**Data Wrangling Python**

**Membaca file dengan menggunakan pandas**

Sebagai salah satu library untuk melakukan proses awal dari analisis data, pandas juga memiliki kemampuan untuk membaca berbagai macam jenis file. Format yang bisa dibaca oleh pandas ada berbagai macam, antara lain .txt, .csv, .tsv, dan lainnya. Pandas tidak hanya bisa membaca file saja, namun juga bisa merubah data dari file menjadi bentuk dataframe yang akhirnya nanti bisa diakses, diagregasi dan diolah. Coba praktikan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data)

Hasil pada panel console akan keluar seperti berikut :

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)

0 1 Male 19 15 39

1 2 Male 21 15 81

2 3 Female 20 16 6

3 4 Female 23 16 77

4 5 Female 31 17 40

5 6 Female 22 17 76

6 7 Female 35 18 6

7 8 Female 23 18 94

8 9 Male 64 19 3

9 10 Female 30 19 72

10 11 Male 67 19 14

11 12 Female 35 19 99

12 13 Female 58 20 15

13 14 Female 24 20 77

14 15 Male 37 20 13

15 16 Male 22 20 79

16 17 Female 35 21 35

17 18 Male 20 21 66

18 19 Male 52 23 29

19 20 Female 35 23 98

20 21 Male 35 24 35

21 22 Male 25 24 73

22 23 Female 46 25 5

23 24 Male 31 25 73

24 25 Female 54 28 14

25 26 Male 29 28 82

26 27 Female 45 28 32

27 28 Male 35 28 61

28 29 Female 40 29 31

29 30 Female 23 29 87

.. ... ... ... ...

## Membaca file dengan menggunakan head()

Pada suatu kasus, data yang kita baca cukup banyak atau loading yang lama. Untuk memastikan data kita terbaca dengan baik dan bisa menampilkan data sebagian untuk ditampilkan secara benar, kita bisa memakai fungsi head(). Bisa dituliskan kode di bawah ini untuk prakteknya:

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data.head())

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)

0 1 Male 19 15 39

1 2 Male 21 15 81

2 3 Female 20 16 6

3 4 Female 23 16 77

4 5 Female 31 17 40

## Melakukan akses data kolom

Dalam suatu analisis data ada kalanya kita hanya butuh melakukan akses beberapa data saja dan tidak perlu harus menampilkan semua data. Pada pandas kita bisa melakukan akses dalam berbagai kebutuhan. Mulai dari hanya akses kolom tertentu ataupun baris tertentu. Pada sesi kali ini kita akan mencoba untuk melakukan akses beberapa kolom tertentu pada suatu dataset.

Pertama yang harus dilakukan untuk melakukan akses kolom adalah mengetahui nama-nama kolom yang ada. Coba ketikkan kode di bawah ini untuk melihat nama kolom yang ada.

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data.columns)

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',

'Spending Score (1-100)'],

dtype='object')

**Note :** Pada dataset ini ada 5 kolom termasuk class, dimana 4 kolom  merupakan data numerik dan 1 kolom merupakan data string. Pada praktek selanjutnya kita akan mencoba mengakses data age. Untuk melakukannya coba tuliskan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data['Age'])

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

0 19

1 21

2 20

3 23

4 31

5 22

6 35

7 23

8 64

9 30

10 67

11 35

12 58

13 24

14 37

15 22

16 35

17 20

18 52

19 35

20 35

21 25

22 46

23 31

24 54

25 29

26 45

27 35

28 40

29 23

..

170 40

171 28

172 36

173 36

174 52

175 30

176 58

177 27

178 59

179 35

180 37

181 32

182 46

183 29

184 41

185 30

186 54

187 28

188 41

189 36

190 34

191 32

192 33

193 38

194 47

195 35

196 45

197 32

198 32

199 30

Name: Age, Length: 200, dtype: int64

## Melakukan akses data melalui baris

Selain melakukan akses data melalui kolom, dengan menggunakan pandas juga bisa melakukan akses dengan menggunakan baris. Berbeda dengan akses melalui kolom, fungsi untuk menampilkan data dari suatu baris adalah fungsi .iloc[i] dimana [i] menunjukan urutan baris yang akan ditampilkan yang dimana indexnya diawali dari 0. Coba ketikan code di bawah ini untuk mempermudah :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data.iloc[5])

