	-99.0	3.2	100.0	0.0	Flower Class V2
3	-99.0	3.1	100.0	0.0	Flower Class V2
4	-99.0	3.6	100.0	0.0	Flower Class V2

## Filtering Dataframe

### Filter menggunakan *iloc*

`df.iloc[[0,3]]` → Mengambil beberapa baris

	sepal_length	sepal_width	petal_length	petal_width	flower_class
0	5.1	3.5	1.4	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa

`df.iloc[0:3]` → Mengambil berdasarkan *range* baris

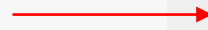
	sepal_length	sepal_width	petal_length	petal_width	flower_class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa



## *Filtering Dataframe*

### **Filter menggunakan *iloc***

```
df.iloc[[0,2],[1,3]]
```



Mengambil baris dan kolom



	sepal_width	petal_width
0	3.5	0.2
2	3.2	0.2

# Creating Additional Column

## Menambah kolom

Cara Pertama

Nama kolom baru

```
df['sepal_length_v2'] = df['sepal_length'] * 100
df
```

	sepal_length	sepal_width	petal_length	petal_width	flower_class	sepal_length_v2
0	5.1	3.5	1.4	0.2	Iris-setosa	510.0
1	4.9	3.0	1.4	0.2	Iris-setosa	490.0
2	4.7	3.2	1.3	0.2	Iris-setosa	470.0
3	4.6	3.1	1.5	0.2	Iris-setosa	460.0
4	5.0	3.6	1.4	0.2	Iris-setosa	500.0

## Creating Additional Column

# Menambah kolom menggunakan loc

Cara Kedua

: berarti memilih  
seluruh data

```
df.loc[:, 'sepal_length_v3'] = df['sepal_length'] * 100
```

	sepal_length	sepal_width	petal_length	petal_width	flower_class	sepal_length_v2	sepal_length_v3
0	5.1	3.5	1.4	0.2	Iris-setosa	510.0	510.0
1	4.9	3.0	1.4	0.2	Iris-setosa	490.0	490.0
2	4.7	3.2	1.3	0.2	Iris-setosa	470.0	470.0
3	4.6	3.1	1.5	0.2	Iris-setosa	460.0	460.0
4	5.0	3.6	1.4	0.2	Iris-setosa	500.0	500.0

## Creating Additional Column

# Menambah kolom berdasarkan kondisi tertentu

Kondisi      Nama kolom baru      Isi value

```
df.loc[ (df.sepal_length < 5), 'sepal_length_type'] = 'Small'
df.loc[ (df.sepal_length > 5), 'sepal_length_type'] = 'Big'
df.loc[ (df.sepal_length == 5), 'sepal_length_type'] = 'Equal'

df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	flower_class	sepal_length_v2	sepal_length_v3	sepal_length_type
0	5.1	3.5	1.4	0.2	Iris-setosa	17.85	510.0	Big
1	4.9	3.0	1.4	0.2	Iris-setosa	14.70	490.0	Small
2	4.7	3.2	1.3	0.2	Iris-setosa	15.04	470.0	Small
3	4.6	3.1	1.5	0.2	Iris-setosa	14.26	460.0	Small
4	5.0	3.6	1.4	0.2	Iris-setosa	18.00	500.0	Equal

## Grouping Dataframe

# Memisahkan data ke dalam group terpisah

**Groupby di Pandas tidak akan mengeluarkan data apapun sampai kita menyatakan atribut *aggregate* setelah groupby.**

Syntax  
groupby  
(nama\_kolom)

Atribut  
aggregate

Karena tidak disebutkan nama kolom yang di-*aggregate*, maka yang ditampilkan *value* dari seluruh kolom

▶ `df.groupby('flower_class').count()` →

📄

	sepal_length	sepal_width	petal_length	petal_width	sepal_length_v2	sepal_length_v3	sepal_length_type
flower_class							
Iris-setosa	50	50	50	50	50	50	50
Iris-versicolor	50	50	50	50	50	50	50
Iris-virginica	50	50	50	50	50	50	50

## Grouping Dataframe

# Groupby pada kolom yang di-aggregate

Syntax  
groupby(nama\_kolom)

Kolom yang  
di-aggregate

Atribut  
aggregate

▶ `df.groupby('flower_class')['sepal_length'].mean()`

