PJJ PENGOLAHAN DATA DENGAN PYTHON

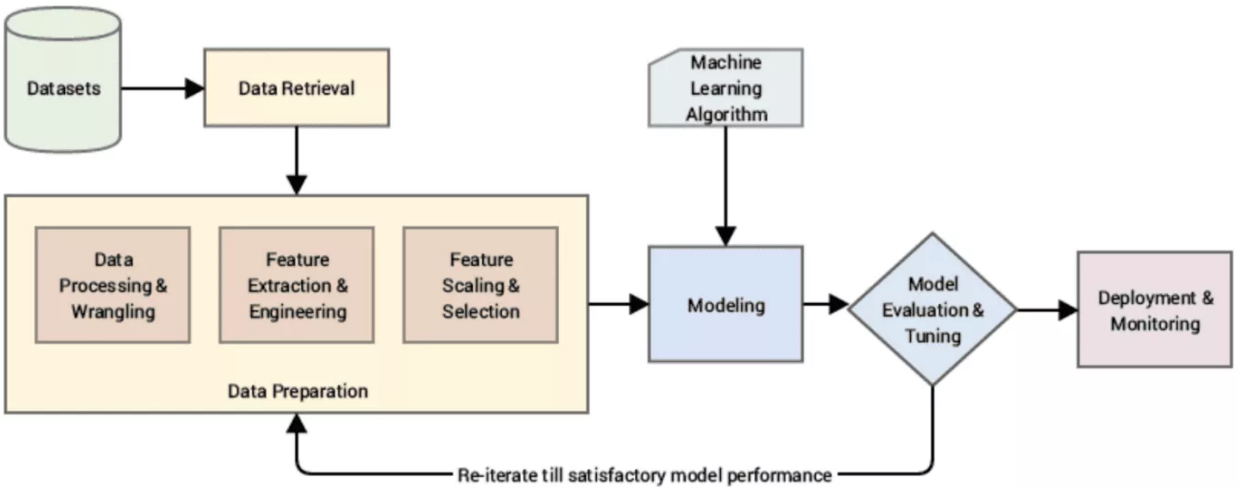
Selasa, 21 Juni 2022

Feature Engineering

Daftar Isi:

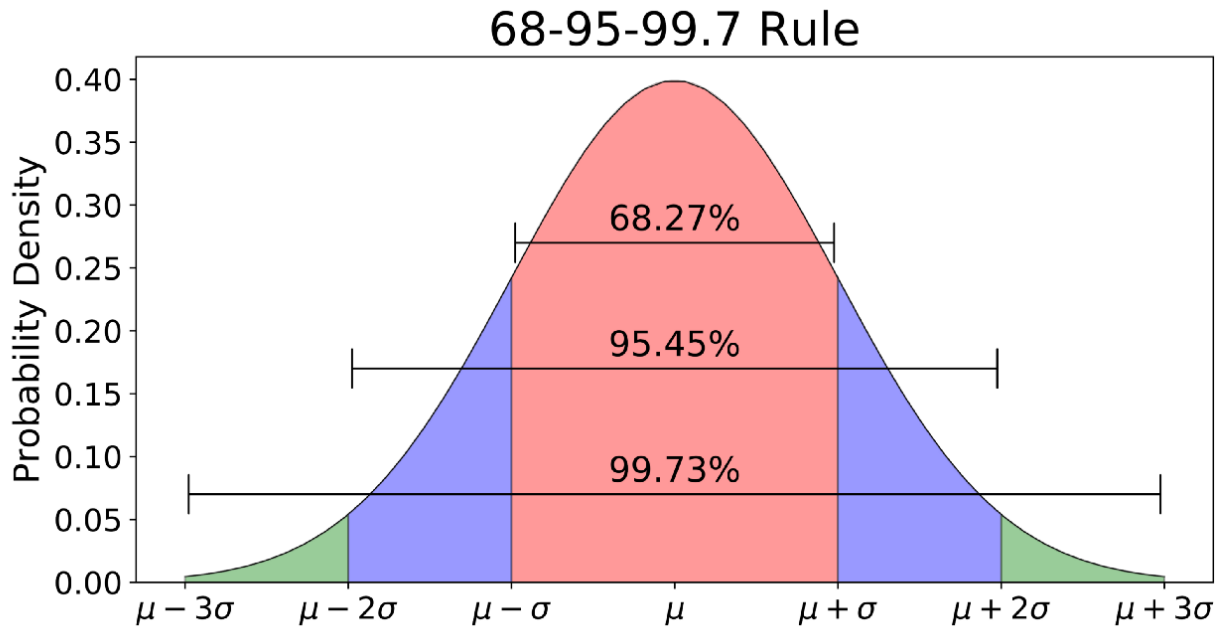
Feature engineering adalah proses menggunakan domain knowledge sehingga meningkatkan kemampuan preediksi dari algoritma machine learning.

- Feature Extraction - membuat feature dari feature-feature yang sudah ada dan menanggalkan feature yang lama untuk mengurangi feature pada dataset.
- Feature Scaling & Selection - proses memilih variabel input untuk model.



1. Data Pencilan (Outlier)

1.1 Three Sigma Rule



$P(a-3\sigma<X<a+3\sigma)=0.99730$

Kriteria Outlier untuk **Three Sigma Rule** :

> $|x_K-\bar{x}|>3\sigma$

Keterangan :

- x_K adalah data ke-k,
- $\bar{x}$ adalah rata-rata data,
- σ adalah standar deviasi

In [1]:
```
# PEMBUATAN FUNGSI UNTUK KRITERIA OUTLIER
def three_sigma_outlier (df):
    if (abs(potassium-df['potassium'].mean())>(3*df['potassium'].std())):
        return "Outlier"
    else:
        return "Not Outlier"
```

In [2]:
```
import pandas as pd
df=pd.read_csv('UScereal.csv')
df.head(8)
```

Out[2]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vit
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	en
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	en
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	en
3	Apple Cinnamon Cheerios	G	146.66667	2.666667	2.666667	240.00000	2.000000	14.00000	13.333333	1	93.33333	en
4	Apple Jacks	K	110.00000	2.000000	0.000000	125.00000	1.000000	11.00000	14.000000	2	30.00000	en

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vit
5	Basic 4	G	173.33333	4.000000	2.666667	280.00000	2.666667	24.00000	10.666667	3	133.33333	en
6	Bran Chex	R	134.32836	2.985075	1.492537	298.50746	5.970149	22.38806	8.955224	1	186.56716	en
7	Bran Flakes	P	134.32836	4.477612	0.000000	313.43284	7.462687	19.40299	7.462687	3	283.58209	en

In [3]:

```
# LOAD FUNGSI UNTUK KRITERIA OUTLIER
def three_sigma_outlier (potassium):
    if (abs(potassium-df['potassium'].mean())>(3*df['potassium'].std())):
        return "Outlier"
    else:
        return "Not Outlier"

    # APLIKASI KE TABEL
df['result_3sigma']=df['potassium'].apply(three_sigma_outlier)
df.head()
```

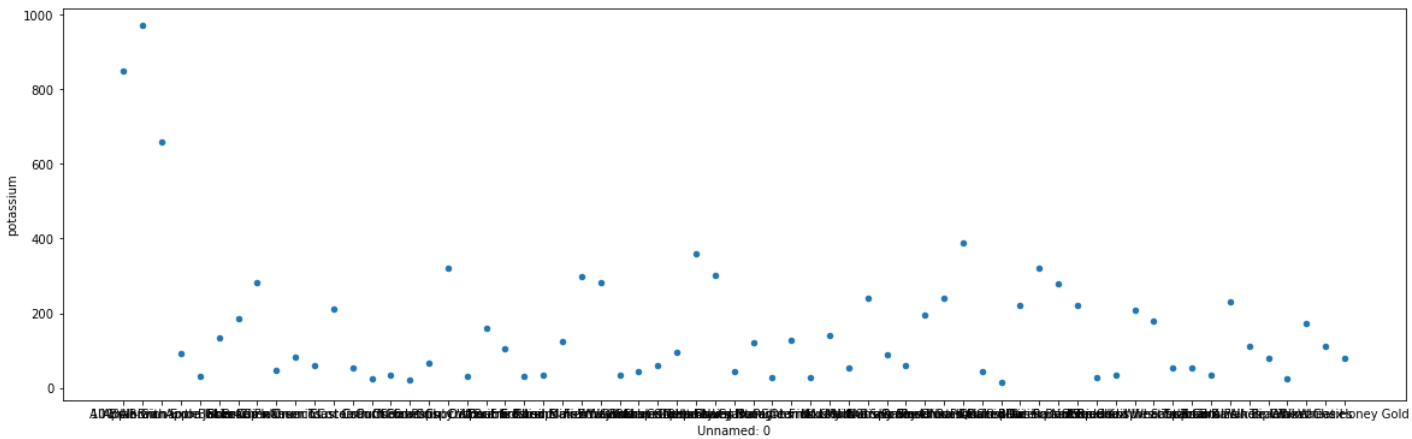
Out[3]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vit
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	en
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	en
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	en
3	Apple Cinnamon Cheerios	G	146.66667	2.666667	2.666667	240.00000	2.000000	14.00000	13.333333	1	93.33333	en
4	Apple Jacks	K	110.00000	2.000000	0.000000	125.00000	1.000000	11.00000	14.000000	2	30.00000	en

In [4]:

```
# VISUALISASI OUTLIER
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
plt.rcParams["figure.figsize"] = (20,6)

df.plot(x ='Unnamed: 0', y='potassium', kind = 'scatter')
plt.show()
```



In [5]:

```
# MELIHAT OUTLIER
df.loc[df['result_3sigma']=='Outlier']
```

Out[5]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vita
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	en
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	en

In [6]:

```
# JIKA HENDAK MEMBUAT DATASET BARU MENGELUARKAN OUTLIER
df_net=df.loc[df['result_3sigma']=='Not Outlier']
df_net.head()
```

Out[6]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vitar
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	enri
3	Apple Cinnamon Cheerios	G	146.66667	2.666667	2.666667	240.00000	2.000000	14.00000	13.333333	1	93.33333	enri
4	Apple Jacks	K	110.00000	2.000000	0.000000	125.00000	1.000000	11.00000	14.000000	2	30.00000	enri
5	Basic 4	G	173.33333	4.000000	2.666667	280.00000	2.666667	24.00000	10.666667	3	133.33333	enri
6	Bran Chex	R	134.32836	2.985075	1.492537	298.50746	5.970149	22.38806	8.955224	1	186.56716	enri

1.2 Hampel Identifier

Kriteria Outlier untuk Hampel Identifier :

> Median Absolute Value from The Median (MADM)

$$MADM(x) = 1.4826 \times median\{|x_K - x^+|\}$$

Keterangan :

- x_K adalah data ke K
- x^+ adalah median dari data

In [7]:

```
# BUAT KOLOM UNTUK $x_k-x+$
med=df['potassium'].quantile(0.55)
def abs_med_dev (potassium):
    return abs(potassium-med)
df['amd']=df['potassium'].apply(abs_med_dev)
df.head()
```

Out[7]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vita
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	en
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	en

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vit
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	en
3	Apple Cinnamon Cheerios	G	146.66667	2.666667	2.666667	240.00000	2.000000	14.00000	13.333333	1	93.33333	en
4	Apple Jacks	K	110.00000	2.000000	0.000000	125.00000	1.000000	11.00000	14.000000	2	30.00000	en

In [8]:

```
med_abs = df['amd'].quantile(0.50)
def hampel_outlier (potassium):
    madm=1.4826*med_abs
    if (potassium>3*madm):
        return "Outlier"
    else:
        return "Not Outlier"
df['result_hampel']=df['potassium'].apply(hampel_outlier)
df.head()
```

Out[8]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vit
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	en
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	en
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	en
3	Apple Cinnamon Cheerios	G	146.66667	2.666667	2.666667	240.00000	2.000000	14.00000	13.333333	1	93.33333	en
4	Apple Jacks	K	110.00000	2.000000	0.000000	125.00000	1.000000	11.00000	14.000000	2	30.00000	en

In [9]:

```
# MELIHAT OUTLIER
df.loc[df['result_hampel']=='Outlier']
```

Out[9]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vi
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	e
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	e
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	e
30	Grape-Nuts	P	440.00000	12.000000	0.000000	680.00000	12.000000	68.00000	12.000000	3	360.00000	e
44	Post Nat. Raisin Bran	P	179.10448	4.477612	1.492537	298.50746	8.955224	16.41791	20.895522	3	388.05970	e

1.3 Boxplot Outlier Rule

Kriteria Outlier untuk **Boxplot Outlier Rule** :

- $x_K > x_U + 1.5Q$
- $x_K > x_L - 1.5Q$

Keterangan :

- x_K adalah data ke-k,
- x_U adalah kuartil ke-1 atau disebut kuartil bawah (*lower quartile*),
- x_L adalah kuartil ke-3 atau disebut kuartil bawah (*upper quartile*),
- Q adalah jangkuan interkuartil (selisih kuartil bawah - kuartil atas)

In [10]:
```
# FUNGSI OUTLIER RULE
low_q=df['potassium'].quantile(0.25)
upr_q=df['potassium'].quantile(0.75)
iq_d=df['potassium'].quantile(0.75)-df['potassium'].quantile(0.25)
def boxplot_outlier (potassium):
    if (potassium>upr_q+1.5*iq_d) or (potassium<low_q-1.5*iq_d):
        return "Outlier"
    else:
        return "Not Outlier"
```

In [11]:
```
df['result_boxplot']=df['potassium'].apply(boxplot_outlier)
df.head()
```

Out[11]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vit
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	en
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	en
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	en
3	Apple Cinnamon Cheerios	G	146.66667	2.666667	2.666667	240.00000	2.000000	14.00000	13.333333	1	93.33333	en
4	Apple Jacks	K	110.00000	2.000000	0.000000	125.00000	1.000000	11.00000	14.000000	2	30.00000	en

In [12]:
```
# MELIHAT OUTLIER
df.loc[df['result_boxplot']=='Outlier']
```

Out[12]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vit
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	en
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	en
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	en

> **PERTANYAAN ISENG-ISENG**
>
> Apakah data yang terdistribusi normal memiliki outlier ? Bila ya mengapa, bila tidak mengapa ?

2. Normalitas Data

Apakah semua algoritma machine learning mengharapkan normalitas data ? --> TIDAK

- Algoritma yang membutuhkan normalisasi :
  - Algoritma yang bersifat ' curve fitting algorithms' seperti regresi linear/non-linear/logistik, KNN, SVM, Neural Networks, clustering algorithms like k-means clustering etc.
  - Algoritma yang menggunakan faktorisasi matriks, dekomposisi, dan reduksi dimensi seperti PCA, SVD, Factorization Machines
- Algoritma yang tidak membutuhkan normalisasi :
  - tree based algorithms -
    - CART,
    - Random Forests,
    - Gradient Boosted Decision Trees etc
- Hati-hati, perhatikan apanya yang butuh normalisasi ? Apakah regresi linear perlu normalisasi data ? TIDAK! Regresi linear memerlukan normalitas pada sisa residu hasil regresi.

> > Perlu pemahaman yang tepat terhadap algoritma yang anda gunakan.

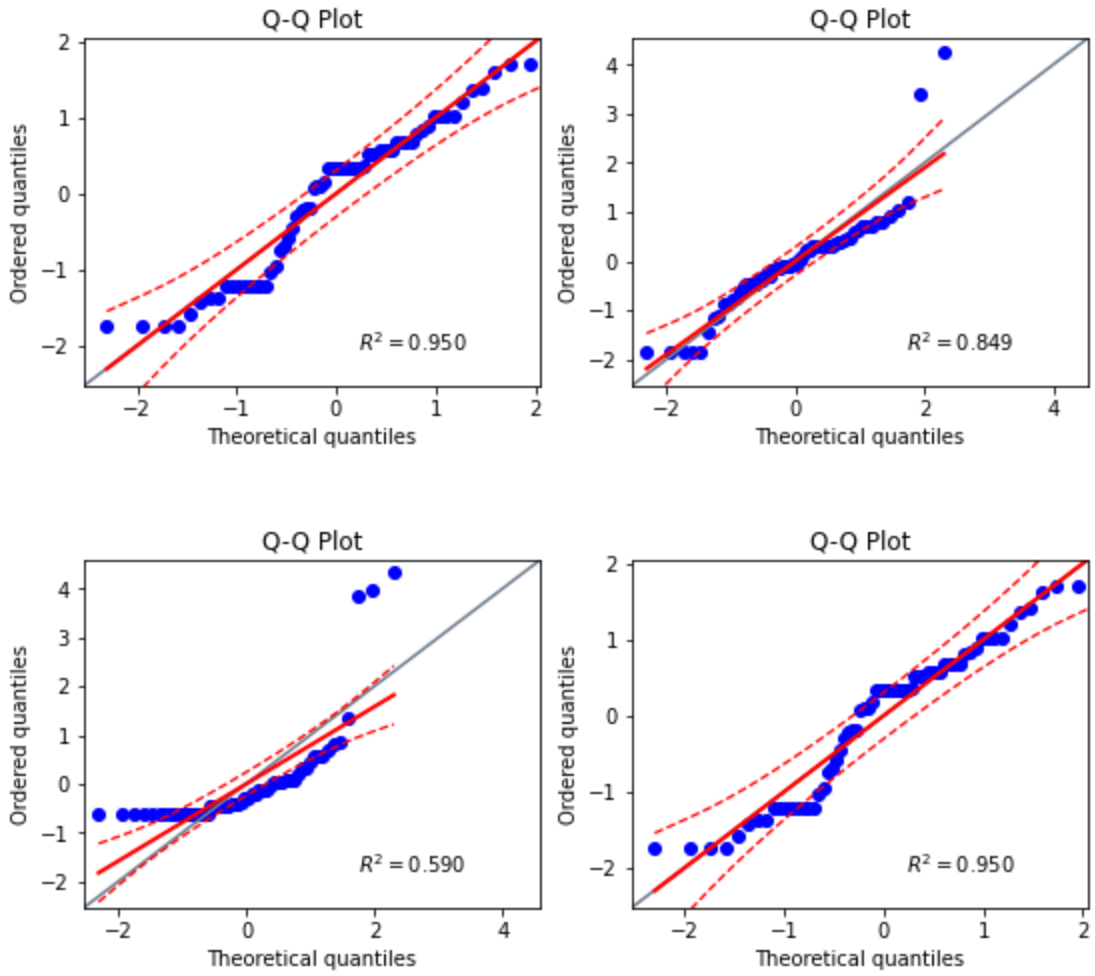
2.1 QQ Plot

QQ Plot **(quantile-quantile plot)** merupakan instrumen grafis yang membantu untuk menilai apakah suatu dataset memenuhi distribusi teoritikal (distribusi normal atau eksponensial). Misalkan, bila kita melakukan analisis statistik yang mengasumsikan variabel dependen distribusi normal kita bisa menyandingkan.