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

CustomerID 6

Genre Female

Age 22

Annual Income (k$) 17

Spending Score (1-100) 76

Name: 5, dtype: object

## Menampilkan suatu data dari baris dan kolom tertentu

Tidak hanya dengan menentukan dari kolom dan baris, dengan menggunakan pandas kita juga bisa memanggil suatu data dari suatu baris dan kolom tertentu dalam satu waktu. Perhatikan dan coba kode di bawah ini:

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data['Age'].iloc[1])  
  
print("Cuplikan Dataset:")  
  
print(csv\_data.head())

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

21

Cuplikan Dataset:

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)

0 1 Male 19 15 39

1 2 Male 21 15 81

2 3 Female 20 16 6

3 4 Female 23 16 77

4 5 Female 31 17 40

## Menampilkan data dalam range tertentu

Setelah menampilkan suatu kelompok data, bagaimana jika ingin menampilkan data dari baris ke 5 sampai ke 20 dari suatu dataset? Untuk mengantisipasi hal tersebut, pandas juga bisa menampilkan data dalam range tertentu, baik range untuk baris saja, kolom saja, dan range untuk baris dan kolom.

Akses range pada suatu kolom dan baris tertentu, untuk mencobanya silahkan ketikkan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print("Menampilkan data ke 5 sampai kurang dari 10 :")  
  
print(csv\_data['Age'].iloc[5:10])

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

Menampilkan data ke 5 sampai kurang dari 10 :

5 22

6 35

7 23

8 64

9 30

Name: Age, dtype: int64

Menampilkan suatu range data tertentu pada suatu baris saja. Cobalah ketikan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print("Menampilkan data ke 5 sampai kurang dari 10 dalam satu baris:")  
  
print(csv\_data.iloc[5:10])

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

Menampilkan data ke 5 sampai kurang dari 10 dalam satu baris:

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)

5 6 Female 22 17 76

6 7 Female 35 18 6

7 8 Female 23 18 94

8 9 Male 64 19 3

9 10 Female 30 19 72

## Menampilkan informasi statistik dengan Numpy

Mengetahui informasi statistik pada suatu data sangat penting. Mulai dari distribusi data, nilai max atau min, hingga standar deviasi dari suatu dataset. Jika datanya berjumlah dibawah 10 mungkin masih dikerjakan secara manual. Namun, bayangkan jika datanya sudah mencapai ratusan bahkan ribuan. Tidak mungkin pastinya untuk dilakukan secara manual. Maka dari itu pentingnya fungsi describe() pada pandas. Fungsi describe() ini memungkinkan untuk mengetahui informasi statistik dari suatu dataset secara cepat. Coba untuk ketikkan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data.describe(include='all'))

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

CustomerID Genre Age Annual Income (k$) \

count 200.000000 200 200.000000 200.000000

unique NaN 2 NaN NaN

top NaN Female NaN NaN

freq NaN 112 NaN NaN

mean 100.500000 NaN 38.850000 60.560000

std 57.879185 NaN 13.969007 26.264721

min 1.000000 NaN 18.000000 15.000000

25% 50.750000 NaN 28.750000 41.500000

50% 100.500000 NaN 36.000000 61.500000

75% 150.250000 NaN 49.000000 78.000000

max 200.000000 NaN 70.000000 137.000000

Spending Score (1-100)

count 200.000000

unique NaN

top NaN

freq NaN

mean 50.200000

std 25.823522

min 1.000000

25% 34.750000

50% 50.000000

75% 73.000000

max 99.000000

**Note :** Banyak nilai NaN yang tampil. Hal itu karena pada dataset ada format data string yang akhirnya memunculkan format NaN.

Untuk meminimalisir hal tersebut dan memfilter hanya data numerical saja, digunakan  exclude=[‘O’], dimana fungsi itu akan mengabaikan data yang non-numerical untuk diproses. Coba implementasikan code di bawah ini:

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data.describe(exclude=['O']))

