CleansingData

September 12, 2022

Nama : Muhammad Reesa Rosyid

Program : Python for Data Science

1 EDA

1.1 Definisi EDA

EDA (Exploratory Data Analisis) adalah proses kritis dalam melakukan investigasi awal pada data untuk menemukan pola, menemukan anomali, menguji hipotesis dan memeriksa asumsi dengan bantuan statistik ringkasan dan representasi grafis (visualisasi).

1.2 Data Cleansing with Pandas

1.2.1 Source of Missing Value

Beberapa alasan mengapa data bisa hilang: * Pengguna lupa untuk mengisi * Data hilang saat transfer dari database * Terdapat program error * Pengguna memilih tidak mengisi

Langkah-langkah menyusun rencana untuk membersihkan data. Kita mulai dari: * Memahami data yang akan kita olah * Apa kegunaannya * Tipe data apa yang seharusnya digunakan * Terdapat missing data yang valuenya dapat dideteksi oleh Pandas * Terdapat pula missing data yang valuenya tidak dapat dideteksi oleh Pandas

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
#Membuka data via URL
URL = "https://raw.githubusercontent.com/ardhiraka/PFDS_sources/master/
 ↪property_data.csv"
df = pd.read_csv(URL)
df
```

[2]:

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3	1	1000
1	100002000.0	197.0	LEXINGTON	N	3	1.5	--
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850
3	100004000.0	201.0	BERKELEY	12	1	NaN	700
4	NaN	203.0	BERKELEY	Y	3	2	1600
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800

```
6  100007000.0     NaN  WASHINGTON          NaN            2   HURLEY   950
7  100008000.0   213.0     TREMONT            Y           --        1   NaN
8  100009000.0   215.0     TREMONT            Y           na        2  1800
```

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   PID           8 non-null      float64
 1   ST_NUM        7 non-null      float64
 2   ST_NAME       9 non-null      object
 3   OWN_OCCUPIED  8 non-null      object
 4   NUM_BEDROOMS  7 non-null      object
 5   NUM_BATH      8 non-null      object
 6   SQ_FT         8 non-null      object
dtypes: float64(2), object(5)
memory usage: 632.0+ bytes
```

Standart Missing Value Standart Missing Value adalah missing value yang dapat dideteksi oleh pandas. Biasanya akan bertuliskan NaN.

Contohnya:

[4]:
```
df['ST_NUM']
```

[4]:
```
0    104.0
1    197.0
2      NaN
3    201.0
4    203.0
5    207.0
6      NaN
7    213.0
8    215.0
Name: ST_NUM, dtype: float64
```

Cara melihat jumlah NaN pada sebuah kolom

[5]:
```
df['ST_NUM'].isnull()
```

[5]:
```
0    False
1    False
2     True
3    False
4    False
5    False
```

```
6     True
7    False
8    False
Name: ST_NUM, dtype: bool
```

[6]:
```
# Per Kolom
kondisi = df['ST_NUM'].isnull()
kondisi.sum()
```

[6]:
```
2
```

[7]:
```
# Semua
df.isnull().sum()
```

[7]:
```
PID             1
ST_NUM          2
ST_NAME         0
OWN_OCCUPIED    1
NUM_BEDROOMS    2
NUM_BATH        1
SQ_FT           1
dtype: int64
```

Non Standard Missing Value Non Standard Missing Value adalah missing value yang tidak dapat terdeteksi oleh pandas

misalnya penulisan : * na * n/a * - * –

Caranya kita dapat mendeklarasikan missing value pada saat memanggil dataset dan menganggapnya sebagai missing value.

[8]:
```
df.NUM_BEDROOMS
```

[8]:
```
0      3
1      3
2    NaN
3      1
4      3
5    NaN
6      2
7     --
8     na
Name: NUM_BEDROOMS, dtype: object
```

Pada NUM_BEDROOM terdapat – dan na yang tidak dapat dibaca oleh pandas sebagai missing value

[9]:
```
kondisi = df['NUM_BEDROOMS'].isnull()
kondisi.sum()
```

[9]:
```
2
```

Selain itu jika mencoba menghitung jumlah missing value dari NUM_BEDROOM yang terhitung hanya 2 yang seharunya terdapat 4 missing value. Oleh karena itu diperlukan pembersihan dengan metode dibawah ini.

