Modul Praktikum
Dasar Pengolahan Data

Program Studi Teknik Elektro
Universitas Pelita Harapan
2021

Daftar Isi

Introduction to Python

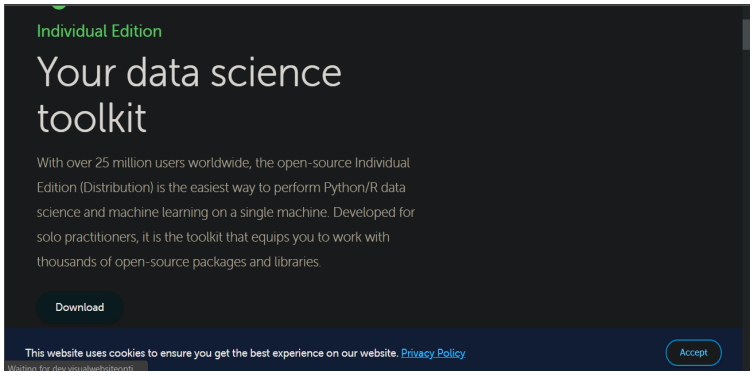
1. Penjelasan Singkat

Anaconda adalah suatu *software* distribusi yang memudahkan kita untuk mengakses *software-software* yang digunakan untuk *scientific programming*. Salah satu *software* yang didistribusikan oleh Anaconda adalah Python. Python adalah bahasa pemrograman interpretatif multiguna, dan dirancang agar memiliki struktur *syntax* yang serupa dengan bahasa Inggris, sehingga mudah dimengerti oleh orang awam.

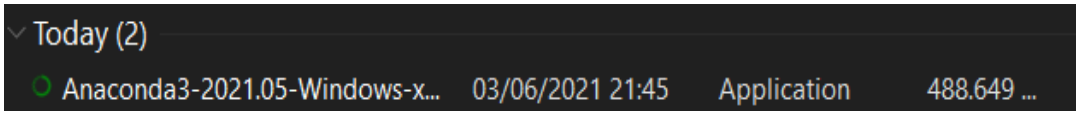
2. Langkah-Langkah Percobaan

Instalasi Anaconda

A. Pertama *download* Anaconda dari https://www.anaconda.com/products/individual ,tekan *download* kemudian pilih sistem operasi yang sesuai dengan *device* yang digunakan.

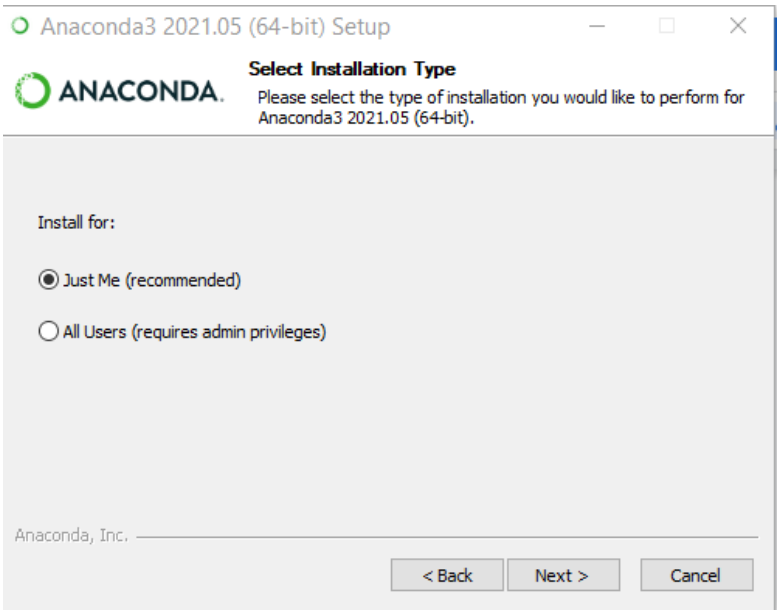


B. Setelah proses *download* selesai, carilah *installer* Anaconda seperti yang terlihat pada gambar di bawah. Klik *installer* untuk memulai proses instalasi Anaconda

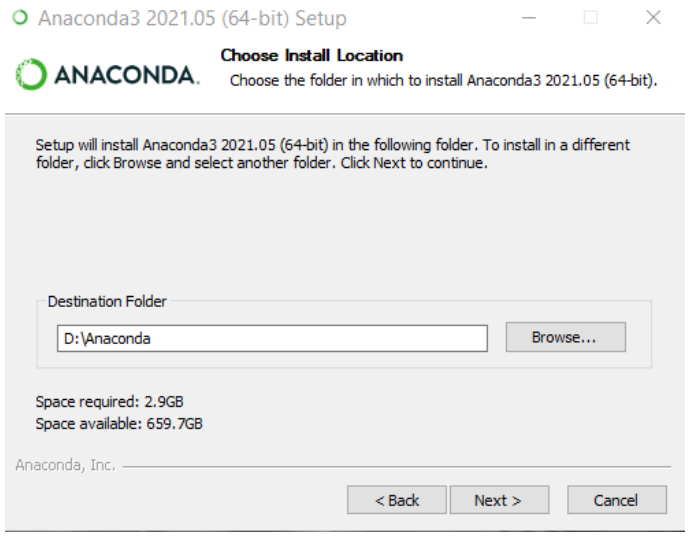


C. Ketika diklik akan muncul tampilan seperti berikut, klik Next.

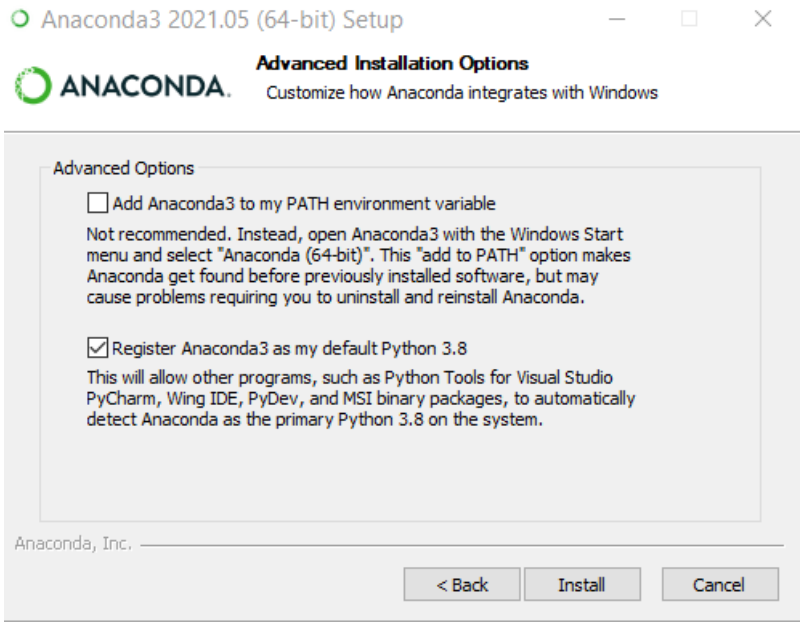
D. Pilihlah instalasi, *just for me*. Klik Next



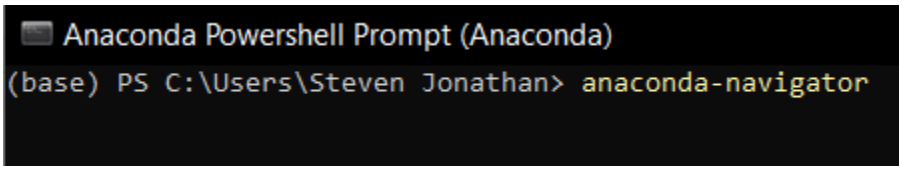
E. *Install* aplikasi pada tempat yang diinginkan, buatlah *destination folder* bernama D:\Anaconda, kemudian *browse* dari pilih *local disk d*.



