Modul 2 *Pre-Processing* Data

Tujuan

Mahasiswa dapat mempraktikkan metode *pre-processing* data dasar berupa *data cleaning*, *data transformation* dan *data reduction* menggunakan Python untuk pembelajaran mesin dengan benar.

Dasar Teori

Pre-processing data adalah proses menyiapkan data mentah (*raw*) untuk membuatnya cocok sebagai model pembelajaran mesin. Metode ini merupakan langkah pertama dan penting saat membuat model pembelajaran mesin. Untuk membangun dan mengembangkan model pembelajaran mesin, terlebih dahulu harus diperoleh dataset yang relevan. Dataset ini terdiri dari data yang dikumpulkan dari berbagai sumber dan berbeda yang kemudian digabungkan dalam format yang tepat untuk membentuk dataset. Format kumpulan data berbeda menurut kasus penggunaan. Misalnya, kumpulan data bisnis akan sangat berbeda dari kumpulan data medis. Sementara kumpulan data bisnis akan berisi data industri dan bisnis yang relevan, kumpulan data medis akan mencakup data terkait perawatan kesehatan.

Pre-processing dibutuhkan karena untuk dapat mencapai hasil yang baik dalam pengembangan model Pembelajaran Mesin, format data harus dikondisikan dengan cara yang tepat. Beberapa model *Machine Learning* membutuhkan informasi dalam format tertentu. Misal algoritma *Random Forest* tidak mendukung nilai nol, sehingga untuk mengeksekusi nilai nol pada *Random Forest* harus dikondisikan dari data mentah yang ada. Aspek lain yaitu daset seharusnya diformat menjadi sebuah bentuk yang dapat dieksekusi oleh lebih dari satu algoritma *Machine Learning* atau *Deep Learning* untuk mendapatkan model terbaik yang dapat dipilih.

Secara garis besar *pre-processing* data dapat dibagi menjadi 3 hal yaitu *data cleaning*, *data transformation* dan *data reduction*. *Data cleaning* adalah proses untuk membersihkan data yang akan ditata kembali dalam bentuk dataset. Berdasarkan kedalamannya, *data cleaning* dapat dibagi menjadi 3 level yang berbeda. Data transformation adalah proses mengubah bentuk/tipe data dalam representasi yang berbeda agar mudah untuk diolah selanjutnya. Sedangkan data reduction adalah proses untuk mengurangi dimensi dari data sehingga lebih sederhana dan tidak rumit (Jafari, 2022).

A. *Data Cleaning*

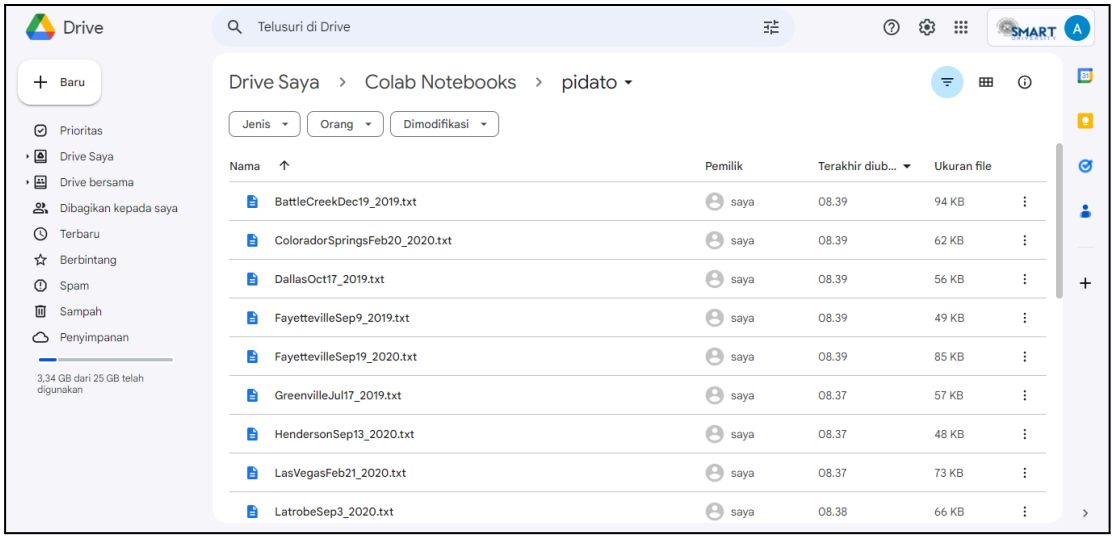
Data cleaning adalah sebuah tahapan yang penting dalam sebelum pengolahan lebih lanjut dilakukan. Tahapan ini memainkan peran yang penting dalam pembuatan model *Machine Learning*. Meskipun bukan merupakan proses yang rumit, tetapi keberhasilan pembuatan model ditentukan oleh *data cleaning* yang benar dan sesuai. Dengan *data cleaning* yang baik, kemungkinan untuk memperoleh hasil yang baik dengan algoritma yang sederhana cukup besar. Bentuk operasi *data cleaning* antara lain sebagai berikut:

1. *Data cleaning* Level 1

   Pada level ini merupakan *data cleaning* dengan tingkat kedalaman terendah. Karakteristik dari *data cleaning* level 1 adalah tersusunnya dataset yang standar sesuai dengan standar struktur data, memiliki judul kolom data yang intuitif, dan memastikan setiap baris memiliki identifier yang unik.

   Contoh 1: Standarisasi data agar terstruktur

   Apabila terdapat kumpulan data dengan format yang tidak terstruktur dan tidak dapat dianalisis secara langsung, seperti pada contoh berikut:



Maka perlu dilakukan penataan ulang data agar lebih mudah untuk dianalisis selanjutnya

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from os import listdir
from google.colab import drive
drive.mount('/content/drive')

FileNames = listdir('/content/drive/MyDrive/Colab Notebooks/pidato')
print(FileNames)

pidato_df = pd.DataFrame(index=range(len(FileNames)),columns=['File
Name','The Content'])
print(pidato_df)
```

```
for i,f_name in enumerate(FileNames):
    f = open('/content/drive/MyDrive/Colab Notebooks/pidato/' +
f_name, "r", encoding='utf-8')
    f_content = f.readlines()
    f.close()

    pidato_df.at[i,'File Name'] = f_name
    pidato_df.at[i,'The Content'] = f_content[0]

pidato_df.columns = ['FileName','Content']

pidato_df
```

Contoh 2: Reindexing

Apabila terdapat data temperatur pada periode waktu tertentu, dapat dilakukan reindexing sesuai dengan tahunnya menggunakan kode berikut.

```
air_df = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/TempData.csv')
air2016_df = air_df.drop(columns=['Year'])
air2016_df.set_index(['Month','Day','Time'],inplace=True)
air2016_df
```

Output:

			Temp
Month	Day	Time	
1	1	00:00:00	79.0
		00:30:00	79.0
		01:00:00	79.0
		01:30:00	77.0
		02:00:00	78.0
...	...	...	...
12	31	22:00:00	77.0
		22:30:00	77.0
		23:00:00	77.0
		23:00:00	77.0
		23:30:00	77.0

20453 rows × 1 columns

Untuk memilih sebuah data tertentu, dapat menambahkan kode berikut.

```
#Slicing data
air2016_df.loc[2,24,'00:30:00']
```

Output:

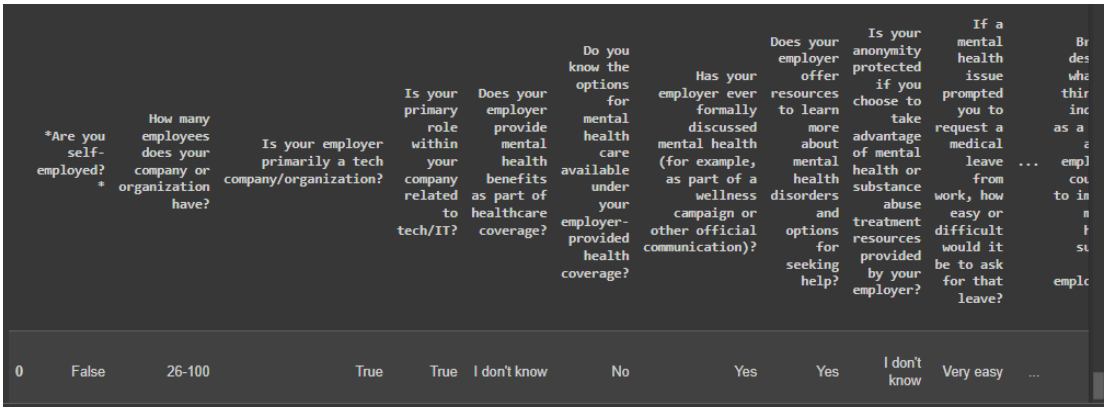
			Temp
Month	Day	Time	
2	24	00:30:00	77.0

Contoh 3: Mengubah Nama Kolom dan Baris

Apabila terdapat suatu data mungkin dari survei yang memiliki nama kolom yang sangat panjang akan lebih baik jika diubah menjadi nama lain yang lebih intuitif untuk memudahkan pengolahan data.

```
response_df = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/OSMI Mental Health in Tech Survey 2019.csv')
response_df.head(1)
```

Output:

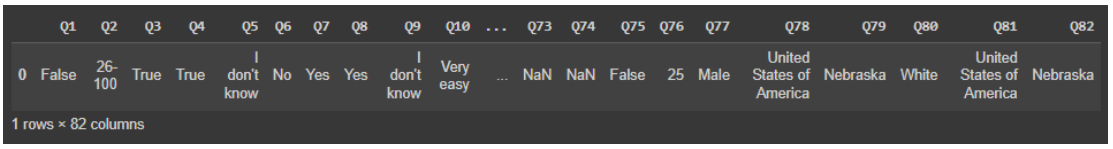


	*Are you self-employed? *	How many employees does your company or organization have?	Is your employer primarily a tech company/organization?	Is your primary role within your company related to tech/IT?	Does your employer provide mental health benefits as part of healthcare coverage?	Do you know the options for mental health care available under your employer-provided health coverage?	Has your employer ever formally discussed mental health (for example, as part of a wellness campaign or other official communication)?	Does your employer offer resources to learn more about mental health disorders and options for seeking help?	Is your anonymity protected if you choose to take advantage of mental health or substance abuse treatment resources provided by your employer?	If a mental health issue prompted you to request a medical leave from work, how easy or difficult would it be to ask for that leave?	...	Br des wha thir inc as a empl cou to in m h su empl
0	False	26-100	True	True	I don't know	No	Yes	Yes	I don't know	Very easy	...	

Untuk mengubah nama kolom menjadi nama yang lebih intuitif dapat menerapkan kode berikut ini.

