

Data Quality dengan Python

Statistics descriptive, handling missing value and finding outliers

Topik

- Data Quality dengan Python: Data Profiling
 - Definisi Data Profiling?
 - Importing Data
 - Inspeksi tipe data
 - Descriptive Statistics
 - Length
 - Count
 - Missing Value
 - Maximum dan Minimum
 - Mean, Medium, Modus dan Standard Deviasi
 - Quantile Statistics
 - Correlation
 - Penggunaan Profiling Libraries
- Data Quality dengan Python: Data Cleansing
 - Definisi Data Cleansing?
 - Missing Data
 - Outliers
 - Deduplikasi Data
 - Case Studi: Data Profiling
 - Case Study: Data Cleansing — Bagian 1
 - Case Study: Data Cleansing — Bagian 2

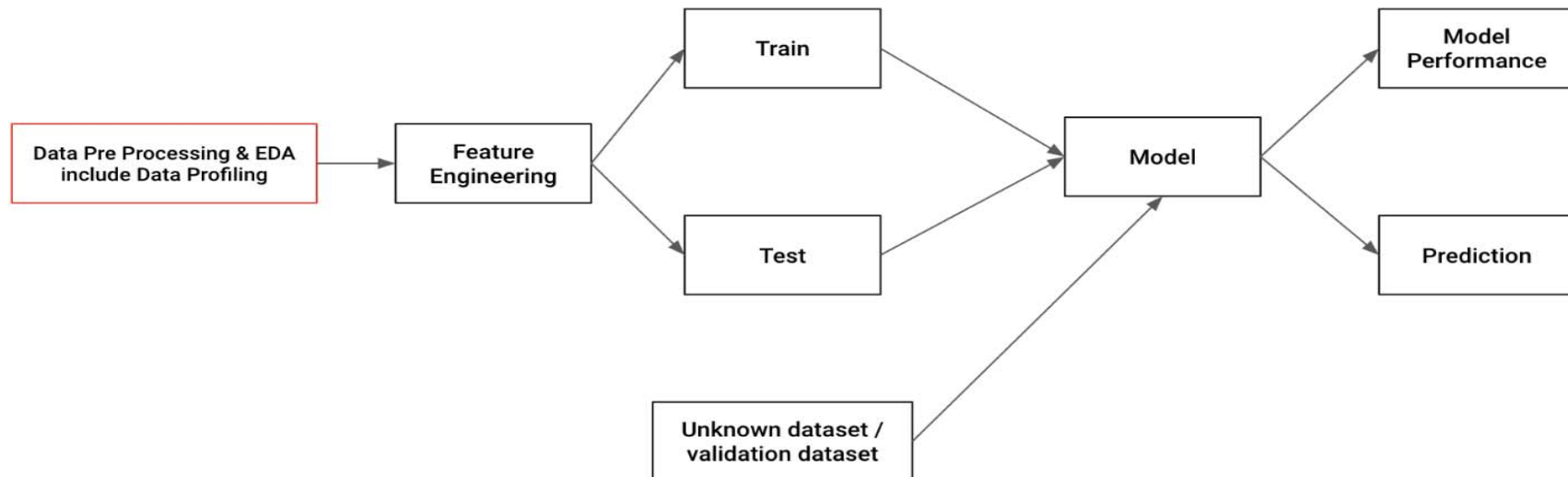
Data Profiling

Data Profiling

- Exploratory Data Analysis (EDA) adalah penggunaan pendekatan statistik yang bertujuan untuk menemukan dan meringkas sebuah dataset, mengetahui struktur dan hubungan antar variabel dalam dataset.
- EDA merupakan proses pre-analysis baik untuk descriptive analysis maupun predictive analysis.
- Data Profiling adalah salah satu bagian dari EDA.

Data Profiling

- Data profiling adalah kegiatan merangkum dataset menggunakan statistik deskriptif.
- Tujuan dari pembuatan data profiling adalah untuk memiliki pemahaman yang kuat tentang data sehingga dapat mulai menyusun framework analisis dan memvisualisasikan data.



Importing Data

- Langkah pertama yang harus dilakukan adalah inisialisasi Library dan mengimport dataset tersebut ke dalam Python menggunakan library Pandas dan diassign sebagai **retail_raw**.
- Library yang perlu diimport adalah:
 - (1) **pandas**,
 - (2) **numpy**, dan
 - (3) **pandas_profiling**.
- Untuk dua library yang pertama importlah sebagai aliasnya.
- Syntax:

```
import pandas as pd
import numpy as np
import pandas_profiling
retail_raw = pd.read_csv('/path/retail_raw_reduced_data_quality.csv')
```

Struktur Dataset

Beberapa cara dibawah ini dapat digunakan untuk mengetahui struktur dataset:

1. Inspeksi tipe data
2. Inspeksi jumlah kolom dan baris
3. Inspeksi kolom

Struktur Dataset: 1. Inspeksi tipe data

- Dengan library Pandas, dapat menjalankan fungsi **.dtypes** untuk melihat data untuk setiap kolom termasuk tipe datanya.
- Bentuk umum:

```
[data_frame] .dtypes
```