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

CustomerID Age Annual Income (k$) Spending Score (1-100)

count 200.000000 200.000000 200.000000 200.000000

mean 100.500000 38.850000 60.560000 50.200000

std 57.879185 13.969007 26.264721 25.823522

min 1.000000 18.000000 15.000000 1.000000

25% 50.750000 28.750000 41.500000 34.750000

50% 100.500000 36.000000 61.500000 50.000000

75% 150.250000 49.000000 78.000000 73.000000

max 200.000000 70.000000 137.000000 99.000000

## Handling Missing Value

Pada suatu dataset, ada kalanya data yang kita akan kita kelola tidak lengkap. Hal ini tentunya akan menyulitkan atau membuat hasil analisa tidak akurat. Penanggulangan akan data yang hilang ini biasa disebut Handling Missing Value. Penanganan dari nilai yang kosong ini banyak caranya. Sebagai seorang data science yang berhubungan dengan data yang real, solusi pertama yang benar-benar kita anjurkan untuk kasus seperti ini adalah melakukan trace kembali ke sumber data atau memerika ulang record. Terutama jika data itu berasal dari human record. Sangat disarankan untuk menelusuri kembali agar tidak terjadi kesalahan ketika sudah mencapai titik analisa. Selain solusi untuk melakukan penelusuran kembali ke sumberdata, pada ilmu data science juga ada beberapa metode yang bisa dijadikan solusi untuk menangani kasus ini.

## Melakukan pengecekan untuk nilai NULL yang ada

Dengan menggunakan fungsi pandas, kita tidak perlu melihat satu persatu baris data untuk mengetahui apakah ada nilai kosong atau NULL/NAN pada suatu dataset. Bayangkan jika kita memilki 1000 baris data. Apakah kita harus melihat semua baris data tersebut? Tentu saja tidak. Maka dari itu di pandas disediakan fungsi untuk mengecek apakah ada data yang kosong. Coba praktikkan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
  
print(csv\_data.isnull().values.any())

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

False

**Note :** data yang digunakan merupakan data yang lengkap, maka dari itu output yang dihasilkan False. Coba Sekarang ganti dengan dataset yang memang terdapat data yang kosong. Coba ketikkan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data\_missingvalue.csv")  
  
print(csv\_data.isnull().values.any())

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

True

## Mengisi dengan Mean

Salah satu metode yang bisa dikatakan sebagai solusi yang umum pada kasus general data science adalah mengisi data kosong dengan menggunakan mean dari masing-masing kolom. Pertama kita harus menentukan mean dari masing-masing kolom. Pada pandas terdapat fungsi mean() untuk menentukan nilai mean dari masing-masing kolom. Mean sendiri digunakan untuk data yang memiliki sedikit sifat outlier/noisy/anomali dalam sebaran datanya maupun isinya.  Coba ketikkan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data\_missingvalue.csv")  
  
print(csv\_data.mean())

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

CustomerID 100.500000

Age 38.939698

Annual Income (k$) 61.005051

Spending Score (1-100) 50.489899

dtype: float64

Fungsi mean sendiri berfungsi untuk menampilkan  nilai mean (rata-rata) dari setiap kolom. Nilai inilah nanti yang akan mengisi nilai kosong dari dataset yang mengalami kasus missing value. Untuk mengisi nilai yang kosong menggunakan fungsi fillna(), coba ketikkan kode di bawah ini :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data\_missingvalue.csv")  
print(csv\_data.mean())  
print("Dataset yang masih terdapat nilai kosong ! :")  
print(csv\_data.head(10))  
  
csv\_data=csv\_data.fillna(csv\_data.mean())  
print("Dataset yang sudah diproses Handling Missing Values dengan Mean :")  
print(csv\_data.head(10))

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

CustomerID 100.500000

Age 38.939698

Annual Income (k$) 61.005051

Spending Score (1-100) 50.489899

dtype: float64

Dataset yang masih terdapat nilai kosong ! :

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)

0 1 Male 19.0 15.0 39.0

1 2 Male NaN 15.0 81.0

2 3 Female 20.0 NaN 6.0

3 4 Female 23.0 16.0 77.0

4 5 Female 31.0 17.0 NaN

5 6 Female 22.0 NaN 76.0

6 7 Female 35.0 18.0 6.0

7 8 Female 23.0 18.0 94.0

8 9 Male 64.0 19.0 NaN

9 10 Female 30.0 19.0 72.0

Dataset yang sudah diproses Handling Missing Values dengan Mean :

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)