[10]:
```
URL = "https://raw.githubusercontent.com/ardhiraka/PFDS_sources/master/
 ↪property_data.csv"
misval = ["--", "na", "-", "n/a"]
df = pd.read_csv(URL, na_values=misval)
df
```

[10]:

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	1.5	NaN
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	12	1.0	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.0	HURLEY	950.0
7	100008000.0	213.0	TREMONT	Y	NaN	1	NaN
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

[11]:
```
df.isnull().sum()
```

[11]:
```
PID             1
ST_NUM          2
ST_NAME         0
OWN_OCCUPIED    1
NUM_BEDROOMS    4
NUM_BATH        1
SQ_FT           2
dtype: int64
```

Unxepected Missing Value Unxepected Missing Value merupakan missing value yang tak terduga. Dapat dilihat pada kolom OWN_OCCUPIED terdapat nilai 12 yang merupakan expexted missing value karena seharusnya user hanya mengisikan data berupa Y atau N. Nilai 12 tersebut dapat dijadikan NaN dengan cara filter manual sebagai berikut.

[13]:
```
# filter manual OWN_OCCUPIED
# Logika:
#  1. Loop pada kolom OWN_OCCUPIED.
#  2. Data yang seharusnya adalah Y atau N.
#     Dengan demikian dapat dibuat kondisi jika
#     filled data tersebut bukan merupakan Y atau N
#     maka akan dianggap missing value.

for idx, row in df['OWN_OCCUPIED'].iteritems():
```

```
    if row == 'N' or row == 'Y':
        pass
    else:
        df.loc[idx, 'OWN_OCCUPIED'] = np.nan
```

[14]:
```
df['OWN_OCCUPIED']
```

[14]:
```
0      Y
1      N
2      N
3    NaN
4      Y
5      Y
6    NaN
7      Y
8      Y
Name: OWN_OCCUPIED, dtype: object
```

[15]:
```
df['OWN_OCCUPIED'].isnull()
```

[15]:
```
0    False
1    False
2    False
3     True
4    False
5    False
6     True
7    False
8    False
Name: OWN_OCCUPIED, dtype: bool
```

[16]:
```
df
```

[16]:

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	1.5	NaN
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.0	HURLEY	950.0
7	100008000.0	213.0	TREMONT	Y	NaN	1	NaN
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

Contoh kedua terdapat pada NUM_BATH dimana terdapat nilai HURLEY dan 1.5 sebagai unexpected missing value yang seharusnya berisi integer. Dapat dibersihkan dengan metode filter manual berikut.

[17]:
```
# Filter manual NUM_BATH
# Logika:
#  1. Loop pada kolom NUM_BATH
#  2. Buat kondisi apakah tipe data setiap baris dari kolom NUM_BATH merupakan␣
 ↪string
#  3. Buat kondisi mengecek apakah setiap data dari kolom NUM_BATH merupakan␣
 ↪numeric atau bukan

for idx, row in df['NUM_BATH'].iteritems():
    if type(row) == str:
        if row.isnumeric():
            pass
        else:
            df.loc[idx, 'NUM_BATH'] = np.nan
```

[18]:
```
df['NUM_BATH']
```

[18]:
```
0      1
1    NaN
2      1
3    NaN
4      2
5      1
6    NaN
7      1
8      2
Name: NUM_BATH, dtype: object
```

[19]:
```
df
```

[19]:

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	NaN	NaN
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.0	NaN	950.0
7	100008000.0	213.0	TREMONT	Y	NaN	1	NaN
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

1.2.2 Replacing / Imputasi

Imputasi dapat dilakukan apabila terdapat missing value yang telah diketahui polanya atau cara yang lebih umum adalah dengan mencari nilai tengah. Contoh:

[20]:
```
# memasukkan static value
val = 100005000
df['PID'].fillna(val, inplace=True)
```

[21]:
```
df['PID']
```

[21]:
```
0    100001000.0
1    100002000.0
2    100003000.0
3    100004000.0
4    100005000.0
5    100006000.0
6    100007000.0
7    100008000.0
8    100009000.0
Name: PID, dtype: float64
```

[22]:
```
df
```

[22]:

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	NaN	NaN
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	100005000.0	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.0	NaN	950.0
7	100008000.0	213.0	TREMONT	Y	NaN	1	NaN
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

[23]:
```
# central tendency
val = df['ST_NUM'].median()
df['ST_NUM'].fillna(val, inplace=True)
```

[24]:
```
df['ST_NUM']
```

[24]:
```
0    104.0
1    197.0
2    203.0
3    201.0
4    203.0
5    207.0
6    203.0
7    213.0
8    215.0
Name: ST_NUM, dtype: float64
```

[25]:
```
df
```

[25]:

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	NaN	NaN
2	100003000.0	203.0	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	100005000.0	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	203.0	WASHINGTON	NaN	2.0	NaN	950.0
7	100008000.0	213.0	TREMONT	Y	NaN	1	NaN
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

[ ]: