

Data Wrangling

Data Wrangling adalah suatu proses mengubah data mentah menjadi sebuah data yang rapi untuk siap dianalisis

Data Wrangling merupakan suatu tahapan penting dalam data preprocessing dan mencakup beberapa proses seperti :

1. data importing
2. data cleaning
3. data structuring
4. string processing
5. HTML parsing
6. handling dates and times
7. handling missing data
8. text mining

Data Importing

```
In [1]: ## Import Package
import pandas as pd
```

```
In [2]: #membaca sebuah data menggunakan pandas
data = pd.read_csv('Downloads\Data\shopping_data.csv')
df = pd.DataFrame(data)
```

```
In [3]: ## Menampilkan informasi statistik dengan Numpy
import numpy as np
df.describe(include='all')
```

	CustomerID	Genre	Age	Annual Income (k\$)	Spending Score (1-100)
count	200.000000	200	200.000000	200.000000	200.000000
unique	NaN	2	NaN	NaN	NaN
top	NaN	Female	NaN	NaN	NaN
freq	NaN	112	NaN	NaN	NaN
mean	100.500000	NaN	38.850000	60.560000	50.200000
std	57.879185	NaN	13.969007	26.264721	25.823522
min	1.000000	NaN	18.000000	15.000000	1.000000
25%	50.750000	NaN	28.750000	41.500000	34.750000
50%	100.500000	NaN	36.000000	61.500000	50.000000
75%	150.250000	NaN	49.000000	78.000000	73.000000
max	200.000000	NaN	70.000000	137.000000	99.000000

setelah menampilkan informasi data di atas, terlihat terdapat format NaN pada dataset tersebut.

Data Cleaning

Untuk meminimalisir hal tersebut dan memfilter hanya data numerical saja, digunakan `exclude=["O"]`, dimana fungsi itu akan mengabaikan data yang non-numerical untuk diproses.

```
In [4]: df.describe(exclude=[ 'O']))
```

Out[4]:

	CustomerID	Age	Annual Income (k\$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

Pengecekan Nilai NULL yang ada

```
In [5]: print(df.isnull().values.any())
```

False

karena output yang dihasilkan false, maka dataset yang digunakan merupakan data yang lengkap.

Kemudian coba kita gunakan data dari dataset milik DQLab berikut

```
In [6]: csv_data = pd.read_csv("https://storage.googleapis.com/dqlab-dataset/shopping_data.csv")
print(csv_data)
```

	CustomerID	Genre	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19.0	15.0	39.0
1	2	Male	NaN	15.0	81.0
2	3	Female	20.0	NaN	6.0
3	4	Female	23.0	16.0	77.0
4	5	Female	31.0	17.0	NaN
..
195	196	Female	35.0	120.0	79.0
196	197	Female	45.0	126.0	28.0
197	198	Male	32.0	126.0	74.0
198	199	Male	32.0	137.0	18.0
199	200	Male	30.0	137.0	83.0

[200 rows x 5 columns]

```
In [7]: print(csv_data.isnull().values.any())
```

True

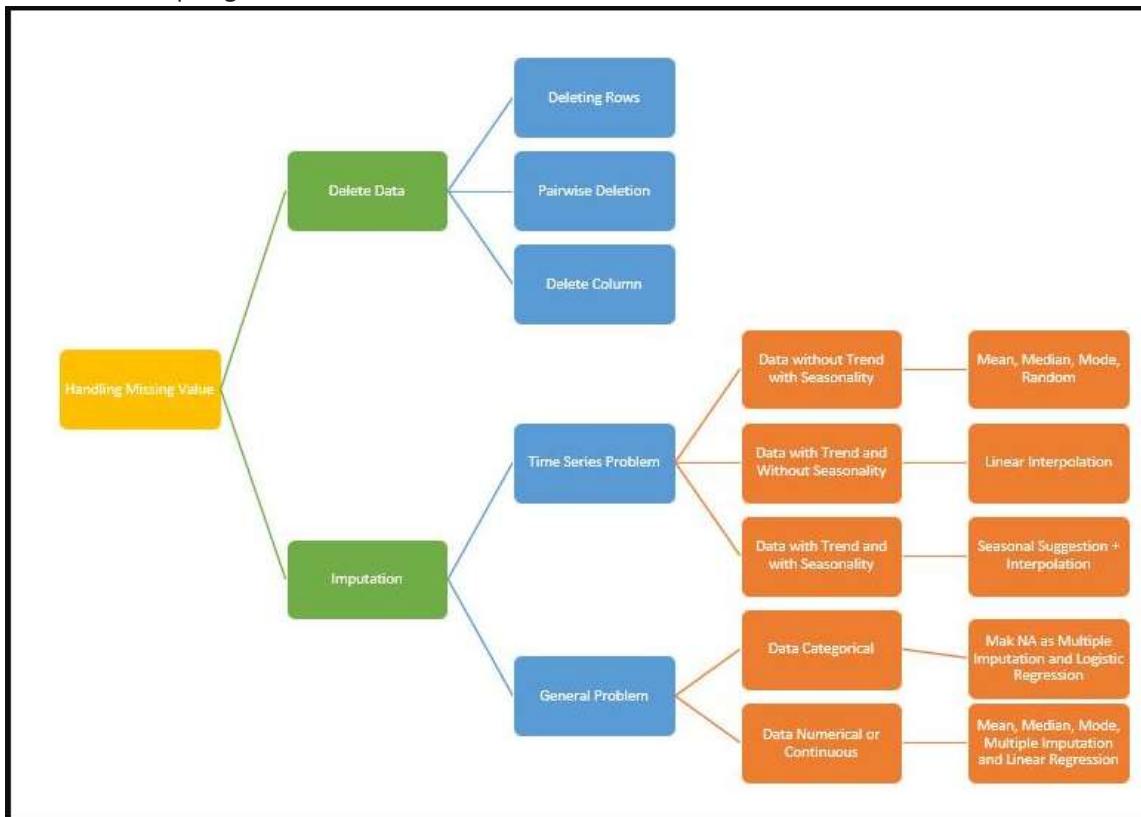
Nah, output yang dihasilkan adalah true kan. Oke sekarang kita proses data yang missing ini

!

```
In [8]: #kita berinama data tersebut dengan Shopping Data
ShoppingData = csv_data
```

Solusi mengisi Missing Value

skema dalam pengolahan data



Dalam diagram diatas, perlu diketahui bahwa kasus kehilangan data bisa diatasi dengan berbagai cara. Bahkan, melakukan penghapusan data juga merupakan solusi yang bisa menjadi pilihan apabila jika dirasa mengisi nilai kosong akan memberikan pengaruh yang kurang bagus terhadap analisa, atau apabila pertimbangan data yang dihapus atau data yang hilang sedikit dan tidak memberikan terlalu banyak sumbangsih untuk analisa yang akan dilakukan. Penghapusan data bisa langsung pada baris data tersebut atau langsung satu kolom data. Pada solusi kedua yaitu menggunakan imputation (pengisian data yang kosong) bisa tergantung dari permasalahannya. Khusus untuk masalah yang berhubungan forecasting atau peramalan tergantung dari data yang ada (lebih lengkap bisa dilihat pada gambar). Khusus untuk general problem tergantung jenis datanya. Jika yang hilang data kategorikal atau bersifat string bisa menggunakan relasi antar kolom dengan Logistic Regression, jika numerical bisa menggunakan statistik sederhana dan linear regression. Pada sesi kali ini kita akan mencoba menangani data hilang dengan statistik sederhana, Mean dan Median.

Mengisi dengan Mean

```
In [9]: print(ShoppingData.mean())
print("dataset yang masih terdapat nilai kosong ! :")
print(ShoppingData.head(10))
```

```
ShoppingData = ShoppingData.fillna(ShoppingData.mean())
print("Dataset yang sudah diproses Handling Missing Values dengan Mean :")
print(ShoppingData.head(10))
```

CustomerID	Genre	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19.0	15.0
1	2	Male	NaN	15.0
2	3	Female	20.0	NaN
3	4	Female	23.0	16.0
4	5	Female	31.0	17.0
5	6	Female	22.0	NaN
6	7	Female	35.0	18.0
7	8	Female	23.0	18.0
8	9	Male	64.0	19.0
9	10	Female	30.0	19.0

dataset yang masih terdapat nilai kosong ! :

CustomerID	Genre	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19.000000	15.000000
1	2	Male	38.939698	15.000000
2	3	Female	20.000000	61.005051
3	4	Female	23.000000	16.000000
4	5	Female	31.000000	17.000000
5	6	Female	22.000000	61.005051
6	7	Female	35.000000	18.000000
7	8	Female	23.000000	18.000000
8	9	Male	64.000000	19.000000
9	10	Female	30.000000	19.000000

C:\Users\HP\AppData\Local\Temp\ipykernel_11444\3616134074.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
print(ShoppingData.mean())
C:\Users\HP\AppData\Local\Temp\ipykernel_11444\3616134074.py:5: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.
```

```
ShoppingData = ShoppingData.fillna(ShoppingData.mean())
```

Mengisi dengan Median

median digunakan untuk data-data yang memiliki sifat outlier yang kuat. Kenapa median dipilih? Median merupakan nilai tengah yang artinya bukan hasil dari perhitungan yang melibatkan data outlier. Pada beberapa kasus, data outlier dianggap mengganggu dan sering dianggap noisy karena bisa mempengaruhi distribusi kelas dan mengganggu analisa pada klasterisasi (clustering).