Tugas 1. Melihat data untuk setiap kolom termasuk tipe datanya

- Lengkapi kode berikut ini:

```
# Cetak tipe data di setiap kolom retail_raw  
print(____.____)
```

- Hasil yang diharapkan:

```
order_id      int64  
order_date    object  
customer_id   int64  
city          object  
province      object  
product_id    object  
brand         object  
quantity      float64  
item_price    float64  
dtype: object
```

Struktur Dataset: 2. Inspeksi jumlah kolom dan baris

- Jumlah baris dan kolom dataset → .shape
- Syntax:

```
data_frame.shape
```

- Kode

```
# Jumlah baris dan kolom  
retail_raw.shape
```

- Pada output, dataset memiliki jumlah baris sebanyak 5000 baris dan 9 kolom

```
(5000, 9)
```

Struktur Dataset: 3. Inspeksi kolom

- Daftar kolom pada dataset, tipe data pada setiap kolom, dan jumlah entries atau row pada masing-masing kolom → `info()`

- Syntax:

```
data_frame.info()
```

- Kode

```
# Daftar kolom, jumlah non-null data, dan tipe data per kolom  
retail_raw.info()
```

#	Column	Non-Null Count		Dtype
0	order_id	5000	non-null	int64
1	order_date	5000	non-null	object
2	customer_id	5000	non-null	int64
3	city	4984	non-null	object
4	province	4988	non-null	object
5	product_id	4989	non-null	object
6	brand	5000	non-null	object
7	quantity	4986	non-null	float64
8	item_price	4987	non-null	float64

Inspeksi kolom

#	Column	Non-Null Count		Dtype
0	order_id	5000	non-null	int64
1	order_date	5000	non-null	object
2	customer_id	5000	non-null	int64
3	city	4984	non-null	object
4	province	4988	non-null	object
5	product_id	4989	non-null	object
6	brand	5000	non-null	object
7	quantity	4986	non-null	float64
8	item_price	4987	non-null	float64

Dari output tersebut dapat diambil kesimpulan bahwa:

- Dataset terdiri dari sembilan kolom
- Memiliki kolom order_id, customer_id bertipe int64 dan quantity, item_price bertipe float64. Tipe int64 dan float64 adalah tipe data **numerik**
- Kolom order_date, city, province, product_id, brand memiliki tipe data **object**. Tipe data objek adalah tipe data yang memiliki kandungan string dan termasuk kedalam tipe data non-numerik
- Terdapat empat kolom tanpa missing value yaitu kolom yang memiliki 5000 non-null. Kolom tersebut order_id, order_date, customer_id, brand. Selebihnya adalah memiliki missing value.

Descriptive Statistics

Berbagai statistik deskriptif yang dapat digunakan untuk lebih memahami struktur data

- Length
- Count
- Missing Value
- Maximum dan Minimum
- Quantile Statistics
- Correlation

Descriptive Statistics – Bagian 1

Length

- Fungsi **len** menghitung jumlah pengamatan dalam suatu *series* / column.
- Fungsi **len** akan menghitung semua pengamatan, terlepas dari apakah ada null-value atau tidak (include missing value).

```
len(nama_dataframe["nama_kolom"])
```

- Contoh penggunaan untuk mengetahui length dari kolom *city* dari dataframe *retail_raw*

```
# Kolom city
length_city = len(retail_raw['city'])
print('Length kolom city:', length_city)
```

- Jika dijalankan akan menghasilkan output:

```
Length kolom city: 5000
```

Tugas 2. Inspeksi length

- Inspeksi length dari kolom **product_id** dari dataframe **retail_raw**.
- Lengkapi kode berikut:

```
length_product_id = ---(---['---'])  
print('Length kolom product_id:', ---)
```

- Output yang diharapkan:

```
Length kolom product_id: 5000
```

Descriptive Statistics – Bagian 2

Count

- Fungsi **count** menghitung jumlah pengamatan non-NA / non-null dalam suatu *series* / column.

```
nama_dataframe["nama_kolom"].count()
```

- Contoh penggunaan untuk mengetahui count dari kolom **city** dari dataframe **retail_raw**

```
1 # Count kolom city
2 count_city = retail_raw['city'].count()
3 print('Count kolom count_city:', count_city)
```

- Jika dijalankan akan menghasilkan output:

```
Count kolom count_city: 4984
```

Descriptive Statistics – Bagian 2

- Jumlah baris dataset yang terisi (tidak termasuk nilai NaN)

```
# Jumlah baris tanpa nilai NaN  
retail_raw.count()
```

```
order_id      5000  
order_date    5000  
customer_id   5000  
city          4984  
province      4988  
product_id    4989  
brand         5000  
quantity      4986  
item_price    4987  
dtype: int64
```

Tugas 3. Menemukan jumlah non-null value

- Temukan jumlah non-null value dari kolom **product_id** dari dataframe **retail_raw**.
- Lengkapi kode berikut:

```
count_product_id = ---['---'].---()  
print('Count kolom product_id:', ---)
```

