

**MODUL PRAKTIKUM DATA SCIENCE :  
CLUSTERING, CLASSIFICATION & REGRESSION**



**PROGRAM STUDI TEKNIK ELEKTRO  
UNIVERSITAS PELITA HARAPAN**

Modul Praktikum Data Science :  
Clustering, Classification & Regression

Program Studi Teknik Elektro  
Universitas Pelita Harapan  
<http://ee.fast.uph.edu>

Tangerang, Mei 2020

Disusun oleh  
Winly Williamdy

## Daftar Isi

	Halaman
1. Pendahuluan : Data Science, Jupyter Notebook & GitHub	1
2. Python for Data Analytics : Data Types, Operations, Logical Operations & Function	6
3. Data Analysis : Clustering & Principal Component Analysis	16
4. Data Analysis : Linear Regression	27
5. Data Analysis : Decision Tree & Random Forest	35
6. Model Deployment : Flask	45
7. Cheat sheet : Numpy	57
8. Cheat sheet : Pandas	58
9. Cheat sheet : Scikit-Learn	59
10. Referensi	60

## **Pendahuluan : Data Science, Jupyter Notebook & GitHub**

### **1. Tujuan**

1. Mengetahui data science dan fungsinya pada berbagai aspek kehidupan.
2. Dapat menginstal Jupyter Notebook dan membuat virtual environment sendiri menggunakan Anaconda Prompt.
3. Dapat melakukan pemrograman pada aplikasi Jupyter Notebook menggunakan bahasa pemrograman Python 3.
4. Dapat membuat repository pada GitHub.

### **2. Penjelasan Singkat**

Data science adalah ilmu pengolahan data multi-disiplin yang menggabungkan beberapa ilmu lain yaitu matematika dan statistika, ilmu komputer, dan wilayah ilmu pengetahuan lain seperti bisnis atau teknik. Data science semakin berkembang di era Industri 4.0 karena dengan semakin bertambahnya jumlah data yang terekam maka diperlukan kemampuan untuk mengolah data tersebut sehingga bermanfaat untuk manusia.

Data Science memiliki tahap – tahap dalam pengerjaannya, yaitu:

1. Data Scraping  
Pada tahap ini, data – data yang telah direkam (baik dari sekumpulan individu atau sensor) akan ‘dibersihkan’ sehingga dataset terbebas dari bias dan outliers.
2. Data Storage  
Tahap ini adalah tahap penyimpanan data yang telah dibersihkan ke dalam suatu database agar bisa dipanggil ketika diinginkan.
3. Exploratory Data Analysis (EDA)  
Tahap EDA adalah tahap untuk menghasilkan hipotesis dan intuisi terhadap dataset yang dikerjakan.
4. Prediction  
Tahap ini adalah tahap di mana machine learning bekerja. Tahap prediction bisa menggunakan beberapa tools seperti clustering, classification, regression, dll untuk membuat suatu prediksi yang berdasar dari dataset yang diberikan.
5. Communication  
Tahap communication adalah tahap visualisasi hasil kerja dengan menampilkan berbagai grafik yang mudah dipahami dan menyampaikan ringkasan dan kesimpulan dari data yang telah diolah. Tahap ini bisa juga dilakukan dengan mengintegrasikan algoritma machine learning dengan aplikasi lain.

Dalam melakukan tahap – tahap di atas, seorang data scientist biasanya menggunakan beberapa tools untuk tahap tertentu. Namun dalam modul ini, yang akan digunakan adalah Jupyter Notebook dengan bahasa pemrograman Python 3 dan GitHub. Dalam mengolah database, biasanya seorang data scientist perlu mempelajari SQL atau Spark. Selain itu, menggunakan machine learning tidak terbatas dengan bahasa

pemrograman Python 3. Bahasa pemrograman lain yang memiliki machine learning adalah R dan Matlab.

Jupyter Notebook adalah aplikasi open-source yang bisa digunakan untuk melakukan pemrograman berbagai bahasa pemrograman, salah satunya adalah Python 3. Jupyter Notebook dapat diinstal melalui Anaconda yang juga menyediakan aplikasi lain untuk mempelajari data science. Dalam menjalankan program, kita bisa membuat suatu virtual environment sendiri. Virtual environment yang dibuat sendiri memungkinkan kita untuk menambahkan library tertentu yang kita inginkan dan melepasnya kapanpun kita mau sehingga performa compiler bisa lebih efisien karena tidak perlu menggunakan library yang tidak dibutuhkan.

GitHub adalah suatu version control systems, suatu software yang digunakan oleh suatu tim developer untuk mengerjakan proyek dan memudahkan mereka ketika code yang dikerjakan akan diubah seiring waktu. Dengan menggunakan GitHub, suatu tim dapat membuat suatu repository bersama sehingga code dapat dikerjakan bersama dan perubahannya dapat dimonitor menggunakan changelogs. Selain itu, kita dapat melakukan 'forking' dari repository orang lain. Forking adalah membuat suatu proyek baru berdasarkan repository orang lain.

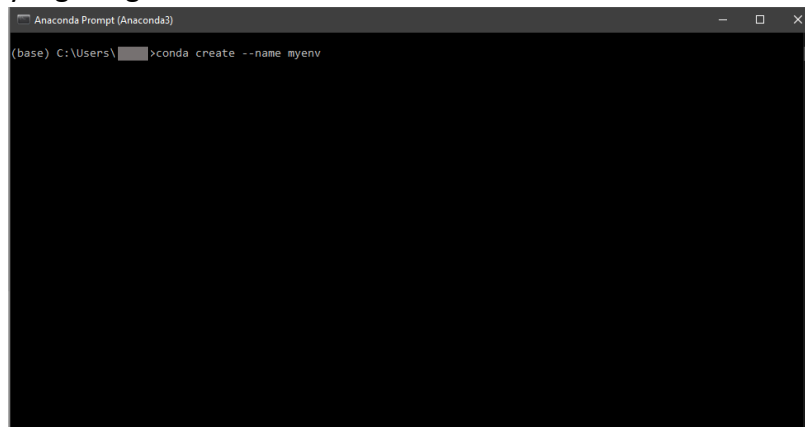
### 3. Langkah Kerja

#### A. Instalasi Jupyter Notebook

1. Masuk ke website <https://www.anaconda.com/> lalu pilih **Individual Edition** pada bagian **Products**.
2. Unduh **Anaconda Installer** dengan bahasa Python 3 sesuai dengan sistem operasi komputer lalu buka aplikasi yang telah diunduh.
3. Akan muncul aplikasi setup. Ikuti arahan dari aplikasi setup dan perhatikan destinasi folder untuk menginstal Anaconda.
4. Setelah instalasi selesai, buka aplikasi **Anaconda Navigator** pada komputer lalu pada aplikasi Anaconda Navigator instal Jupyter Notebook dengan menekan tombol **Install** pada icon **Jupyter Notebook**.
5. Setelah instalasi selesai, tombol **Install** akan berubah menjadi **Launch**. Klik tombol **Launch** untuk membuka Jupyter Notebook.

## B. Membuat Customized Virtual Environment

1. Buka Anaconda Prompt pada komputer.
2. Ketik code seperti ini dan ubah **myenv** dengan nama virtual environment yang diinginkan lalu tekan enter.



3. Anaconda Prompt akan membalas dengan pertanyaan **'Proceed ([y]/n)?'** lalu ketik 'y' dan tekan enter.
4. Untuk mengaktifkan virtual environment yang sudah dibuat, ketikan :  
`conda activate [NAMA VIRTUAL ENVIRONMENT]`  
Untuk mematikan virtual environment, ketikan :  
`conda deactivate`
5. Virtual environment yang dibuat menggunakan kode di atas akan menggunakan versi Python bawaan dari komputer. Jika ingin menggunakan environment dengan versi Python berbeda, ketikan :  
`conda create -n [NAMA VIRTUAL ENVIRONMENT] python=[VERSI]`
6. Untuk menginstal library pada virtual environment, aktifkan virtual environment yang diinginkan, lalu ketikkan:  
`conda install [NAMA LIBRARY]`, atau  
`pip install [NAMA LIBRARY]`
7. Untuk melihat libraries dan packages yang telah terinstal pada virtual environment, aktifkan virtual environment lalu ketikkan :  
`conda list`
8. Untuk menghapus virtual environment, ketikkan :  
`conda env remove -n [NAMA VIRTUAL ENVIRONMENT]`
9. Virtual environment yang telah dibuat bisa dibagikan ke perangkat lain dengan menggunakan kode berikut :  
`conda env export > [NAMA FILE].yaml`
10. Untuk membuat virtual environment dari file .yaml, gunakan kode berikut :  
`conda env create -f [NAMA FILE].yaml`

Untuk dokumentasi lebih lanjut dari Anaconda dapat dilihat pada <https://docs.conda.io/en/latest/>.

### C. Membuat Repository pada GitHub

1. Buka website <https://github.com/> lalu log in atau buat akun jika belum memiliki akun GitHub.
2. Setelah selesai log in, klik tombol **New** pada bagian repository.
3. Berikan nama untuk repository baru, lalu berikan deskripsi jika perlu lalu centang checkbox **Initialize this repository with a README** dan tekan tombol **Create repository**. Pilihan **Public** atau **Private** dipilih sesuai dengan kebutuhan.
4. Repository sudah berhasil dibuat. Edit file **README.md** untuk mengubah tampilan utama repository. File README.md bisa diisi instruksi untuk menggunakan file – file proyek di dalam repository dan bisa ditambahkan gambar atau GIF.
5. Untuk memasukkan file ke dalam repository, gunakan tombol **Upload** lalu drag file yang ingin diupload.  
Cara lain bisa dilakukan dengan tombol **Create New File** lalu ketikkan kode yang diinginkan. Pada saat memberikan nama diakhiri dengan “/”, maka secara otomatis nama tersebut akan dijadikan nama folder.
6. Gunakan fitur **Branch** untuk memisahkan pekerjaan pada file master sehingga kita bisa mengerjakan file baru tanpa perlu mengubah file master. Branch biasa digunakan untuk membuat update dari kode sehingga bisa dilakukan troubleshooting dan jika sudah tidak ada bug baru disatukan dengan file master.
7. Ketika melakukan perubahan pada file di dalam repository, kita bisa melakukan **Pull request**. Pull request adalah fitur pada GitHub untuk mereview perubahan yang terjadi pada file, lalu jika perubahan sesuai dengan kehendak, pemilik repository bisa mengkonfirmasi perubahan tersebut.
8. Untuk melakukan forking, buka repository orang lain, lalu tekan tombol **Fork**. Setelah beberapa saat, repository tersebut akan muncul pada repository kita. Lakukan beberapa perubahan lalu gunakan Pull request untuk menawarkan perubahan pada pemilik repository utama.

### 4. Pertanyaan

1. Download dan instal Jupyter Notebook pada komputer!
2. Buatlah akun GitHub lalu buat repository dengan format nama :  
**Data-Science\_[Nama]\_[Angkatan]\_EE-UPH**

Setelah repository dibuat, buatlah 6 folder dengan format nama :  
**Modul [n] – [Judul Modul]**

Hint :

n = 1 sampai 6, dan sesuaikan Judul Modul dengan Daftar Isi.

Pada GitHub, folder tidak bisa dibentuk jika tidak ada file di dalamnya. Buat file .md agar folder bisa dibentuk.

3. Buatlah sebuah virtual environment dengan spesifikasi sebagai berikut :  
Python : 3.8  
Libraries : numpy, pandas, plotly, seaborn, sklearn, scipy  
Setelah selesai membuat virtual environment tersebut, export environment tersebut ke dalam file .yml lalu upload file tersebut pada GitHub di folder Modul 1.
4. Buka repository modul data science Elektro UPH pada link berikut :  
<https://github.com/elektrouphDS>  
Download repository dalam bentuk Zip dengan menggunakan fitur **Download Zip**.  
Buka folder **Modul 1 - Pendahuluan : Data Science, Python, Jupyter Notebook & GitHub**. Buat virtual environment pada komputer dengan file **Viren01.yml** lalu tuliskan spesifikasi dari virtual environment tersebut!
5. Tulislah apa yang Anda ketahui tentang aplikasi data science untuk Teknik Elektro pada file **README.md** di repository Anda! Tambahkan gambar agar penjelasan lebih detail.



# Python for Data Analytics : Data Types, Operations, Logical Operations & Function

## 1. Tujuan

1. Mengetahui lingkungan kerja Jupyter Notebook.
2. Mengetahui data types yang tersedia pada Python 3 beserta built-in function yang sering digunakan.
3. Dapat melakukan function print dan print with format pada Python 3.
4. Dapat melakukan logical operation menggunakan function if, elif, dan else.
5. Dapat melakukan iterasi menggunakan function for dan while.
6. Dapat membuat function sendiri.
7. Dapat membuat lambda expression dari function yang telah dibuat.

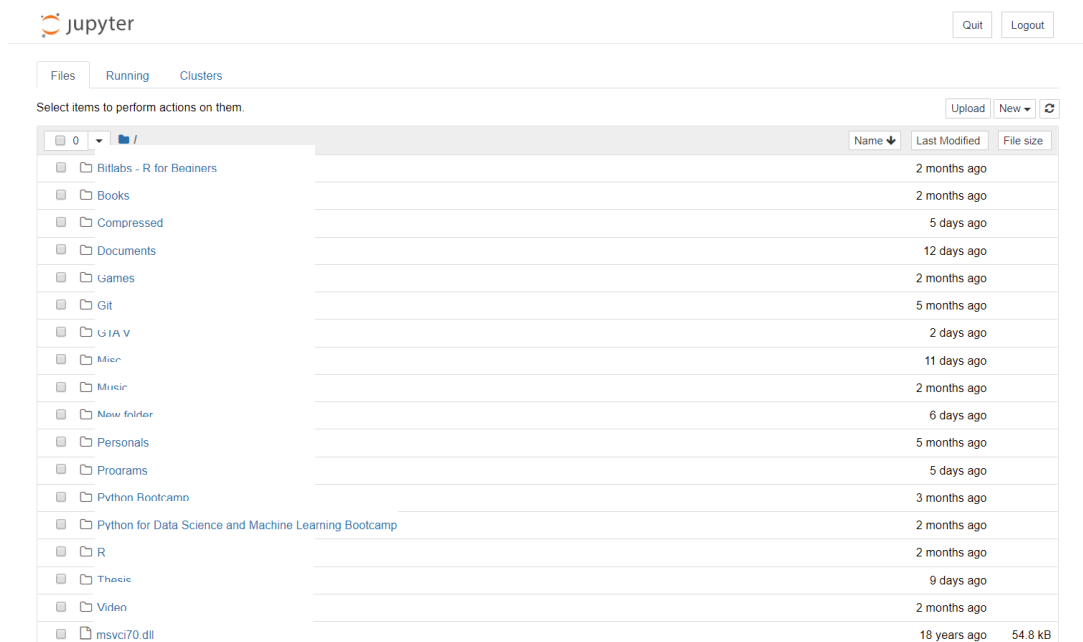
## 2. Penjelasan Singkat

Jupyter Notebook adalah aplikasi yang sangat populer untuk digunakan pada data science. Jupyter notebook dapat menjalankan bahasa pemrograman Python untuk versi 2 dan 3. Keunggulan Jupyter Notebook dibandingkan dengan aplikasi compiler lain adalah Jupyter Notebook dapat mengeluarkan output langsung dalam bentuk tab dan bisa digunakan juga untuk menuliskan catatan dengan menggunakan markdown.

### CATATAN

Jupyter Notebook dapat menjalankan bahasa pemrograman R. Untuk menggunakan bahasa pemrograman R, install packages untuk R pada saat membuat environment.

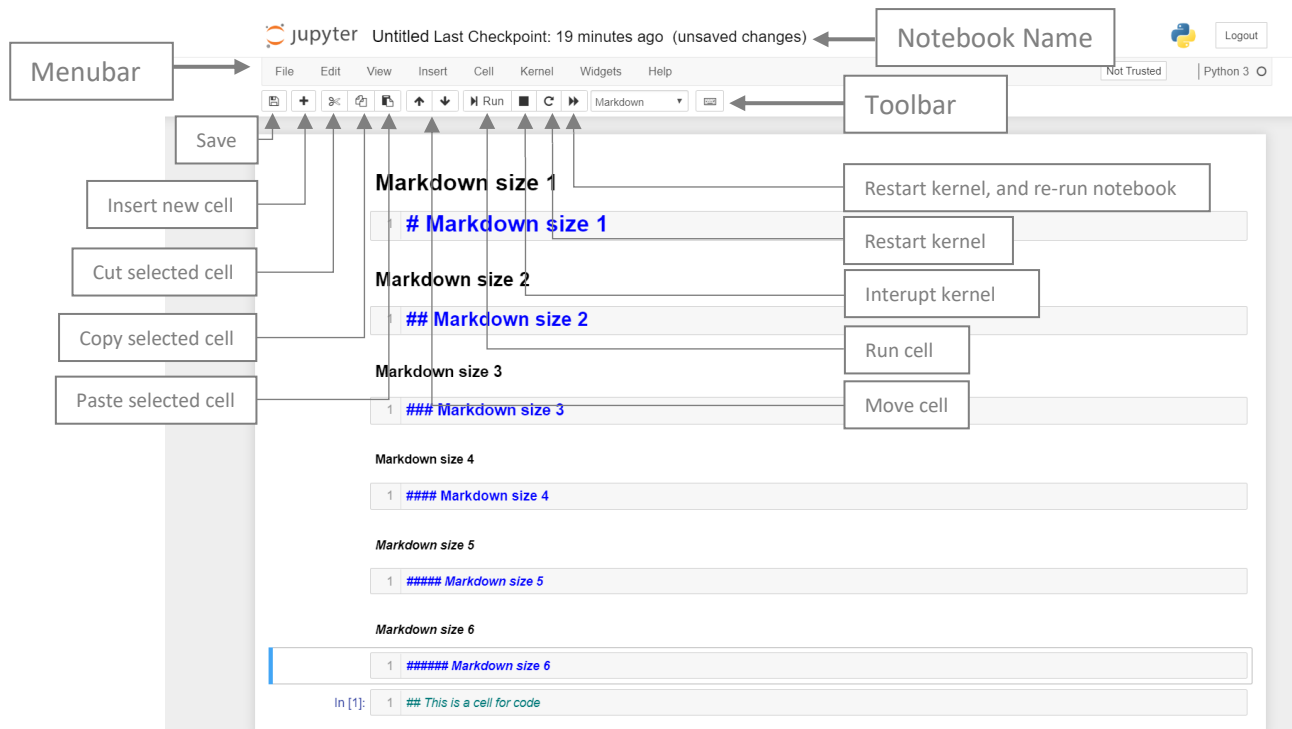
<https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/>



Gambar 2.1. Halaman utama Jupyter Notebook

Untuk membuat sebuah file baru, tekan tombol New lalu pilih bahasa pemrograman yang diinginkan. Directory file yang dibuat akan berada pada directory di mana Anaconda diinstal. Ketika bekerja dengan beberapa notebook sekaligus, kita dapat memonitor notebook – notebook yang dibuka pada tab **Running**. Pada bagian tab

Running, kita dapat memonitor seluruh notebook yang terbuka dan kita bisa mematikan notebook ketika tidak digunakan dengan tombol **Shutdown**. Melakukan Shutdown kepada notebook yang tidak digunakan dapat menghemat komputasi yang dilakukan interpreter.



Gambar 2.2. Lingkungan Kerja Jupyter Notebook

## Data Types

Python 3 memiliki banyak data types. Data types yang tersedia pada Python 3 adalah sebagai berikut :

Data type	Keterangan	Syntax	Contoh
<b>Integer</b>	Bilangan bulat	a = int (BILANGAN BULAT)	a = int (9)
		a = BILANGAN BULAT	a = 9
<b>Float</b>	Bilangan desimal	b = float (BILANGAN DESIMAL)	b = float(20.1)
		b = BILANGAN DESIMAL	b = 20.1
<b>Complex</b>	Bilangan kompleks	c = complex (BILANGAN KOMPLEKS)	c = complex (2,1)
		c = (BILANGAN KOMPLEKS)	c = (2+1j)
<b>String</b>	Seurutan karakter	d = 'KARAKTER ATAU KATA'	d = 'UPH'
		d = "KARAKTER ATAU KATA"	d = "UPH"
<b>List</b>	Sekumpulan objek yang terurut oleh index, bisa reassign elemen didalam	e = list((OBJEK))	e = list (('ha', 'he', 'ho', 1, 2, 3))
		e = [OBJEK]	e = ['ha', 'he', 'ho', 1, 2, 3 ]
<b>Tuple</b>	Sekumpulan objek yang terurut oleh index, tidak bisa	f = tuple ((OBJEK))	f = tuple((1,2,3))

	reassign elemen di dalam	f = (OBJEK)	f = (1,2,3)
<b>Boolean</b>	Preposisi logis, 'TRUE' atau 'FALSE'	g = bool(0 atau 1)	g = bool(0)
		g = True atau g = False	g = False
<b>Dictionary</b>	Sekumpulan objek yang terurut oleh key	h = dict ({key 1: value1, key2 : value2})	h = dict ({'a' : 1, 'b' : 2})
		h = {key 1: value1, key2 : value2}	h = {'a' : 1, 'b' : 2}
<b>Sets</b>	Sekumpulan objek yang tidak berulang (unik)	i = set(OBJEK)	i = set(1,2,3,3,4,5,6)
			>> output : (1,2,3,4,5,6)

Data types jenis numerik (int, float, complex) memiliki beberapa operasi sebagai berikut :

Operasi	Syntax	Hasil
<b>Penambahan</b>	a+b	Penjumlahan a dan b
<b>Pengurangan</b>	a-b	Pengurangan a oleh b
<b>Perkalian</b>	a*b	Perkalian a dan b
<b>Pembagian</b>	a/b	Pembagian a oleh b
<b>Hasil bagi</b>	a//b	Mengambil quotient dari hasil pembagian
<b>Sisa bagi</b>	a%b	Mengambil remainder dari hasil pembagian
<b>Perkalian berpangkat</b>	pow (a,b)	Perkalian a dengan pangkat b
<b>Absolut</b>	abs(a)	Nilai mutlak dari a
<b>Konjugasi</b>	a.conjugate()	Konjugasi dari bilangan kompleks a

Data type string memiliki beberapa built-in function dan teknik yang cukup sering digunakan, yaitu :

mystring :	T	E	K	N	I	K		E	L	E	K	T	R	O		U	P	H
Index :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Negative Index :	0	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Built-in Function	Syntax	Contoh	Hasil
<b>Calling by index</b>	mystring [n]	mystring [3]	'N'
<b>Calling by negative index</b>	mystring [-n]	mystring [-5]	'O'
<b>Slicing</b>	mystring [Start : Stop : Step]	mystring [0:5:1]	'TEKNI'
<b>String length</b>	len(mystring)	len(mystring)	18
<b>Uppercase</b>	mystring.upper()	mystring.upper()	'TEKNIK ELEKTRO UPH'
<b>Lowercase</b>	mystring.lower()	mystring.lower()	'teknik elektro uph'
<b>Splitting</b>	mystring.split('splitter')	mystring.split( )	['TEKNIK','ELEKTRO','UPH']
<b>Reverse string</b>	mystring [ : : -1]	mystring [ : : -1]	'HPU ORTKELE KINKET'

Untuk menampilkan suatu data type atau hasil operasi, bisa digunakan function **print()**. Function ini bisa juga menggunakan **format** sehingga ketika digabungkan dengan fungsi iterasi akan menampilkan hasil iterasi yang lebih rapi. Syntax untuk melakukan function print dan format adalah sebagai berikut :

Print & Format Syntax	CATATAN
<pre> print (mystring) &gt;&gt; out : TEKNIK ELEKTRO UPH  print ('Nama saya adalah {}'.format(('Budi')))  &gt;&gt; out : Nama saya adalah Budi.  print ('Angka ke- {0} adalah {1}'.format(3,1))  &gt;&gt; out : Angka ke- 3 adalah 1.  print ('Harga {a} {b} adalah {c}'.format(a = 10, b = 'Apel', c = 50000)) &gt;&gt; out : Harga 10 Apel adalah 50000 </pre>	<p>Seluruh kode yang dituliskan dengan Python 3 bersifat case sensitive yang artinya huruf kapital sangat mempengaruhi.</p> <p>.format() akan dikenali oleh interpreter, sedangkan .Format() tidak akan dikenali.</p>

### Sequential Data Types : List, Tuple, Dictionary, Set

Sequential Data types terbagi menjadi dua jenis, mutable dan immutable. Secara sederhana, data types mutable adalah data types yang elemen di dalamnya dapat diubah setelah dibuat sedangkan immutable tidak bisa. List dan set adalah objek mutable sedangkan tuple dan dictionary adalah immutable.

Berikut adalah built-in function untuk data type list:

mylist :	1	'hai'	'hello'	55	[1,3]	{'a': 1, 'b': 2}	1	'hahaha'
Index :	0	1	2	3	4	5	6	7

Built-in Function	Syntax	Contoh	Hasil
Calling by Index	mylist [n]	mylist [1]	'hai'
Multi-index Calling	mylist [n][m]	mylist [4][1]	3
Length of List	len[mylist]	len[mylist]	6
Appending	mylist.append(OBJECT)	mylist.append('hoho')	[1, 'hai', 'hello', 55, [1,3], {'a':1, 'b':2}, 1, 'hahaha', 'hoho']
Slicing	mylist [Start : Stop : Step]	mylist [0:2:1]	1, 'hai'

### Conditional Statement : If, Elif, Else

Conditional statement adalah fitur populer yang tersedia di berbagai bahasa pemrograman, termasuk Python 3. Conditional statement memanfaatkan booleans

(True or False) untuk menguji suatu kondisi terpenuhi atau tidak. Data types booleans pada Python 3 memiliki operasi sebagai berikut :

Operasi	Syntax	Hasil
<b>OR</b>	A or B	Jika A adalah False, maka B. Jika tidak maka A
<b>AND</b>	A and B	Jika A adalah False, maka A. Jika tidak maka B
<b>NOT</b>	not A	Jika A adalah False, maka True. Jika tidak maka False
<b>Bitwise Complement</b>	~A	A complement
<b>Bitwise OR</b>	A   B	Sama dengan OR, namun bitwise
<b>Bitwise AND</b>	A & B	Sama dengan AND, namun bitwise

Conditional statement memerlukan comparison operation. Berikut adalah comparison operation untuk Python 3 :

Operasi	Syntax	Hasil
<b>Lebih Kecil</b>	A < B	Jika A lebih kecil dari B, maka True
<b>Lebih kecil sama dengan</b>	A <= B	Jika A lebih kecil atau sama dengan B, maka True
<b>Lebih besar</b>	A > B	Jika A lebih besar dari B, maka True
<b>Lebih besar sama dengan</b>	A >= B	Jika A lebih besar atau sama dengan B, maka True
<b>Sama dengan</b>	A == B	Jika A sama dengan B, maka True
<b>Tidak sama dengan</b>	A != B	Jika A tidak sama dengan B, maka True
<b>Adalah</b>	A is B	Jika A adalah B, maka True
<b>Bukanlah</b>	A is not B	Jika A bukanlah B, maka True

Conditional Statement Syntax	Contoh Conditional Statement
<pre>if (KONDISI 1):     KODE 1 elif (KONDISI 2):     KODE 2 else:     KODE 3</pre>	<pre>a = 7 if (a == 10):     num_1 = 101     num_2 = 201 elif (a &gt; 10):     num_1 = 102     num_2 = 202 else:     num_1 = 103     num_2 = 203 print (num_1, num_2)  &gt;&gt; out : 103, 203</pre>
CATATAN	
<p>Indentasi dari kode pada Python 3 sangat berpengaruh terutama saat menggunakan conditional statement. Serangkaian kode pada satu kondisi harus memiliki indentasi yang sama agar bisa berjalan bersama.</p>	

Conditional statement if adalah statement yang menguji apakah suatu kondisi terpenuhi atau tidak. Jika terpenuhi (True), maka serangkaian kode akan dijalankan. Jika tidak terpenuhi, maka serangkaian kode tersebut akan dilewati dan pembacaan akan dilanjutkan. Statement if biasanya diikuti dengan elif atau else. Elif adalah statement lanjutan ketika statement if di atasnya tidak terpenuhi sedangkan else adalah statement paling akhir ketika seluruh if dan elif tidak terpenuhi.

## Iteration : For & While

Iterasi adalah fitur yang sangat berguna untuk melakukan proses yang berulang. Iterasi biasa digunakan untuk melakukan print berulang ataupun perhitungan berulang. Ada 2 jenis iteration yang bisa digunakan yaitu for dan while. For biasa digunakan untuk melakukan iterasi pada setiap elemen di dalam suatu list, tuple, set, atau dictionary sedangkan while biasanya digunakan untuk melakukan iterasi sampai suatu kondisi terpenuhi.

Berikut adalah syntax untuk menggunakan for dan while:

Syntax For Loop	Contoh For Loop
<pre>for i in x:     print (i)</pre>	<pre>x = [1,2,3,3,4]  for i in x:     print (i) &gt;&gt; out : 1 2 3 3 4</pre>
Catatan	
<p>Variabel i pada syntax bisa diubah dengan nama variabel apa saja.</p> <p>Secara otomatis, setiap iterasi nilai variabel i akan bertambah dengan 1.</p>	
Syntax While Loop	Catatan
<pre>times = 1 while (CONDITION):     CODE</pre>	<p>Indentasi juga sangat penting pada iterasi. Fungsi indentasi pada iterasi sama dengan indentasi pada conditional statement yaitu untuk membedakan kode yang termasuk ke dalam perintah atau tidak.</p>
Contoh While Loop	
<pre>times = 1 while times != 6:     print ('Iterasi ke : {}'.format (times))  &gt;&gt; out : Iterasi ke : 1 Iterasi ke : 2 Iterasi ke : 3 Iterasi ke : 4 Iterasi ke : 5</pre>	

## Function, Mapping, Lambda Expression

Function sering digunakan untuk meringkas kode untuk dipanggil saat dibutuhkan. Suatu function dapat disesuaikan sesuai kebutuhan. Dengan adanya function, keseluruhan kode dapat diperpendek dengan meringkas sebagian kode yang berulang. Perlu diingat bahwa function berada pada scope yang lebih kecil dari keseluruhan code sehingga objek yang dideklarasikan di dalam function tidak akan dikenali di luar scope function. Suatu function dapat digunakan untuk mengembalikan

suatu nilai dengan fitur function. Walaupun begitu, suatu function bisa juga tidak mengembalikan value melainkan sekadar melakukan serangkaian code.

Mapping adalah fitur yang bisa digunakan untuk menerapkan function ke beberapa objek sekaligus tanpa perlu melakukan iterasi. Mapping biasa digunakan untuk menerapkan function kepada beberapa elemen di dalam sequential data type seperti list.

Lambda expression biasa juga disebut anonymous function. Artinya, lambda expression berfungsi seperti function pada umumnya tapi tidak didefinisikan seperti function. Lambda expression cocok digunakan ketika serangkaian code tersebut terlalu pendek jika dibuat menjadi function. Lambda expression bisa digunakan untuk meringkas code menjadi satu line code dan mengurangi nesting.

Berikut syntax function, mapping, dan lambda expression :

Function Syntax	Mapping Syntax	Lambda Exp. Syntax
<pre>def func_name(a):     result = a**2     return a</pre>	<pre>map (function,target)</pre>	<pre>lambda item: operation</pre>
Contoh Function		
<pre>def volume_of_cube (a):     volume = a**3     return volume  &gt;&gt;&gt; volume_of_cube(4) 64</pre>		
Contoh Mapping		
<pre>my_list = [1,2,3,4,5] list (map (volume_of_cube, my_list))  &gt;&gt;&gt; [1, 8, 27, 64, 125]</pre>		
Contoh Lambda Expression		
<pre>list (map(lambda number: number**3, my_list))  &gt;&gt;&gt; [1, 8, 27, 64, 125]</pre>		

### 3. Langkah Kerja

#### 1. Built-in Function : input()

1. Buka tautan berikut dan pelajari mengenai built-in function input().  
<https://docs.python.org/3/library/functions.html#input>
2. Bukalah Jupyter Notebook, lalu buatlah notebook dengan file name :  
Latihan Modul 2 – [NAMA] – [NIM]
3. Buatlah sebuah program yang akan meminta user untuk memasukkan :  
Nama :  
NIM :  
Umur :  
Fakultas :  
Program Studi :

Setelah user selesai memasukkan seluruh informasi, maka program akan mengeluarkan hasil sebagai berikut menggunakan function print:

Mahasiswa bernama **[Nama]** berumur **[Umur]** tahun. **[Nama]** adalah mahasiswa **[Fakultas]** dan mengambil program studi **[Program Studi]**. NIM **[Nama]** adalah **[NIM]**.

#### B. Nested If

1. Buatlah program menggunakan if, elif, dan else untuk melakukan sebuah decision making.

Dengan ketentuan sebagai berikut :

- i. User diminta untuk memasukkan 3 angka yang akan dimasukkan ke dalam variabel a,b, dan c.
- ii. Program akan melakukan print alfabet seperti berikut jika syarat – syarat dipenuhi.

Hasil	Syarat
<b>V</b>	Jika a lebih besar dari 10 dan b lebih besar sama dengan 5
<b>W</b>	Jika a lebih besar dari 10 dan b lebih kecil dari 5
<b>X</b>	Jika a lebih kecil dari 10 dan c lebih besar dari 7
<b>Y</b>	Jika a lebih kecil dari 10 dan c lebih kecil sama dengan 7
<b>Z</b>	Jika a sama dengan 10

#### ■ Note :

- Buatlah terlebih dahulu pohon keputusan dari syarat.
- Buat program tersebut menggunakan function If di dalam If (nested If) dan tidak diperbolehkan menggunakan operator AND dan OR.
- Asumsikan user pasti akan memasukkan angka dan bukan huruf.



- Jika muncul error ketika operator perbandingan tidak bisa membandingkan string, konversi terlebih dahulu input yang dimasukkan user menjadi variabel numerik.

### C. Nested Loop

1. Buatlah suatu program untuk membuat tampilan segi empat dengan menggunakan function print di dalam suatu loop.
2. Hint :

- User akan memasukkan dua buah angka yang akan ditampung pada variabel panjang dan lebar.
- Lakukan print karakter '\*' (asterik dengan dua space) pada function loop agar hasil program menyerupai persegi.
- Agar function print tidak secara otomatis memulai line baru setelah melakukan print, gunakan syntax sebagai berikut :

```
print ('* ', end='')
```

- Jika muncul error pada function for di mana string tidak dapat diinterpretasikan sebagai integer, ubahlah terlebih dahulu input dari user menjadi integer.

```
panjang persegi : 10
lebar persegi : 10
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
panjang persegi : 6
lebar persegi : 6
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
panjang persegi : 3
lebar persegi : 2
* * *
```

### D. Function, Mapping, Lambda Expression

1. Buatlah program nested loop menjadi suatu function dengan nama square\_maker.
2. Buatlah suatu lambda expression untuk menghitung volume suatu bola dengan panjang jari – jari sesuai user inginkan.
  - **Hint : Rumus volume bola :  $\frac{4}{3}\pi r^3$**
  - Lambda expression yang dibuat hanya 1 line code, dan map lambda expression tersebut kepada list dengan isi : [1,2,3,4,5,6]

### 4. Pertanyaan

1. Pada program A, jika pada function input NIM dimasukkan 02030001 lalu dikonversi menjadi suatu integer, apa hasil print yang dikeluarkan pada bagian NIM?
2. Ubahlah program A yang sudah diselesaikan sehingga keluaran dari print merupakan kalimat yang seluruh hurufnya adalah huruf kapital

3. Tentukan keluaran program pada program B jika user memasukkan input dengan skenario sebagai berikut :

Input			Hasil
a	b	c	
20	3	7	
15	2	8	
9	2	4	
14	15	3	
1	7	10	
10	1	6	
2	3	1	
11	5	8	
4	6	7	
9	1	9	
10	9	9	
3	7	4	

4. Buatlah program untuk menampilkan bentuk jajaran genjang. User akan memasukkan dua variabel yaitu panjang dan tinggi.

```

panjang : 10
tinggi : 10
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

panjang : 5
tinggi : 7
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

panjang : 5
tinggi : 5
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

```

5. Buatlah suatu readme file pada folder Modul 2 GitHub pribadi anda dan jelaskan apa yang membedakan antara function dan lambda expression. Buatlah penjelasan dengan menyertakan gambar contoh function yang disederhanakan menjadi suatu lambda expression.
6. Pada file readme yang sama, jelaskan perbedaan antara function, method, dan property!
7. Upload seluruh program yang telah dikerjakan pada GitHub pribadi pada folder **Modul 2 – Python untuk Data Analytics : Data Types, Operations, Logical Operations & Function** dengan format .ipynb.

## Data Analysis : Clustering & Principal Component Analysis

### 1. Tujuan

- Dapat melakukan data cleansing menggunakan library Pandas.
- Dapat melakukan Exploratory Data Analysis (EDA) yang divisualisasikan dengan library Seaborn.
- Dapat melakukan data preprocessing menggunakan library Scikit-learn.
- Dapat menggunakan algoritma machine learning K-Means Clustering untuk mengelompokkan data.

### 2. Penjelasan Singkat

Algoritma machine learning terbagi menjadi banyak jenis seperti supervised learning, unsupervised learning, semi-supervised learning, deep learning, dan reinforced learning. Untuk pembelajaran awal, algoritma yang akan dipelajari adalah **unsupervised learning** dan **supervised learning**. Yang membedakan kedua algoritma ini adalah data yang dianalisis. Untuk supervised learning, data yang digunakan harus memiliki variabel target sehingga supervised learning biasa digunakan untuk melakukan **prediksi**. Sebaliknya, unsupervised learning menggunakan data yang tidak perlu ada variabel target. Unsupervised learning sering digunakan untuk mengelompokkan data ke dalam **segmen** atau **klaster**. Unsupervised learning tidak memiliki *ground truth* akibat tidak adanya variabel target sehingga untuk menilai keberhasilan algoritma diperlukan visualisasi dari hasil pengelompokan.

Dalam menggunakan algoritma machine learning jenis apapun, sudah seharusnya data yang dimasukkan dibersihkan terlebih dahulu. Proses pembersihan data disebut dengan **data cleansing**. Proses data cleansing ditujukan untuk menghilangkan dataset dari sampel yang tidak memiliki nilai pada satu atau lebih kolom (NULL). Sampel dengan value NULL dapat dibuang atau jika ukuran dataset sangat kecil, value NULL dapat diimputasi dengan nilai lain seperti mean atau median. Metode imputasi NULL yang lebih kompleks dapat dilihat pada link berikut :

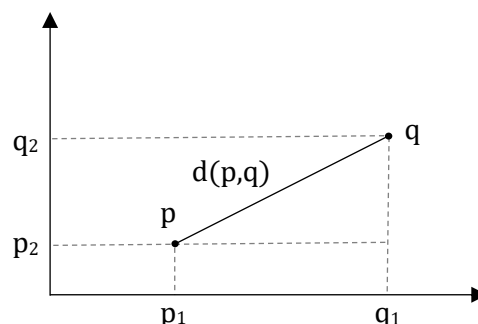
<https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>

Langkah selanjutnya adalah **Exploratory Data Analysis (EDA)**. Langkah ini dilakukan untuk melihat karakteristik dataset secara umum. EDA meliputi menampilkan data type tiap kolom, jumlah sampel yang mengandung NULL, ringkasan statistik tiap kolom, dan visualisasi distribusi data menggunakan histogram atau boxplot. Dengan melakukan EDA, outliers bisa ditemukan sehingga outliers bisa dibuang dan tidak mengganggu performa machine learning. **Urutan tahap EDA dan data cleansing kadang dipertukarkan tergantung dengan data yang dianalisis.**

Sebelum memasukkan data ke algoritma machine learning, proses yang perlu dilakukan adalah **standarisasi data**. Mengingat machine learning hanya menerima data numerik, maka agar setiap data memiliki skala yang sama, maka dilakukan standarisasi menggunakan function **StandardScaler** pada library Scikit-learn.

Untuk dataset dengan ukuran kolom besar (lebih dari 3 kolom), maka sebaran data tidak bisa lagi dipetakan pada koordinat Kartesian. Oleh karena itu, digunakanlah suatu metode bernama **Principal Component Analysis (PCA)**. Metode ini digunakan untuk menurunkan dimensi data menjadi dua atau tiga dimensi sehingga data bisa dipetakan sedangkan informasi yang terbuang seminimal mungkin. Prinsip utama PCA adalah menentukan komponen baru yang memiliki **variance** data sedekat mungkin dengan dataset asli dengan menggunakan **eigen value**.

Setelah data melewati seluruh tahap di atas, maka data siap dimasukkan ke algoritma machine learning. Pada kasus ini, algoritma **K-Means Clustering** akan mengelompokkan data ke dalam K klaster di mana nilai K ditentukan oleh user sendiri. K-Means Clustering bekerja dengan cara menyebar titik secara acak sejumlah K. Titik – titik ini disebut centroid dan akan dianggap sebagai titik tengah sebaran data. Centroid yang disebar kemudian akan dihitung jarak Euclidean distancenya terhadap setiap data dan data tersebut akan diberikan label sesuai centroid terdekat. Setelah seluruh sampel diberikan label, maka centroid akan dipindahkan lokasinya sesuai dengan jumlah jarak dari sampel yang termasuk pada centroid tersebut dibagi dengan jumlah sampel yang termasuk. Proses ini akan dilakukan terus menerus hingga tidak ada perubahan lokasi dari centroid.



$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2 + \dots + (p_n - q_n)^2}$$

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Dalam menentukan nilai K, maka diperkenalkan metode **Elbow Method**. Metode ini akan mencoba setiap nilai K pada rentang tertentu dan membandingkannya dengan klaster K-1. Ketika terjadi perubahan klaster dari suatu data, maka perubahannya akan dihitung lalu diakumulasi dan ditampilkan dalam bentuk grafik. Grafik yang dihasilkan

akan menyerupai bentuk siku tangan (elbow) dan titik siku tersebut yang akan digunakan sebagai nilai K pada algoritma.