```
response_df['Do you know the options for mental health care available
under your employer-provided health coverage?']
keys = ['Q{}'.format(i) for i in range(1,83)]
columns_dic = pd.Series(response_df.columns,index=keys)
columns_dic['Q4']
response_df.columns = keys
response_df.head(1)
```

Output:



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	...	Q73	Q74	Q75	Q76	Q77	Q78	Q79	Q80	Q81	Q82
0	False	26-100	True	True	I don't know	No	Yes	Yes	I don't know	Very easy	...	NaN	NaN	False	25	Male	United States of America	Nebraska	White	United States of America	Nebraska

1 rows × 82 columns

2. *Data cleaning* Level 2

Data cleaning level 2 mencakup membongkar, restrukturisasi, dan merumuskan ulang tabel.

Contoh 4: Membongkar tabel

Dengan menggunakan kode dari contoh 1, tambahkan kode berikut untuk membongkar nama file.

```
#Unpacking FileName
Months =
['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Oct','Sep','Nov','D
ec']
def SeparateCity(v):
```

```
    for mon in Months:
        if (mon in v):
            return v[:v.find(mon)]
pidato_df['City'] = pidato_df.FileName.apply(SeparateCity)

def SeparateDate(r):
    return r.FileName[len(r.City):r.FileName.find('.txt')]

pidato_df['Date'] = pidato_df.apply(SeparateDate,axis=1)
pidato_df.Date = pd.to_datetime(pidato_df.Date,format='%b%d_%Y')

def extractDMY(r):
    r['Day'] = r.Date.day
    r['Month'] = r.Date.month
    r['Year'] = r.Date.year
    return r
pidato_df = pidato_df.apply(extractDMY,axis=1)

pidato_df.drop(columns=['FileName'],inplace=True)

pidato_df.head()
```

Output:

	Content	City	Date	Day	Month	Year
0	Thank you very much. Thank you. Thank you. Tha...	Greenville	2019-07-17	17	7	2019
1	Thank you, thank you. Wow. Wow, and I'm thrill...	Henderson	2020-09-13	13	9	2020
2	Well, thank you very much. And hello Las Vegas...	LasVegas	2020-02-21	21	2	2020
3	So thank you Pennsylvania, very much. I'm thri...	Latrobe	2020-09-03	3	9	2020
4	Thank you very much and thank you to the origi...	Lexington	2019-11-04	4	11	2019

Selanjutnya untuk membongkar konten dari tabel dengan kriteria tertentu dapat menggunakan kode berikut.

```
#Unpacking content
Words = ['vote','tax','campaign','economy']

def FindWordRatio(row):
    total_n_words = len(row.Content.split(' '))
    for w in Words:
        row['r_{}'.format(w)] = row.Content.count(w)/total_n_words
    return row

pidato_df = pidato_df.apply(FindWordRatio,axis=1)
pidato_df.head()
```

Output:

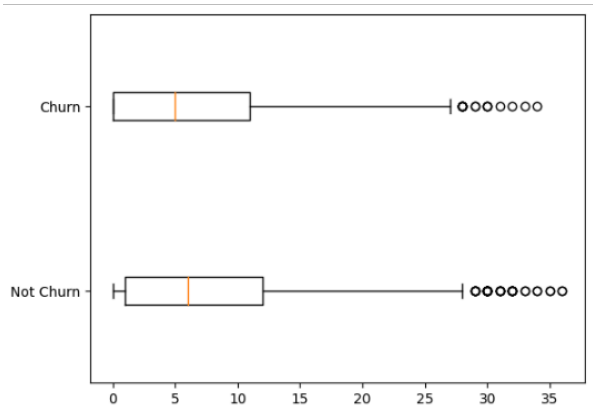
	Content	City	Date	Day	Month	Year	r_vote	r_tax	r_campaign	r_economy
0	Thank you very much. Thank you. Thank you. Tha...	Greenville	2019-07-17	17	7	2019	0.001603	0.000660	0.000283	0.000660
1	Thank you, thank you. Wow. Wow, and I'm thrill...	Henderson	2020-09-13	13	9	2020	0.001568	0.000560	0.000224	0.000448
2	Well, thank you very much. And hello Las Vegas...	LasVegas	2020-02-21	21	2	2020	0.000652	0.000434	0.000507	0.000072
3	So thank you Pennsylvania, very much. I'm thri...	Latrobe	2020-09-03	3	9	2020	0.001687	0.000161	0.000080	0.000482
4	Thank you very much and thank you to the origi...	Lexington	2019-11-04	4	11	2019	0.003017	0.000447	0.000670	0.000559

Contoh 5: Restrukturisasi tabel

Dalam contoh ini, akan digunakan dataset Customer Churn.csv. Dataset ini berisi catatan 3.150 pelanggan sebuah perusahaan telekomunikasi. Data mencakup informasi demografi seperti jenis kelamin dan usia, dan kolom aktivitas seperti jumlah panggilan berbeda dalam 9 bulan. Kumpulan data menentukan apakah setiap pelanggan melakukan penggantian atau tidak dalam 3 bulan setelah 9 bulan pengumpulan data aktivitas pelanggan. Perpindahan pelanggan, dari sudut pandang perusahaan telekomunikasi, berarti pelanggan berhenti menggunakan layanan perusahaan dan menerima layanan dari pesaing perusahaan. Data tersebut kemudian ditampilkan dalam bentuk boxplot sebagai berikut.

```
customer_df = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/Customer Churn.csv')
customer_df.head(1)

customer_df.columns = ['Call_Failure', 'Complains',
'Subscription_Length', 'Seconds_of_Use',
       'Frequency_of_use', 'Frequency_of_SMS',
'Distinct_Called_Numbers',
       'Status', 'Churn']

churn_possibilities = customer_df.Churn.unique()

box_sr = pd.Series('',index = churn_possibilities)

for poss in churn_possibilities:
    BM = customer_df.Churn == poss
    box_sr[poss] = customer_df[BM].Call_Failure.values

#Plot data origin
print(box_sr)

plt.boxplot(box_sr,vert=False)
plt.yticks([1,2],['Not Churn','Churn'])
plt.show()
```

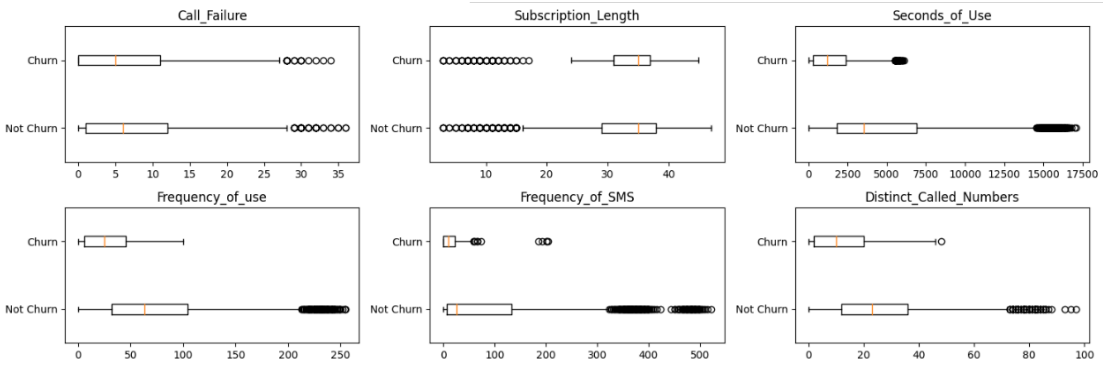
Output:



Selanjutnya dilakukan restrukturisasi data sehingga diperoleh representasi informasi dalam bentuk lain tetapi berasal dari data yang sama menggunakan kode berikut.

```
#Restrukturisasi data
select_columns = ['Call_Failure', 'Subscription_Length',
'Seconds_of_Use',
       'Frequency_of_use', 'Frequency_of_SMS',
'Distinct_Called_Numbers']
churn_possibilities = customer_df.Churn.unique()

plt.figure(figsize=(15,5))
for i,sc in enumerate(select_columns):
    for poss in churn_possibilities:
        BM = customer_df.Churn == poss
        box_sr[poss] = customer_df[BM][sc].values
    plt.subplot(2,3,i+1)
    plt.boxplot(box_sr,vert=False)
    plt.yticks([1,2],['Not Churn','Churn'])
    plt.title(sc)
plt.tight_layout()
plt.show()
```

Output:



Contoh 6: Merumuskan ulang tabel

Dalam contoh ini, digunakan data Electric_Production.csv untuk membuat prediksi. Pada kasus ini akan dibuat prediksi berapa kebutuhan listrik bulanan 1 bulan dari sekarang. Model prediksi dibuat berdasarkan data yang tersedia, berikut kodenya.

```
#Read RAW Data
month_df = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/Electric_Production.csv')
month_df
```

Output:

	DATE	IPG2211A2N
0	1/1/1985	72.5052
1	2/1/1985	70.6720
2	3/1/1985	62.4502
3	4/1/1985	57.4714
4	5/1/1985	55.3151
...	...	...
392	9/1/2017	98.6154
393	10/1/2017	93.6137
394	11/1/2017	97.3359
395	12/1/2017	114.7212
396	1/1/2018	129.4048

397 rows × 2 columns

Selanjutnya dilakukan penerapan *data cleaning* level 1 berupa penggantian judul kolom kedua agar lebih intuitif, penggantian tipe data kolom “DATE” menjadi datetime, dan indeks bawaan berubah ke kolom “DATE” untuk menyederhanakan data.

```
#Data Cleaning Level 1
month_df.columns = ['Date','Demand']
month_df.set_index(pd.to_datetime(month_df.Date,format='%m/%d/%Y'),in
place=True)
month_df.drop(columns=['Date'],inplace=True)
month_df
```

Output:

	Demand
Date	
1985-01-01	72.5052
1985-02-01	70.6720
1985-03-01	62.4502
1985-04-01	57.4714
1985-05-01	55.3151
...	...
2017-09-01	98.6154
2017-10-01	93.6137
2017-11-01	97.3359
2017-12-01	114.7212
2018-01-01	129.4048

397 rows × 1 columns

```
#Data Cleaning Level 2 - Merumuskan Ulang Tabel
attributes_dic={'IA1':'Average demand of the month',
            'IA2':'Slope of change for the demand of the month',
            'IA3': 'Average demands of months t-2, t-3 and t-4',
            'DA': 'Demand of month t'}
predict_df =
pd.DataFrame(index=month_df.iloc[24:].index,columns=attributes_dic.keys())
predict_df
```

Output:

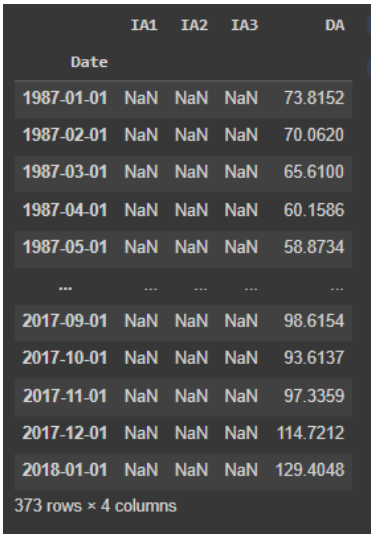
Date	IA1	IA2	IA3	DA
1987-01-01	NaN	NaN	NaN	NaN
1987-02-01	NaN	NaN	NaN	NaN
1987-03-01	NaN	NaN	NaN	NaN
1987-04-01	NaN	NaN	NaN	NaN
1987-05-01	NaN	NaN	NaN	NaN
...	...	...	...	...
2017-09-01	NaN	NaN	NaN	NaN
2017-10-01	NaN	NaN	NaN	NaN
2017-11-01	NaN	NaN	NaN	NaN
2017-12-01	NaN	NaN	NaN	NaN
2018-01-01	NaN	NaN	NaN	NaN

373 rows × 4 columns

Selanjutnya kolom atribut yang masih tidak memiliki nilai (NaN) ini diprediksikan nilainya dengan menerapkan model regresi linear. Berikut contoh prediksi untuk kolom DA.

```
#Mengisi NaN pada kolom DA
predict_df.DA = month_df.loc['1987-01-01':].Demand
predict_df
```

Output:



Date	IA1	IA2	IA3	DA
1987-01-01	NaN	NaN	NaN	73.8152
1987-02-01	NaN	NaN	NaN	70.0620
1987-03-01	NaN	NaN	NaN	65.6100
1987-04-01	NaN	NaN	NaN	60.1586
1987-05-01	NaN	NaN	NaN	58.8734
...	...	...	...	...
2017-09-01	NaN	NaN	NaN	98.6154
2017-10-01	NaN	NaN	NaN	93.6137
2017-11-01	NaN	NaN	NaN	97.3359
2017-12-01	NaN	NaN	NaN	114.7212
2018-01-01	NaN	NaN	NaN	129.4048

373 rows × 4 columns

3. *Data cleaning* Level 3

   Data cleaning level 3 merupakan proses pembersihan dataset yang paling dalam karena lebih detail dari pada tingkat sebelumnya. Berikut merupakan bentuk *data cleaning* level 3:

   a. Menghapus data pengamatan yang tidak diinginkan

   Ini termasuk menghapus nilai duplikat/redundan atau tidak relevan dari kumpulan dataset. Dalam pengumpulan dan pengamatan data, duplikat data pengamatan adalah kasus yang paling sering muncul. Selain itu adanya pengamatan yang tidak relevan dengan pengamatan yang sesuai dengan masalah spesifik yang ingin dipecahkan juga termasuk data pengamatan yang tidak diinginkan. Pengamatan yang berlebihan mengubah sebagian besar efisiensi karena data berulang dapat ditambahkan ke sisi yang benar atau ke sisi yang salah, sehingga menghasilkan hasil yang tidak tepat. Pengamatan yang tidak relevan adalah semua jenis data yang tidak berguna dan dapat dihapus secara langsung.

   Contoh 7: Menghapus semua baris yang punya nilai nol menggunakan `dropna()`