- Output yang diharapkan:

```
Count kolom product_id: 4989
```

Descriptive Statistics – Bagian 3

Missing Value

- Dengan memanfaatkan **Length** dan **Count**, jumlah missing-value dapat dihitung.
- Jumlah nilai yang hilang (**Missing Value**) adalah perbedaan antara **Length** dan **Count**.
- Contoh penggunaan untuk mengetahui count dari kolom **city** dari dataframe **retail_raw**

```
# Length kolom city
length_city = len(retail_raw['city'])
print('Length kolom city:', length_city)
```

```
# Count kolom city
count_city = retail_raw['city'].count()
print('Count kolom count_city:', count_city)
```

```
1 # Missing value pada kolom city
2 number_of_missing_values_city = length_city - count_city
3 float_of_missing_values_city = float(number_of_missing_values_city/length_city)
4 pct_of_missing_values_city = '{0:.1f}%'.format(float_of_missing_values_city * 100)
5 print('Persentase missing value kolom city:', pct_of_missing_values_city)
```

Descriptive Statistics – Bagian 3

- Jika dijalankan akan menghasilkan output:

```
Length kolom city: 5000  
Count kolom count_city: 4984  
Persentase missing value kolom city: 0.3%
```

Terlihat persentase ***missing value*** sebanyak **0.3%** dari total *entry* atau baris **pada kolom city**.

Tugas 4. Menemukan jumlah missing value

- Temukan jumlah missing value dari kolom **product_id** dari dataframe **retail_raw**.
- Lengkapi kode berikut:

```
# Length kolom product_id
length_product_id = ---(---['---'])
print('Length kolom product_id :', ---)

# Count kolom product_id
count_product_id = ---['---'].---()
print('Count kolom product_id :', ---)

# Persentase Missing Value
number_of_missing_values_product_id = --- - ---
float_of_missing_values_product_id = ---
pct_of_missing_values_product_id = '{0:.1f}%'.format( --- * 100)
print('Persentase missing value kolom product_id:', --- )
```

- Output yang diharapkan: `Persentase missing value kolom product_id: 0.2%`

Descriptive Statistics – Bagian 4

Maximum dan Minimum

- Fungsi **max** dan **min** digunakan untuk mengetahui element terbesar dan terkecil dari suatu kolom di dataframe.

```
nama_dataframe["nama_kolom"].max()  
nama_dataframe["nama_kolom"].min()
```

Mean, Medium, Modus dan Standard Deviasi

- Fungsi **mean**, **medium**, **modus** dan **standard deviasi** digunakan untuk mengetahui pemusatan data dan persebarannya.

```
nama_dataframe["nama_kolom"].mean()  
nama_dataframe["nama_kolom"].median()  
nama_dataframe["nama_kolom"].mode()  
nama_dataframe["nama_kolom"].std()
```

Descriptive Statistics – Bagian 4

- Contoh

```
1 # Deskriptif statistics kolom quantity
2 print('Kolom quantity')
3 print('Minimum value: ', retail_raw['quantity'].min())
4 print('Maximum value: ', retail_raw['quantity'].max())
5 print('Mean value: ', retail_raw['quantity'].mean())
6 print('Mode value: ', retail_raw['quantity'].mode())
7 print('Median value: ', retail_raw['quantity'].median())
8 print('Standard Deviation value: ', retail_raw['quantity'].std())
```

- Jika dijalankan akan menghasilkan:

```
Kolom quantity
Minimum value: 1.0
Maximum value: 720.0
Mean value: 11.423987164059366
Mode value: 0 1.0
dtype: float64
Median value: 5.0
Standard Deviation value: 29.44202501081146
```

Tugas 5. Membuat statistics deskriptif

- Buatlah statistics deskriptif untuk data **item_price** dari dataframe **retail_raw**! Berupa **max**, **min**, **mean**, **median**, dan **std**!
- Lengkapi kode berikut:

```
print('Kolom item_price ')\nprint('Minimum value: ', __['item_price'].__())\nprint('Maximum value: ', __['__'].__())\nprint('Mean value: ', retail_raw['__'].__())\nprint('Mode value: ', retail_raw['__'].__())\nprint('Median value: ', __[item_price '].__())\nprint('Standard Deviation value: ', __[item_price '].__())
```

- Output yang diharapkan:

```
Kolom item_price\nMinimum value: 26000.0\nMaximum value: 29762000.0\nMean value: 933742.7311008623\nMedian value: 604000.0\nStandard Deviation value: 1030829.8104242847
```

Descriptive Statistics – Bagian 5

Quantile Statistics

- Quantiles adalah titik potong yang membagi distribusi dalam ukuran yang sama.
- Jika akan membagi distribusi menjadi empat grup yang sama, kuantil yang dibuat dinamai *quartile*.
- Jika dibagi kedalam 10 sepuluh group yang sama dinamakan *percentile*.
- Dalam kasus di bawah ini, ingin membagi distribusi menjadi empat grup atau quartile.

```
nama_dataframe["nama_kolom"].quantile([.25, .5, .75])
```

- Contoh:

```
1 # Quantile statistics kolom quantity
2 print('Kolom quantity:')
3 print(retail_raw['quantity'].quantile([0.25, 0.5, 0.75]))
```