### 3. Langkah Kerja

#### A. Importing Library & Reading Dataset

- 1) Buka folder yang berisikan modul praktikum dari GitHub Elektro UPH yang telah didownload saat mengerjakan Modul 1.
- 2) Buka Jupyter Notebook, buat notebook baru dengan bahasa pemrograman Python 3 lalu import library berikut:
  - a. pandas
  - b. numpy
  - c. seaborn
  - d. matplotlib.pyplot
  - e. sklearn

Note :

- Pastikan seluruh library tersebut telah diinstal pada virtual environment.
  - Fitur **as** digunakan untuk mempersingkat nama library ketika dipanggil.
- 3) Gunakan built-in function `read_csv()` pada library pandas untuk membaca file dataset dengan format `.csv`. Perhatikan filename dataset yang akan dibaca dan directory file. Untuk mempermudah, letakkan file dataset pada folder yang sama dengan notebook.

```
In [1]: 1 import numpy as np
        2 import pandas as pd
        3 import seaborn as sns
        4 import matplotlib.pyplot as plt
        5 import sklearn
        6 pd.set_option('display.max_columns', None)
        7 data = pd.read_csv('wine-clustering.csv')
        8 data
```

161	13.69	3.26	2.54	20.0	107	1.83	0.56	0.50	0.80	5.8
162	12.85	3.27	2.58	22.0	106	1.65	0.60	0.60	0.96	5.5
163	12.96	3.45	2.35	18.5	106	1.39	0.70	0.40	0.94	5.2
164	13.78	2.76	2.30	22.0	90	1.35	0.68	0.41	1.03	9.5
165	13.73	4.36	2.26	22.5	88	1.28	0.47	0.52	1.15	6.6
166	13.45	3.70	2.60	23.0	111	1.70	0.92	0.43	1.46	10.6
167	12.82	3.37	2.30	19.5	88	1.48	0.66	0.40	0.97	10.2
168	13.58	2.58	2.69	24.5	105	1.55	0.84	0.39	1.54	8.6
169	13.40	4.60	2.86	25.0	112	1.98	0.96	0.27	1.11	8.5
170	12.20	3.03	2.32	19.0	96	1.25	0.49	0.40	0.73	5.5
171	12.77	2.39	2.28	19.5	86	1.39	0.51	0.48	0.64	9.8

## B. Exploratory Data Analysis & Data Cleansing

- 1) EDA dapat dimulai dengan melihat datatype setiap kolom pada dataset. Gunakan method **info()** pada library pandas untuk menampilkan datatype setiap kolom dan menampilkan jumlah value non-NULL.

```
In [4]: 1 data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 13 columns):
Alcohol           178 non-null float64
Malic_Acid        178 non-null float64
Ash               178 non-null float64
Ash_Alcanity      178 non-null float64
Magnesium         178 non-null int64
Total_Phenols     178 non-null float64
Flavanoids        178 non-null float64
Nonflavanoid_Phenols 178 non-null float64
Proanthocyanins   178 non-null float64
Color_Intensity   178 non-null float64
Hue               178 non-null float64
OD280             178 non-null float64
Proline           178 non-null int64
dtypes: float64(11), int64(2)
memory usage: 18.2 KB
```

- 2) Periksa ukuran dataset dengan menggunakan property **shape**. Property shape akan menampilkan ukuran dataset. Cocokkan jumlah non-NULL untuk setiap kolom dengan ukuran dataset.

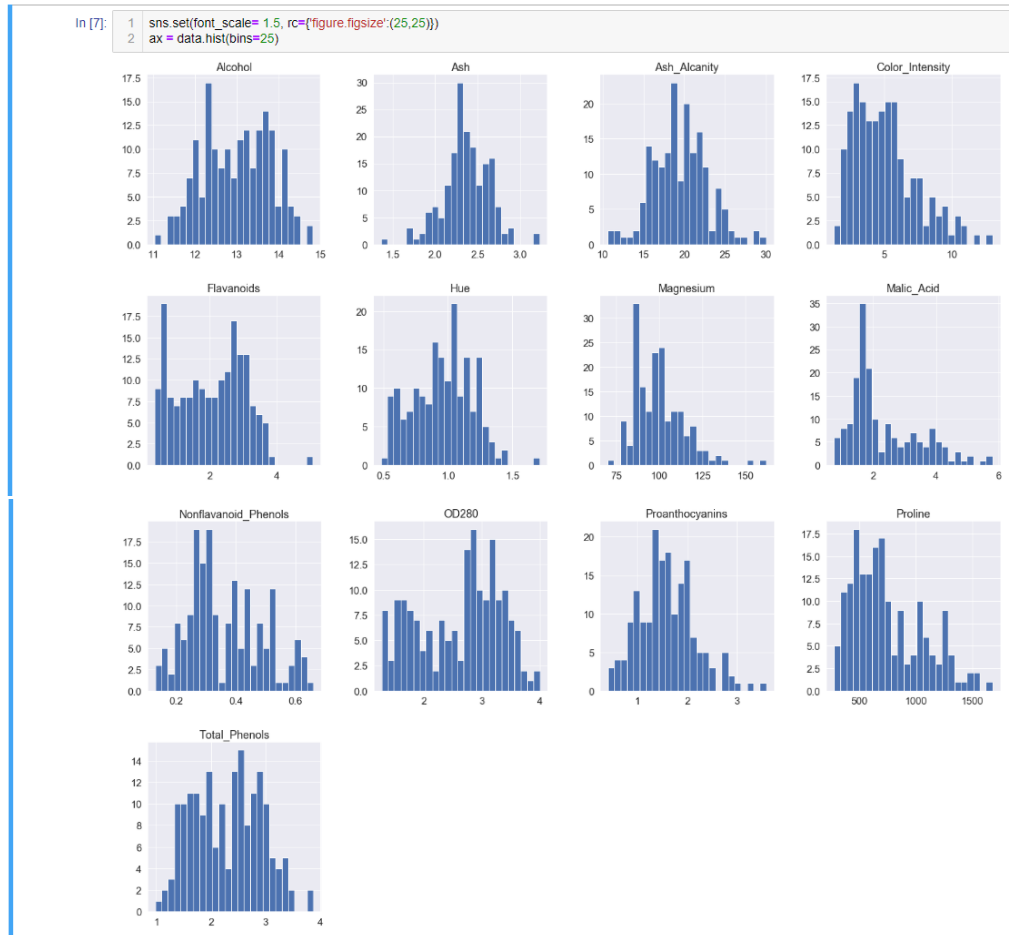
```
In [3]: 1 data.shape
Out[3]: (178, 13)
```

- 3) Karena tidak ada NULL pada dataset, maka null imputation tidak dilakukan dan proses dilanjutkan dengan menampilkan ringkasan informasi statistik setiap kolom dengan menggunakan method **describe()**.

```
In [6]: 1 data.describe()
Out[6]:
```

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Inten
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000

- 4) Visualisasikan sebaran data dengan histogram untuk melihat distribusinya menggunakan method **hist()**. Gunakan juga library Seaborn agar penampilan grafik lebih rapih.



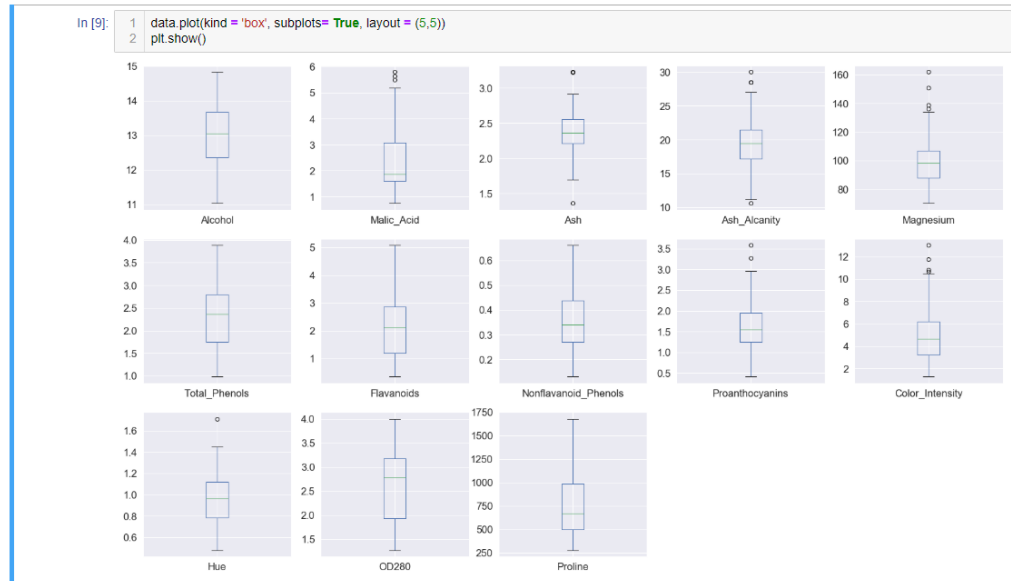
- 5) Hitung skewness (kecondongan / kemiringan) distribusi data menggunakan method **skew()**.

```
In [39]: 1 skew = pd.DataFrame(data.skew(), columns = ['Skewness'])
        2 skew
```

Out[39]:

	Skewness
Alcohol	-0.051482
Malic_Acid	1.039651
Ash	-0.176699
Ash_Alcanity	0.213047
Magnesium	1.098191
Total_Phenols	0.086639
Flavanoids	0.025344
Nonflavanoid_Phenols	0.450151
Proanthocyanins	0.517137
Color_Intensity	0.868585
Hue	0.021091
OD280	-0.307285
Proline	0.767822

- 6) Visualisasikan sebaran data dengan boxplot untuk melihat outliers menggunakan method **plot()**. Masukkan parameter **kind = 'box'** pada method untuk menampilkan boxplot.



- 7) Hitung simpangan interkuartil (Interquartile Range / IQR) untuk setiap kolom lalu buang seluruh data yang melewati rentang :

- $Q1 - 1.5 \times IQR$
- $Q3 + 1.5 \times IQR$

```
In [10]: 1 q1 = data.quantile(0.25)
        2 q3 = data.quantile(0.75)
        3 iqr = q3-q1
        4 print(iqr)
```

Alcohol 1.3150  
Malic\_Acid 1.4800  
Ash 0.3475  
Ash\_Alcanity 4.3000  
Magnesium 19.0000  
Total\_Phenols 1.0575  
Flavonoids 1.6700  
Nonflavanoid\_Phenols 0.1675  
Proanthocyanins 0.7000  
Color\_Intensity 2.9800  
Hue 0.3375  
OD280 1.2325  
Proline 484.5000  
dtype: float64

```
In [11]: 1 condition_1 = (q1 - 1.5 * iqr)
        2 condition_2 = (q3 + 1.5 * iqr)
        3
        4 data_cleaned = data[~((data < condition_1) | (data > condition_2)).any(axis=1)]
        5 data_cleaned
```

167	12.82	3.37	2.30	19.5	88	1.48	0.66	0.40	0.97	10.2
168	13.58	2.58	2.69	24.5	105	1.55	0.84	0.39	1.54	8.6
169	13.40	4.60	2.86	25.0	112	1.98	0.96	0.27	1.11	8.5
170	12.20	3.03	2.32	19.0	96	1.25	0.49	0.40	0.73	5.5
171	12.77	2.39	2.28	19.5	86	1.39	0.51	0.48	0.64	9.8
172	14.16	2.51	2.48	20.0	91	1.68	0.70	0.44	1.24	9.7
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.3
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.2
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.3
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.2

161 rows x 13 columns



### C. Data Preprocessing & Principal Component Analysis

- 1) Mulai tahap preprocessing dengan import library **StandardScaler** lalu lakukan standarisasi dataset yang sudah dibersihkan. Cek kembali apakah standarisasi sudah sesuai dengan melihat mean dan standar deviasi setiap kolom. Jika standarisasi berhasil, maka nilai mean akan sama dengan 0, dan standar deviasi akan sama dengan 1.

```
In [11]: 1 from sklearn.preprocessing import StandardScaler
2
3 std_scaler = StandardScaler()
4 data_cluster = data_cleaned.copy()
5 data_cluster[data_cluster.columns] = std_scaler.fit_transform(data_cluster)
6 data_cluster.describe()
```

Out[11]:

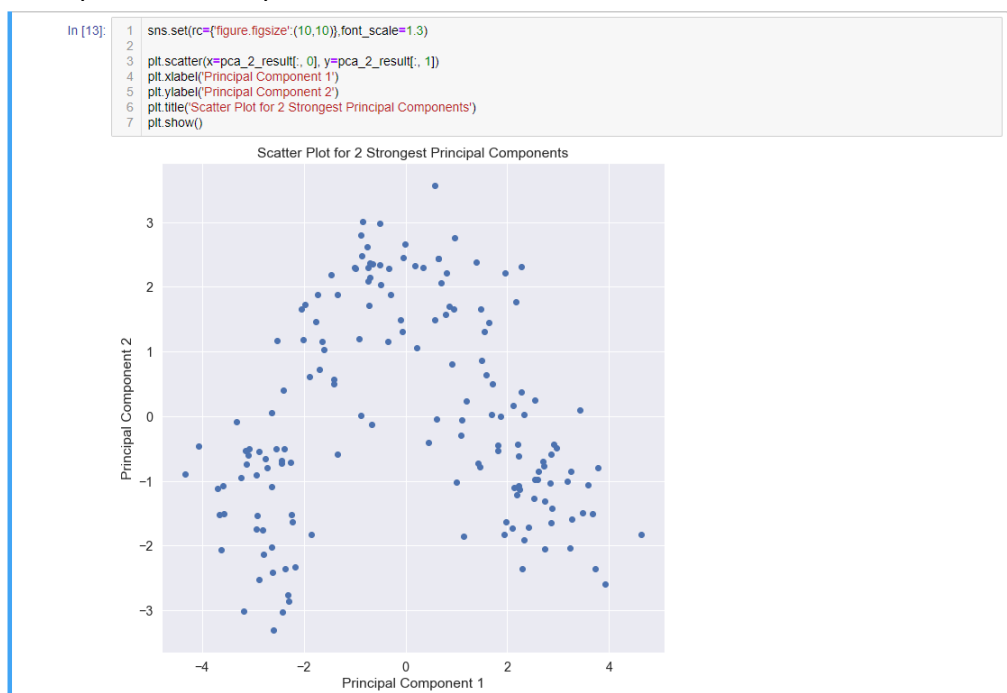
	Alcohol	Malic_Acid	Ash	Ash_Alcalinity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Pr
count	1.610000e+02	1.610000e+02	1.610000e+02	1.610000e+02	1.610000e+02	1.610000e+02	1.610000e+02	1.610000e+02	
mean	-4.551225e-16	-6.757879e-17	1.558450e-16	4.316768e-16	-2.063567e-16	5.861426e-16	-5.102888e-17	2.620402e-17	
std	1.003120e+00	1.003120e+00	1.003120e+00	1.003120e+00	1.003120e+00	1.003120e+00	1.003120e+00	1.003120e+00	
min	-2.049857e+00	-1.499357e+00	-2.758493e+00	-2.672685e+00	-2.314802e+00	-2.078638e+00	-1.733988e+00	-1.878728e+00	
25%	-8.428553e-01	-6.668235e-01	-5.543333e-01	-7.095058e-01	-8.545719e-01	-9.047796e-01	-8.383905e-01	-7.258140e-01	
50%	3.725025e-02	-4.275897e-01	-1.369043e-02	-7.729542e-02	-4.333325e-02	9.458657e-02	9.791652e-02	-1.493571e-01	
75%	8.419182e-01	6.920248e-01	6.933041e-01	5.881892e-01	6.056577e-01	8.084196e-01	8.815648e-01	5.918018e-01	
max	2.250087e+00	2.759005e+00	2.315233e+00	2.584643e+00	2.877126e+00	2.521619e+00	1.919644e+00	2.485875e+00	

- 2) Lakukan PCA dengan import library **PCA** terlebih dahulu lalu transform data yang telah distandarisasi menjadi **2** komponen saja. Hitung variance yang terwakilkan oleh kedua komponen tersebut dengan menjumlahkan kedua variance ratio tiap komponen.

```
In [12]: 1 from sklearn.decomposition import PCA
2
3 pca_2 = PCA(2)
4 pca_2_result = pca_2.fit_transform(data_cluster)
5 print('Cumulative variance of 2 principal components: {:.2%}'.format(np.sum(pca_2.explained_variance_ratio_)))
```

Cumulative variance of 2 principal components: 59.34%

- 3) Visualisasikan sebaran data yang sudah didekomposisi menjadi 2 komponen menggunakan **scatterplot**. Gunakan library **Seaborn** agar grafik yang ditampilkan lebih rapih.



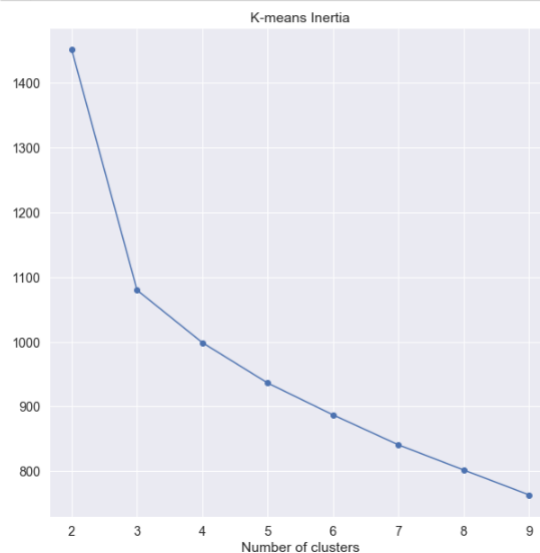
## D. Elbow Method & K-Means Clustering

- 1) Import algoritma **K-Means Clustering** dari Scikit-learn, lalu hitung pergantian kluster dari data ke dalam list inertia. Gunakan nilai **K** dari **2 sampai 10** dengan menggunakan **for** loop.

```
In [14]: 1 import sklearn.cluster as cluster
2
3 inertia = []
4 for i in range(2,10):
5     kmeans = cluster.KMeans(n_clusters=i,
6                             init='k-means++',
7                             n_init=15,
8                             max_iter=500,
9                             random_state=17)
10    kmeans.fit(data_cluster)
11    inertia.append(kmeans.inertia_)
```

- 2) Visualisasikan hasil inertia tiap K ke dalam grafik.

```
In [22]: 1 sns.set(rc={'figure.figsize':(10,10)},font_scale=1.3)
2
3 plt.plot(range(2,len(inertia)+2), inertia, marker='o')
4 plt.xlabel('Number of clusters')
5 plt.title('K-means Inertia')
6 plt.grid(True)
```



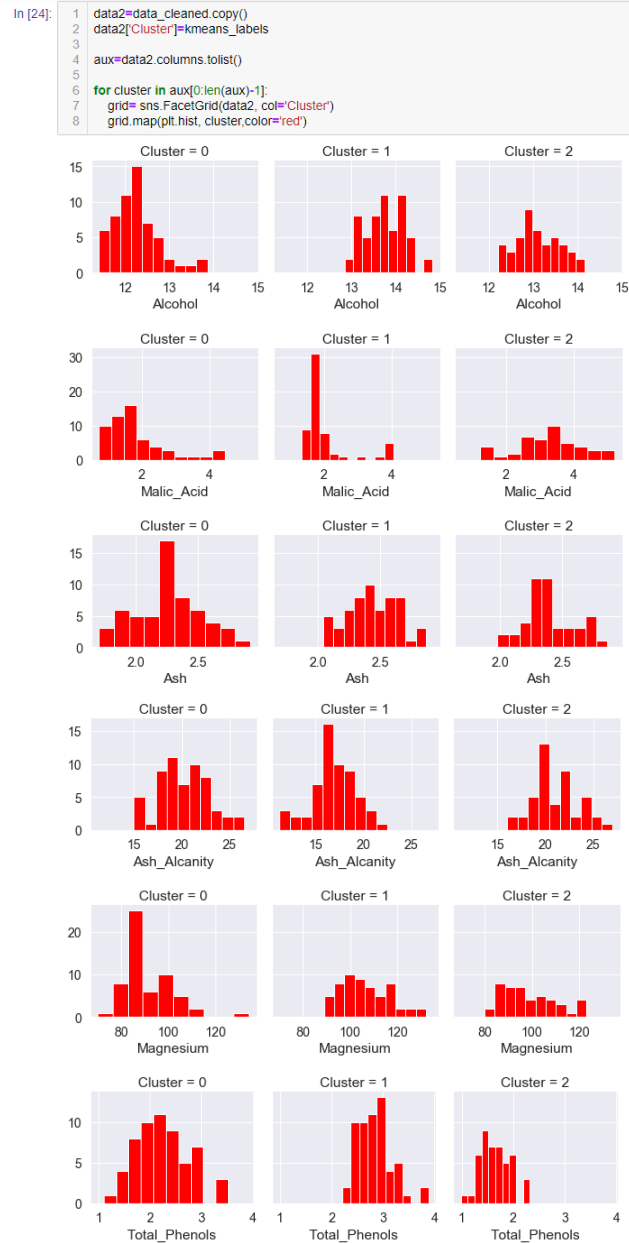
- 3) Kelompokkan data ke dalam K kluster sesuai dengan hasil **Elbow Method**. Pada kasus ini, nilai K yang digunakan adalah K=3. Hitung jumlah anggota setiap kluster dan tentukan letak **centroid** (pusat kluster) untuk tiap kluster.

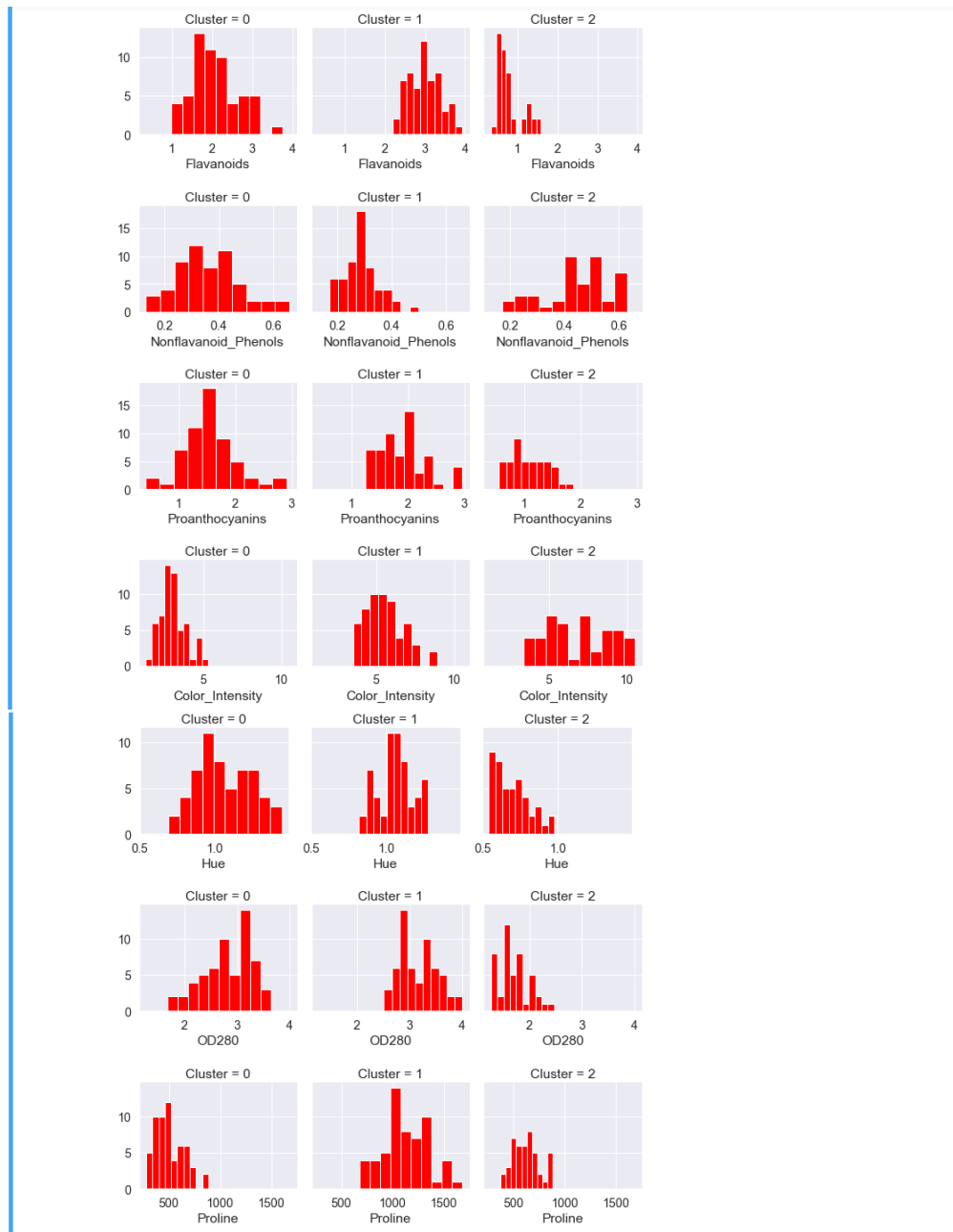
```
In [17]: 1 kmeans = cluster.KMeans(n_clusters=3,random_state=17,init='k-means++')
2 kmeans_labels = kmeans.fit_predict(data_cluster)
3
4 centroids = kmeans.cluster_centers_
5 centroids_pca = pca_2.transform(centroids)
6
7 pd.Series(kmeans_labels).value_counts()
```

```
Out[17]: 1 58
0 58
2 45
dtype: int64
```

## E. Visualization

- 1) Visualisasikan data sesuai dengan klasternya. Copy data\_cleaned ke dalam dataframe baru, lalu tambahkan kolom 'Cluster'. Isi kolom Cluster dengan label klaster tiap data.
- 2) Visualisasikan karakteristik setiap klaster. Gunakan **FacetGrid** pada library Seaborn untuk mempermudah proses visualisasi.





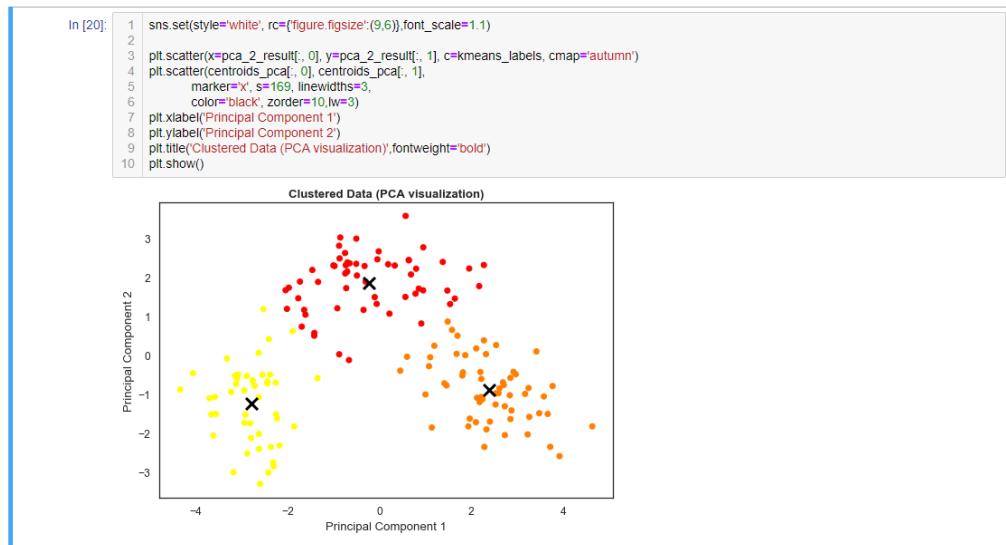
3) Tampilkan karakteristik centroid dengan melakukan **inverse transform** pada standard scaler.

```
In [19]: 1 centroids_data=pd.DataFrame(data=std_scaler.inverse_transform(centroids), columns=data.columns)
          2 centroids_data.head()
```

Out[19]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_I
0	12.277069	1.845345	2.259310	20.187931	91.086207	2.241034	2.067414	0.359310	1.560862	
1	13.756724	2.010000	2.442414	16.900000	106.034483	2.843793	2.987586	0.286897	1.898966	
2	13.100889	3.284222	2.395333	21.006667	98.466667	1.640667	0.796889	0.448444	1.079333	

- 4) Visualisasikan sebaran data pada dua principal component untuk melihat hasil klusterisasi. Tampilkan sebaran data dengan warna yang berbeda sesuai dengan label klusternya. Tampilkan juga centroid agar kluster terlihat jelas.



#### 4. Pertanyaan

- 1) Apa yang dimaksud dengan standar deviasi? Apa hubungan antara besar standar deviasi dan variance data?
- 2) Apa itu skewness? Apa makna dari angka skewness tersebut dan hubungannya dengan angka yang positif dan negatif?
- 3) Hitung jumlah outliers yang berada pada data jika menggunakan metode IQR!
- 4) Apa yang membedakan standarisasi dan normalisasi?
- 5) Apa yang terjadi jika standar deviasi pada data diubah menjadi 1?
- 6) Jika dekomposisi principal component diubah menjadi 3 principal component, berapa total variance ratio yang diwakilkan ketiga komponen tersebut?
- 7) Apa yang dimaksud dengan inertia pada algoritma K-Means Clustering? Apakah ada metric untuk menyatakan bahwa nilai K yang dipilih merupakan K yang cocok untuk algoritma ini?
- 8) Buatlah suatu analisis clustering dengan menggunakan PCA yang memiliki 3 principal component! Visualisasikan hasil clustering dan karakteristik tiap kluster. Kerjakan seluruh pertanyaan di atas dan analisis pada file notebook baru, lalu upload file tersebut pada GitHub pribadi di folder **Modul 3 - Clustering & Principal Component Analysis**.

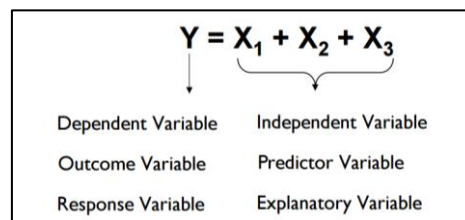
## Data Analysis : Linear Regression

### 1. Tujuan

- Dapat melakukan Exploratory Data Analysis (EDA) yang divisualisasikan dengan library Seaborn.
- Dapat melakukan Train-Test-Split lalu membentuk model prediksi linear regression.
- Dapat mengevaluasi model prediksi yang dibentuk menggunakan residual, mean square error, dan residual plot.

### 2. Penjelasan Singkat

**Linear regression** adalah metode yang sering digunakan dalam statistika untuk melakukan analisis yang sifatnya prediktif. Ide utama dari regresi adalah untuk melihat seberapa penting peran suatu **variabel bebas** terhadap **variabel terikat**.



Dalam membahas machine learning, linear regression sering juga disebut **linear model**. Linear regression merupakan algoritma jenis **supervised**, sehingga diperlukan variabel **target** dalam bekerja. Dalam pembentukan model, dilakukan proses **fitting** untuk mencari hubungan linear yang paling mendekati hubungan antara variabel bebas dan variabel terikat. Proses fitting dari linear regression menggunakan **Least Square Method**. Metode ini membentuk suatu garis linear yang memiliki jarak dengan data sampai terbentuk garis dengan square error paling kecil sehingga bisa menghasilkan suatu persamaan yang menggambarkan relasi antara variabel bebas dan variabel terikat. Secara matematika, Least Square Method dirumuskan sebagai berikut :

$$f(a, b) = a + b x,$$

so

$$R^2(a, b) \equiv \sum_{i=1}^n [y_i - (a + b x_i)]^2$$

$$\frac{\partial(R^2)}{\partial a} = -2 \sum_{i=1}^n [y_i - (a + b x_i)] = 0$$

$$\frac{\partial(R^2)}{\partial b} = -2 \sum_{i=1}^n [y_i - (a + b x_i)] x_i = 0.$$

These lead to the equations

$$n a + b \sum_{i=1}^n x_i = \sum_{i=1}^n y_i$$

$$a \sum_{i=1}^n x_i + b \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i.$$

$$b = \frac{\text{cov}(x, y)}{\sigma_x^2}$$

$$a = \bar{y} - b \bar{x}.$$

Dalam pembentukan model, dataset yang digunakan akan dibagi menjadi dua, yaitu **dataset untuk model training** dan **dataset untuk model testing**. Walaupun tidak ada ketetapan dalam ratio pembagian dataset, namun perlu diingat agar membagi dataset training lebih besar dibandingkan dataset testing agar hasil prediksi lebih akurat. Setelah dataset dibagi, maka langkah selanjutnya adalah melakukan training model dengan dataset training. Model yang terbentuk kemudian digunakan untuk melakukan prediksi pada dataset test. Hasil prediksi tersebut dibandingkan dengan nilai sesungguhnya dari target sehingga terlihat perbandingannya. Untuk menilai performa model linear regression, maka digunakan metric **mean square error** dan **residual**. Metric residual dapat divisualisasikan menggunakan **residual plot** sehingga terlihat tren dari residual itu sendiri.

### 3. Langkah Kerja

#### A. Importing Library & Reading Dataset

- 1) Buka Jupyter Notebook, buat notebook baru dengan bahasa pemrograman Python 3 lalu import library berikut:
  - a. pandas
  - b. numpy
  - c. seaborn
  - d. matplotlib.pyplot
  - e. sklearn
- 2) Import dataset dari kumpulan datasets yang tersedia di Scikit-learn. Pilih library Boston House Prices Dataset dengan melakukan import **load\_boston**. Tampilkan deskripsi dataset dengan menggunakan function **DESCR**.

```
In [1]: 1 import numpy as np
        2 import pandas as pd
        3 import seaborn as sns
        4 import matplotlib.pyplot as plt
        5 import sklearn
        6
        7 from sklearn.datasets import load_boston
        8 boston = load_boston()
        9 print(boston.DESCR)

..._boston_dataset:
Boston house prices dataset
-----

**Data Set Characteristics:**

: Number of Instances: 506

: Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

: Attribute Information (in order):
- CRIM    per capita crime rate by town
- ZN      proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS   proportion of non-retail business acres per town
- CHAS    Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX     nitric oxides concentration (parts per 10 million)
- RM      average number of rooms per dwelling
- AGE     proportion of owner-occupied units built prior to 1940
- DIS     weighted average distance to five Boston urban centers
```

- Ubah data yang terkandung dalam dataset tersebut (masih dalam format text) menjadi pandas dataframe. Ubah nama kolom dari dataframe menjadi nama kolom yang sebenarnya menggunakan function **feature\_names** lalu tampilkan sebagian dataset.

```
In [2]: 1 data = pd.DataFrame(boston.data)
        2 data.columns = boston.feature_names
        3 data.head()
```

```
Out[2]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

- Tambahkan kolom PRICE pada dataframe sebelumnya dengan menggunakan function **target**. Kolom PRICE adalah variabel target dari regresi.

```
In [3]: 1 data['PRICE'] = boston.target
        2 data.head()
```

```
Out[3]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

## B. Exploratory Data Analysis

- EDA dapat dimulai dengan melihat datatype setiap kolom pada dataset. Gunakan method **info()** pada library pandas untuk menampilkan datatype setiap kolom dan menampilkan jumlah value non-NULL.

```
In [4]: 1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
CRIM      506 non-null float64
ZN        506 non-null float64
INDUS     506 non-null float64
CHAS      506 non-null float64
NOX       506 non-null float64
RM        506 non-null float64
AGE       506 non-null float64
DIS       506 non-null float64
RAD       506 non-null float64
TAX       506 non-null float64
PTRATIO   506 non-null float64
B         506 non-null float64
LSTAT     506 non-null float64
PRICE     506 non-null float64
dtypes: float64(14)
memory usage: 55.4 KB
```

- Periksa ukuran dataset dengan menggunakan property **shape**. Property shape akan menampilkan ukuran dataset. Cocokkan jumlah non-NULL untuk setiap kolom dengan ukuran dataset.

```
In [5]: 1 data.shape
```

```
Out[5]: (506, 14)
```



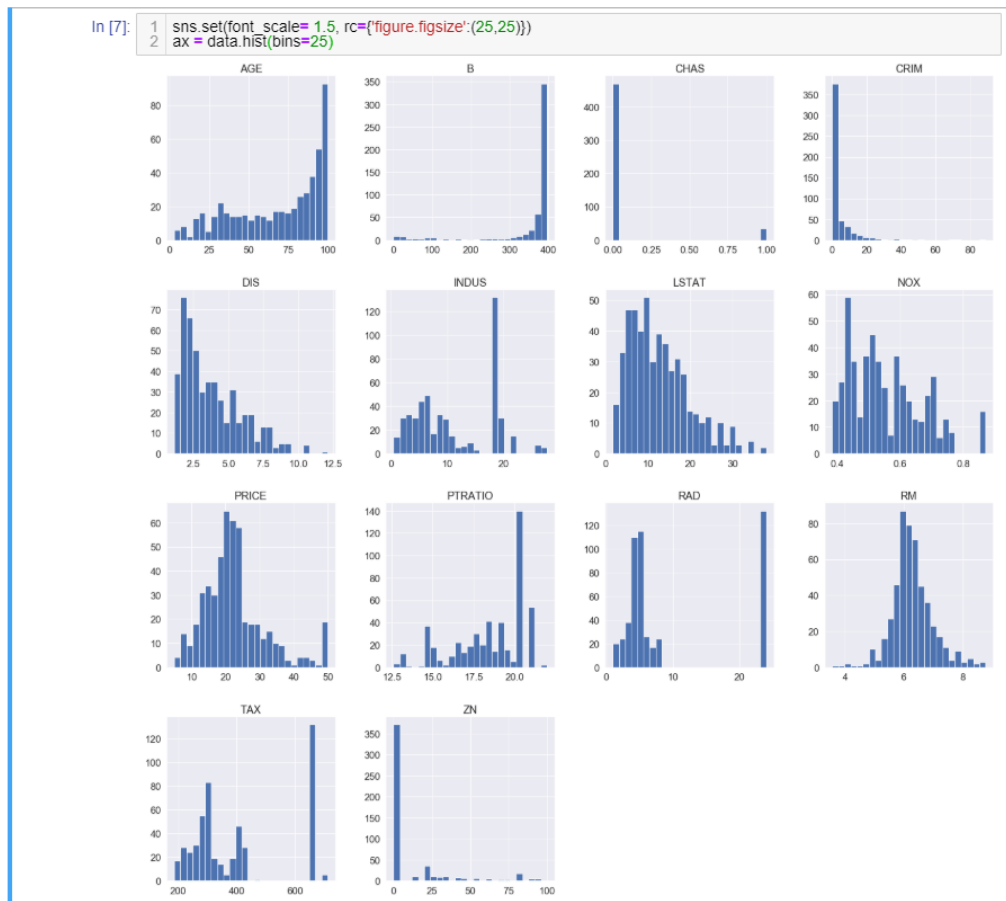
- 3) Karena tidak ada NULL pada dataset, maka null imputation tidak dilakukan dan proses dilanjutkan dengan menampilkan ringkasan informasi statistik setiap kolom dengan menggunakan method **describe()**.

```
In [6]: 1 data.describe()
```

Out[6]:

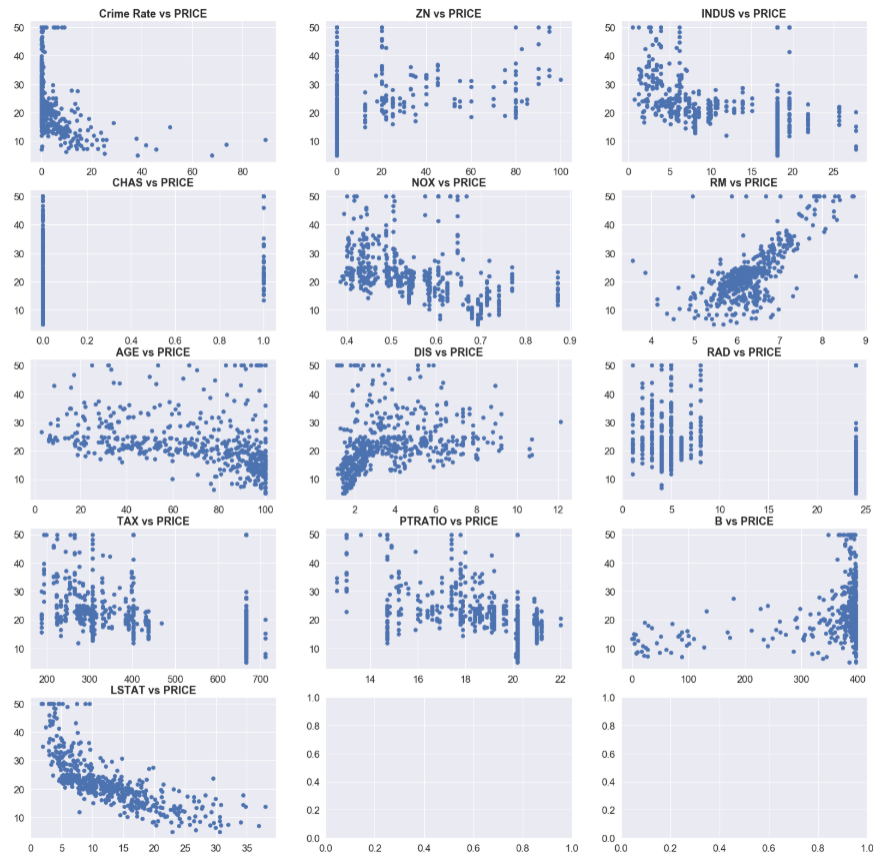
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.45553
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.16494
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000

- 4) Visualisasikan sebaran data dengan histogram untuk melihat distribusinya menggunakan method **hist()**. Gunakan juga library Seaborn agar penampilan grafik lebih rapih.



5) Visualisasikan hubungan antara setiap variabel bebas dengan variabel terikat (PRICE) menggunakan **scatterplot**.

```
In [8]: 1 fig, axes = plt.subplots(5,3)
2
3 # Crime Rate vs PRICE
4 axes[0,0].scatter(data['CRIM'], data['PRICE'])
5 axes[0,0].set_title('Crime Rate vs PRICE', fontweight = 'bold')
6
7 # ZN vs PRICE
8 axes[0,1].scatter(data['ZN'], data['PRICE'])
9 axes[0,1].set_title('ZN vs PRICE', fontweight = 'bold')
10
11 # INDUS vs PRICE
12 axes[0,2].scatter(data['INDUS'], data['PRICE'])
13 axes[0,2].set_title('INDUS vs PRICE', fontweight = 'bold')
14
15 # CHAS vs PRICE
16 axes[1,0].scatter(data['CHAS'], data['PRICE'])
17 axes[1,0].set_title('CHAS vs PRICE', fontweight = 'bold')
18
19 # NOX vs PRICE
20 axes[1,1].scatter(data['NOX'], data['PRICE'])
21 axes[1,1].set_title('NOX vs PRICE', fontweight = 'bold')
22
23 # RM vs PRICE
24 axes[1,2].scatter(data['RM'], data['PRICE'])
25 axes[1,2].set_title('RM vs PRICE', fontweight = 'bold')
26
27 # AGE vs PRICE
28 axes[2,0].scatter(data['AGE'], data['PRICE'])
29 axes[2,0].set_title('AGE vs PRICE', fontweight = 'bold')
30
31 # DIS vs PRICE
32 axes[2,1].scatter(data['DIS'], data['PRICE'])
33 axes[2,1].set_title('DIS vs PRICE', fontweight = 'bold')
34
35 # RAD vs PRICE
36 axes[2,2].scatter(data['RAD'], data['PRICE'])
37 axes[2,2].set_title('RAD vs PRICE', fontweight = 'bold')
38
39 # TAX vs PRICE
40 axes[3,0].scatter(data['TAX'], data['PRICE'])
41 axes[3,0].set_title('TAX vs PRICE', fontweight = 'bold')
42
43 # PTRATIO vs PRICE
44 axes[3,1].scatter(data['PTRATIO'], data['PRICE'])
45 axes[3,1].set_title('PTRATIO vs PRICE', fontweight = 'bold')
46
47 # B vs PRICE
48 axes[3,2].scatter(data['B'], data['PRICE'])
49 axes[3,2].set_title('B vs PRICE', fontweight = 'bold')
50
51 # LSTAT vs PRICE
52 axes[4,0].scatter(data['LSTAT'], data['PRICE'])
53 axes[4,0].set_title('LSTAT vs PRICE', fontweight = 'bold')
54
55 plt.show()
```



### C. Train-Test Split

- 1) Lakukan Train-Test Split untuk dataset dengan ratio Train : Test = 0,7 : 0,3. Gunakan fitur **random state** agar hasil split benar – benar random.

a. Note :

- Splitting dilakukan menjadi 4 dataframe: X\_train, X\_test, Y\_train, dan Y\_test. Dataframe X\_train dan Y\_train digunakan untuk fitting / melatih model regressor sedangkan X\_test dan Y\_test digunakan untuk evaluasi model.
- X\_train dan X\_test berisikan 13 variabel bebas sedangkan Y\_train dan Y\_test berisikan variabel terikat (PRICE).

```
In [10]: 1 X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(
2         data.drop("PRICE", axis=1), data["PRICE"], test_size=0.3, random_state=5)
3
4 print(X_train.shape)
5 print(X_test.shape)
6 print(Y_train.shape)
7 print(Y_test.shape)

(354, 13)
(152, 13)
(354,)
(152,)
```

### D. Linear Regression

- 1) Import algoritma Linear Regression dari Scikit-learn. Lakukan fitting / training model dengan menggunakan dataframe **X\_train** dan **Y\_train**.
- 2) Setelah training model selesai, lakukan prediksi dengan model tersebut menggunakan dataframe X\_train dan X\_test lalu masukkan hasil prediksi ke dalam list.

```
In [11]: 1 from sklearn.linear_model import LinearRegression
2 model = LinearRegression()
3 model.fit(X_train, Y_train)
4
5 pred_train = model.predict(X_train)
6 pred_test = model.predict(X_test)
```

- 3) Tampilkan koefisien dari model dengan menggunakan function **coef\_**. Tampilkan koefisien dari setiap variabel bebas ke dalam dataframe.

```
In [12]: 1 coef = pd.DataFrame(zip(X_train.columns, model.coef_), columns=["Features", "Coefficients"])
2 coef
```

```
Out[12]:
```

	Features	Coefficients
0	CRIM	-0.154486
1	ZN	0.041395
2	INDUS	-0.025377
3	CHAS	0.786055
4	NOX	-12.936584
5	RM	4.039523
6	AGE	-0.010535
7	DIS	-1.334986
8	RAD	0.318273
9	TAX	-0.012643
10	PTRATIO	-0.977288
11	B	0.012671
12	LSTAT	-0.462052

- 4) Tampilkan hasil prediksi dari  $X_{test}$  lalu bandingkan dengan nilai  $Y_{test}$ . Tampilkan keduanya pada dataframe yang sama untuk membandingkan.

```
In [13]: 1 diff = pd.DataFrame(zip(pred_test, Y_test), columns = ['Predicted Prize', 'Real Prize'])
          2 diff
```

Out[13]:

	Predicted Prize	Real Prize
0	37.389977	37.6
1	31.567942	27.9
2	27.133739	22.6
3	6.551176	13.8
4	33.693108	35.2
5	5.549194	10.4
6	27.100056	23.9
7	29.829810	29.0
8	26.446224	22.8
9	22.388735	23.2

## E. Evaluation

- 1) Visualisasikan hasil prediksi model pada sumbu X dan nilai sesungguhnya ( $Y_{test}$ ) dalam satu grafik. Gunakan function **regplot** pada library Seaborn untuk melihat perbandingan antara Predicted Price dan Real Price. Jika model linear regression bekerja dengan baik, maka data akan berkumpul sekitar garis.

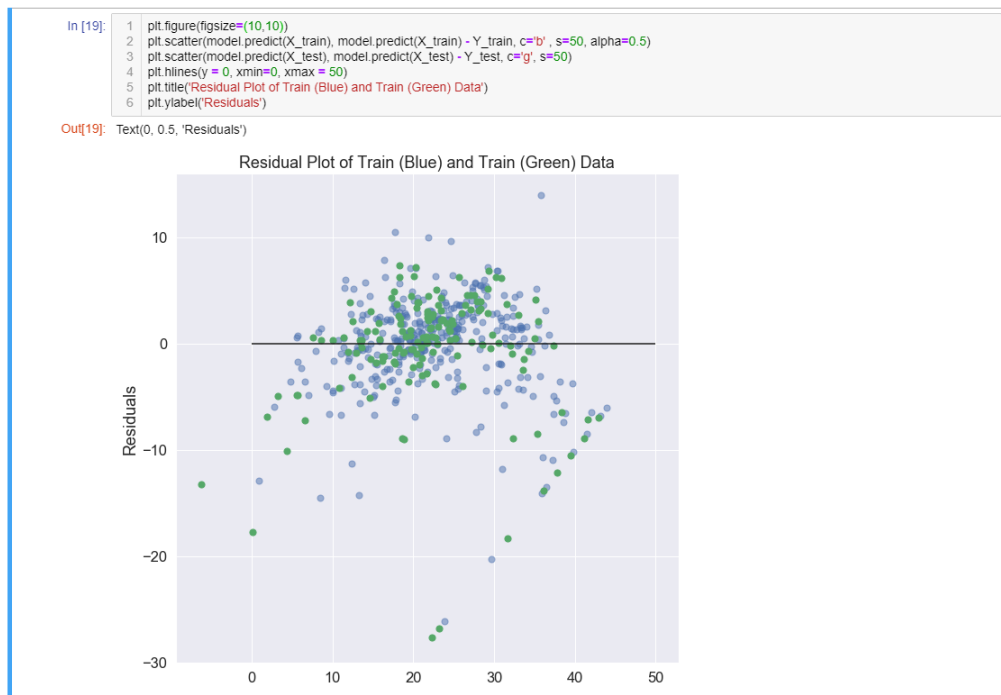


- 2) Hitung residual dari prediksi. Setelah itu, hitung **Mean Square Error** dari prediksi menggunakan rumus yang ada pada penjelasan singkat.

```
In [18]: 1 residu = np.sum((Y_test - pred_test) ** 2)
          2 mse = np.mean((Y_test - pred_test) ** 2)
          3 print(mse, residu)
```

30.697037704088608 4665.949731021471

- 3) Buat **residual plot** dari model dengan menggunakan **scatter plot** dan garis lurus ditengahnya.



#### 4. Pertanyaan

- 1) Dari seluruh variabel bebas, mana saja yang merupakan binary variable?
- 2) Variabel bebas mana yang mempengaruhi variabel terikat secara signifikan? Mengapa?
- 3) Apa maksud dari nilai koefisien yang positif dan negatif?
- 4) Apa itu Sum Residual dan Mean Square Error? Apa makna dari nilai yang ditampilkan?
- 5) Apa itu residual plot? Residual plot seperti apa yang dikatakan bagus?
- 6) Buatlah suatu analisis regresi baru dengan melakukan data cleansing untuk data yang tidak sesuai syarat IQR! Visualisasikan hasil analisis dan bandingkan dengan analisis tanpa melakukan data cleansing! Bandingkan juga metric evaluasi kedua model dan residual plot-nya. Kerjakan seluruh pertanyaan di atas dan analisis pada file notebook baru, lalu upload file tersebut pada GitHub pribadi di folder **Modul 4 - Linear Regression**.

## Data Analysis : Decision Tree & Random Forest

### 1. Tujuan

- Dapat melakukan Exploratory Data Analysis (EDA) yang divisualisasikan dengan library Seaborn.
- Melakukan encoding menggunakan one-hot encoding untuk categorical variable.
- Dapat membentuk model prediksi menggunakan algoritma Decision Tree dan Random Forest.
- Dapat mengevaluasi model prediksi yang dibentuk menggunakan confusion matrix.

### 2. Penjelasan Singkat

**Decision tree** adalah algoritma supervised learning yang dapat digunakan untuk melakukan klasifikasi dan prediksi. Algoritma ini melakukan klasifikasi sesuai dengan percabangan dan setiap percabangan diatur oleh rules terkait suatu variabel. Semakin banyak variabel yang dipertimbangkan, maka percabangan dari decision tree akan semakin banyak. Oleh karena banyaknya percabangan yang muncul, maka percabangan tersebut dibatasi dengan parameter **max\_depth** pada algoritma. Dengan adanya parameter tersebut, algoritma decision tree hanya akan mengambil variabel yang paling berperan dalam klasifikasi. Urutan pengambilan variabel sebagai cabang didasarkan oleh **gini index** variabel tersebut. Variabel dengan gini index paling rendah akan diambil sebagai variabel paling pertama sebagai cabang.

Gini index adalah ukuran ketidaksamaan sampel di dalam dataset. Gini index memiliki rumus sebagai berikut :

$$Gini\ index = 1 - \sum_{i=1}^n p(i)^2$$

$$p(i) = \text{probability of chosen class}$$

Di mana n adalah jumlah kelas yang terdapat pada dataset, dan p(i) adalah probabilitas kelas tersebut terpilih. Nilai gini index = 0 berarti sampel bersifat homogen, dan index = 1 sebaliknya.

Algoritma **Random Forest** termasuk ke dalam algoritma jenis **ensemble**. Algoritma jenis ini menggabungkan beberapa model yang bekerja secara independen agar menghasilkan prediksi yang lebih baik dibandingkan dengan satu model saja. Untuk prediksi akhir, maka digunakanlah **majority voting**. Random Forest akan membentuk beberapa decision tree untuk melakukan ini. Untuk menghindari korelasi yang tinggi dari setiap Decision Tree yang dibentuk, maka digunakanlah **meta-algorithm Bagging (Bootstrap Aggregating)**. Dengan menggunakan meta-algorithm ini, setiap Decision Tree pada ensemble tidak berkorelasi tinggi dan meningkatkan variance dari tiap model yang di-ensemble.

### 3. Langkah Kerja

#### A. Importing Library & Reading Dataset

- 1) Buka Jupyter Notebook, buat notebook baru dengan bahasa pemrograman Python 3 lalu import library berikut:
  - a. pandas
  - b. numpy
  - c. seaborn
  - d. matplotlib.pyplot
  - e. sklearn
- 2) Buka dataset menggunakan library pandas. Dataset yang digunakan adalah dataset **Telecom\_Churn.csv**.

```
In [1]: 1 import numpy as np
        2 import pandas as pd
        3 import matplotlib.pyplot as plt
        4 import seaborn as sns
        5 import sklearn
        6
        7 data = pd.read_csv('Telecom_Churn.csv')
```

#### B. Exploratory Data Analysis

- 1) Mulai proses EDA dengan melihat datatype dan jumlah non-null menggunakan method **info()**.

```
In [2]: 1 data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7032 entries, 0 to 7031
Data columns (total 22 columns):
Unnamed: 0      7032 non-null int64
customerID     7032 non-null object
gender         7032 non-null object
SeniorCitizen  7032 non-null int64
Partner        7032 non-null object
Dependents     7032 non-null object
tenure         7032 non-null int64
PhoneService   7032 non-null object
MultipleLines   7032 non-null object
InternetService 7032 non-null object
OnlineSecurity 7032 non-null object
OnlineBackup    7032 non-null object
DeviceProtection 7032 non-null object
TechSupport     7032 non-null object
StreamingTV     7032 non-null object
StreamingMovies 7032 non-null object
Contract        7032 non-null object
PaperlessBilling 7032 non-null object
PaymentMethod   7032 non-null object
MonthlyCharges  7032 non-null float64
TotalCharges    7032 non-null float64
Churn           7032 non-null object
dtypes: float64(2), int64(3), object(17)
memory usage: 1.2+ MB
```

- 2) Periksa ukuran dataset dengan menggunakan property **shape**. Property shape akan menampilkan ukuran dataset. Cocokkan jumlah non-NULl untuk setiap kolom dengan ukuran dataset.

```
In [3]: 1 data.shape

Out[3]: (7032, 22)
```

- 3) Tampilkan seluruh **unique value** dari tiap kolom pada dataset untuk membedakan antara variabel kategorikal dan variabel numerik lalu pisahkan antara variabel kategorikal dan variabel numerik ke dalam dataframe yang berbeda.

```
In [4]: 1 for column in data:
2         print('%a' % format(a = column, data[column].unique()))
```

Unnamed: 0 : [ 0 1 2 ... 7029 7030 7031]  
customerID : ['7590-VHVEG' '5575-GNVDE' '3668-QPYBK' ... '4801-JJAZL' '8361-LTMKD'  
'3186-AJIEK']  
gender : ['Female' 'Male']  
SeniorCitizen : [0 1]  
Partner : ['Yes' 'No']  
Dependents : ['No' 'Yes']  
tenure : [ 1 34 2 45 8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27  
5 46 11 70 63 43 15 60 18 66 9 3 31 50 64 56 7 42 35 48 29 65 38 68  
32 55 37 36 41 6 4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]  
PhoneService : ['No' 'Yes']  
MultipleLines : ['No phone service' 'No' 'Yes']  
InternetService : ['DSL' 'Fiber optic' 'No']  
OnlineSecurity : ['No' 'Yes' 'No internet service']  
OnlineBackup : ['Yes' 'No' 'No internet service']  
DeviceProtection : ['No' 'Yes' 'No internet service']  
TechSupport : ['No' 'Yes' 'No internet service']  
StreamingTV : ['No' 'Yes' 'No internet service']  
StreamingMovies : ['No' 'Yes' 'No internet service']  
Contract : ['Month-to-month' 'One year' 'Two year']  
PaperlessBilling : ['Yes' 'No']  
PaymentMethod : ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'  
'Credit card (automatic)']  
MonthlyCharges : [29.85 56.95 53.85 ... 63.1 44.2 78.7]  
TotalCharges : [ 29.85 1889.5 108.15 ... 346.45 306.6 6844.5]  
Churn : ['No' 'Yes']

```
In [5]: 1 unique = ['customerID']
2 cat = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService',
3        'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
4        'Contract', 'PaperlessBilling', 'PaymentMethod', 'Churn']
5 num = ['tenure', 'MonthlyCharges', 'TotalCharges']
6
7 df_uni = data[unique]
8 df_cat = data[cat]
9 df_num = data[num]
```

- 4) Visualisasikan sebaran data untuk **variabel numerik** dengan histogram untuk melihat distribusinya menggunakan method **hist()**. Gunakan juga library Seaborn agar penampilan grafik lebih rapih. Tampilkan juga ringkasan statistika dari variabel numerikal menggunakan method describe().



```
In [6]: 1 df_num.describe()
```

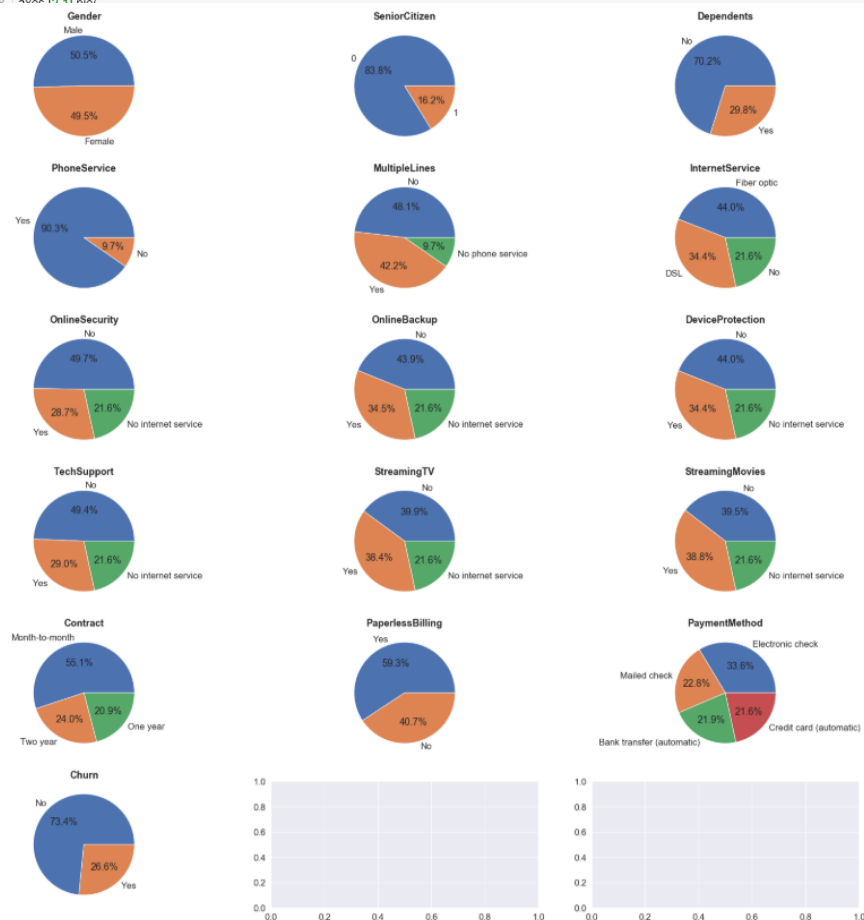
Out[6]:

	tenure	MonthlyCharges	TotalCharges
count	7032.000000	7032.000000	7032.000000
mean	32.421786	64.798208	2283.300441
std	24.545260	30.085974	2266.771362
min	1.000000	18.250000	18.800000
25%	9.000000	35.587500	401.450000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.862500	3794.737500
max	72.000000	118.750000	8684.800000



- 5) Visualisasikan sebaran data untuk **variabel kategorikal** dengan **pie chart** yang terdapat pada library plotly. Gunakan fitur **subplots** agar grafik yang dihasilkan lebih rapih.

```
In [8]: 1 fig, axes = plt.subplots(6,3, figsize=(30,30))
2
3 ##
4 axes[0,0].pie(
5     x = df_cat['gender'].value_counts().values,
6     labels = df_cat['gender'].value_counts().keys(),
7     autopct='%1.1f%%')
8 axes[0,0].set_title('Gender', fontweight = 'bold')
9
10 axes[0,1].pie(
11     x = df_cat['SeniorCitizen'].value_counts().values,
12     labels = df_cat['SeniorCitizen'].value_counts().keys(),
13     autopct='%1.1f%%')
14 axes[0,1].set_title('SeniorCitizen', fontweight = 'bold')
15
16 axes[0,2].pie(
17     x = df_cat['Dependents'].value_counts().values,
18     labels = df_cat['Dependents'].value_counts().keys(),
19     autopct='%1.1f%%')
20 axes[0,2].set_title('Dependents', fontweight = 'bold')
21
22 ##
23 axes[1,0].pie(
24     x = df_cat['PhoneService'].value_counts().values,
25     labels = df_cat['PhoneService'].value_counts().keys(),
26     autopct='%1.1f%%')
27 axes[1,0].set_title('PhoneService', fontweight = 'bold')
28
29 axes[1,1].pie(
30     x = df_cat['MultipleLines'].value_counts().values,
31     labels = df_cat['MultipleLines'].value_counts().keys(),
32     autopct='%1.1f%%')
33 axes[1,1].set_title('MultipleLines', fontweight = 'bold')
34
35 axes[1,2].pie(
36     x = df_cat['InternetService'].value_counts().values,
37     labels = df_cat['InternetService'].value_counts().keys(),
38     autopct='%1.1f%%')
39 axes[1,2].set_title('InternetService', fontweight = 'bold')
40
41 ##
42 axes[2,0].pie(
43     x = df_cat['OnlineSecurity'].value_counts().values,
44     labels = df_cat['OnlineSecurity'].value_counts().keys(),
45     autopct='%1.1f%%')
46 axes[2,0].set_title('OnlineSecurity', fontweight = 'bold')
47
48 axes[2,1].pie(
49     x = df_cat['OnlineBackup'].value_counts().values,
50     labels = df_cat['OnlineBackup'].value_counts().keys(),
51     autopct='%1.1f%%')
52 axes[2,1].set_title('OnlineBackup', fontweight = 'bold')
53
54 axes[2,2].pie(
55     x = df_cat['DeviceProtection'].value_counts().values,
56     labels = df_cat['DeviceProtection'].value_counts().keys(),
57     autopct='%1.1f%%')
58 axes[2,2].set_title('DeviceProtection', fontweight = 'bold')
59
60 ##
61 axes[3,0].pie(
62     x = df_cat['TechSupport'].value_counts().values,
63     labels = df_cat['TechSupport'].value_counts().keys(),
64     autopct='%1.1f%%')
65 axes[3,0].set_title('TechSupport', fontweight = 'bold')
66
67 axes[3,1].pie(
68     x = df_cat['StreamingTV'].value_counts().values,
69     labels = df_cat['StreamingTV'].value_counts().keys(),
70     autopct='%1.1f%%')
71 axes[3,1].set_title('StreamingTV', fontweight = 'bold')
72
73 axes[3,2].pie(
74     x = df_cat['StreamingMovies'].value_counts().values,
75     labels = df_cat['StreamingMovies'].value_counts().keys(),
76     autopct='%1.1f%%')
77 axes[3,2].set_title('StreamingMovies', fontweight = 'bold')
78
79 ##
80 axes[4,0].pie(
81     x = df_cat['Contract'].value_counts().values,
82     labels = df_cat['Contract'].value_counts().keys(),
83     autopct='%1.1f%%')
84 axes[4,0].set_title('Contract', fontweight = 'bold')
85
86 axes[4,1].pie(
87     x = df_cat['PaperlessBilling'].value_counts().values,
88     labels = df_cat['PaperlessBilling'].value_counts().keys(),
89     autopct='%1.1f%%')
90 axes[4,1].set_title('PaperlessBilling', fontweight = 'bold')
91
92 axes[4,2].pie(
93     x = df_cat['PaymentMethod'].value_counts().values,
94     labels = df_cat['PaymentMethod'].value_counts().keys(),
95     autopct='%1.1f%%')
96 axes[4,2].set_title('PaymentMethod', fontweight = 'bold')
97
98 ##
99 axes[5,0].pie(
100     x = df_cat['Churn'].value_counts().values,
101     labels = df_cat['Churn'].value_counts().keys(),
102     autopct='%1.1f%%')
103 axes[5,0].set_title('Churn', fontweight = 'bold')
104
105 axes[5,1].pie(
106     x = df_cat['Churn'].value_counts().values,
107     labels = df_cat['Churn'].value_counts().keys(),
108     autopct='%1.1f%%')
109 axes[5,1].set_title('Churn', fontweight = 'bold')
110
111 axes[5,2].pie(
112     x = df_cat['Churn'].value_counts().values,
113     labels = df_cat['Churn'].value_counts().keys(),
114     autopct='%1.1f%%')
115 axes[5,2].set_title('Churn', fontweight = 'bold')
```

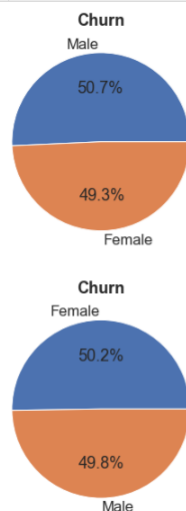


- 6) Pisahkan dataset menjadi dua, yaitu untuk pelanggan yang melakukan churn dan tidak melakukan churn (no\_churn). Setelah dipisahkan, perlihatkan distribusi data tiap kolom untuk dataset **churn** dan **no\_churn**.

```
In [9]: 1 churn = data[data['Churn'] == 'No']
        2 no_churn = data[data['Churn'] == 'Yes']

In [10]: 1 def churn_nochurn (column_name):
        2     fig, axes = plt.subplots(2, figsize = (10,10))
        3     axes[0].pie(
        4         x=list(churn[column_name].value_counts().values),
        5         labels = list(churn[column_name].value_counts().keys()),
        6         autopct='%1.1f%%')
        7     axes[0].set_title('Churn', fontweight = 'bold')
        8
        9     axes[1].pie(
        10        x=list(no_churn[column_name].value_counts().values),
        11        labels = list(no_churn[column_name].value_counts().keys()),
        12        autopct='%1.1f%%')
        13    axes[1].set_title('Churn', fontweight = 'bold')
```

```
In [11]: 1 churn_nochurn('gender')
```



### C. Preprocessing

- 1) Lakukan standarisasi untuk variabel numerikal menggunakan StandardScaler. **Standarisasi dilakukan pada dataframe numerikal.**

```
In [12]: 1 from sklearn.preprocessing import OneHotEncoder
        2 from sklearn.preprocessing import StandardScaler

In [13]: 1 ss = StandardScaler()
        2 scaled_num = pd.DataFrame(ss.fit_transform(df_num), columns = num)
```

- 2) Gunakan method **describe()** untuk melihat apakah standarisasi berhasil. Pastikan mean = 0 dan standar deviasi = 1.

```
In [14]: 1 scaled_num.describe()

Out[14]:
```

	tenure	MonthlyCharges	TotalCharges
count	7.032000e+03	7.032000e+03	7.032000e+03
mean	-1.214741e-16	9.652878e-17	-1.172113e-16
std	1.000071e+00	1.000071e+00	1.000071e+00
min	-1.280248e+00	-1.547283e+00	-9.990692e-01
25%	-9.542963e-01	-9.709769e-01	-8.302488e-01
50%	-1.394171e-01	1.845440e-01	-3.908151e-01
75%	9.199259e-01	8.331482e-01	6.668271e-01
max	1.612573e+00	1.793381e+00	2.824261e+00

- 3) Lakukan encoding untuk variabel kategorikal menggunakan **OneHotEncoder**. Lakukan encoding ini pada dataframe kategorikal. Variabel terikat (Churn) tidak perlu di-encode karena variabel ini adalah variabel target. Tampilkan sebagian data menggunakan method **head()** untuk melihat apakah encoding berhasil.

```
In [15]: 1 encoder = OneHotEncoder()
2 encoded_cat = pd.DataFrame(encoder.fit_transform(df_cat).toarray(), columns = encoder.get_feature_names())
3 encoded_cat.drop(columns = {
4     'x16_No', 'x16_Yes'}, inplace = True)
5 encoded_cat.head()
```

```
Out[15]:
```

	x0_Female	x0_Male	x1_0	x1_1	x2_No	x2_Yes	x3_No	x3_Yes	x4_No	x4_Yes	...	x12_Yes	x13_Month- to-month	x13_One year	x13_...
0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	...	0.0	1.0	0.0	
1	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	...	0.0	0.0	1.0	
2	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	...	0.0	1.0	0.0	
3	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	1.0	
4	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	...	0.0	1.0	0.0	

5 rows x 43 columns

- 4) Satukan dataframe numerikal dan kategorikal menjadi satu dataset menggunakan method **merge()** dari library pandas. Ubah value dalam kolom Churn dari string menjadi binary (Yes = 1, No = 0).

```
In [16]: 1 cleaned = encoded_cat.merge(scaled_num, left_index=True, right_index=True, how = "left")
2 cleaned["Churn"] = data["Churn"]
3 cleaned["Churn"].replace(['Yes': 1, 'No': 0], inplace = True)
4 cleaned.head()
```

```
Out[16]:
```

	x0_Female	x0_Male	x1_0	x1_1	x2_No	x2_Yes	x3_No	x3_Yes	x4_No	x4_Yes	...	x14_No	x14_Yes	x15_Bank transfer (automatic)	x15_C auton
0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	...	0.0	1.0	0.0	
1	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	...	1.0	0.0	0.0	
2	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	...	0.0	1.0	0.0	
3	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	...	1.0	0.0	1.0	
4	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	...	0.0	1.0	0.0	

5 rows x 47 columns

- 5) Lakukan Train-Test Split untuk dataset dengan ratio Train : Test = 0,7 : 0,3. Gunakan fitur **random state** agar hasil split benar – benar random.

a. Note :

- Splitting dilakukan menjadi 4 dataframe: X\_train, X\_test, Y\_train, dan Y\_test. Dataframe X\_train dan Y\_train digunakan untuk fitting / melatih model regressor sedangkan X\_test dan Y\_test digunakan untuk evaluasi model.
- X\_train dan X\_test berisikan 46 variabel bebas sedangkan Y\_train dan Y\_test berisikan variabel terikat (Churn).

```
In [17]: 1 from sklearn.model_selection import train_test_split
2 X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(
3     cleaned.drop("Churn", axis = 1), cleaned["Churn"], test_size=0.3, random_state = 5)
4
5 print(X_train.shape)
6 print(X_test.shape)
7 print(Y_train.shape)
8 print(Y_test.shape)
```

```
(4922, 46)
(2110, 46)
(4922,)
(2110,)
```

## D. Decision Tree

- 1) Import algoritma Decision Tree dari Scikit-learn. Lakukan fitting / training model dengan menggunakan dataframe **X\_train** dan **Y\_train**. Perhatikan parameter **max\_depth** pada algoritma ini. **Max\_depth** akan membatasi percabangan dari pohon keputusan. Membiarkan **max\_depth** sesuai default akan menjadikan pohon keputusan membagi keputusan sejumlah sampel sehingga terjadi **overfitting**.

Setelah training model selesai, lakukan prediksi dengan model tersebut menggunakan dataframe **X\_train** dan **X\_test** lalu masukkan hasil prediksi ke dalam list.

```
In [18]: 1 from sklearn import tree
2 model = tree.DecisionTreeClassifier(max_depth=2)
3 model.fit(X_train, Y_train)
4
5 dt_pred_train = model.predict(X_train)
6 dt_pred_test = model.predict(X_test)
```

- 2) Tampilkan hasil prediksi dari **X\_test** lalu bandingkan dengan nilai **Y\_test**. Tampilkan keduanya pada dataframe yang sama untuk membandingkan.

```
In [19]: 1 dt_pred = pd.DataFrame(zip(dt_pred_train, Y_train), columns = ['Prediction', 'Actual'])
2 dt_pred
```

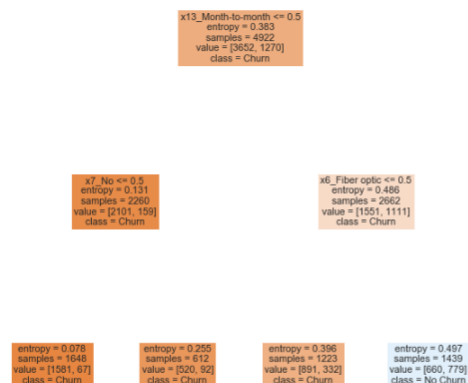
Out[19]:

	Prediction	Actual
0	1	1
1	0	1
2	0	0
3	1	0
4	0	1
5	0	0
6	0	0
7	1	1
8	0	0
9	0	0

## E. Evaluation – Decision Tree

- 1) Lakukan evaluasi model decision tree dengan menampilkan pohon keputusan terlebih dahulu. Gunakan function **plot\_tree** dari library Scikit-learn.

```
In [20]: 1 tree.plot_tree(model,
2 filled=True,
3 feature_names = list(X_train.columns),
4 class_names=['Churn', 'No Churn'],)
```



- 2) Lakukan evaluasi prediksi decision tree menggunakan **evaluation metrics**. Tampilkan **accuracy**, **precision**, **recall**, **f1-score**, dan **support** dengan menggunakan function **classification\_report** dari Scikit-learn.

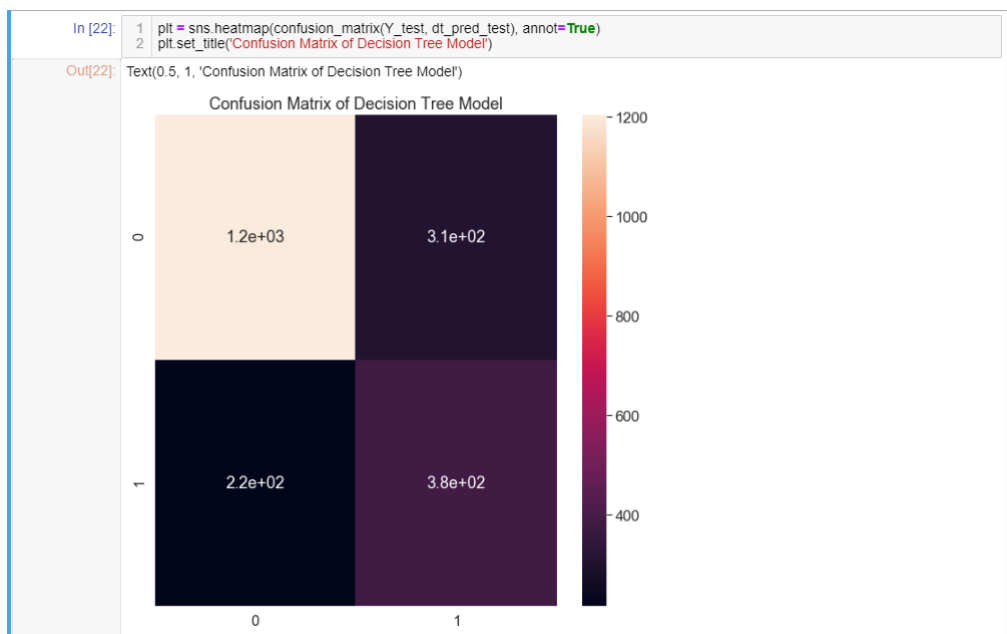
```
In [21]: 1 from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
2 from sklearn import metrics
3 print("\n Classification report : \n", classification_report(Y_test, dt_pred_test))
4 print("Accuracy Score : ", accuracy_score(Y_test, dt_pred_test))
```

Classification report :  

	precision	recall	f1-score	support
0	0.85	0.80	0.82	1511
1	0.56	0.64	0.59	599
accuracy			0.75	2110
macro avg	0.70	0.72	0.71	2110
weighted avg	0.77	0.75	0.76	2110

Accuracy Score : 0.75260663507109

- 3) Lakukan evaluasi prediksi decision tree menggunakan **confusion matrix**. Plot confusion matrix menggunakan **heatmap** dari library Seaborn.



## F. Random Forest

- 1) Import algoritma Random Forest dari Scikit-learn. Lakukan fitting / training model dengan menggunakan dataframe **X\_train** dan **Y\_train**. Perhatikan parameter **max\_depth** dan **n\_estimators** pada algoritma ini. Setelah training model selesai, lakukan prediksi dengan model tersebut menggunakan dataframe **X\_train** dan **X\_test** lalu masukkan hasil prediksi ke dalam list.

```
In [23]: 1 from sklearn.ensemble import RandomForestClassifier
2
3 rf = RandomForestClassifier(n_estimators=20, random_state=1, max_depth=3)
4 rf.fit(X_train, Y_train)
5 rf_pred_train = rf.predict(X_train)
6 rf_pred_test = rf.predict(X_test)
```

- 2) Tampilkan hasil prediksi dari X\_test lalu bandingkan dengan nilai Y\_test. Tampilkan keduanya pada dataframe yang sama untuk membandingkan.

```
In [24]: 1 rf_pred = pd.DataFrame(zip(rf_pred_train, Y_train), columns = ['Prediction', 'Actual'])
         2 rf_pred

Out[24]:
```

	Prediction	Actual
0	0	1
1	0	1
2	0	0
3	0	0
4	1	1
5	0	0
6	0	0
7	0	1
8	0	0
9	0	0

## E. Evaluation – Random Forest

- 1) Lakukan evaluasi prediksi random forest menggunakan **evaluation metrics**. Tampilkan **accuracy, precision, recall, f1-score, dan support** dengan menggunakan function **classification\_report** dari Scikit-learn.

```
In [25]: 1 print("\n Classification report : \n", classification_report(Y_test, rf_pred_test))
         2 print("Accuracy Score : ", accuracy_score(Y_test, rf_pred_test))

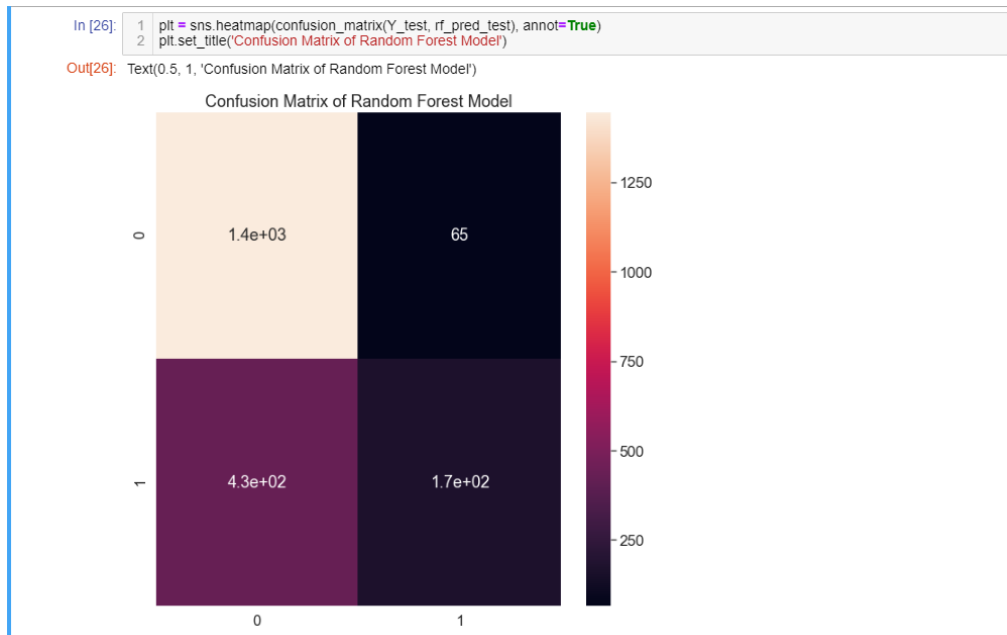
Classification report :
      precision    recall  f1-score   support

    0       0.77       0.96       0.85       1511
    1       0.72       0.28       0.40         599

 accuracy          0.76       2110
 macro avg       0.75       0.62       0.63       2110
 weighted avg    0.76       0.76       0.73       2110

Accuracy Score : 0.7649289099526067
```

- 2) Lakukan evaluasi prediksi random forest menggunakan **confusion matrix**. Plot confusion matrix menggunakan **heatmap** dari library Seaborn.



#### 4. Pertanyaan

- 1) Apa itu One Hot Encoder? Encoder jenis apa saja yang tersedia dalam library Scikit-learn dan apa kelebihan dan kekurangan dari masing – masing encoder?
- 2) Apa itu entropy pada Decision Tree? Apa yang membedakan entropy dan gini index?
- 3) Apa itu meta-algorithm Bagging? Bagaimana cara kerjanya dan mengapa algoritma itu bisa sangat powerful untuk ensemble learning?
- 4) Apa itu overfitting? Apa yang membedakan overfitting dengan underfitting?
- 5) Apa saja metric yang terkandung pada classification report? Jelaskan masing – masing metric tersebut!
- 6) Apa saja angka yang ditampilkan pada confusion matrix? Apa maksud dari setiap angka tersebut?

Buatlah suatu analisis baru (menggunakan Decision Tree dan Random Forest) dengan melakukan data cleansing untuk data yang tidak sesuai syarat IQR (untuk numerical variable)! Lakukan juga data cleansing dengan membuang kolom (untuk categorical variable) yang dirasa tidak relevan dengan kemungkinan pengguna melakukan churn! Visualisasikan hasil analisis dan bandingkan dengan analisis tanpa melakukan data cleansing! Bandingkan juga metric evaluasi kedua analisis.

Kerjakan seluruh pertanyaan di atas dan analisis pada file notebook baru, lalu upload file tersebut pada GitHub pribadi di folder **Modul 5 - Decision Tree & Random Forest**.

## Model Deployment : Flask

### 1. Tujuan

- Dapat melakukan Exploratory Data Analysis (EDA) yang divisualisasikan dengan library Seaborn.
- Dapat melakukan Train-Test-Split lalu membentuk model prediksi lalu menyimpannya dalam bentuk pickle.
- Dapat mengevaluasi model prediksi yang dibentuk menggunakan evaluation metrics.
- Dapat melakukan model deployment menggunakan Flask.

### 2. Penjelasan Singkat

Tahap terakhir dalam membuat suatu model machine learning adalah **model deployment**. Model deployment adalah tahap mengintegrasikan model yang sudah dibuat dengan aplikasi bisnis yang sudah ada. Dengan adanya model deployment, model prediksi yang telah dibentuk bisa menjadi lebih praktis ketika digunakan. Model deployment dapat dilakukan pada berbagai aplikasi yang ada. Contoh paling mudah melakukan model deployment adalah dengan mengaplikasikannya menjadi **web application**. Web application yang dibuat menggunakan modul ini akan berbasis **Flask**. Flask merupakan micro web framework yang diprogram menggunakan Python 3.

Agar suatu model machine learning dapat diintegrasikan menjadi web application, maka model tersebut harus diubah menjadi bentuk **pickle**. Pickling adalah proses mengubah suatu objek pemrograman pada Python menjadi bit-stream. Dengan mengubahnya menjadi bit-stream, maka objek tersebut dapat disimpan atau disalurkan di dalam jaringan. Proses mengubah objek menjadi bit-stream disebut **serializing**, sedangkan proses sebaliknya disebut dengan **deserializing**.

Untuk mengintegrasikan model menjadi web application, maka perlu dilakukan sedikit pemrograman menggunakan HTML di bagian akhir.

### 3. Langkah Kerja

#### A. Importing Library & Reading Dataset

- 1) Buka Jupyter Notebook, buat notebook baru dengan bahasa pemrograman Python 3 lalu import library berikut:
  - a. pandas
  - b. numpy
  - c. seaborn
  - d. matplotlib.pyplot
  - e. sklearn

```
In [1]: 1 import numpy as np
        2 import pandas as pd
        3 import matplotlib.pyplot as plt
        4 import seaborn as sns
        5 import sklearn
        6
        7 data = pd.read_csv('Iris.csv')
```



- 2) Buka dataset **Iris.csv** yang terdapat pada folder **Modul 6** menggunakan pandas lalu tampilkan sebagian dataset menggunakan method **head()**.

```
In [2]: 1 data.head()
Out[2]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

- 3) Buang kolom **Id** dari dataset.