```
↳ flower_class
Iris-setosa      5.006
Iris-versicolor  5.936
Iris-virginica   6.588
Name: sepal_length, dtype: float64
```

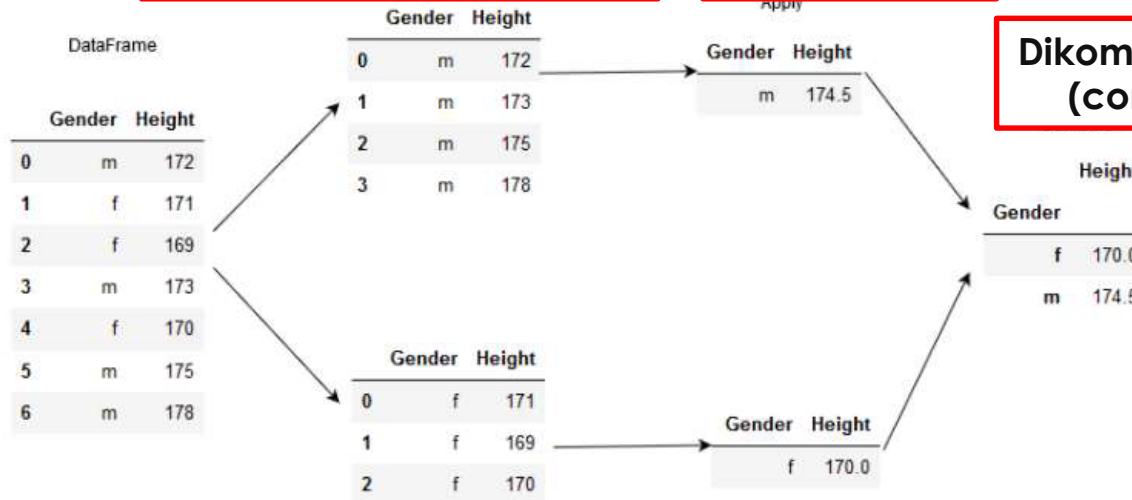
# Grouping Dataframe

## Konsep grouping dataframe

Dipisahkan berdasarkan  
jenis value didalamnya  
(*split*)

Aggregate  
diaplikasikan  
(*apply*)

Dikombinasikan  
(*combine*)



Sumber:  
analyticsvidhya

## Grouping Dataframe

# Mengkombinasikan Groupby dan atribut aggregate lebih dari satu

Atribut aggregate  
agg(nama\_kolom\_hasil\_  
aggregate)

Atribut aggregate  
yang diinginkan

```
df.groupby('flower_class').agg(average_sepal_length_per_class = ('sepal_length', 'mean'),
                              median_sepal_length_per_class = ('sepal_length', 'median'),
                              sum_sepal_length_per_class = ('sepal_length', 'sum'))
```

↳

	average_sepal_length_per_class	median_sepal_length_per_class	sum_sepal_length_per_class
flower_class			
Iris-setosa	5.006	5.0	250.3
Iris-versicolor	5.936	5.9	296.8
Iris-virginica	6.588	6.5	329.4



## Grouping Dataframe

# Grouping menggunakan *pivot\_table* dengan 1 value

Data : DataFrame      Values : kolom yang di-aggregate, optional      index : kolom, grouper      aggfunc : daftar fungsi

```
pd.pivot_table(df, values = 'sepal_width', index = 'flower_class', aggfunc = ['mean', 'median'])
```

```
↗
```

	mean	median
	sepal_width	sepal_width
flower_class		
Iris-setosa	3.418	3.4
Iris-versicolor	2.770	2.8
Iris-virginica	2.974	3.0

## Grouping Dataframe

# Grouping menggunakan *pivot\_table* dengan lebih dari 1 value

Data : DataFrame

Values : kolom yang di-aggregate, optional

index : kolom, grouper

aggfunc : daftar fungsi

```
pd.pivot_table(df, values = ['sepal_width', 'sepal_length'], index = 'flower_class', aggfunc = ['mean', 'median'])
```

flower_class	mean		median	
	sepal_length	sepal_width	sepal_length	sepal_width
Iris-setosa	5.006	3.418	5.0	3.4
Iris-versicolor	5.936	2.770	5.9	2.8
Iris-virginica	6.588	2.974	6.5	3.0

## Grouping Dataframe

# Groupby dengan lebih dari 1 kolom

Nama kolom-kolom  
yang di-groupby

Atribut aggregate

```
df.groupby(['flower_class', 'sepal_length_type']).agg(average_sepal_length_per_class = ('sepal_length', 'mean'),
median_sepal_length_per_class = ('sepal_length', 'median'),
sum_sepal_length_per_class = ('sepal_length', 'sum') )
```

		average_sepal_length_per_class	median_sepal_length_per_class	sum_sepal_length_per_class
flower_class	sepal_length_type			
Iris-setosa	Big	5.313636	5.25	116.9
	Equal	5.000000	5.00	40.0
	Small	4.670000	4.70	93.4
Iris-versicolor	Big	5.997872	6.00	281.9
	Equal	5.000000	5.00	10.0
	Small	4.900000	4.90	4.9
Iris-virginica	Big	6.622449	6.50	324.5
	Small	4.900000	4.90	4.9

# Merging Dataframe

## Concatenate : menggabungkan tabel

Syntax :

```
pd.concat([df1, df2, df3])
```

Menggabungkan tabel  
ke bawah

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

df3				
	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Result					
		A	B	C	D
x	0	A0	B0	C0	D0
x	1	A1	B1	C1	D1
x	2	A2	B2	C2	D2
x	3	A3	B3	C3	D3
y	4	A4	B4	C4	D4
y	5	A5	B5	C5	D5
y	6	A6	B6	C6	D6
y	7	A7	B7	C7	D7
z	8	A8	B8	C8	D8
z	9	A9	B9	C9	D9
z	10	A10	B10	C10	D10
z	11	A11	B11	C11	D11

## Merging Dataframe

### Concatenate : menggabungkan tabel

Syntax :

`pd.concat([df1, df4, axis = 1])`

Menggabungkan  
tabel ke samping

df1					df4				Result							
										A	B	C	D	B	D	F
0	A0	B0	C0	D0	2	B2	D2	F2	0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	3	B3	D3	F3	1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	6	B6	D6	F6	2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	7	B7	D7	F7	3	A3	B3	C3	D3	B3	D3	F3
									6	NaN	NaN	NaN	NaN	B6	D6	F6
									7	NaN	NaN	NaN	NaN	B7	D7	F7

## Merging Dataframe

### Syntax untuk menggabungkan (merge)

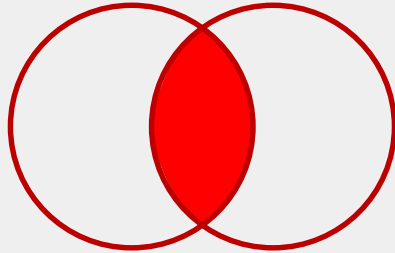
```
DataFrame.merge(right, how='inner', on=None, left_on=None,  
right_on=None, left_index=False, right_index=False, sort=False,  
suffixes=('_x', '_y'), copy=True, indicator=False, validate=None)
```

Parameter:

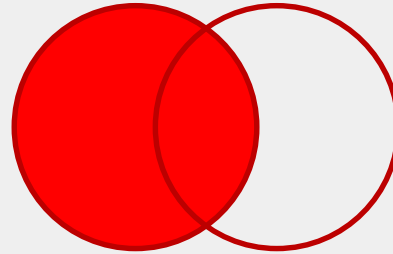
- *right* : DataFrame
- *how* : tipe *merge* yang akan dipakai
- *on* : kolom atau *index* yang digabung
- *left\_on* : nama kolom atau *index* yang digabung ke sisi kiri
- *right\_on* : nama kolom atau *index* yang digabung ke sisi kanan

# Merging Dataframe

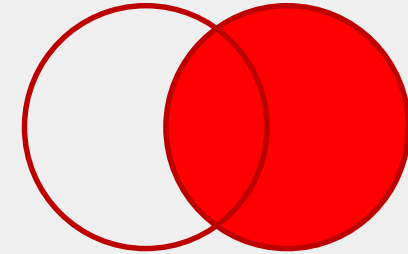
## Tip –tipe merge



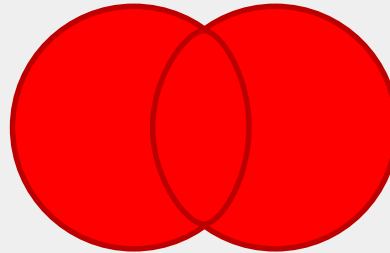
Inner Join



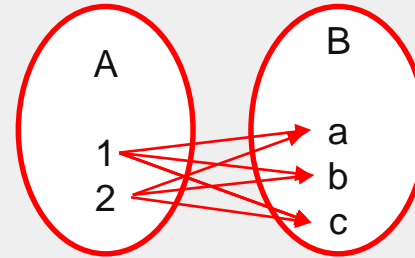
Left Join



Right Join



Outer Join



Cross Join

Special Thanks to :



Slide template by SlideCarnival