> > **Berhenti dan berfikir** </br> Data seperti apakah yang bisa diuji normalitasnya menggunakan qqplot?

Untuk memvisualisasikan QQ Plot kita bisa menggunakan library pingouin. Jangan lupa install dulu librarynya : </br> `pip install pingouin`

In [13]:
```
import pingouin as pg
import pandas as pd
import matplotlib.pyplot as plt
fig, ((qplot_sugars, qplot_sodium), (qplot_fibre, qplot_protein)) = plt.subplots(2, 2, fig
fig.subplots_adjust(hspace=0.5)
qp_sugars = pg.qqplot(df['sugars'], ax=qplot_sugars, dist='norm')
qp_sodium = pg.qqplot(df['sodium'],ax=qplot_sodium, dist='norm')
qp_fibre = pg.qqplot(df['fibre'], ax=qplot_fibre, dist='norm')
qp_protein = pg.qqplot(df['sugars'], ax=qplot_protein,dist='norm')
```



Untuk mengatasi ketaknormalan data bisa menggunakan transformasi variabel, menggunakan

- log transformation,
- square root transformation,
- reciprocal transformation,

> **Berhenti dan berfikir** </br> Mengapa transformasi data umumnya menggunakan log/square root transformation ?

3. Data Hilang (Missing Value)

Misalkan anda melakukan analisis data terhadap data peminjaman BMN di suatu kantor. Dari daftar peminjaman BMN tersebut anda melihat ada suatu list peminjaman sbb :

No	BMN	Nama Peminjam	Jumlah Dipinjam
1	Mobil Avanza	Budi	4
2	Laptop Dell	Ali	3
3	Proyektor Panasonic	Iqbal	8
4	Laptop Asus	Susi	
5	Mesin Fotokopi Konica	Ani	6
...	...	...	...
21	Laptop Acer	Rudi	
22	Mesin Fotokopi	Ali	6

No	BMN	Nama Peminjam	Jumlah Dipinjam
23	Proyektor Panasonic	Mina	4
...	...	...	...
87	Mobil Alphard	Susi	3
88	Laptop Lenovo	Mikail	9
100	Mobil Nissan	Joni	

Dari data di atas terlihat ada peminjaman yang hari peminjamannya tidak tercatat. Anda mencari tahu mengapa data tersebut hilang. Operator BMN mengatakan bahwa memang sekitar 5% pencatatan lupa dicatat jumlah barang yang dipinjam. Anda menanyakan apakah 5% tersebut spesifik pada jenis barang tertentu yang dipinjam atau user tertentu. Petugas mengatakan tidak, itu terjadi merata di seluruh pencatatan.

- **Missing Completely at Random (MCAR)** </br> Dengan mengasumsikan pernyataan petugas benar, maka kasus di atas dapat dianggap sebagai MCAR, yakni data yang hilang tidak memiliki relasi dengan variabel yang sedang diobservasi.
- **Missing at Random (MAR)** </br> Data tidak hilang secara acak tetapi bisa diprediksi dari variabel-variabel lain yang sedang diobservasi maka kasus tersebut disebut sebagai MAR. Misalkan, data yang hilang bisa diprediksi dari jenis BMN yang diprediksi atau dari orang yang meminjam.
- **Missing Not At Random (MNAR)** </br> Bila memiliki pola yang dapat dijelaskan namun diluar variabel yang diobservasi disebut MNAR . Misalkan, data yang hilang dapat diprediksi dari apakah pada tanggal peminjaman hari cerah atau hujan, tetapi kita tidak mengobservasi hal tersebut.

Lebih jauh tentang ini :

- Youtube - Missing Data Analysis, Mplus Short Course

Cara untuk melihat data kosong :

- Menggunakan library missingno (jangan lupa install terlebih dahulu)
- menggunakan df.isnull()

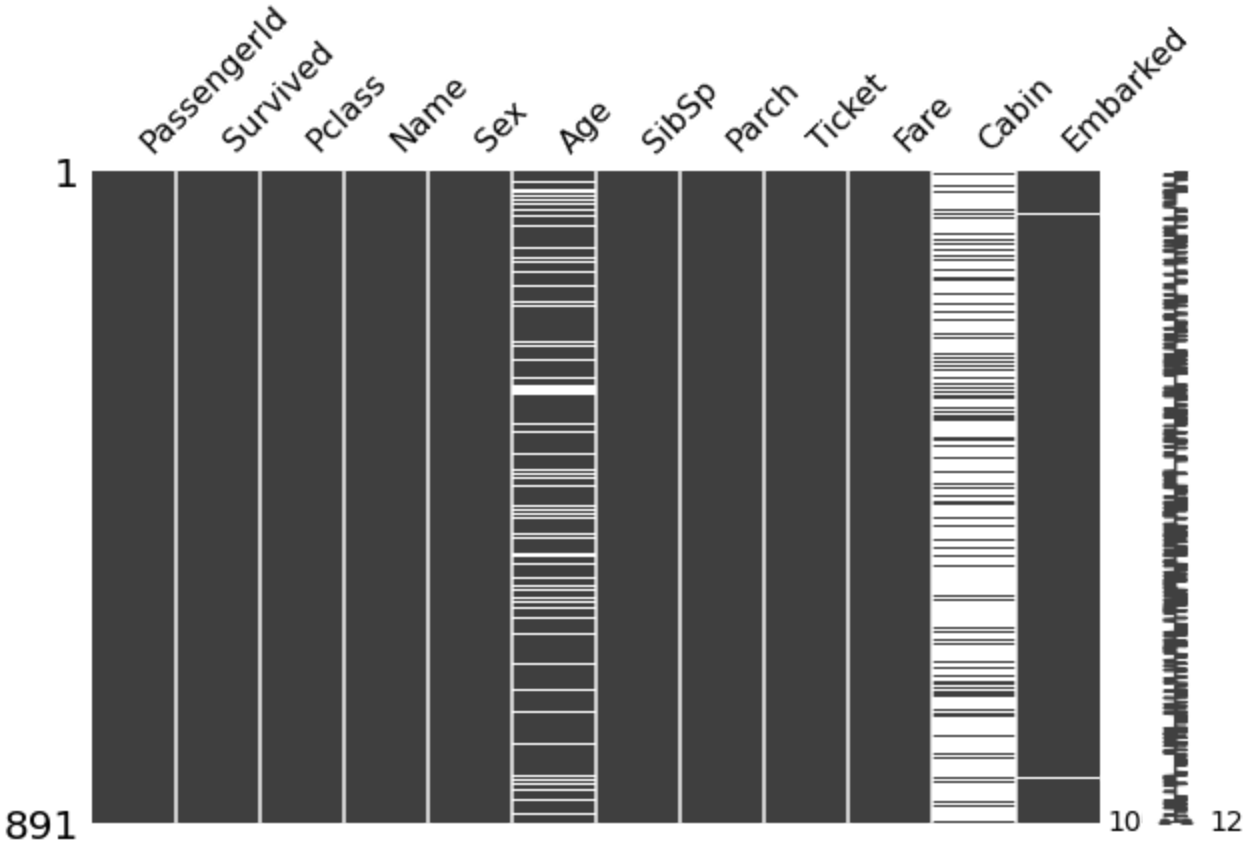
In [14]:
```
df_titanic = pd.read_csv('https://raw.githubusercontent.com/audit-ti/pjj-pengolahan-data-p
df_titanic.isnull().sum()
```

Out[14]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [15]:
```
# visualiasi missing value dengan bantuan library
import pandas as pd
import missingno as msno
df_titanic.head()
msno.matrix(df_titanic, figsize=(10, 6))
```

```
<AxesSubplot:>
```

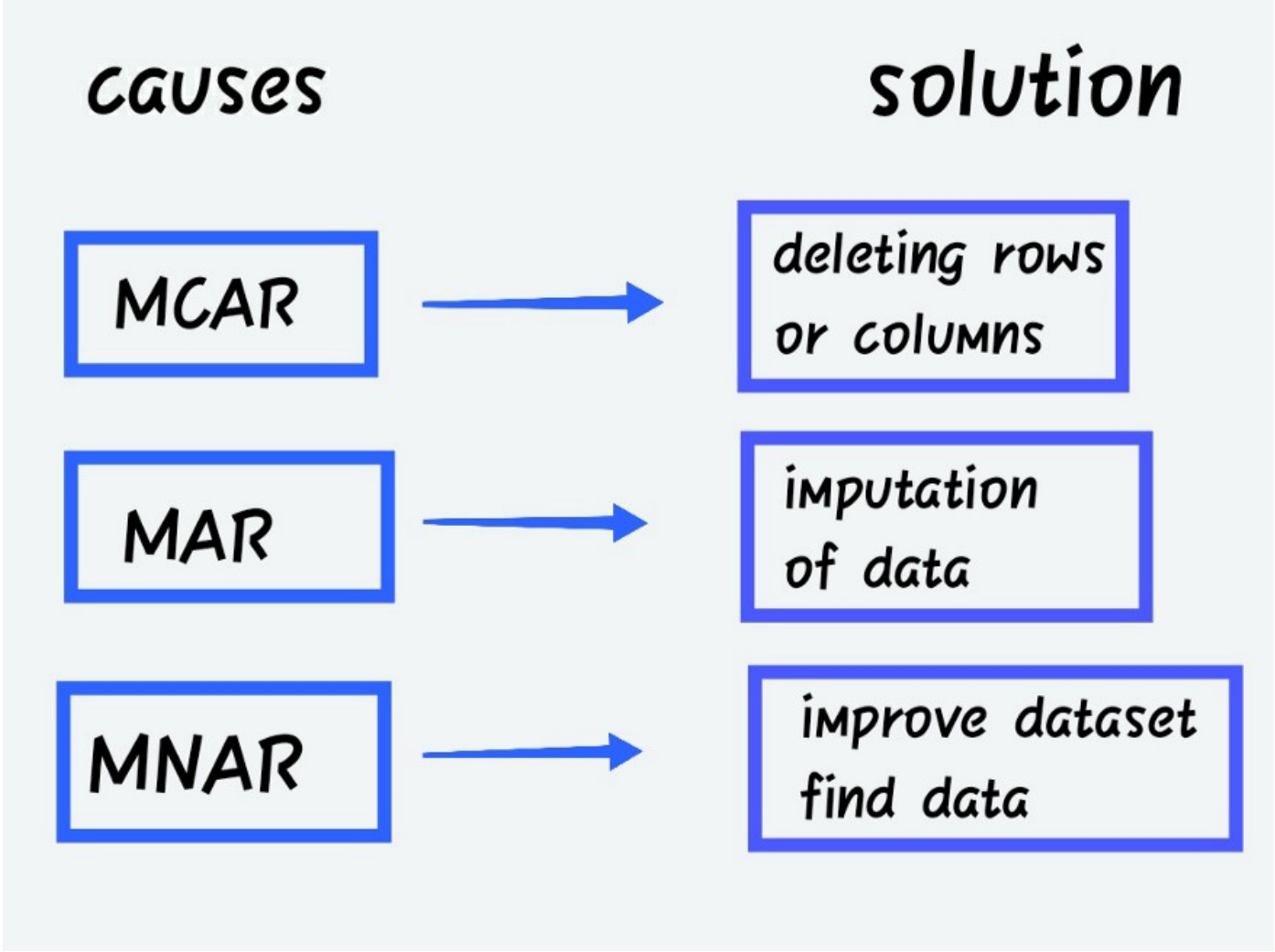
Out[15]:



3. Penanganan atas Data yang Hilang

Ada banyak cara untuk menangani data yang hilang. Sebelum melakukan penanganan perhatikan baik-baik mengapa data hilang?

1. Melakukan penghapusan pada observasi yang datanya hilang
2. Melakukan imputasi (Imputation berarti menggunakan variabel-variabel data yang ada untuk memprediksi data yang hilang.)



3.1 Menghapus Observasi yang Datanya Hilang

Ini bisa dilakukan dengan list-wise deletion ataupun pair-wise deletion.

In [16]:
```
# Kondisi sebelum penghapusan
df_titanic.describe()
```

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

3.1.1 Listwise Deletion

Pada listwise deletion suatu row akan dihapus bila salah satu variabel memiliki nilai yang hilang.

In [17]:
```
df_titanic.dropna(subset=['Age'],how='any',inplace=True)
df_titanic.describe()
```

Out[17]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	714.000000	714.000000	714.000000	714.000000	714.000000	714.000000	714.000000
mean	448.582633	0.406162	2.236695	29.699118	0.512605	0.431373	34.694514
std	259.119524	0.491460	0.838250	14.526497	0.929783	0.853289	52.918930
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	222.250000	0.000000	1.000000	20.125000	0.000000	0.000000	8.050000
50%	445.000000	0.000000	2.000000	28.000000	0.000000	0.000000	15.741700
75%	677.750000	1.000000	3.000000	38.000000	1.000000	1.000000	33.375000
max	891.000000	1.000000	3.000000	80.000000	5.000000	6.000000	512.329200

3.1.2 Pairwise Deletion

Parwise Deletion digunakan saat prosedur statistik yang anda butuhkan tidak memperbolehkan adanya nilai yang hilang. Misalkan anda memiliki 6 variabel, VAR1, VAR2, ..., VAR6. Secara MCAR, pada VAR3 dan VAR5 terdapat data yang hilang. Untuk itu, pada prosedur statistik sum untuk variabel 3 dan variabel 5, data yang hilang tidak diikutkan. </br> `This is a problematic way to solve a problem.` </br> Namun, metode ini logis ketika misalkan anda memerlukan regresi yang tidak membutuhkan VAR 3 dan VAR 5.

3.2 Single Imputation

Single imputation merupakan prosedur dimana suatu nilai dari elemen data yang hilang digantikan tanpa mendefinisikan model eksplisit untuk memperkirakan data yang hilang tersebut.

3.2.1 Mean/Median/Mode Imputation

mengganti data yang hilang dengan menggunakan rata-rata dari data variabel (exclude data yang hilang).

In [18]:
```
# fungsi fillna di pandas dapat digunakan untuk imputasi missing values
# taruh hasilnya di variable baru untuk mempermudah pengamatan
df_titanic['Age_mean_uni'] = df_titanic.Age.fillna(df_titanic.Age.mean())
df_titanic['Age_median_uni'] = df_titanic.Age.fillna(df_titanic.Age.median())
df_titanic['Age_mode_uni'] = df_titanic.Age.fillna(df_titanic.Age.median())
```

In [19]:
```
# mengetahui mean per kelompok sex, kita menggunakan groupby
df_grouped = df_titanic[['Sex', 'Age']].groupby(['Sex']).mean()
df_grouped
# imputasi untuk male group
# filter kondisi
condition = df_titanic['Sex'] == 'male'
# tangkap nilai mean untuk laki2
mean_male = df_grouped.loc['male','Age']
# imputasi
df_titanic.loc[condition, 'Age_mean_multi'] = df_titanic.loc[condition,'Age'].fillna(mean_
```

3.3 Multiple Imputation

Multiple imputation merupakan cara untuk memperkirakan data yang hilang dengan memperhatikan faktor ketidakpastian.

Adanya beberapa metode untuk melakukan multiple imputation antara lain :

- KNN or K-Nearest Neighbor imputation
- Multiple Imputation by Chained Equations (MICE) with random forests

3.3.1 Multivariate Imputation via Chained Equations (MICE)

Misalkan kita memiliki variabel $x_1, x_2, x_3, \ldots x_n$. Pada dataset anda, variabel x_2 dan x_3, dan x_5 mengalami beberapa data yang hilang.

- Pada saat anda ingin memunculkan nilai yang hilang dari x_2 maka variabel yang hilang akan diregres terhadap variabel x_1, x_3, $x_4 - x_5$
- Pada saat anda ingin memunculkan nilai yang hilang dari x_3 maka variabel yang hilang akan diregres terhadap variabel x_1, x_2, $x_4 - x_5$
- dst

Secara default, regresi linear digunakan untuk memprediksi nilai yang hilang bila variabel yang diprediksi adalah variabel kontinu dan regresi logistik bila variabel yang hilang adalah variabel kategorikal.

In [20]:
```
import pandas as pd
import numpy as np
# importing the MICE from fancyimpute library
from fancyimpute import IterativeImputer

df = pd.DataFrame([[np.nan, 2, np.nan, 0],
                   [3, 4, np.nan, 1],
                   [np.nan, np.nan, np.nan, 5],
                   [np.nan, 3, np.nan, 4],
                   [5,      7,  8,     2],
                   [2,      5,  7,     9]],
                  columns = ['var1', 'var2', 'var3', 'var4'])
df
```

Out[20]:

	var1	var2	var3	var4
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4
4	5.0	7.0	8.0	2
5	2.0	5.0	7.0	9

In [21]:
```
# calling the  MICE class
mice_imputer = IterativeImputer()

# imputing the missing value with mice imputer
df = mice_imputer.fit_transform(df)
df_mice = pd.DataFrame(df, columns=['var1', 'var2', 'var3', 'var4'])
df_mice
```

Out[21]:

	var1	var2	var3	var4
0	1.739135	2.000000	7.906731	0.0
1	3.000000	4.000000	7.919353	1.0
2	2.304308	4.235237	7.441668	5.0
3	1.608699	3.000000	7.481066	4.0
4	5.000000	7.000000	8.000000	2.0
5	2.000000	5.000000	7.000000	9.0

Referensi Lain :

- https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/advanced-imputation-techniques?ex=1
- https://towardsdatascience.com/handling-missing-data-like-a-pro-part-3-model-based-multiple-imputation-methods-bdfe85f93087

4. Penanganan Outlier

Ada tiga metode dasar untuk menangani data pencilan :

1. **Menghapus Pencilan**
2. **Mengganti Nilai Pencilan**
3. **Mengestimasi Nilai Pencilan**

Untuk 1 dan 3 bisa dilakukan dengan metode penanganan yang sama dengan penanganan data yang hilang. Untuk metode ke-2 ada beberapa cara yang dapat dilakukan, salah satunya adalah Winsorization.

4.1 Winsorization

merupakan salah satu metode transformasi statistik dengan membatasi nilai-nilai ekstrim data. Disebut Winsorization karena metode ini ditemukan oleh biostatistisi Charles P. Winsor (1895–1951). Strategi umum dari Winsorization adalah dengan menspesifikasi persentil data. Winsorization 90% berarti

- mengganti semua data di bawah 5% persentil menjadi 5%
- mengganti semua data di atas 95% persentil menjadi 95%

In [22]:

```
# Misalkan kita memiliki list data :
dataA=[3, 14, 16, 16, 17, 29, 34, 36, 39, 47, 59, 64, 65, 66, 68, 79, 91, 98]
import numpy as np
q05=np.quantile(dataA, .05)
q95=np.quantile(dataA, .95)
print(q05)
print(q95)
```

```
12.350000000000001
92.04999999999998
```

Maka :

- semua data di bawah 12 akan diganti dengan 12
- semua data di atas 92 akan diganti dengan 92

4.2 Estimasi/Penggantian Outlier Multivariate

Deteksi outlier bedasarkan pengamatan terhadap lebih dari 1 features biasa dikenal dengan multivariate outlier detections.