- Jika dijalankan akan menghasilkan:

```
Kolom quantity:
0.25      2.0
0.50      5.0
0.75     12.0
Name: quantity, dtype: float64
```

Tugas 6. Membuat distribusi quartile

- Buatlah distribusi quartile dari **item_price** dari dataframe **retail_raw**.
- Lengkapi kode berikut:

```
print('Kolom item_price:')  
print(__['item_price'].__([__, __, __]))
```

- Output yang diharapkan:

```
Kolom item_price:  
0.25    450000.0  
0.50    604000.0  
0.75   1045000.0  
Name: item_price, dtype: float64
```

Descriptive Statistics – Bagian 6

Correlation

- Korelasi adalah cara yang tepat untuk menemukan hubungan antara variabel numerik.
- Koefisien korelasi berkisar antara -1 hingga 1.
- Semakin tinggi nilai *correlation*, nilai tersebut berpengaruh untuk proses *training* pada model.
- Nilai korelasi:
 - 1 adalah korelasi positif total
 - -1 adalah korelasi negatif total
 - 0 adalah korelasi non-linear.

```
nama_dataframe[["nama_kolom", "nama_kolom", ".."]].corr()
```

Tugas 7. Menampilkan korelasi data

- Tampilkan korelasi antara **quantity** dan **item_price** .
- Lengkapi kode berikut:

```
print('Korelasi quantity dengan item_price')  
print(etail_raw[['__', '__']].__())
```

- Output yang diharapkan:

```
Korelasi quantity dengan item_price  
          quantity  item_price  
quantity    1.000000   -0.133936  
item_price  -0.133936    1.000000
```

Penggunaan Profiling Libraries

- Melakukan statistik deskriptif dapat menjadi proses yang panjang.
- **Pandas Profiling library** memiliki function yang dapat membuat [profiling data secara otomatis](#).
- Untuk dapat menggunakannya, cukup dengan memanggil library

```
import pandas_profiling
```

- Syntax:

```
pandas_profiling.ProfileReport(nama\_dataframe)
```

- Contoh penggunaan untuk dataset **retail_raw**:

```
# Penggunaan ProfileReport pada dataframe: retail_raw  
pandas_profiling.ProfileReport(retail_raw)
```

Penggunaan Profiling Libraries

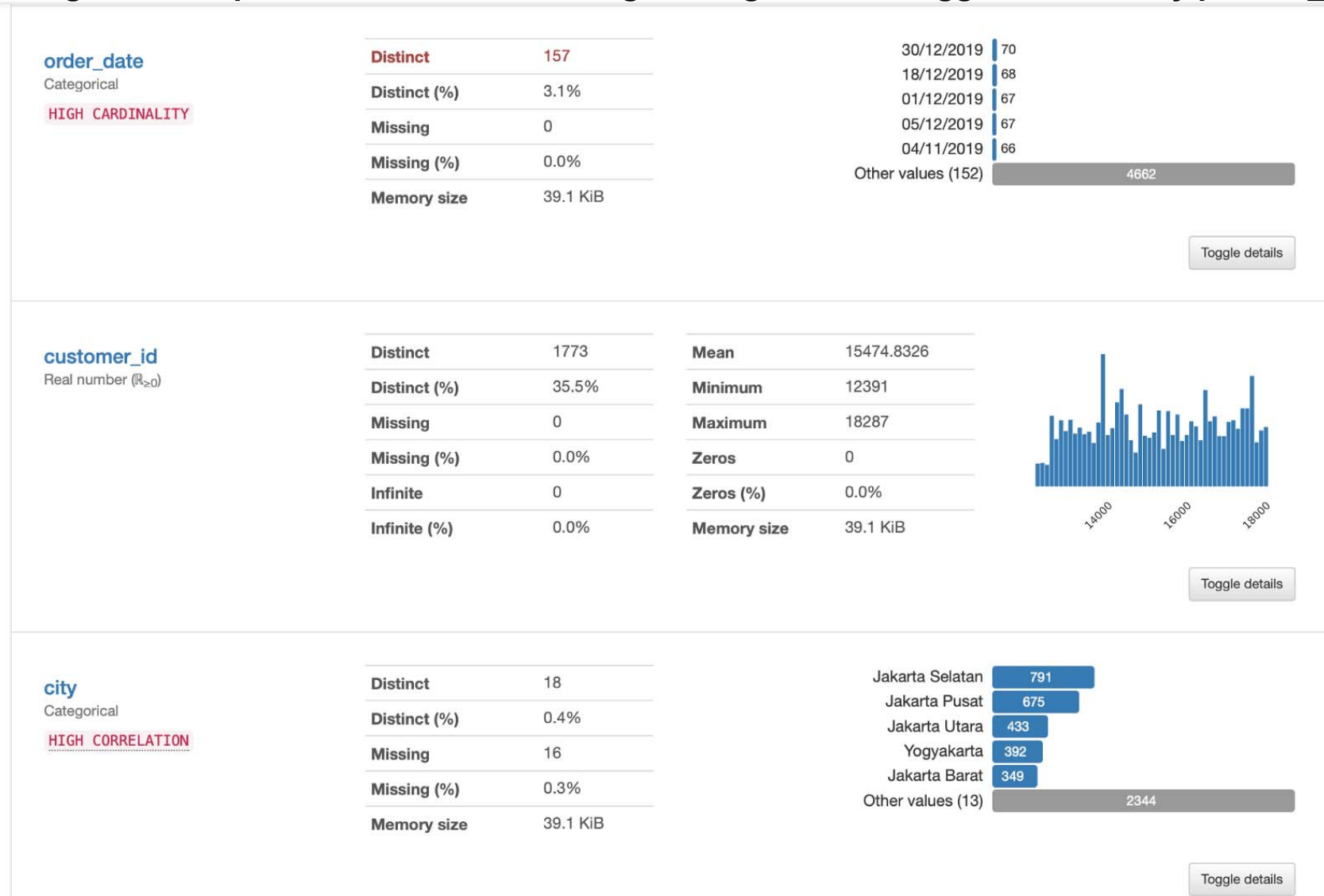
```
import pandas_profiling
```

```
retail_raw_profile = pandas_profiling.ProfileReport(retail_raw)
```

```
# Simpan hasil dalam file html (opsional)
```

```
retail_raw_profile.to_file('retail_raw_report.html')
```