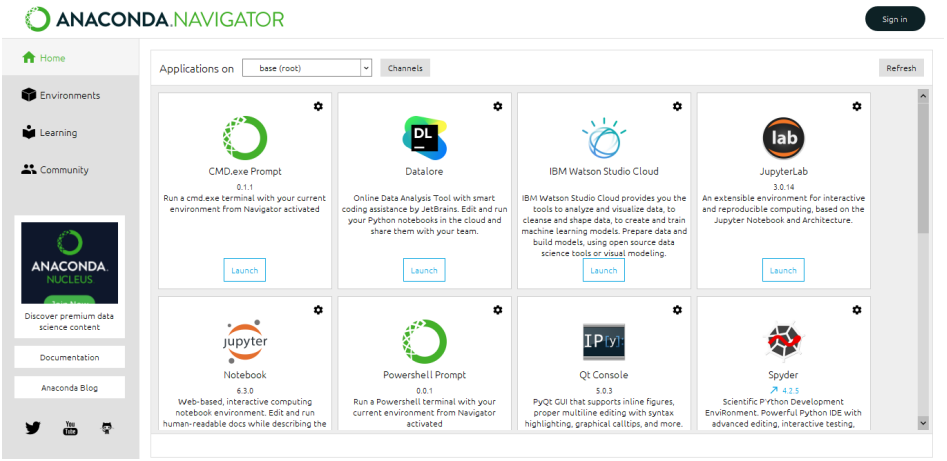
F. Kemudian klik opsi " *register Anaconda as my default Python 3.8* ", klik Next dan tunggu instalasi hingga selesai.



G. Ketika instalasi selesai, carilah aplikasi bernama **anaconda powershell prompt**. Klik, dan ketik **anaconda-navigator,** kemudian *enter* untuk membuka tampilan Anaconda Navigator

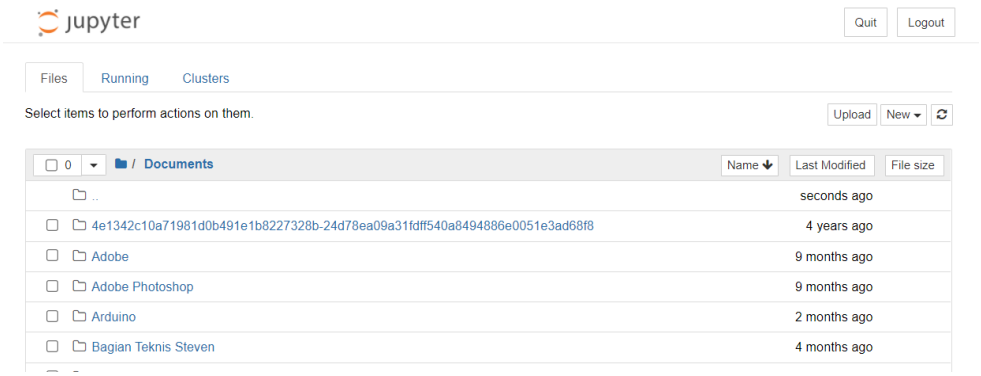


H. Carilah Jupyter Notebook pada Anaconda Navigator, kemudian klik *launch*.

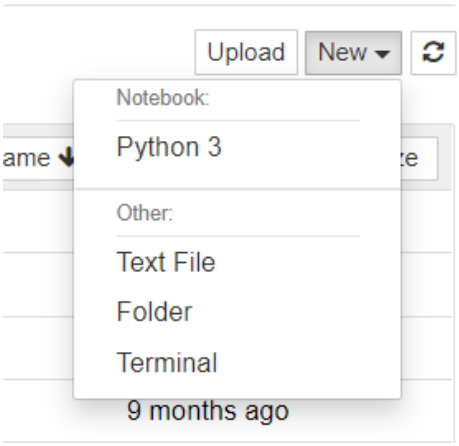


Inisiasi Jupyter Notebook

A. *Launch* Jupyter Notebook di Anaconda, sehingga akan muncul tampilan sebagai berikut



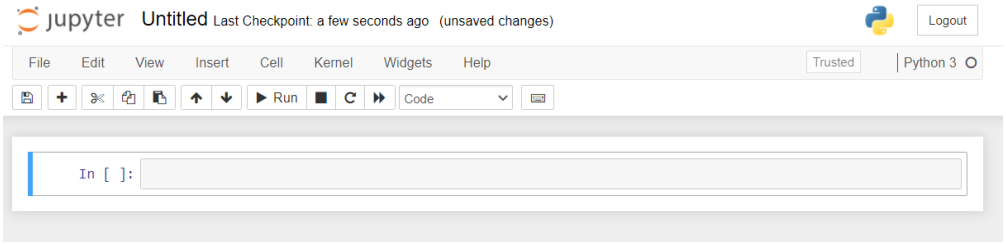
B. Klik kolom box *New*, dan pilih opsi *Notebook* - Python 3



C. Ini adalah tampilan yang menunjukan *Notebook* baru sudah terbentuk, sebelum mendalami pemrograman Python perhatikan hal sebagai berikut:

   a. Untuk menambahkan *cell,* klik *ribbon* dengan logo '+' sehingga kolom akan bertambah pada row selanjutnya di *notebook.*

b. Untuk mengeksekusi program pada kolom *box*, klik *ribbon* yang bertuliskan *Run*.
c. Pastikan kolom pada *notebook*, bertuliskan kode yang menandakan *box* dalam kondisi menerima kode program.