```
import pandas as pd
df = pd.read_csv('data.csv')
df.dropna(inplace = True)
print(df.to_string())
```

   b. Membetulkan galat struktural

   Galat struktural adalah galat yang timbul saat pengukuran data, transfer data, ataupun situasi yang mirip. Galat ini meliputi kesalahan ketik nama fitur, atribut yang sama dengan nama yang berbeda, nama label dalam kelas, adanya kelas yang terpisah padahal sama, ataupun penggunaan huruf besar yang tidak konsisten.

Contoh 8: Konversi data menjadi format yang benar

```
# Convert to_datetime()
import pandas as pd
df = pd.read_csv('data.csv')
df['Date'] = pd.to_datetime(df['Date'])
print(df.to_string())
```

Contoh 9: Menemukan data duplikat menggunakan `duplicate()`

```
# Discover duplicate
print(df.duplicated())
# Remove all duplicates
df.drop_duplicates(inplace = True)
```

c. Mengelola outlier yang tidak diinginkan

Outlier dapat menyebabkan masalah pada jenis model *Machine Learning* tertentu apabila tidak dikelola dengan tepat. Contohnya, pengaruh adanya *outlier* pada model regresi linear tidak begitu terpengaruh dibandingkan dengan model Decision Tree. Logikanya *outlier* tidak boleh dihapus hingga ada rasionalisasi yang kuat untuk menghapusnya. Karena dengan menghapus *outlier* bisa jadi menyebabkan peningkatan kinerja atau justru menurunkan. Alasan yang dapat membuat data *outlier* dihapus seperti adanya kesalahan pengukuran data. Terdapat beberapa alternatif penanganan outlier mulai dari dibiarkan saja, mengganti nilai dengan nilai batas atas atau bawahnya, melakukan transformasi logaritmik, hingga menghapus data yang mengandung outlier.

	Music	Slow songs or fast songs	Dance	Folk	Country	Classical music	Musical	Pop	Rock	Metal or Hardrock	...	Age	Height	Weight	Number of siblings	Gender	Left - right handed	Education	Only child	Village - town	House - block of flats
292	5.0	4.0	5.0	2.0	3.0	2.0	5.0	5.0	4.0	1.0	...	21.0	184.0	120.0	1.0	female	right handed	secondary school	no	city	house/bungalow
612	5.0	3.0	5.0	1.0	1.0	3.0	2.0	3.0	4.0	3.0	...	23.0	172.0	110.0	2.0	male	right handed	secondary school	no	village	block of flats
715	4.0	3.0	2.0	3.0	3.0	3.0	4.0	2.0	5.0	5.0	...	29.0	183.0	111.0	1.0	male	left handed	secondary school	no	village	house/bungalow
793	2.0	3.0	3.0	1.0	2.0	3.0	3.0	3.0	2.0	4.0	...	24.0	185.0	120.0	2.0	male	right handed	doctorate degree	no	city	house/bungalow
796	3.0	5.0	3.0	2.0	2.0	4.0	3.0	5.0	1.0	4.0	...	27.0	189.0	113.0	3.0	male	left handed	doctorate degree	no	city	house/bungalow
859	5.0	3.0	4.0	4.0	2.0	3.0	3.0	4.0	3.0	2.0	...	20.0	190.0	125.0	1.0	male	right handed	secondary school	no	city	block of flats
885	3.0	4.0	3.0	2.0	2.0	2.0	3.0	4.0	4.0	4.0	...	23.0	NaN	165.0	0.0	female	right handed	secondary school	yes	city	house/bungalow
973	5.0	3.0	2.0	5.0	5.0	4.0	5.0	2.0	5.0	5.0	...	20.0	175.0	120.0	2.0	female	right handed	secondary school	no	city	block of flats
992	4.0	4.0	4.0	1.0	4.0	4.0	1.0	3.0	4.0	4.0	...	30.0	200.0	150.0	1.0	male	right handed	masters degree	no	city	block of flats

9 rows × 150 columns

Contoh 10: Mendiagnosa *outlier*

Dua contoh berikut menampilkan deteksi *outlier* univariat. Dalam contoh ini, digunakan file responses.csv dan columns.csv. Kedua file tersebut berisi catatan tanggal survei yang dilakukan di Slovakia.

```
#Open data "Columns"
```

```
column_df = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/columns.csv')
column_df.head(2)
```

Output:

	original	short
0	I enjoy listening to music.	Music
1	I prefer.	Slow songs or fast songs