Preview singkat Descriptive Statistics dari masing-masing kolom menggunakan library pandas_profiling



Correlations

Visualisasi heatmap fitur correlation antar variable (berbagai fungsi) menggunakan library pandas_profiling

Pearson's r

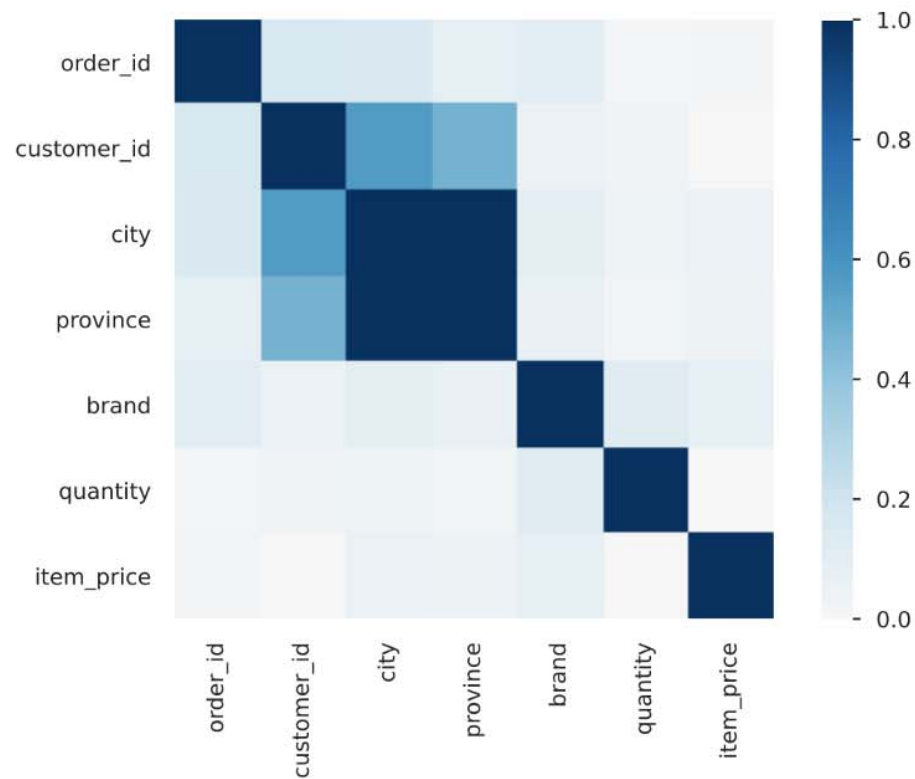
Spearman's ρ

Kendall's τ

Phik (ϕ_k)

Cramér's V (ϕ_c)

Toggle correlation descriptions



Data Cleansing

Data Cleansing

- Data Cleansing adalah kegiatan dasar dari Data Science.
- Data Cleansing adalah proses mengidentifikasi bagian data yang:
 - salah
 - tidak lengkap
 - tidak akurat
 - tidak relevan
 - hilang

dan kemudian memodifikasi, mengganti atau menghapusnya sesuai dengan kebutuhan.

- Selanjutnya akan dibahas data cleansing yang meliputi treatment terhadap:
 1. missing data
 2. treatment outliers
 3. deduplikasi data

Data Cleansing: 1. Missing Data

- Dataset yang ditemui di *real-world* biasanya memiliki banyak *missing value*.
- Kemampuan untuk *treatment missing value* sangat penting karena jika *missing value* dibiarkan dapat memengaruhi analisis dan *machine learning model* yang dihasilkan.
- Cara mengecek kolom yang mempunyai *missing value*:

```
nama_dataframe.isnull().any()
```

- Cara *treatment* terhadap missing-value antara lain:
 - *Leave as it is* (dibiarkan)
 - *Filling the missing value* (imputasi)
 - *Drop them* (hapus row yang mengandung *missing value*)

Data Cleansing: 1. Missing Data

- Mengetahui jumlah *missing value* dari masing-masing kolom

```
# Get Missing Value per Column
```

```
retail_raw.isnull().sum()
```