0 1 Male 19.000000 15.000000 39.000000

1 2 Male 38.939698 15.000000 81.000000

2 3 Female 20.000000 61.005051 6.000000

3 4 Female 23.000000 16.000000 77.000000

4 5 Female 31.000000 17.000000 50.489899

5 6 Female 22.000000 61.005051 76.000000

6 7 Female 35.000000 18.000000 6.000000

7 8 Female 23.000000 18.000000 94.000000

8 9 Male 64.000000 19.000000 50.489899

9 10 Female 30.000000 19.000000 72.000000

## Mengisi dengan Median

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

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data\_missingvalue.csv")  
  
print(csv\_data.median())

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

CustomerID 100.5

Age 36.0

Annual Income (k$) 62.0

Spending Score (1-100) 50.0

dtype: float64

Sama dengan sesi sebelumnya dengan mean(), gunakan kode di bawah ini untuk mengisi nilai yang kosong menggunakan fungsi fillna() :

import pandas as pd  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data\_missingvalue.csv")  
print("Dataset yang masih terdapat nilai kosong ! :")  
print(csv\_data.head(10))  
  
csv\_data=csv\_data.fillna(csv\_data.median())  
print("Dataset yang sudah diproses Handling Missing Values dengan Median :")  
print(csv\_data.head(10))

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

Dataset yang masih terdapat nilai kosong ! :

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)

0 1 Male 19.0 15.0 39.0

1 2 Male NaN 15.0 81.0

2 3 Female 20.0 NaN 6.0

3 4 Female 23.0 16.0 77.0

4 5 Female 31.0 17.0 NaN

5 6 Female 22.0 NaN 76.0

6 7 Female 35.0 18.0 6.0

7 8 Female 23.0 18.0 94.0

8 9 Male 64.0 19.0 NaN

9 10 Female 30.0 19.0 72.0

Dataset yang sudah diproses Handling Missing Values dengan Median :

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)

0 1 Male 19.0 15.0 39.0

1 2 Male 36.0 15.0 81.0

2 3 Female 20.0 62.0 6.0

3 4 Female 23.0 16.0 77.0

4 5 Female 31.0 17.0 50.0

5 6 Female 22.0 62.0 76.0

6 7 Female 35.0 18.0 6.0

7 8 Female 23.0 18.0 94.0

8 9 Male 64.0 19.0 50.0

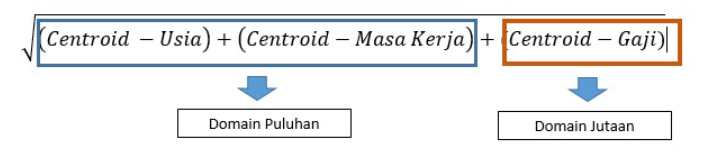
9 10 Female 30.0 19.0 72.0

Normalisasi Data

Terkadang pada beberapa kasus, 1 kolom dengan kolom yang lain memiliki skala yang berbeda. Seperti cuplikan gambar di bawah ini :

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Usia** | **Masa Kerja (Tahun)** | **Gaji** |
| 1 | 50 | 30 | 10000000 |
| 2 | 30 | 10 | 5000000 |

Antara Usia dan Masa Kerja masih memiliki range yang sama dalam skala puluhan. Namun, jika kolom Usia dan Masa Kerja dibandingkan dengan Gaji memiliki range nilai yang berbeda, dimana Usia dan Masa Kerja memiliki range puluhan dan Gaji mempunyai range nilai jutaan. Memang terlihat sederhana, namun hal ini bisa menjadi masalah besar dalam contoh kasus klasterisasi atau klasifikasi. Masuk pada kasus K-means yang sudah pernah dibahas sebelumnya. K-means merupakan algoritma klasterisasi (clustering) yang menggunakan perhitungan jarak dalam prosesnya. Sekarang coba bayangkan :



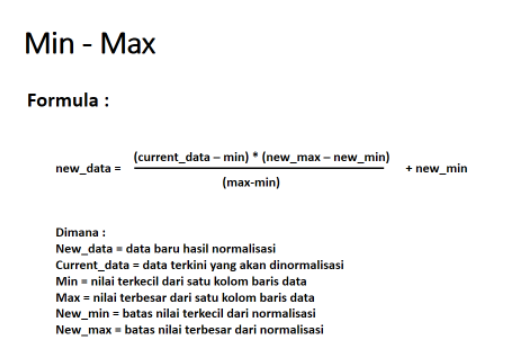
Jika tidak ada normalisasi, maka jelas perhitungan kmeans diatas akan tergantung pada Gaji. Kenapa? Karena gaji berdomain jutaan dan 2 kolom lainnya hanya berdomain puluhan. Berapapun usia dan masa kerja seseorang tidak akan berpengaruh terhadap penilaian suatu perusahaan. Perbedaan skala pada setiap kolom ini merupakan hal yang sangat wajar dan sering terjadi dan inilah pentingnya normalisasi. Normalisasi sangat penting, terutama untuk yang menggunakan perhitungan jarak dengan menggunakan metode apapun.