```
In [3]: 1 data = data.drop(columns='Id')
```

## B. Exploratory Data Analysis

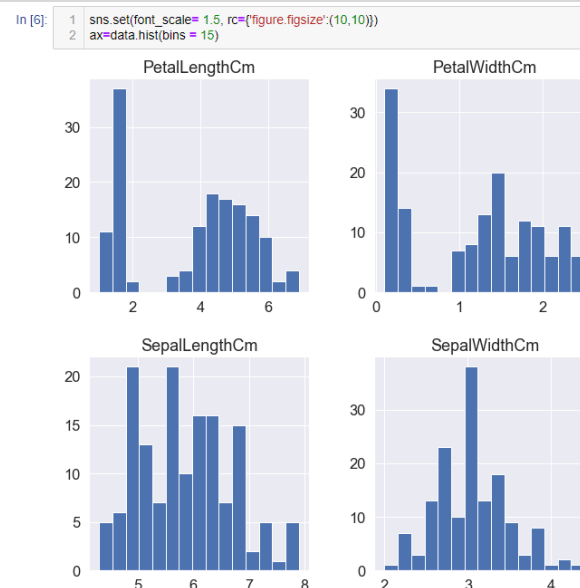
- 1) EDA dapat dimulai dengan melihat datatype setiap kolom pada dataset. Gunakan method **info()** pada library pandas untuk menampilkan datatype setiap kolom dan menampilkan jumlah value non-NULL.

```
In [4]: 1 data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
Sepal.LengthCm    150 non-null float64
Sepal.WidthCm     150 non-null float64
Petal.LengthCm    150 non-null float64
Petal.WidthCm     150 non-null float64
Species           150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

- 2) Periksa ukuran dataset dengan menggunakan property **shape**. Property shape akan menampilkan ukuran dataset. Cocokkan jumlah non-NULL untuk setiap kolom dengan ukuran dataset.

```
In [5]: 1 data.shape
Out[5]: (150, 5)
```

- 3) Visualisasikan sebaran data dengan histogram untuk melihat distribusinya menggunakan method **hist()**. Gunakan juga library Seaborn agar penampilan grafik lebih rapih.



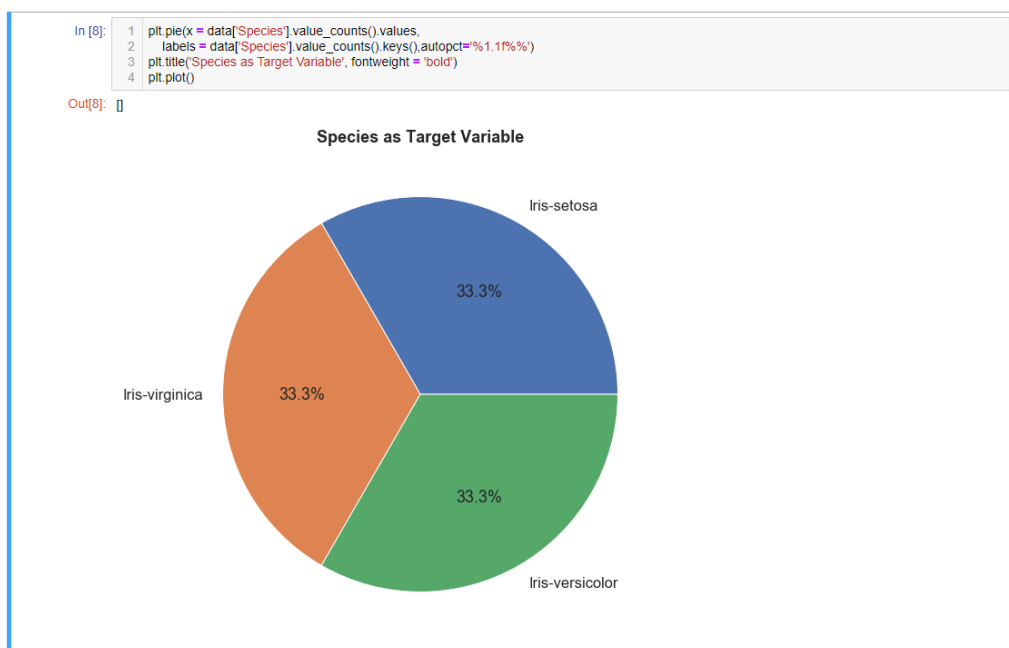
- 4) Karena tidak ada NULL pada dataset, maka null imputation tidak dilakukan dan proses dilanjutkan dengan menampilkan ringkasan informasi statistik setiap kolom dengan menggunakan method **describe()**.

```
In [7]: 1 data.describe()
```

Out[7]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

- 5) Visualisasikan persentase jumlah data tiap spesies dengan menggunakan pie-chart.

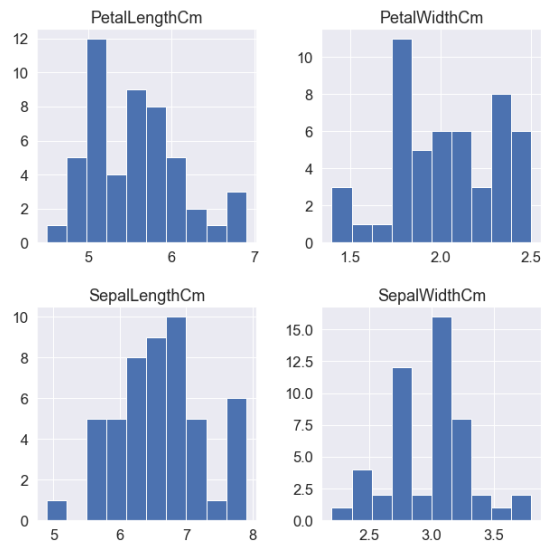


- 6) Pisahkan dataset sesuai dengan spesiesnya lalu visualisasikan karakteristik setiap spesies menggunakan histogram dan ringkasan statistiknya.

### Spesies : Irisi Virginia

```
In [9]: 1 virginica = data[data['Species'] == 'Iris-virginica']
2 versicolor = data[data['Species'] == 'Iris-versicolor']
3 setosa = data[data['Species'] == 'Iris-setosa']
```

```
In [10]: 1 sns.set(font_scale=1.5, rc={'figure.figsize':(10,10)})
2 ax=virginica.hist(bins = 10)
```

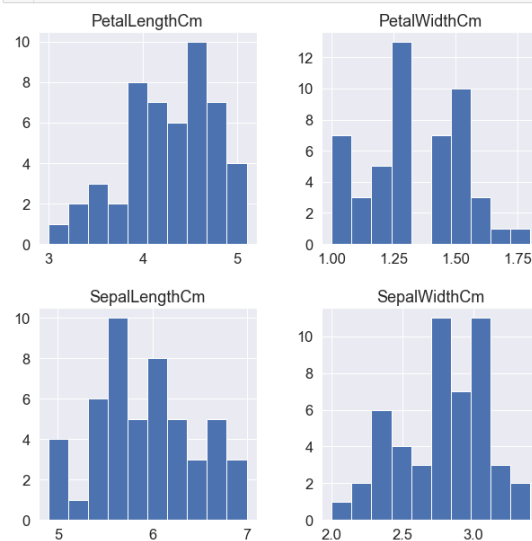


```
In [11]: 1 virginica.describe()
```

```
Out[11]:
```

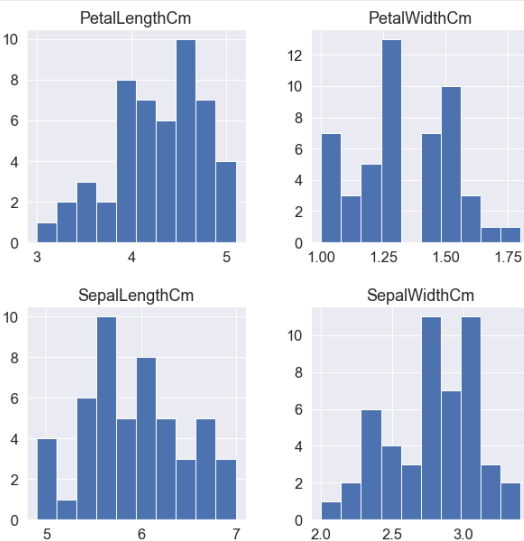
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.00000	50.00000	50.00000	50.00000
mean	6.58800	2.97400	5.55200	2.02600
std	0.63588	0.322497	0.551895	0.27465
min	4.90000	2.20000	4.50000	1.40000
25%	6.22500	2.80000	5.10000	1.80000
50%	6.50000	3.00000	5.55000	2.00000
75%	6.90000	3.17500	5.87500	2.30000
max	7.90000	3.80000	6.90000	2.50000

```
In [12]: 1 sns.set(font_scale=1.5, rc={'figure.figsize':(10,10)})
2 ax=versicolor.hist(bins = 10)
```



## Species : Iris Versicolor

```
In [12]: 1 sns.set(font_scale=1.5, rc={"figure.figsize":(10,10)})
2 ax=versicolor.hist(bins=10)
```



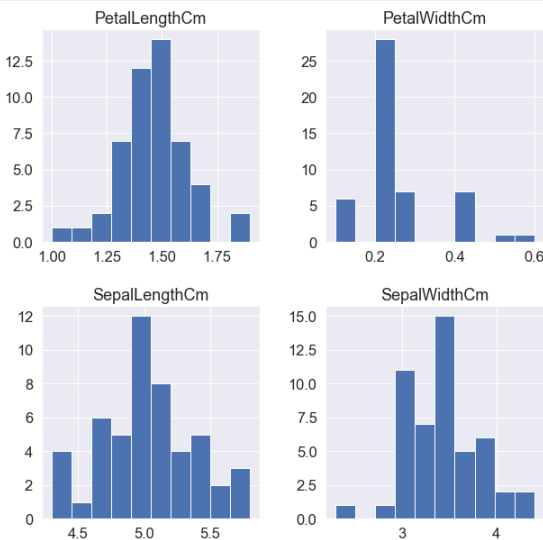
```
In [13]: 1 versicolor.describe()
```

Out[13]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.000000	50.000000	50.000000	50.000000
mean	5.936000	2.770000	4.260000	1.326000
std	0.516171	0.313798	0.469911	0.197753
min	4.900000	2.000000	3.000000	1.000000
25%	5.600000	2.525000	4.000000	1.200000
50%	5.900000	2.800000	4.350000	1.300000
75%	6.300000	3.000000	4.600000	1.500000
max	7.000000	3.400000	5.100000	1.800000

## Species : Iris Setosa

```
In [14]: 1 sns.set(font_scale=1.5, rc={"figure.figsize":(10,10)})
2 ax=setosa.hist(bins=10)
```



In [15]:

1 setosa.describe()

Out[15]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.00000	50.000000	50.000000	50.00000
mean	5.00600	3.418000	1.464000	0.24400
std	0.35249	0.381024	0.173511	0.10721
min	4.30000	2.300000	1.000000	0.10000
25%	4.80000	3.125000	1.400000	0.20000
50%	5.00000	3.400000	1.500000	0.20000
75%	5.20000	3.675000	1.575000	0.30000
max	5.80000	4.400000	1.900000	0.60000

### C. Preprocessing

- 1) Gunakan **standard scaler** untuk melakukan standarisasi kolom numerikal. Buang terlebih dahulu kolom target.

In [16]:

```
1 from sklearn.preprocessing import StandardScaler
```

In [17]:

```
1 target = data['Species']
2 data = data.drop (columns='Species')
```

In [18]:

```
1 ss = StandardScaler()
2 scaled = pd.DataFrame (ss.fit_transform(data), columns = data.columns)
```

In [19]:

```
1 scaled.describe()
```

Out[19]:

	SepalLengthCm	SepalWidthCm	Petal.LengthCm	Petal.WidthCm
count	1.500000e+02	1.500000e+02	1.500000e+02	1.500000e+02
mean	-2.775558e-16	-5.140333e-16	1.154632e-16	9.251859e-16
std	1.003350e+00	1.003350e+00	1.003350e+00	1.003350e+00
min	-1.870024e+00	-2.438987e+00	-1.568735e+00	-1.444450e+00
25%	-9.006812e-01	-5.877635e-01	-1.227541e+00	-1.181504e+00
50%	-5.250608e-02	-1.249576e-01	3.362659e-01	1.332259e-01
75%	6.745011e-01	5.692513e-01	7.627586e-01	7.905908e-01
max	2.492019e+00	3.114684e+00	1.786341e+00	1.710902e+00

- 2) Lakukan Train-Test Split untuk dataset dengan ratio Train : Test = 0,7 : 0,3. Gunakan fitur **random state** agar hasil split benar – benar random.

#### a. Note :

- Splitting dilakukan menjadi 4 dataframe: X\_train, X\_test, Y\_train, dan Y\_test. Dataframe X\_train dan Y\_train digunakan untuk fitting / melatih model decision tree sedangkan X\_test dan Y\_test digunakan untuk evaluasi model.
- X\_train dan X\_test berisikan 4 variabel bebas sedangkan Y\_train dan Y\_test berisikan variabel target (Species).

In [21]:	1	from sklearn.model_selection import train_test_split
	2	X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(
	3	data.drop('Species', axis = 1), data['Species'], test_size=0.3, random_state = 5)
	4	
	5	print(X_train.shape)
	6	print(X_test.shape)
	7	print(Y_train.shape)
	8	print(Y_test.shape)
		(105, 4)
		(45, 4)
		(105,)
		(45,)

#### D. Decision Tree

- 1) Import algoritma Decision Tree dari Scikit-learn. Lakukan fitting / training model dengan menggunakan dataframe **X\_train** dan **Y\_train**. Perhatikan parameter **max\_depth** pada algoritma ini. Max\_depth akan membatasi percabangan dari pohon keputusan. Membiarkan max\_depth sesuai default akan menjadikan pohon keputusan membagi keputusan sejumlah sampel sehingga terjadi **overfitting**.

Setelah training model selesai, lakukan prediksi dengan model tersebut menggunakan dataframe X\_train dan X\_test lalu masukkan hasil prediksi ke dalam list.

```
In [18]: 1 from sklearn import tree
          2 model = tree.DecisionTreeClassifier(max_depth=2)
          3 model.fit(X_train, Y_train)
          4
          5 dt_pred_train = model.predict(X_train)
          6 dt_pred_test = model.predict(X_test)
```

- 2) Tampilkan hasil prediksi dari X\_test lalu bandingkan dengan nilai Y\_test. Tampilkan keduanya pada dataframe yang sama untuk membandingkan.

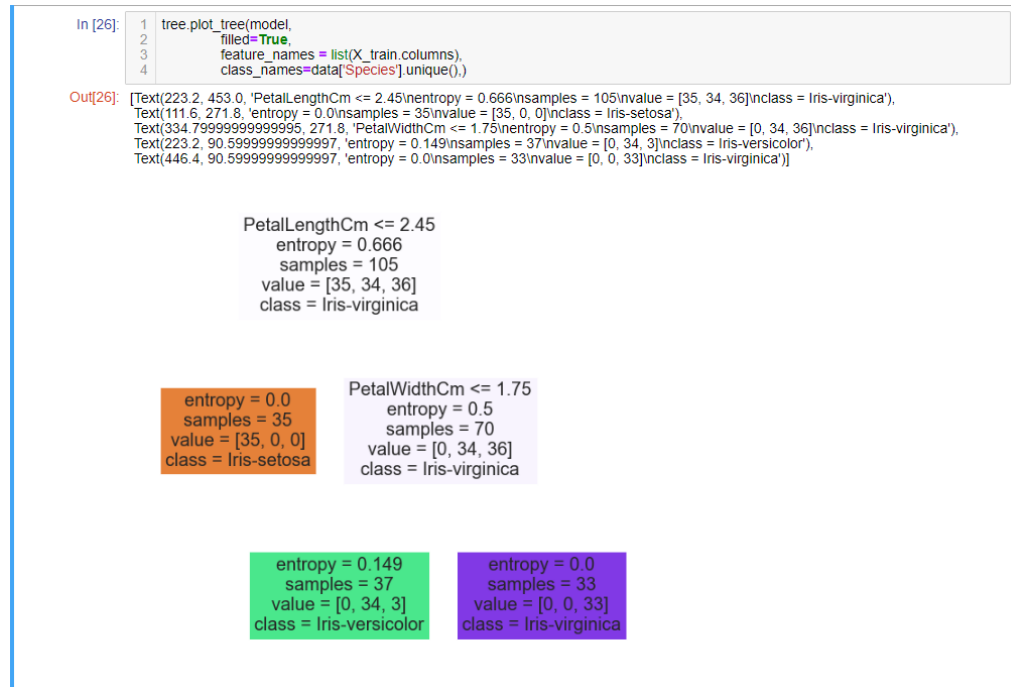
```
In [23]: 1 dt_pred = pd.DataFrame(zip(dt_pred_train, Y_train), columns = ['Prediction', 'Actual'])
          2 dt_pred
```

Out[23]:

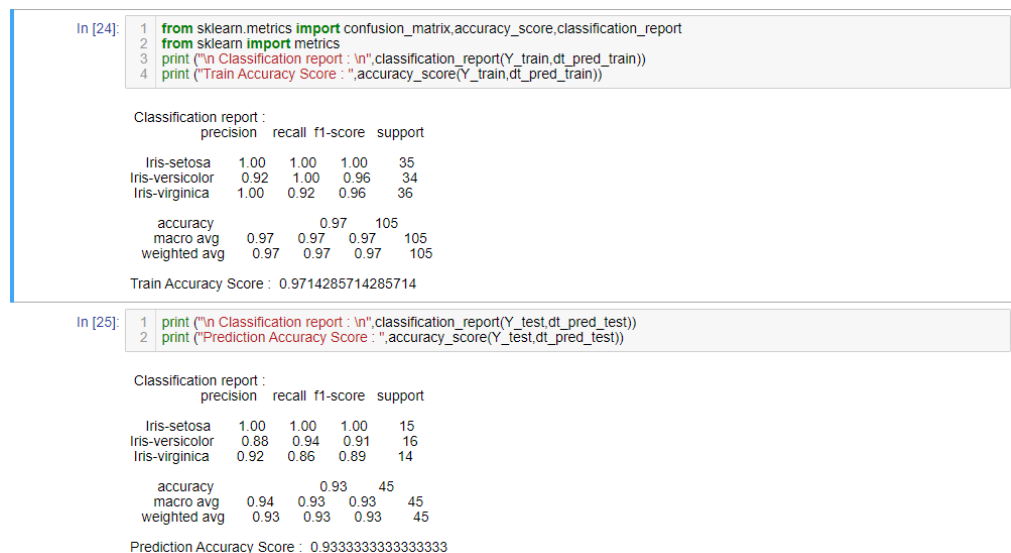
	Prediction	Actual
0	Iris-virginica	Iris-virginica
1	Iris-versicolor	Iris-versicolor
2	Iris-virginica	Iris-virginica
3	Iris-virginica	Iris-virginica
4	Iris-setosa	Iris-setosa
5	Iris-virginica	Iris-virginica
6	Iris-setosa	Iris-setosa
7	Iris-setosa	Iris-setosa
8	Iris-setosa	Iris-setosa
9	Iris-setosa	Iris-setosa

## Evaluation – Decision Tree

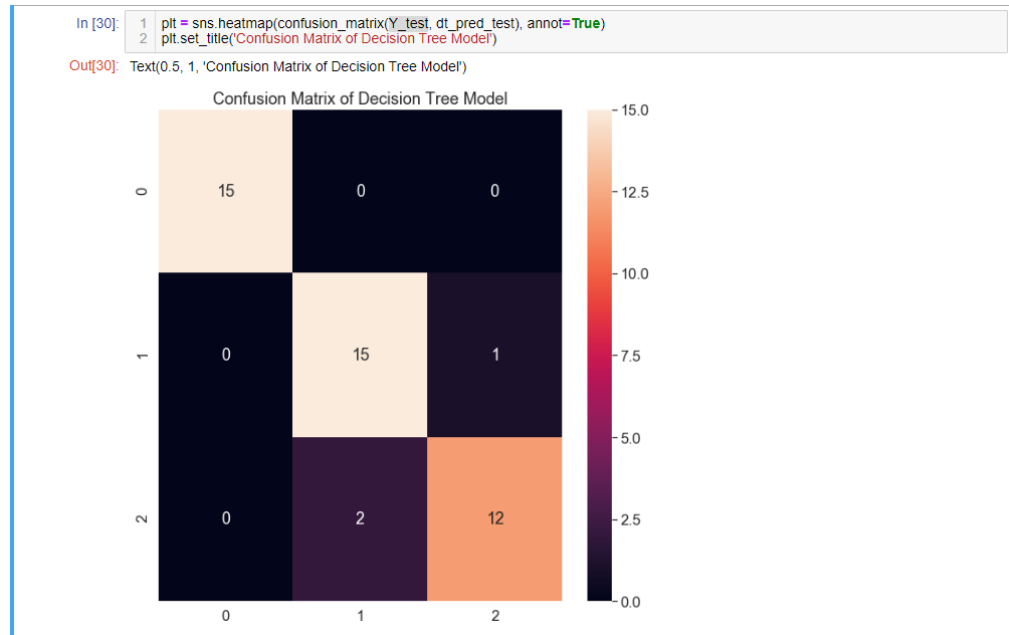
- 1) Lakukan evaluasi model decision tree dengan menampilkan pohon keputusan terlebih dahulu. Gunakan function **plot\_tree** dari library Scikit-learn.