```
#Open data "Responses"
response_df = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/responses.csv')
response_df.head(2)
```

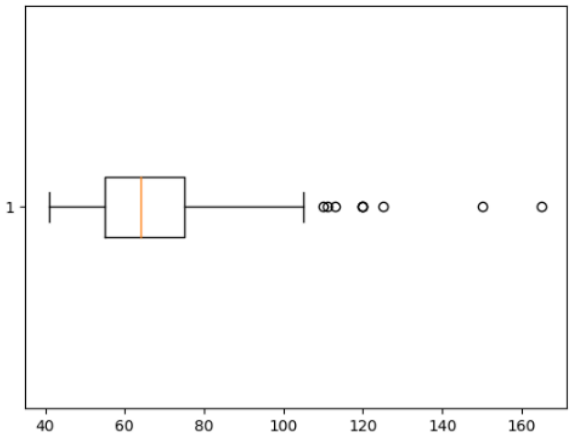
Output:

	Music	Slow songs or fast songs	Dance	Folk	Country	Classical music	Musical	Pop	Rock	Metal or Hardrock	...	Age	Height	Weight	Number of siblings	Gender	Left - right handed	Education	Only child	Village - town	House - block of flats
0	5.0	3.0	2.0	1.0	2.0	2.0	1.0	5.0	5.0	1.0	...	20.0	163.0	48.0	1.0	female	right handed	college/bachelor degree	no	village	block of flats
1	4.0	4.0	2.0	1.0	1.0	1.0	2.0	3.0	5.0	4.0	...	19.0	163.0	58.0	2.0	female	right handed	college/bachelor degree	no	city	block of flats

2 rows × 150 columns

```
#Mendeteksi outlier dari satu atribut numerik
fig = plt.boxplot(response_df.Weight.dropna(),vert=False)
response_df[response_df.Weight>105]
```

Output:



Lingkaran yang berada sebelum batas bawah dan setelah batas atas mewakili objek data dalam data yang secara statistik terlalu berbeda dari angka lainnya, sehingga bisa digolongkan sebagai outlier. Lingkaran-lingkaran ini disebut “fliers” dalam konteks analisis boxplot. Sehingga data outliers yang diperoleh sebagai berikut.

d. Menangani data yang hilang

Data yang hilang (*missing data*) adalah data yang seharusnya memiliki informasi penting sebagai anggota dari *dataset*, tetapi tidak eksis dalam *dataset*. Langkah pertama yang perlu dilakukan untuk menangani data yang hilang adalah dengan menyadari, mendeteksi, dan menandai data yang hilang dalam *dataset*. Selanjutnya, data yang hilang tersebut perlu diisi dengan nilai tertentu. Untuk data numerik, *missing data* dapat diisi dengan nilai nol, rerata, median, atau modus dari keseluruhan *dataset*.

Contoh 11: Menggantikan nilai nol dengan nilai tertentu menggunakan `fillna()`

```
import pandas as pd
df = pd.read_csv('data.csv')
df.fillna(100, inplace = True)
print(df.to_string())
```

Contoh 12: Menggantikan nilai pada kolom tertentu menggunakan `fillna()`

```
import pandas as pd
df = pd.read_csv('data.csv')
df["Calories"].fillna(100, inplace = True)
print(df.to_string())
```

Contoh 13: Menggantikan nilai pada sel yang kosong dengan rerata menggunakan `mean()`

```
import pandas as pd
df = pd.read_csv('data.csv')
x = df["Calories"].mean()
df["Calories"].fillna(x, inplace = True)
print(df.to_string())
```

Untuk mengubah nilai dengan nilai median dan modus dapat menggunakan syntax `median()` dan `mode()`.

B. Data Reduction

Data reduction adalah proses untuk mereduksi data dengan tujuan agar pengolahan dan ekstraksi fitur bisa lebih sederhana tetapi dengan tetap mempertahankan hasil yang ingin dicapai. Bentuk data reduction ada 2 yaitu secara numerik dan dimensi. Reduksi data secara numerik dapat dilakukan dalam bentuk:

- Random Sampling: Memilih beberapa objek data secara acak untuk menghindari biaya komputasi yang tidak terjangkau.
- Pengambilan Sampel Berstrata (Stratified Sampling): Memilih beberapa objek data secara acak untuk menghindari biaya komputasi yang tidak terjangkau, sambil mempertahankan representasi rasio sub-populasi dalam sampel.

- Random Over/Under Sampling: Memilih beberapa objek data secara acak untuk menghindari biaya komputasi yang tidak terjangkau, sambil menciptakan representasi sub-populasi yang ditentukan dalam sampel.

Berikut ini adalah metode reduksi dimensi:

- Regresi Linier: Menggunakan analisis regresi untuk menyelidiki kekuatan prediksi atribut independen untuk memprediksi atribut dependen tertentu.
- *Decision Tree*: Menggunakan algoritma pohon keputusan untuk menyelidiki kekuatan prediksi atribut independen untuk memprediksi atribut dependen tertentu.
- *Random Forest*: Menggunakan algoritma random forest untuk menyelidiki kekuatan prediksi atribut independen untuk memprediksi atribut dependen tertentu.
- *Brute-force Computational Dimension Reduction*: Eksperimen komputasi untuk mencari subset terbaik dari atribut independen yang menghasilkan prediksi paling sukses dari atribut dependen.
- *Principal Component Analysis* (PCA): Merepresentasikan data dengan mentransformasikan sumbu sedemikian rupa sehingga sebagian besar variasi data dijelaskan oleh atribut pertama dan atribut-atribut tersebut ortogonal satu sama lain.
- *Functional Data Analysis* (FDA): Mewakili data menggunakan lebih sedikit titik menggunakan representasi fungsional.

C. Data Transformation: Rescale Data

Rescaling data atau penskalaan data dilakukan ketika atribut rentang data bervariasi antara satu data dengan lainnya. Hal ini perlu dilakukan karena keseragaman rentang data mempengaruhi proses selanjutnya seperti ekstraksi fitur. Selain itu juga pada algoritma yang menggunakan pembobotan input seperti regresi, neural network, ataupun algoritma yang menerapkan pengukuran jarak seperti K-Nearest Neighbor, perlu dataset yang merupakan normalisasi dari rentang data yang heterogen. Penerapan penskalaan data pada Python dapat dilakukan menggunakan kelas `MinMaxScaler` dari Scikit-Learn.

Contoh 14: *Rescale Data* antara 0 dan 1