```
In [10]: print(ShoppingData.median())
print("dataset yang masih terdapat nilai kosong ! :")
print(ShoppingData.head(10))

ShoppingData = ShoppingData.fillna(ShoppingData.median())
print("Dataset yang sudah diproses Handling Missing Values dengan Mean :")
print(ShoppingData.head(10))
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11444\480445917.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.
print(ShoppingData.median())
CustomerID          100.500000
Age                 36.000000
Annual Income (k$)   61.502525
Spending Score (1-100) 50.000000
dtype: float64
dataset yang masih terdapat nilai kosong ! :
   CustomerID  Genre      Age  Annual Income (k$)  Spending Score (1-100)
0            1  Male  19.000000           15.000000          39.000000
1            2  Male  38.939698           15.000000          81.000000
2            3 Female  20.000000          61.005051           6.000000
3            4 Female  23.000000          16.000000          77.000000
4            5 Female  31.000000          17.000000         50.489899
5            6 Female  22.000000          61.005051          76.000000
6            7 Female  35.000000          18.000000           6.000000
7            8 Female  23.000000          18.000000          94.000000
8            9  Male  64.000000           19.000000         50.489899
9           10 Female  30.000000           19.000000          72.000000
Dataset yang sudah diproses Handling Missing Values dengan Mean :
   CustomerID  Genre      Age  Annual Income (k$)  Spending Score (1-100)
0            1  Male  19.000000           15.000000          39.000000
1            2  Male  38.939698           15.000000          81.000000
2            3 Female  20.000000          61.005051           6.000000
3            4 Female  23.000000          16.000000          77.000000
4            5 Female  31.000000          17.000000         50.489899
5            6 Female  22.000000          61.005051          76.000000
6            7 Female  35.000000          18.000000           6.000000
7            8 Female  23.000000          18.000000          94.000000
8            9  Male  64.000000           19.000000         50.489899
9           10 Female  30.000000           19.000000          72.000000
C:\Users\HP\AppData\Local\Temp\ipykernel_11444\480445917.py:5: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.
ShoppingData = ShoppingData.fillna(ShoppingData.median())
```

Praktek Normalisasi menggunakan Scikit Learn pada Python

```
In [11]: from sklearn import preprocessing
array = df.values

X = array[:,2:5] #memisahkan fitur dari dataset.
Y = array[:,0:1] #memisahkan class dari dataset

dataset = pd.DataFrame({'Customer ID':array[:,0], 'Gender':array[:,1], 'Age':array[:,2], 'Income':array[:,3], 'Spending Score':array[:,4]})
print("dataset sebelum dinormalisasi :")
print(dataset.head(10))

min_max_scaler = preprocessing.MinMaxScaler(feature_range=(0,1)) #inisialisasi normalisasi
data = min_max_scaler.fit_transform(X) #transformasi MinMax untuk fitur
dataset = pd.DataFrame({'Age':data[:,0], 'Income':data[:,1], 'Spending Score':data[:,2], 'Genre':data[:,3], 'Customer ID':data[:,4]})

print("dataset setelah dinormalisasi :")
print(dataset.head(10))
```

dataset sebelum dinormalisasi :

	Customer ID	Gender	Age	Income	Spending Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

dataset setelah dinormalisasi :

	Age	Income	Spending Score	Customer ID	Gender
0	0.019231	0.000000	0.387755	1	Male
1	0.057692	0.000000	0.816327	2	Male
2	0.038462	0.008197	0.051020	3	Female
3	0.096154	0.008197	0.775510	4	Female
4	0.250000	0.016393	0.397959	5	Female
5	0.076923	0.016393	0.765306	6	Female
6	0.326923	0.024590	0.051020	7	Female
7	0.096154	0.024590	0.948980	8	Female
8	0.884615	0.032787	0.020408	9	Male
9	0.230769	0.032787	0.724490	10	Female

Created by : Riki Dian Pratama

Source Dataset : DQLab Academy