- 2) Lakukan evaluasi prediksi decision tree menggunakan **evaluation metrics**. Tampilkan **accuracy**, **precision**, **recall**, **f1-score**, dan **support** dengan menggunakan function **classification\_report** dari Scikit-learn.



- 3) Lakukan evaluasi prediksi decision tree menggunakan **confusion matrix**. Plot confusion matrix menggunakan **heatmap** dari library Seaborn.



## E. Pickling

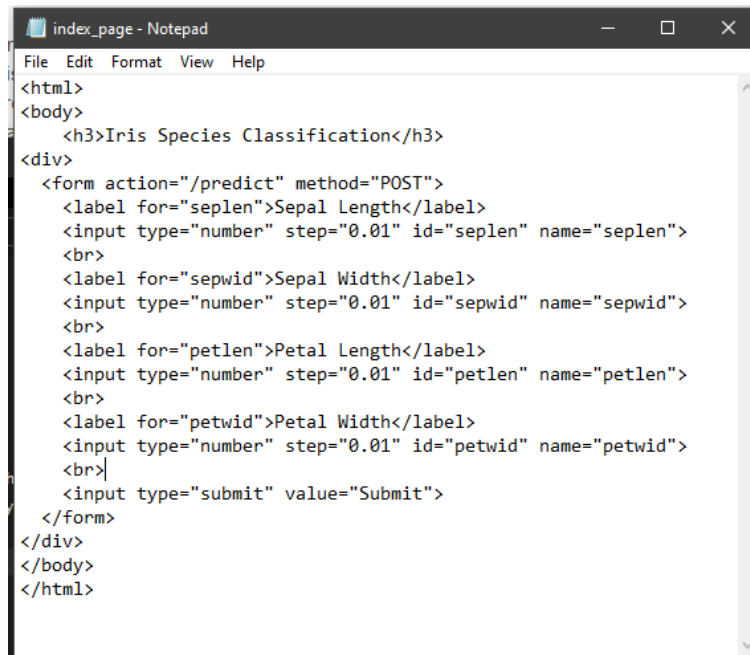
- 1) Lakukan pickling dengan import library **pickle** terlebih dahulu. Tentukan nama file untuk model yang akan di-serialize. Extension untuk file pickle adalah **.pkl**. Lakukan pickling menggunakan function **dump()**. Isi parameter obj dengan model yang akan di-serialize, lalu isi parameter file dengan function **open()**. Parameter 'wb' pada function open() menandakan file yang disimpan bersifat open for **writing (w)** dan **binary (b)**.

```
In [27]: 1 import pickle
         2 filename = 'decision_tree.pkl'
         3 pickle.dump(model, open(filename, 'wb'))
```



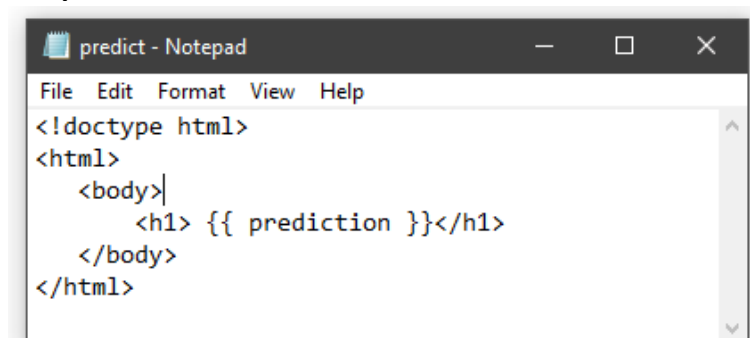
## F. HTML : Index Page & Prediction Result Page

- 1) Buat sebuah folder pada directory pengerjaan model dengan nama folder **templates**.
- 2) Bukalah notepad pada komputer lalu ketikkan program berikut. Setelah selesai, simpan program dengan nama file **index\_page.html** pada folder **templates**.



```
<html>
<body>
  <h3>Iris Species Classification</h3>
  <div>
    <form action="/predict" method="POST">
      <label for="seplen">Sepal Length</label>
      <input type="number" step="0.01" id="seplen" name="seplen">
      <br>
      <label for="sepwid">Sepal Width</label>
      <input type="number" step="0.01" id="sepwid" name="sepwid">
      <br>
      <label for="petlen">Petal Length</label>
      <input type="number" step="0.01" id="petlen" name="petlen">
      <br>
      <label for="petwid">Petal Width</label>
      <input type="number" step="0.01" id="petwid" name="petwid">
      <br>
      <input type="submit" value="Submit">
    </form>
  </div>
</body>
</html>
```

- 3) Bukalah notepad pada komputer lalu ketikkan program berikut. Setelah selesai, simpan program dengan nama file **prediction.html** pada folder **templates**.



```
<!doctype html>
<html>
  <body>
    <h1> {{ prediction }}</h1>
  </body>
</html>
```

## G. Model Deployment : Flask

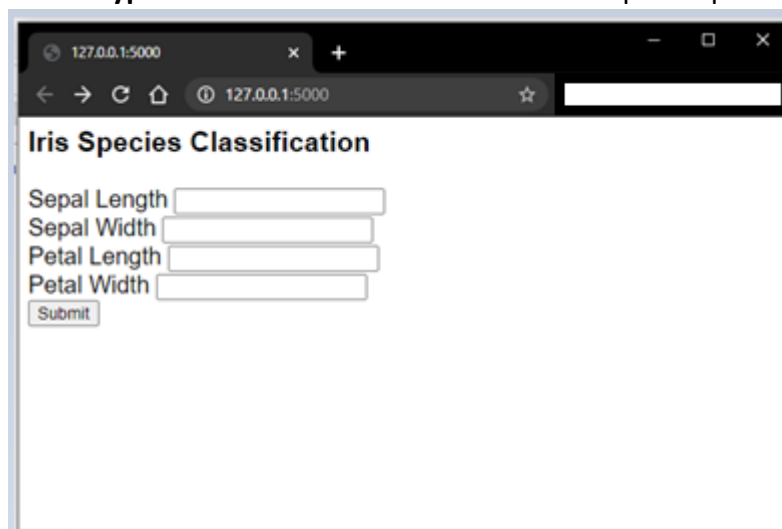
- 1) Buatlah sebuah file notebook baru pada directory yang sama dengan pickle dan templates HTML.
- 2) Import libraries berikut pada notebook :
  - a. os
  - b. pandas
  - c. numpy
  - d. pickle
  - e. flask
- 3) Ketikkan kode berikut untuk menjalankan web apps berbasis **Flask**.

```
In [ ]: 1 import os
2 import pandas as pd
3 import numpy as np
4 import flask
5 import pickle
6 from flask import Flask, render_template, request
7
8 app=Flask(__name__)
9 @app.route("/")
10
11 def index():
12     return flask.render_template("index_page.html")
13
14 def ValuePredictor(to_predict_list):
15     to_predict = np.array(to_predict_list).reshape(1,4)
16     loaded_model = pickle.load(open("decision_tree.pkl","rb"))
17     result = loaded_model.predict(to_predict)
18     return result[0]
19
20 @app.route("/predict",methods = ['POST'])
21 def result():
22     if request.method == "POST":
23         to_predict_list = request.form.to_dict()
24         to_predict_list=list(to_predict_list.values())
25         to_predict_list = list(map(float,to_predict_list))
26         result = ValuePredictor(to_predict_list)
27         prediction = str(result)
28         return render_template("predict.html",prediction=prediction)
29
30 if __name__ == '__main__':
31     app.run()
```

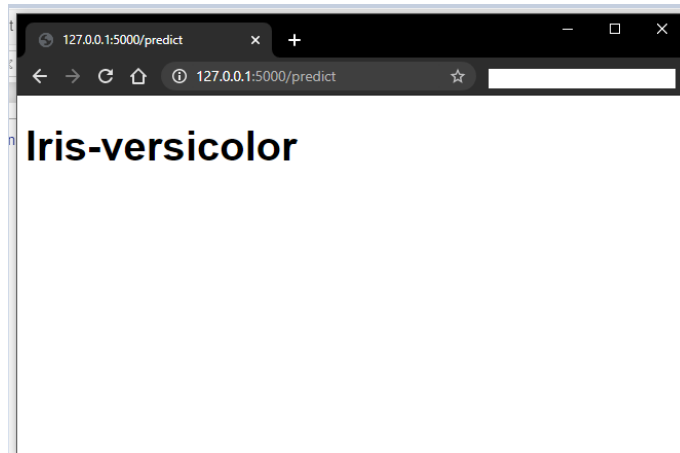
\* Serving Flask app "\_\_main\_\_" (lazy loading)  
\* Environment: production  
WARNING: This is a development server. Do not use it in a production deployment.  
Use a production WSGI server instead.  
\* Debug mode: off

\* Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)

- 4) Tekan **hyperlink** berwarna biru untuk melihat aplikasi pada web.



- 5) Isi setiap field pada web apps lalu tekan tombol **Submit**. Ketika tombol submit ditekan, maka hasil prediksi akan muncul.



#### 4. Pertanyaan

- 1) Apakah ada cara lain untuk storing model selain dengan pickle? Bagaimana cara kerjanya dan apa keuntungannya?
- 2) Buatlah sebuah web apps untuk melakukan prediksi harga rumah berdasarkan dataset Boston Housing yang telah dikerjakan pada Modul 4. Gunakan pickle dan Flask untuk membuat web application dan bentuk model yang sudah siap digunakan dengan Jupyter Notebook. Unggah hasil pekerjaan baru tersebut pada folder Modul 6 – Model Deployment : Flask pada GitHub pribadi.

## CHEAT SHEET : NUMPY

Syntax	Fungsi
import numpy	Import library numpy
a = np.array([LIST])	Membuat array
np.zeros(n)	Membuat array 1 dimensi sejumlah n dengan value 0
np.zeros((n,m))	Membuat array 2 dimensi berukuran nxm dengan value 0
np.zeros((n,m,p))	Membuat array 3 dimensi berukuran nxm xp dengan value 0
np.full((n,m), x)	Membuat array 2 dimensi berukuran nxm dengan value x
np.random.rand(n,m)	Membuat array 2 dimensi berukuran nxm dengan value random (0 - 1)
np.eye(n)	Membuat array identitas berukuran nxn
array.size	Property untuk menampilkan ukuran array (baris x kolom)
array.shape	Property untuk menampilkan dimensi array (baris, kolom)
array.dtype	Property untuk menampilkan data type untuk elemen di dalam array
np.copy(ARRAY)	Copy array ke variabel baru
array.sort ()	Mengurutkan value array
np.append(ARRAY, VALUE)	Menambahkan value baru ke dalam array
np.delete(ARRAY, n, axis = 0)	Menghapus baris ke-n dari array
np.delete(ARRAY, n, axis = 1)	Menghapus kolom ke-n dari array
np.concatenate(ARRAY1, ARRAY2, axis = 0)	Menyatukan Array 1 dan 2 secara baris
np.concatenate(ARRAY1, ARRAY2, axis = 1)	Menyatukan Array 1 dan 2 secara kolom
np.split(ARRAY, n)	Membagi array menjadi sub-array sebanyak n
a[n]	Mengambil elemen pada index n
a[n,m]	Mengambil elemen pada index n,m
a[x:y]	Mengambil seluruh elemen dari index x sampai y
a[x:y, u]	Mengambil seluruh elemen dari index x sampai y pada kolom u
a[ : x]	Mengambil seluruh elemen dari index 0 sampai x
a.sum()	Menambahkan seluruh elemen array
a.min()	Menampilkan elemen paling kecil dari array
a.max()	Menampilkan elemen paling besar dari array
a.mean()	Menampilkan rata - rata elemen dari array
a.median()	Menampilkan median dari array

## CHEAT SHEET : PANDAS

Syntax	Fungsi
import pandas	Import library Pandas
df = pd.read_csv('filename.csv')	Loading data dengan format csv
df.loc[:, (df>n).any()]	Memilih kolom dengan value >n
df.loc[:, (df>n).all()]	Memilih kolom tanpa value >n
df.loc[:, (df.isnull()).any()]	Memilih kolom dengan null
df.loc[:, (df.notnull()).all()]	Memilih kolom tanpa null
df.reset_index()	Memulai ulang index dari 0
df = df.rename(columns = { 'NAMA 1' : 'NAMA 2'})	Mengubah nama kolom dari NAMA1 ke NAMA2
df.unique()	Menampilkan value unik
df.duplicated('KOLOM')	Mengecek kolom duplikat
df.drop_duplicates('KOLOM')	Membuang kolom duplikat
df.groupby(by = 'KOLOM').mean()	Mengelompokkan dataset sesuai dengan value yang sama pada KOLOM
df.dropna()	Membuang value NaN
df.fillna(VALUE)	Mengganti value NaN dengan VALUE
df.replace('VALUE1', 'VALUE2')	Mengganti VALUE1 dengan VALUE2

## CHEAT SHEET : SCIKIT-LEARN

Syntax	Fungsi
from sklearn.linear_model import LinearRegression	Import algoritma Linear Regression
from sklearn.svm import SVC	Import algoritma Support Vector Classifier
from sklearn.naive_bayes import GaussianNB	Import algoritma Gaussian Naïve Bayes
from sklearn import neighbors	Import algoritma K-Nearest Neighbor
from sklearn import tree	Import algoritma Decision Tree
from sklearn.decomposition import PCA	Import algoritma Principal Component Analysis
from sklearn.cluster import KMeans	Import algoritma K-Means Clustering
from sklearn.preprocessing import StandardScaler	Standarisasi dataset
scaler = preprocessing.StandardScaler()	
scaled = scaler.fit_transform(DATA FRAME)	
from sklearn.preprocessing import Normalizer	Normalisasi kolom
normalized = preprocessing.normalize(COLUMN)	
from sklearn.model_selection import train_test_split	Train-Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state = 0)	
supervised_model.fit(X,Y)	Fitting model supervised
unsupervised_model.fit(X)	Fitting model unsupervised
from sklearn.metrics import confusion_matrix	Confusion Matrix
print(confusion_matrix(Y_test, Y_pred))	
from sklearn.metrics import accuracy_score	Evaluation Metrics
accuracy_score(Y_test, Y_pred)	
from sklearn.metrics import mean_absolute_error	Mean Absolute Error
mean_absolute_error (Y_true, Y_predict)	
from sklearn.metrics import mean_squared_error	Mean Squared Error
mean_squared_error (Y_true, Y_predict)	
from sklearn.metrics import r2_score	R Squared Score
r2_score (Y_true, Y_predict)	

## REFERENSI

- [1] Foster Provost and Tom Fawcett, 2013, "Data Science for Business: What you need to know about data mining and data-analytic thinking (1st. ed.)". O'Reilly Media, Inc.
- [2] Rollins John, 2015, "*Foundational Methodology for Data Science*", IBM OpenSource.
- [3] "Explained K-Means + PCA Visualization for Wine Dataset" [Online]. Available : <https://www.kaggle.com/rodrigofragoso/explained-k-means-pca-visualization>. [Accessed : 16-Jun-2020]
- [4] "Do we do data cleansing or EDA first?" [Online]. Available : <https://www.kaggle.com/questions-and-answers/103089>. [Accessed : 16-Jun-2020]
- [5] "Importance of Feature Scaling." [Online]. Available : [https://scikit-learn.org/stable/auto\\_examples/preprocessing/plot\\_scaling\\_importance.html#sphx-glr-auto-examples-preprocessing-plot-scaling-importance-py](https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html#sphx-glr-auto-examples-preprocessing-plot-scaling-importance-py). [Accessed : 17-Jun-2020]
- [6] "Scikit-learn Decomposition PCA." [Online]. Available : <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>. [Accessed : 17-Jun-2020]
- [7] "6 Different Ways to Compensate for Missing Value in a Dataset." [Online]. Available : <https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>. [Accessed : 17-Jun-2020]
- [8] "Scale, Standardize, or Normalize with Scikit-Learn." [Online]. Available : <https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02>. [Accessed : 18-Jun-2020]
- [9] "Understanding Boxplot." [Online]. Available : <https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51#:~:text=A%20boxplot%20is%20a%20standardized,and%20what%20their%20values%20are>. [Accessed : 19-Jun-2020]
- [10] "Ways to Detect and Remove the Outliers." [Online]. Available : <https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba>. [Accessed : 19-Jun-2020]
- [11] "How to Determine the Optimal K for K-Means?" [Online]. Available : <https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb#:~:text=End%20Notes,for%20finding%20the%20optimal%20K>. [Accessed : 22-Jun-2020]
- [12] "Medical Cost Dataset EDA and Regression." [Online]. Available : <https://www.kaggle.com/hely333/eda-regression>. [Accessed : 23-Jun-2020]

- [13] "Linear Regression on Boston Housing Dataset." [Online]. Available : <https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155> [Accessed : 23-Jun-2020]
- [14] "Linear Regression for Machine Learning." [Online]. Available : <https://machinelearningmastery.com/linear-regression-for-machine-learning/> [Accessed : 23-Jun-2020]
- [15] "Linear Regression, Back to Basic." [Online]. Available : <https://medium.com/data-science-group-iitr/linear-regression-back-to-basics-e4819829d78b> [Accessed : 23-Jun-2020]
- [16] "Using Residuals to Identify a Line of Good Fit." [Online]. Available : <http://www.shodor.org/interactivate/discussions/UsingResiduals/> [Accessed : 29-Jun-2020]
- [17] "Understanding Random Forest." [Online]. Available : <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>. [Accessed : 02-Jul-2020]
- [18] "Gini Index vs Information Entropy" [Online]. Available : <https://towardsdatascience.com/gini-index-vs-information-entropy-7a7e4fed3fcb>. [Accessed : 02-Jul-2020]
- [19] "Decision Trees Documentation" [Online]. Available : <https://scikit-learn.org/stable/modules/tree.html>. [Accessed : 02-Jul-2020]
- [20] "Bagging and Random Forest Ensemble Algorithms for Machine Learning." [Online]. Available : <https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/>. [Accessed : 03-Jul-2020]
- [21] "Churn Prediction by Selecting from 11 Tuned Models" [Online]. Available : <https://www.kaggle.com/berkanacar/churn-prediction-by-selecting-from-11-tuned-models> [Accessed : 07-Jul-2020]
- [22] "Telecom Customer Churn Prediction". [Online]. Available : <https://www.kaggle.com/pavanraj159/telecom-customer-churn-prediction> [Accessed : 07-Jul-2020]
- [23] "Label Encoder vs. One Hot Encoder in Machine Learning" [Online]. Available : <https://medium.com/@contactsunny/label-encoder-vs-one-hot-encoder-in-machine-learning-3fc273365621#:~:text=What%20one%20hot%20encoding%20does,which%20column%20has%20what%20value.&text=So%2C%20that's%20the%20difference%20between%20Label%20Encoding%20and%20One%20Hot%20Encoding>. [Accessed : 08-Jul-2020]



- [24] "Metrics to Evaluate your Machine Learning Algorithm." [Online]. Available : <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>. [Accessed : 08-Jul-2020]
- [25] "Overfitting and Underfitting with Machine Learning Algorithms" [Online]. Available : <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/> [Accessed : 08-Jul-2020]
- [26] "What Does it Mean to Deploy A Machine Learning Model?" [Online]. Available : <https://towardsdatascience.com/what-does-it-mean-to-deploy-a-machine-learning-model-dddb983ac416#:~:text=Deploying%20a%20machine%20learning%20model,input%20and%20return%20an%20output.> [Accessed : 09-Jul-2020]
- [27] "How to deploy your ML model in Jupyter Notebook to your Flask App?" [Online]. Available : <https://medium.com/techcrush/how-to-deploy-your-ml-model-in-jupyter-notebook-to-your-flask-app-d1c4933b29b5>. [Accessed : 09-Jul-2020]
- [28] "Iris Species Dataset." [Online]. Available : <https://www.kaggle.com/uciml/iris> [Accessed : 09-Jul-2020]
- [29] "Understanding Python Pickling and How To Use It Securely" [Online]. Available : <https://www.synopsys.com/blogs/software-security/python-pickling/> [Accessed : 10-Jul-2020]
- [30] "Pickle vs JSON." [Online] Available : <https://www.educba.com/python-pickle-vs-json/>. [Accessed : 10-Jul-2020]
- [31] "What is Statistic?" [Online]. Available : <https://www.stat.uci.edu/what-is-statistics/>. [Accessed : 10-Jul-2020]
- [32] "Python General Information." [Online] Available : <https://www.python.org/about/>. [Accessed : 10-Jul-2020]
- [33] "Numerical Python." [Online]. Available : <https://numpy.org/> [Accessed : 10-Jul-2020]
- [34] "Pandas Library" [Online] Available : <https://pandas.pydata.org/>. [Accessed : 11-Jul-2020]
- [35] "Matplotlib Library." [Online]. Available : <https://matplotlib.org/>. [Accessed : 11-Jul-2020]
- [36] "Seaborn Library." [Online]. Available : <https://seaborn.pydata.org/>. [Accessed : 12-Jul-2020]
- [37] "Scikit-learn Library." [Online]. Available : <https://scikit-learn.org/>. [Accessed : 12-Jul-2020]
- [38] "Jupyter Notebook." [Online]. Available : <https://jupyter.org/about>. [Accessed : 13-Jul-2020]
- [39] "Micro Web Framework : Flask." [Online]. Available : <https://www.fullstackpython.com/flask.html>. [Accessed : 13-Jul-2020]
- [40] "Pickle." [Online]. Available : <https://docs.python.org/3/library/pickle.html>. [Accessed : 13-Jul-2020]