- k-nearest neighbours
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise-DBSCAN)
- isolation forests

5. Transformasi Variabel

5.1 Encoding

Encoding merupakan metode untuk mengubah data kategorikal ke dalam format bilangan bulat. Ada beberapa macam encoding, antara lain label encoding, binary encoding, hash encoding, target encoding, dll. Untuk melakukan encoding kita bisa menggunakan :

- library LabelEncoder.
- dictionary {}.
- df.Series.map().
- dst.

5.1.1 Label Encoding

Digunakan untuk mengubah sekumpulan label ke dalam format integer sehingga bisa dilakukan komputasi.

In [23]:
```
import pandas as pd

city_name = ['Jakarta', 'Bandung', 'Surabaya', 'Jakarta', 'Bandung', 'Surabaya', 'Pontiana
df = pd.DataFrame(city_name, columns=['Kota'])
df
```

Out[23]:

	Kota
0	Jakarta
1	Bandung
2	Surabaya
3	Jakarta
4	Bandung
5	Surabaya
6	Pontianak
7	Medan
8	Makassar
9	Jayapura

In [24]:
```
# Import label encoder
from sklearn import preprocessing
```

```
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['Kota']= label_encoder.fit_transform(df['Kota'])
df
```

Out[24]:

	Kota
0	1
1	0
2	6
3	1
4	0
5	6
6	5
7	4
8	3
9	2

5.2.2 One Hot Encoding

- Pada one hot encoding, kita mengkonversi satu feature menjadi beberapa fitur yang nilainya 0 atau 1.
- Salah satu penggunaaan hot encoding adalah pada saat menggunakan regresi dengan variabel dummy.

In [25]:

```
import pandas as pd

nama_pegawai = ['Ninuk', 'Marwoto', 'Saefudin', 'Hanafi', 'Sawiyah', 'Cecek', 'Lintang']
kota_asal = ['Jakarta', 'Bandung', 'Surabaya', 'Jakarta', 'Medan', 'Makassar', 'Medan']
df = pd.DataFrame(list(zip(nama_pegawai, kota_asal)), columns=['nama_pegawai', 'kota_asal'
df
```

Out[25]:

	nama_pegawai	kota_asal
0	Ninuk	Jakarta
1	Marwoto	Bandung
2	Saefudin	Surabaya
3	Hanafi	Jakarta
4	Sawiyah	Medan
5	Cecek	Makassar
6	Lintang	Medan

In [26]:

```
df_ohe = pd.get_dummies(df['kota_asal'])
df_ohe
# Join the encoded df
df_ohe2 = df.join(df_ohe)
df_ohe2
```

Out[26]:

	nama_pegawai	kota_asal	Bandung	Jakarta	Makassar	Medan	Surabaya
0	Ninuk	Jakarta	0	1	0	0	0
1	Marwoto	Bandung	1	0	0	0	0
2	Saefudin	Surabaya	0	0	0	0	1
3	Hanafi	Jakarta	0	1	0	0	0
4	Sawiyah	Medan	0	0	0	1	0
5	Cecek	Makassar	0	0	1	0	0
6	Lintang	Medan	0	0	0	1	0

In [27]:

```
from sklearn.preprocessing import OneHotEncoder

y = OneHotEncoder(sparse=False)
y = y.fit_transform(df[['kota_asal']])
df_y = pd.DataFrame(y)
df_y
```

Out[27]:

	0	1	2	3	4
0	0.0	1.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	1.0
3	0.0	1.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0
5	0.0	0.0	1.0	0.0	0.0
6	0.0	0.0	0.0	1.0	0.0

5.2. Binning

Metode transformasi variabel lainnya adalah binning yakni mengkonversi data numeris menjadi kategoris, dengan menggunakan range.

5.2.1 Menggunakan qcut

The pandas documentation describes qcut as a “Quantile-based discretization function.” This basically means that qcut tries to divide up the underlying data into equal sized bins. The function defines the bins using percentiles based on the distribution of the data, not the actual numeric edges of the bins. qcut() Selain fungsi cut(), ada juga fungsi qcut() yang dapat digunakan untuk melakukan binning data. Menurut dokumentasi pandas, qcut digambarkan sebagai Quantile-based discretization function. Singkatnya fungsi qcut() ini akan membagi data ke dalam jumlah yang sama. Karena itu, jarak untuk masing-masing bin boleh jadi berbeda satu sama lain.

Di Python, jika kita akan melakukan binning data menjadi 3 bin menggunakan qcut() dapat ditulis sebagai berikut.

data['Harga_binned'] = pd.qcut(data['Harga'], 3)

In [28]:
```
import pandas as pd
df=pd.read_csv('UScereal.csv')
df.head(4)
```

Out[28]:

	Unnamed: 0	mfr	calories	protein	fat	sodium	fibre	carbo	sugars	shelf	potassium	vit
0	100% Bran	N	212.12121	12.121212	3.030303	393.93939	30.303030	15.15152	18.181818	3	848.48485	en
1	All-Bran	K	212.12121	12.121212	3.030303	787.87879	27.272727	21.21212	15.151515	3	969.69697	en
2	All-Bran with Extra Fiber	K	100.00000	8.000000	0.000000	280.00000	28.000000	16.00000	0.000000	3	660.00000	en
3	Apple Cinnamon Cheerios	G	146.66667	2.666667	2.666667	240.00000	2.000000	14.00000	13.333333	1	93.33333	en

6. Menciptakan Feature dari Data Waktu

- Terdapat masa dimana pada suatu masalah prediktif, variabel waktu memiliki kontribusi tertentu.
- Misalkan, pada data time series kita menemukan adanya siklus pada hari/bulan/jam tertentu. Untuk itu, kita perlu memunculkan potongan informasi waktu tersebut.

6.1 Mengekstrak Potongan Tanggal atau Waktu dari Timestamp

In [29]:
```
import pandas as pd
data_waktu=pd.read_csv('TPIA.csv')
data_wkt=data_waktu[['date', 'previous', 'open_price', 'close']].copy()
data_wkt['date'] = pd.to_datetime(data_wkt['date'])
data_wkt.head()
```

Out[29]:

	date	previous	open_price	close
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0

In [30]:
```
# Mengekstrak tanggal saja
data_wkt['tanggal'] = data_wkt['date'].dt.date
data_wkt.head()
```

Out[30]:

	date	previous	open_price	close	tanggal
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	2019-07-29
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	2019-08-19
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	2019-09-19
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	2019-09-20

	date	previous	open_price	close	tanggal
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	2019-10-21

In [31]:
```
# Mengekstrak jam saja
data_wkt['jam'] = data_wkt['date'].dt.time
data_wkt.head()
```

Out[31]:

	date	previous	open_price	close	tanggal	jam
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	2019-07-29	09:00:28
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	2019-08-19	10:00:00
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	2019-09-19	11:01:00
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	2019-09-20	10:44:00
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	2019-10-21	11:00:00

In [32]:
```
# Mengekstrak tahun
data_wkt['tahun'] = data_wkt['date'].dt.year
data_wkt.head()
```

Out[32]:

	date	previous	open_price	close	tanggal	jam	tahun
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	2019-07-29	09:00:28	2019
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	2019-08-19	10:00:00	2019
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	2019-09-19	11:01:00	2019
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	2019-09-20	10:44:00	2019
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	2019-10-21	11:00:00	2019

In [33]:
```
# Mengekstrak kuartal
data_wkt['kuartal'] = data_wkt['date'].dt.quarter
data_wkt.head()
```

Out[33]:

	date	previous	open_price	close	tanggal	jam	tahun	kuartal
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	2019-07-29	09:00:28	2019	3
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	2019-08-19	10:00:00	2019	3
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	2019-09-19	11:01:00	2019	3
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	2019-09-20	10:44:00	2019	3
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	2019-10-21	11:00:00	2019	4

In [34]:
```
# Mengekstrak tahun
data_wkt['bulan'] = data_wkt['date'].dt.month
data_wkt.head()
```

Out[34]:

	date	previous	open_price	close	tanggal	jam	tahun	kuartal	bulan
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	2019-07-29	09:00:28	2019	3	7

	date	previous	open_price	close	tanggal	jam	tahun	kuartal	bulan
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	2019-08-19	10:00:00	2019	3	8
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	2019-09-19	11:01:00	2019	3	9
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	2019-09-20	10:44:00	2019	3	9
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	2019-10-21	11:00:00	2019	4	10

In [35]:
```
#Mengekstrak Hari (Langsung di enkoding)
data_wkt['hari'] = data_wkt['date'].dt.dayofweek
data_wkt.head()
```

Out[35]:

	date	previous	open_price	close	tanggal	jam	tahun	kuartal	bulan	hari
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	2019-07-29	09:00:28	2019	3	7	0
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	2019-08-19	10:00:00	2019	3	8	0
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	2019-09-19	11:01:00	2019	3	9	3
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	2019-09-20	10:44:00	2019	3	9	4
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	2019-10-21	11:00:00	2019	4	10	0

In [36]:
```
# Mengoperasikan tanggal
from datetime import timedelta
data_wkt['tgl_migdep'] = data_wkt['date'] + timedelta(days=7)
data_wkt.head()
```

Out[36]:

	date	previous	open_price	close	tanggal	jam	tahun	kuartal	bulan	hari	tgl_migdep
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	2019-07-29	09:00:28	2019	3	7	0	2019-08-05 09:00:28
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	2019-08-19	10:00:00	2019	3	8	0	2019-08-26 10:00:00
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	2019-09-19	11:01:00	2019	3	9	3	2019-09-26 11:01:00
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	2019-09-20	10:44:00	2019	3	9	4	2019-09-27 10:44:00
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	2019-10-21	11:00:00	2019	4	10	0	2019-10-28 11:00:00

In [37]:
```
# Menghitung jarak
data_wkt['delta_hari'] = (data_wkt['tgl_migdep'] - data_wkt['date']).dt.days
data_wkt.head()
```

Out[37]:

	date	previous	open_price	close	tanggal	jam	tahun	kuartal	bulan	hari	tgl_migdep	delta_hari
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	2019-07-29	09:00:28	2019	3	7	0	2019-08-05 09:00:28	7
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	2019-08-19	10:00:00	2019	3	8	0	2019-08-26 10:00:00	7

	date	previous	open_price	close	tanggal	jam	tahun	kuartal	bulan	hari	tgl_migdep	delta_hari
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	2019-09-19	11:01:00	2019	3	9	3	2019-09-26 11:01:00	7
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	2019-09-20	10:44:00	2019	3	9	4	2019-09-27 10:44:00	7
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	2019-10-21	11:00:00	2019	4	10	0	2019-10-28 11:00:00	7

7. Menciptakan Feature dari Komputasi Matematis

In [38]:
```
data_cmp=data_waktu[['date', 'previous', 'open_price', 'close']].copy()
data_cmp.head()
```

Out[38]:

	date	previous	open_price	close
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0

7.1 Operasi antar kolom

In [39]:
```
data_cmp['Selisih']=data_cmp['close']-data_cmp['open_price']
data_cmp.head()
```

Out[39]:

	date	previous	open_price	close	Selisih
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	50.0
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	175.0
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	-75.0
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	-25.0
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	0.0

7.2 Operasi dengan skalar

In [40]:
```
data_cmp['add_close']=data_cmp['close']+20000
data_cmp.head()
```

Out[40]:

	date	previous	open_price	close	Selisih	add_close
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	50.0	26075.0
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	175.0	27925.0

	date	previous	open_price	close	Selisih	add_close
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	-75.0	28425.0
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	-25.0	28350.0
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	0.0	29450.0

7.3 Memberikan kategori

In [41]:
```
def ur (selisih):
    if selisih>0 :
        return 'Untung'
    elif selisih==0:
        return 'Sama'
    else:
        return 'Rugi'
data_cmp['kategori']=data_cmp['Selisih'].apply(ur)
data_cmp.head()
```

Out[41]:

	date	previous	open_price	close	Selisih	add_close	kategori
0	2019-07-29 09:00:28	6100.0	6025.0	6075.0	50.0	26075.0	Untung
1	2019-08-19 10:00:00	7750.0	7750.0	7925.0	175.0	27925.0	Untung
2	2019-09-19 11:01:00	8475.0	8500.0	8425.0	-75.0	28425.0	Rugi
3	2019-09-20 10:44:00	8425.0	8375.0	8350.0	-25.0	28350.0	Rugi
4	2019-10-21 11:00:00	9450.0	9450.0	9450.0	0.0	29450.0	Sama