```
# importing libraries
import pandas
import scipy
import numpy
from sklearn.preprocessing import MinMaxScaler

# Open data set
url = 'data.csv'
# data parameters
names = ['Duration','Date','Pulse','Maxpulse','Calories']

# preparation of dataframe using the data at given link and defined
columns list
dataframe = pandas.read_csv(url, names = names)
array = dataframe.values
```

```
# Select column to rescale
calories = array[:,4].reshape(-1,1)

# Rescaling data
scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX = scaler.fit_transform(calories)
```

D. *Data Transformation: Binarize Data*

Binerisasi data dilakukan untuk mengubah data menjadi sebuah nilai biner yang dikategorikan sesuai ambang (*threshold*). Semua nilai di atas ambang akan diberi nilai 1 dan seluruh data yang kurang dari atau sama dengan ambang dinilai 0 (nol).

Contoh 15: Binerisasi Data

```
from sklearn.preprocessing import Binarizer
import pandas
import numpy

# Open data set
url = 'data.csv'
# data parameters
names = ['Duration','Date','Pulse','Maxpulse','Calories']

# preparation of dataframe using the data at given link and defined
columns list
dataframe = pandas.read_csv(url, names = names)
array = dataframe.values

# Select column to rescale
pulse = array[:,2]

# Binarize data
binarizer = Binarizer(threshold = 100.0).fit(pulse)
binaryX = binarizer.transform(pulse)
```

E. *Data Transformation: Standardize Data*

Standarisasi data merupakan teknik yang berguna untuk mengubah atribut data nilai rerata dan standar deviasi sesuai standar distribusi Gaussian, yaitu rerata 0 dan standar deviasi 1.

Contoh 16: Standarisasi Data

```
# importing libraries
from sklearn.preprocessing import StandardScaler
import pandas
import numpy

# Open data set
```

```
url = 'data.csv'
# data parameters
names = ['Duration','Date','Pulse','Maxpulse','Calories']

# preparation of dataframe using the data at given link and defined
columns list
dataframe = pandas.read_csv(url, names = names)
array = dataframe.values

# Select column to rescale
calories = array[:,4]

# Standardize data
scaler = StandardScaler().fit(calories)
rescaledX = scaler.transform(calories)
```

Tugas Praktikum

1. Cobalah seluruh contoh 1 hingga 16 menggunakan dataset yang telah disediakan.
2. Lakukan data *cleaning* pada dataset Auto MPG yang dapat diakses pada link (http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data). Identifikasilah:
   - Mean, max, min, modus untuk kolom atribut 1-8.
   - Diagnosa *missing value* dan data NaN untuk setiap kolom.

Tuliskan dalam tabel berikut.

Atribut	Mean	Max	Min	Modus	Missing Value/NaN
1					
2					
…					
8					

Note:

Berikut informasi singkat mengenai atribut dataset Auto MPG.

Attribute Information:

1. mpg: continuous
2. cylinders: multi-valued discrete
3. displacement: continuous
4. horsepower: continuous
5. weight: continuous
6. acceleration: continuous
7. model year: multi-valued discrete
8. origin: multi-valued discrete
9. car name: string (unique for each instance)

Untuk mengetahui representasi data Silahkan print data yang ada.

3. Lakukan penanganan pada *missing value* yang terdeteksi dengan menggantikannya dengan nilai yang paling banyak muncul. Cek ulang apakah masih ada *missing value* yang terdeteksi.
4. Buatlah salah satu penerapan algoritma dari teknik *data reduction* regresi linier menggunakan dataset Auto MPG, visualisasikan dan analisis hasilnya.
5. Coba implementasikan teknik rescale, binarize, dan standardize data pada dataset Auto MPG. Silahkan pilih parameter yang cocok untuk penerapan setiap teknik di atas, analisislah hasilnya.

Referensi

Ben Auffarth, 2021, "*Machine Learning for Time-Series with Python*", PACKT Publishing.

Gopal Sakarkar, Gaurav Patil, dan Prateek Dutta, 2021, "*Machine Learning Algorithms Using Python Programming*", Nova Science Publishers, Inc.

Roy Jafari, 2022, "*Hands-On Data Preprocessing in Python - Learn How to Effectively Prepare Data for Successful Data Analytics*", PACKT Publishing.