Basic Syntax Python

A. Perintah *print* berfungsi untuk mencetak *output* pada *jupyter notebook*. Output yang dicetak tidak terbatas pada kata-kata, biasanya *print* pada *data science* banyak digunakan untuk mencetak keluaran dari pengolahan analisa yang dilakukan pada *dataset*. Penulisan *print* dapat dilakukan dengan menuliskan perintah sesuai contoh gambar.

Contoh Penulisan :

```
print("Isi Kalimat")
```

Tuliskan kata Hello World pada perintah *print*, kemudian *run* pada Python

Jawaban :

```
print("Hello World")
```

```
Hello World
```

B. Variabel bertugas untuk menyimpan nilai, atau data yang di *input*. Nilai variabel dapat berubah, jika kita *assign* nilai pada variabel yang sama. Pada *data science*, variabel sangat membantu ketika kita ingin menyimpan hasil pengolahan yang ingin kita gunakan berulang kali, hal ini memudahkan kita untuk mengakses hasil pengolahan tersebut tanpa harus mengetik keseluruhan perintah. Penulisan variabel bisa dilakukan sesuai contoh gambar di bawah.

Contoh Penulisan :

```
x1 = 6
x4 = 5
T = 7
```

Buatlah bentuk program seperti perintah yang diberikan:

1. **Buatlah variabel alpha, yang akan berisi nilai 7**
2. **Buatlah variabel romeo akan berisi kalimat Ada Apa**
3. ***print* kedua hasil tersebut untuk mengetahui isi variabel.**

Jawaban :

```
alpha = 7

romeo = "Ada Apa"

print(alpha)
print(romeo)
```

```
7
Ada Apa
```

C. Python juga dapat melakukan operasi matematika seperti, penambahan, pengurangan, dan perkalian pada angka seperti contoh yang diberikan. Pada *data science*, ini sangat berguna untuk membuat rumus-rumus matematika, seperti mencari nilai rata-rata (*mean*). Contoh penggunaan dapat dilihat pada gambar dibawah.

Contoh Jawaban :

```
x1 = 7
x2 = 6

print(x1 + x2) // 6 + 7 = 13
print(x1 * x2) // 7 * 6 = 42
print(x1 - x2) // 7 - 6 = 1
```

Buatlah bentuk program seperti perintah yang diberikan:

1. **Buatlah variabel x1, yang akan berisi nilai 7**
2. **Buatlah variabel x2, yang akan berisi nilai 6**

3. ***Print*** **hasil operasi pertambahan x1 dan x2**
4. ***Print*** **hasil operasi perkalian x1 dan x2**
5. ***Print*** **hasil operasi pengurangan x1 dan x2**

Jawaban :

```
x1 = 7
x2 = 6

print(x1 + x2) // 6 + 7 = 13
print(x1 * x2) // 7 * 6 = 42
print(x1 - x2) // 7 - 6 = 1
```

```
9
240
2
```

Data Type Python

A. Berikut adalah tipe-tipe data yang terdapat pada Python

1. **Integers : suatu bentuk bilangan bulat. Contoh : 1, 30, 54**
2. **Float : suatu bentuk bilangan bulat yang memiliki desimal. Contoh : 4.16 , 3.25, 1.7**
3. **String : suatu bentuk kumpulan dari banyak karakter. Contoh : Saya Steven, Aku Lapar**