## Metode Normalisasi

Ada berbagai macam metode normalisasi, seperti MinMax, Zscore, Decimal Scaling, Sigmoid, dan Softmax. Pemakaiannya tergantung pada kebutuhan dari dataset dan jenis analisa yang dilakukan.

### **MinMax**

Metode Min-Max merupakan metode yang cukup bisa dibayangkan karena termasuk metode normalisasi yang bersifat linier dengan data aslinya. Namun, metode ini bisa menyebabkan out of bound pada beberapa kasus.

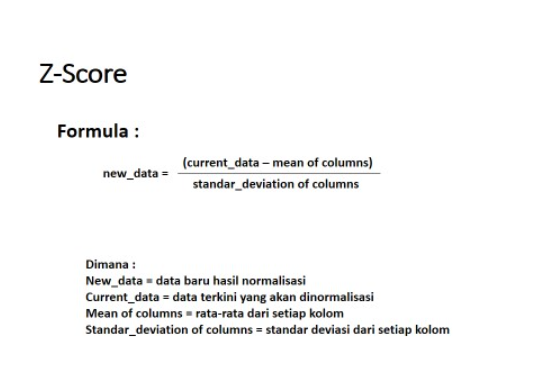


Kenapa bisa terjadi out of bound? Out of Bound terjadi apabila ada data baru masuk, dimana data tersebut melebihi nilai maksimal atau nilai minimal dari data yang sudah ada. Secara otomatis, perhitungan yang berlaku pada data yang sudah diperoleh tadi harus diulangi lagi semuanya dengan data baru yang masuk atau data baru yang mempunyai nilai maksimal/minimum yang melebihi tadi tidak bisa diproses. Karena kekurangan inilah MinMax tidak cocok untuk analisa real time / evolving system. Dimungkinkan dalam kasus-kasus terjadi kasus out of bound pada MinMax.

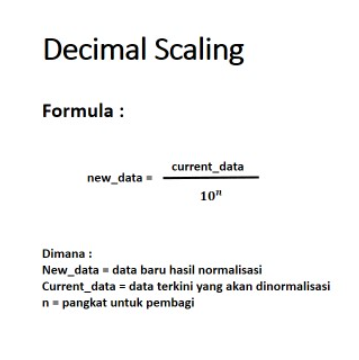
MinMax sangat dianjurkan untuk kasus-kasus berbasis time frame analisis dan forecasting. Perhitungan dari metode ini cukup mengurangi data yang asli dengan nilai minimal dari fitur tersebut, kemudian hasil tersebut dikalikan dari hasil pengurangan nilai maximal yang baru dengan nilai minimal yang baru dan kemudian dibagi dengan nilai max dan min data di setiap fitur terakhir ditambah dengan nilai min yang baru.

### **Z-Score**

Zscore adalah metode yang sering digunakan dalam berbagai penelitian berbasis data mining atau data science. Z-score merupakan metode normalisasi yang berdasarkan mean (nilai rata-rata) dan standard deviation (deviasi standar) dari data. Kenapa Z-Score sangat populer? Selain tidak banyak variabel yang diset dalam perhitungannya. Z-Score sangat dinamis dalam melakukan perhitungan normalisasi. Kelemahan dari Z-Score adalah prosesnya akan terulang lagi jika ada data baru yang masuk. Selain itu elemen yang dibutuhkan untuk perhitungan Z-Score juga membutuhkan proses yang cukup lama baik standar deviation ataupun rata-rata dari setiap kolom.