- Jumlah missing value masing-masing kolom

```
order_id      0
order_date    0
customer_id   0
city          16
province      12
product_id    11
brand         0
quantity      14
item_price    13
dtype: int64
```

Data Cleansing: 1. Missing Data

Beberapa metode yang bisa digunakan untuk menghilangkan *missing value* pada data:

- **Membiarkan *missing value*:** biasanya dilakukan dengan pertimbangan data yang mengandung *missing value* tersebut berpengaruh cukup besar dan tidak bisa digantikan nilainya.
- ***Impute missing value*:** mengisi nilai NaN dengan nilai baru (seperti *mean* dan *median* pada data numerik) dan ('unknown' pada data non-numerik)
- ***Drop missing value*:** menghapus data *missing value* berdasarkan baris maupun kolom. Biasanya kolom yang dihapus adalah kolom yang tidak memiliki korelasi yang kuat terhadap hasil akurasi model atau sebagian besar data memiliki banyak *missing value*.

Data Cleansing: Penanganan Missing Data

IMPUTASI

- **Imputasi** merupakan suatu metode *treatment* terhadap *missing value* dengan mengisinya menggunakan teknik tertentu.
- Bisa menggunakan **mean, modus** ataupun menggunakan *predictive modelling*.
- Pada modul ini akan membahas mengenai pemanfaatan *function fillna* dari Pandas untuk imputasi ini, yaitu

```
nama_dataframe['nama_kolom'].fillna(nama_dataframe.nama_kolom.function())
```

- **.function()** yang dimaksud pada syntax di atas adalah penggunaan fungsi **.mean()** atau **.mode()**.
- Penggunaan fungsi **.mean()** atau **.mode()** ini bergantung pada kondisi yang mengharuskan menggunakan nilai rata - rata atau modus dari kolom yang akan diimputasi, seperti

```
nama_dataframe['nama_kolom'].fillna(nama_dataframe.nama_kolom.mean())
```

atau

```
nama_dataframe['nama_kolom'].fillna(nama_dataframe.nama_kolom.mode())
```

Data Cleansing: Penanganan Missing Data

DROP

- **Drop** row yang mengandung missing value.
- Dapat menggunakan function **dropna** dari Pandas.

```
nama_dataframe['nama_kolom'].dropna()
```

Tugas 8. Menangani missing data dengan drop missing value

Menangani missing data pada retail_raw,

- Cek apakah terdapat missing value pada variabel dataframe, dan kemudian cetak ke console
- Imputasi missing value pada kolom quantity dengan menggunakan nilai rata-rata (mean), dan kemudian cetak ke console
- Drop missing value pada kolom quantity, dan kemudian cetak ke console

Lengkapi kode berikut ini:

```
# Check kolom yang memiliki missing data
print('Check kolom yang memiliki missing data:')
print(__.__.())

# Filling the missing value (imputasi)
print('\nFilling the missing value (imputasi):')
print(__['__'].__(__.__.()))

# Drop missing value
print('\nDrop missing value:')
print(__['__'].__())
```

Tugas 8. Diharapkan diperoleh output sebagai berikut:

<pre> Check kolom yang memiliki missing data: order_id False order_date False customer_id False city True province True product_id True brand False quantity True item_price True dtype: bool </pre>	<pre> Filling the missing value (imputasi): 0 10.000000 1 2.000000 2 8.000000 3 4.000000 4 2.000000 5 4.000000 6 1.000000 7 2.000000 8 32.000000 9 2.000000 10 6.000000 11 1.000000 12 3.000000 13 4.000000 14 11.423987 15 1.000000 16 10.000000 17 1.000000 18 4.000000 19 6.000000 20 12.000000 21 24.000000 22 10.000000 23 6.000000 24 12.000000 25 12.000000 26 96.000000 27 1.000000 28 10.000000 29 18.000000 ... </pre>	<pre> Drop missing 0 10.0 1 2.0 2 8.0 3 4.0 4 2.0 5 4.0 6 1.0 7 2.0 8 32.0 9 2.0 10 6.0 11 1.0 12 3.0 13 4.0 15 1.0 16 10.0 17 1.0 18 4.0 19 6.0 20 12.0 21 24.0 22 10.0 23 6.0 24 12.0 25 12.0 26 96.0 27 1.0 28 10.0 29 18.0 30 36.0 ... </pre>
---	--	---

Tugas 9. Menangani missing data dengan mengganti dengan nilai mean

Hasil yang diharapkan

- Lengkapi missing value pada **item_price** dengan **mean**.
- Lengkapi kode berikut ini:

```
print(__['__'].__(__['__'].__()))
```

0	7.400000e+05
1	6.040000e+05
2	1.045000e+06
3	2.050000e+05
4	9.337427e+05
5	5.200000e+05
6	1.465000e+06
7	2.050000e+05
8	4.500000e+05
9	1.490000e+05
10	1.045000e+06
11	6.040000e+05
12	2.200000e+06
13	1.745000e+06
14	6.950000e+05
15	9.040000e+05
16	7.400000e+05
17	1.748000e+06
18	1.395000e+06
19	7.400000e+05
20	5.200000e+05
21	1.130000e+05
22	7.400000e+05
23	1.045000e+06
24	5.200000e+05
25	3.100000e+05
26	9.337427e+05
27	2.190000e+05
28	1.490000e+05
29	1.150000e+06
	...