4. **Boolean : dua nilai yang terdiri dari 1 dan 0, atau benar dan salah. Contoh : 1 , 0**
5. **List : suatu kumpulan data yang diatur secara teratur, dimana nilai didalamnya dapat berubah. Contoh : [“saya”, 5]**
6. **Tuples : suatu kumpulan data yang diatur secara urutan, dimana nilai didalamnya tidak dapat berubah. Contoh : (10, “saya”, 11)**
7. **Dictionary : suatu kumpulan data yang yang disimpan dalam bentuk “key”. Contoh : [Saya : “mau makan”, Ini : 2]**
8. **Set : suatu kumpulan data yang tidak diurutkan, tapi mampu melakukan operasi tanpa menyimpannya terlebih dahulu. Contoh : A : {0,1,2,3} , B : {2,0,4)**

Kita akan membahas tipe data yang paling banyak dipakai pada *data science* penjelasan sebagai berikut

B. Numerik adalah tipe data yang terdiri dari 3 hal, yaitu integers, float, dan complex., tidak hanya itu kita juga bisa mengubah isi data yang disimpan menjadi bentuk tipe data lain. Kita dapat mengetahui tipe data dari suatu variabel dengan menuliskan kode type, sementara untuk mengubah isi data bisa dilakukan dengan mengetik tipe data yang ingin kita ubah dari tipe awal. Contoh penulisan dapat dilihat di bawah

Contoh Penulisan :

```
type(variable)
```

```
float(variable)
```

Buatlah bentuk program sebagai berikut:

1. **Buatlah variabel a, nilai a adalah 10+3j**
2. **Buatlah variabel b, nilai b adalah 20**
3. **Buatlah variabel d, nilai d adalah nilai pada tipe data variabel b**
4. **Keluarkan isi data pada setiap variabel dengan perintah *print*.**

Jawaban :

```
a = 10 # integeres
print(type(a))

b = 1.7 # Float
print(type(b))

c = 1+3j # Complex
print( c, "Apakah bilangan Kompleks ?", isinstance(1+3j,complex))
```

```
<class 'int'>
<class 'float'>
(1+3j) Apakah bilangan Kompleks ? True
```

C. List adalah kumpulan order sequence, artinya dia adalah jenis variabel kumpulan data yang terurut. List dapat diakses dengan menuliskan angka untuk mengetahui elemen yang disimpan, elemen pada List dapat diubah setelah di *declare*. Penulisan dilakukan dengan memberikan variabel, suatu simbol [ ] seperti contoh dibawah.

Contoh Penulisan :

```
x = [5, 10, 15]
print(x[0]) // ketika di print akan menghasilkan nilai 5
print(x[-1]) // ketika di print akan menghasilkan nilai 15
print(x[0:2]) // ketika di print akan menghasilkan nilai 5, dan nilai 10

x[2] = 10
print(x) // akan dihasilkan [5, 10, 10]
```

Buatlah bentuk program sebagai berikut:

1. **Buatlah variabel x, variabel x adalah suatu *list***
2. **Variabel x akan berisi nilai 5, 10, 15, 20, 25, 30, 35, 40**
3. **Keluarkan hasil elemen pertama**
4. **Keluarkan hasil elemen terakhir**
5. **Keluarkan hasil elemen diantara elemen 3 dan elemen 5**

Jawaban :

```
x = [5,10,15,20,25,30,35,40]
print(x[0])
print(x[-1])
print(x[3:5])
```

```
5
40
[20, 25]
```

D. Dictionary adalah kumpulan pasangan kunci dan nilai yang tidak harus berurutan. Dictionary dapat digunakan untuk menyimpan data kecil hingga besar, untuk mengakses data kita harus mengetahui kuncinya seperti contoh yang diberikan. Key dapat berupa angka, atau kalimat, syaratnya dia harus diletakan di paling depan.

Contoh Penulisan :

```
d = {30:"Keluarkan 30",'kata kunci':"Kunci akan membuka pintu"}
print(d[30])
print(d['kata kunci'])
```

```
Keluarkan 30
Kunci akan membuka pintu
```

Buatlah bentuk program sebagai berikut:

1. **Buat variabel d, variabel d adalah *dictionary***
2. **Terdapat 2 key**
3. ***Key* pertama bernama 30, dan berisi Keluarkan 30**
4. ***Key* kedua bernama kata kunci, dan berisi Kunci akan membuka pintu**
5. **Keluarkan nilai *key* pertama**
6. **Keluarkan nilai *key* kedua**

Jawaban :

```
d = {30:"Keluarkan 30",'kunci':"Kunci akan membuka pintu"}
print(d[30])
print(d['kunci'])
```

```
Keluarkan 30
Kunci akan membuka pintu
```

Conditional dan Loops

A. *Conditional* pada Python merupakan operasi logika dengan syarat yang harus dipenuhi sebelum dapat dijalankan. Perintah ini pada Python dimulai dari If,

elif (else-if), dan else jika pilihan tersebut adalah opsi terakhir. Penulisan dilakukan dengan contoh sebagai berikut.