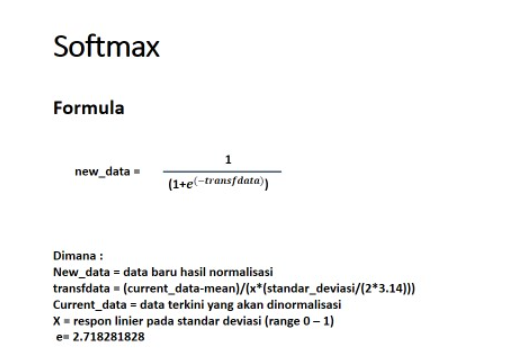


### **Decimal Scaling**



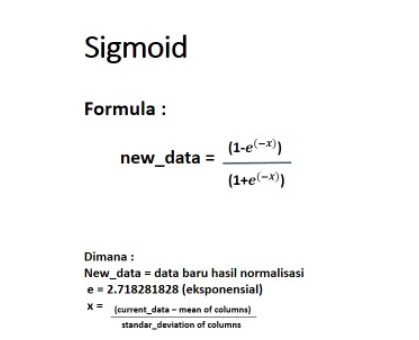
### **Softmax**

Softmax merupakan metode normalisasi pengembangan transformasi secara linier. Output range-nya adalah 0-1. Metode ini sangat berguna pada saat data yang ada melibatkan data outlier.



### **Sigmoid**

Sigmoidal merupakan metode normalization melakukan normalisasi data secara nonlinier ke dalam range -1 s/d 1 dengan menggunakan fungsi sigmoid. Metode ini sangat berguna pada saat data yang ada melibatkan data outlier. Data outlier adalah data yang keluar jauh dari jangkauan data lainnya



## Praktek Normalisasi menggunakan Scikit Learn pada Python

Scikit Learn merupakan library pada python yang digunakan untuk machine learning dan data science. Salah satu library yang selalu menjadi favorit dan komunitasnya sangat kuat. Scikit-learn sendiri tidak hanya untuk analytics saja, namun juga untuk pre-processing, feature selection, dan proses analysis lainnya. Melanjutkan dari sesi normalisasi data, mari kita praktekan kode di bawah ini :

import pandas as pd  
import numpy as np  
from sklearn import preprocessing  
  
csv\_data = pd.read\_csv("https://storage.googleapis.com/dqlab-dataset/shopping\_data.csv")  
array = csv\_data.values

X merupakan matriks yang berisi fitur dataset yang akan digunakan dalam machine learning, baik untuk regresi, klasifikasi, pengklusteran, atau normalisasi

Pada kasus kita, X berisi fitur-fitur yang digunakan untuk dinormalisasi dengan teknik min-max scaler

Ketik lanjutan dari kode di atas:

X = array[:,2:5] #memisahkan fitur dari dataset.   
Y = array[:,0:1]  #memisahkan class dari dataset  
  
dataset=pd.DataFrame({'Customer ID':array[:,0],'Gender':array[:,1],'Age':array[:,2],'Income':array[:,3],'Spending Score':array[:,4]})  
print("dataset sebelum dinormalisasi :")  
print(dataset.head(10))  
  
min\_max\_scaler = preprocessing.MinMaxScaler(feature\_range=(0,1)) #inisialisasi normalisasi MinMax  
data = min\_max\_scaler.fit\_transform(X) #transformasi MinMax untuk fitur  
dataset = pd.DataFrame({'Age':data[:,0],'Income':data[:,1],'Spending Score':data[:,2],'Customer ID':array[:,0],'Gender':array[:,1]})  
  
print("dataset setelah dinormalisasi :")  
print(dataset.head(10))

Klik Tombol ****

Hasil pada panel console akan keluar seperti berikut :

dataset sebelum dinormalisasi :

Age Customer ID Gender Income Spending Score

0 19 1 Male 15 39

1 21 2 Male 15 81

2 20 3 Female 16 6

3 23 4 Female 16 77

4 31 5 Female 17 40

5 22 6 Female 17 76

6 35 7 Female 18 6

7 23 8 Female 18 94

8 64 9 Male 19 3

9 30 10 Female 19 72

dataset setelah dinormalisasi :

Age Customer ID Gender Income Spending Score

0 0.019231 1 Male 0.000000 0.387755

1 0.057692 2 Male 0.000000 0.816327

2 0.038462 3 Female 0.008197 0.051020

3 0.096154 4 Female 0.008197 0.775510

4 0.250000 5 Female 0.016393 0.397959

5 0.076923 6 Female 0.016393 0.765306

6 0.326923 7 Female 0.024590 0.051020

7 0.096154 8 Female 0.024590 0.948980

8 0.884615 9 Male 0.032787 0.020408

9 0.230769 10 Female 0.032787 0.724490