Data Cleansing: 2. Outliers

- ***Outliers*** merupakan data observasi yang muncul dengan nilai-nilai ekstrim.
- Yang dimaksud dengan nilai-nilai ekstrim dalam observasi adalah nilai yang jauh atau beda sama sekali dengan sebagian besar nilai lain dalam kelompoknya.
- Cara *treatment* terhadap *outliers* antara lain:
 - Remove the outliers (dibuang)
 - Filling the missing value (imputasi)
 - Capping / Flooring
 - Prediction

Data Cleansing: 2. Outliers

- Pada umumnya, *outliers* dapat ditentukan dengan metric **IQR** (*interquartile range*).
- Rumus dasar dari IQR: $Q3 - Q1$, dimana
 - $Q1$: *quantile* 25%
 - $Q3$: *quantile* 75%
- Data suatu observasi dapat dikatakan *outliers* jika memenuhi kedua syarat dibawah ini:
 - $data < Q1 - 1.5 * IQR$
 - $data > Q3 + 1.5 * IQR$

- Syntax di Python: $Q1 = \text{nama_dataframe.quantile}(0.25)$

$Q3 = \text{nama_dataframe.quantile}(0.75)$

$IQR = Q3 - Q1$

- Untuk membuang outliers:

```
nama_dataframe = nama_dataframe[~(((nama_dataframe < (Q1 - 1.5*IQR)) | (nama_dataframe > (Q3 + 1.5*IQR)))]
```

Untuk melihat outlier pada dataset dapat menggunakan grafik boxplot.

```
import matplotlib.pyplot as plt

# Visualize Outliers with Boxplot
retail_raw.boxplot()
plt.show()
```

- Grafik boxplot memvisualisasi untuk **kolom bertipe numerik**.
- Dari visualisasi tersebut outliers yang ditandai dengan simbol o pada grafik.
- Outliers banyak terlihat pada kolom item_price.



Data Cleansing: 2. Outliers

- Mencari nilai Q1, Q3, dan IQR. Kemudian menghilangkan *outlier* pada kolom quantity.
- Penggunaan pada dataframe `retail_raw` untuk kolom quantity:

```
1 # Q1, Q3, dan IQR
2 Q1 = retail_raw['quantity'].quantile(0.25)
3 Q3 = retail_raw['quantity'].quantile(0.75)
4 IQR = Q3 - Q1
5
6 # Check ukuran (baris dan kolom) sebelum data yang outliers dibuang
7 print('Shape awal: ', retail_raw.shape)
8
9 # Removing outliers
10 retail_raw = retail_raw[~((retail_raw['quantity'] < (Q1 - 1.5 * IQR)) |
11 (retail_raw['quantity'] > (Q3 + 1.5 * IQR)))]
12
13 # Check ukuran (baris dan kolom) setelah data yang outliers dibuang
14 print('Shape akhir: ', retail_raw.shape)
```

- Ukuran dataframe `retail_raw` sebelum dan setelah dibuang outliers pada kolom quantity yaitu:

```
Shape awal: (5000, 9)
Shape akhir: (4699, 9)
```

Tugas 10. Membuang outliers kolom quantity

- Buang outliers pada kolom quantity
- Lengkapi kode berikut

```
# Q1, Q3, dan IQR
Q1 = __['__'].__(__)
Q3 = __['__'].__(__)
IQR = __ - __
# Check ukuran (baris dan kolom) sebelum data yang outliers dibuang
print('Shape awal: ', __.__(__))

# Removing outliers
retail_raw = __[~((__['__'] < (__)) | (__['__'] > (__)))]

# Check ukuran (baris dan kolom) setelah data yang outliers dibuang
print('Shape akhir: ', __.__(__))
```

- Hasil yang diharapkan:

```
Shape awal:  (5000, 9)
Shape akhir:  (4699, 9)
```

Tugas 11. Membuang outliers kolom item_price

- Menemukan dan membuang sejumlah outliers menggunakan IQR dari variabel **item_price**.
- Lengkapi kode berikut:

```
# Q1, Q3, dan IQR
Q1 = ____['____'].____(____)
Q3 = ____['____'].____(____)
IQR = ____ - ____

# Check ukuran (baris dan kolom) sebelum data yang outliers dibuang
print('Shape awal: ', retail_raw.____)

# Removing outliers
retail_raw = ____[~((____['____'] < (____)) | (____['____'] > (____)))]

# Check ukuran (baris dan kolom) setelah data yang outliers dibuang
print('Shape akhir: ', retail_raw.____)
```

- Hasil yang diharapkan:

Data Cleansing: 3. Deduplikasi data

DUPLIKASI DATA

- Duplikasi data juga mempengaruhi tingkat performa akurasi dari model yang dibangun.
- **Duplikasi data** merupakan data dengan kondisi pada *row-row* tertentu memiliki kesamaan data di seluruh kolomnya.
- Tentunya ada kemungkinan data yang duplikat dalam dataset yang dimiliki.
- Kondisi duplikasi harus diatasi dengan jalan mengeliminir baris yang mengalami duplikasi, sehingga proses ini dikenal dengan **deduplikasi**.
- **Untuk mengecek duplikasi data:**

```
nama_dataframe.duplicated(subset=None)
```

- **Syntax untuk membuang duplikasi:**

```
nama_dataframe.drop_duplicates()
```

Tugas 12. Membuang duplikasi data

- Membuang duplikasi data dari dataset **retail_raw**.
- Lengkapi kode berikut:

```
# Check ukuran (baris dan kolom) sebelum data duplikasi dibuang
print('Shape awal: ', __.__)

# Lihat entries yang duplikasi
print('')
print(retail_raw[retail_raw.duplicated(subset=None)])

# Buang data yang terduplikasi
__.__(inplace=True)

# Check ukuran (baris dan kolom) setelah data duplikasi dibuang
print('Shape akhir: ', __.__)
```

Tugas 12. Membuang duplikasi data

Hasil yang diharapkan

```
order_id  order_date  customer_id  ...  brand  quantity  item_price
954      1737007    22/12/2019    14093  ...  BRAND_H    1.0    450000.0
2441     1737424    23/12/2019    17019  ...  BRAND_R    1.0    590000.0
2460     1710733    03/11/2019    14514  ...  BRAND_H   12.0    590000.0
2517     1736995    22/12/2019    15436  ...  BRAND_R    1.0   1045000.0
3329     1718197    19/11/2019    18079  ...  BRAND_C   12.0    159000.0
3579     1736779    22/12/2019    18156  ...  BRAND_R    1.0   1500000.0
3750     1720504    25/11/2019    18055  ...  BRAND_A    2.0   3144000.0

[7 rows x 9 columns]
```

Tugas 13. Data Profiling

- Lakukan data profiling dengan menggunakan data **uncleaned_raw.csv**
- Ikuti langkah-langkah berikut:
 1. Import dataset csv ke variable bernama **uncleaned_raw**
 2. Inspeksi dataframe **uncleaned_raw**
 3. Check kolom yang mengandung missing value, jika ada, kolom apakah itu dan berapa persen missing value pada kolom tersebut?
 4. Mengisi missing value tersebut dengan mean dari kolom tersebut!

Lengkapi kode berikut

```
# Baca dataset uncleaned_raw.csv
uncleaned_raw = __('/path/uncleaned_raw.csv')

#inspeksi dataframe uncleaned_raw
print('Lima data teratas:')
print(uncleaned_raw.__)

#Check kolom yang mengandung missing value
print('\nKolom dengan missing value:')
print(uncleaned_raw.__.__)

#Persentase missing value
length_qty = __(__['__'])
count_qty = __['__'].__
```

```
#mengurangi length dengan count
number_of_missing_values_qty = __ - __

#mengubah ke bentuk float
float_of_missing_values_qty = __(__ / __)

#mengubah ke dalam bentuk persen
pct_of_missing_values_qty =
'{0:.1f}%'.format(__*__)

#print hasil percent dari missing value
print('Persentase missing value kolom
Quantity:', __)

#Mengisi missing value tersebut dengan mean
dari kolom tersebut
uncleaned_raw['Quantity'] =
__['__'].__(__['__'].__)
```

Tugas 14. Data Cleansing

Lanjutan langkah data profiling:

5. Mengetahui kolom yang memiliki outliers. Gunakan visualisasi dengan boxplot pada dataframe `uncleaned_raw`.

- Lengkapi kode berikut:

```
import ___. pyplot as ____

#Mengetahui kolom yang memiliki outliers!
uncleaned_raw.____
plt.show()
```

Tugas 15. Data Cleansing

Lanjutan langkah data profiling:

6. Melakukan proses removing outliers pada kolom UnitPrice.
7. Checking duplikasi and melakukan deduplikasi dataset tersebut!

Lengkapi kode berikut:

```
#Check IQR
Q1 = ____['____'].____(____)
Q3 = uncleaned_raw['____'].____(____)
IQR = ____ - ____

#removing outliers
uncleaned_raw = ____[~((____[['____']] < (____)) | (____[['____']] > (____)))]

#check for duplication
print(uncleaned_raw.____(subset=None))

#remove duplication
uncleaned_raw = uncleaned_raw.____
```

Tugas 16. Profiling otomatis

- Buatlah profiling otomatis untuk data frame `uncleaned_raw` yang sudah dibuat di Tugas 13.
- Dihasilkan file html.

Penutup

- Telah dipelajari
 - *Profiling* data baik untuk data angka maupun teks
 - Membuat rule untuk data *cleansing* (standarisasi data, value lookup, dan treatment duplikasi data)

Referensi

- Data Quality with Python for Beginner - DQLab.
- <https://www.dqlab.id/akses-module-baru-data-quality-with-python-for-beginner>
- <https://towardsdatascience.com/create-and-customize-boxplots-with-pythons-matplotlib-to-get-lots-of-insights-from-your-data-d561c9883643>