Contoh Penulisan :

```
if nilai>80:
   print("Selamat! Anda mendapat nilai A")

elif nilai>70:
   print("Selamat! Anda mendapat nilai B")

elif nilai>60:
   print("Selamat! Anda mendapat nilai C")

else:
   print("Selamat! Anda mendapat nilai D")
```

Buatlah bentuk program sebagai berikut:

1. **Buat variabel nilai, variabel nilai berisi 70**
2. **Jika nilai lebih besar 80, maka *print* Anda mendapat nilai A**
3. **Jika nilai lebih besar 80, maka *print* Anda mendapat nilai A**
4. **Jika nilai tidak memenuhi keduanya, maka *print* Gagal**

Jawaban :

```
nilai = 70

if nilai>80:
   print("Anda mendapat nilai A")

elif nilai>60:
   print(" Anda mendapat nilai C")

else:
   print("Gagal")
```

```
Anda mendapat nilai C
```

B. Bentuk Perulangan pada Python salah satunya dapat dilakukan dengan memakai For, For digunakan untuk melakukan looping pada *list*. Pada contoh yang diberikan kita melakukan looping dengan for sehingga menghasilkan *output*.

Contoh Penulisan :

```
flowers = ['mawar', 'melati', 'anggrek']
for flower in flowers:  # Contoh kedua
    print("Flower ini bernama ", flower)
```

Buatlah bentuk program sebagai berikut:

1. **Buat variabel bernama flowers, variabel berjenis *list***
2. **Flowers akan berisi sunflower, melati, dan anggrek**
3. **Lakukan *for loop* untuk mengeluarkan isi nilai dari flowers**
4. **Keluarkan hasil flower pada setiap iterasi dengan perintah *print*.**

Jawaban :

```
flowers = ['sunflower', 'melati', 'anggrek']
for x in flowers:  # Contoh kedua
    print(x)
```

```
sunflower
melati
anggrek
```

C. While digunakan untuk melakukan *looping*, dan mengeksekusi statement jika syarat awal terpenuhi. While dapat ditulis dengan contoh pada gambar.

Contoh Penulisan :

```
count = 0
while (count < 7):
    print("Looping nomor ", count)
    count = count + 1
```

Buatlah bentuk program sebagai berikut:

1. **Buat variabel bernama d, d akan bernilai 0**
2. **Jika variabel d memiliki nilai yang lebih kecil daripada 7, maka algoritma akan *print* nilai d**
3. **Nilai d kemudian akan ditambah d, kemudian algoritma akan melakukan berhenti hingga d lebih besar dari 7**

Jawaban :

```
d = 0
while (d < 7):
    print(d)
    d = d + 1
```

```
0
1
2
3
4
5
6
```

Pertanyaan

1. **Buatlah suatu algoritma yang melakukan konversi dari *celcius* menjadi *fahrenheit*, dan sebaliknya.**
   Rumus konversi $\frac{C}{5} = \frac{F-32}{9}$, dimana C adalah *celcius* dan F adalah *fahrenheit.*
   1. **Buatlah dua variabel bernama degree, dan simbol.**
   2. **Pada degree akan berisi 50**
   3. **Simbol akan berisi F.**
   4. **Jika simbol sama dengan C, maka akan dilakukan konversi menjadi nilai Fahrenheit**
   5. **Jika simbol sama dengan F, maka dilakukan konversi menjadi nilai Celcius**
   6. **Jika tidak mempunyai isi pada simbol, maka *print* Penamaan Tidak Dapat Dikonversi**

   Keluaran yang diinginkan

   The temperature in Celcius is 10 degrees

   Jawaban :

```
degree = 50
simbol = 'F'

if simbol == "C":
  Hasil = int(round((9 * degree) / 5 + 32))
  Simbol = "Fahrenheit"
elif simbol == "F":
  Hasil = int(round((degree - 32) * 5 / 9))
  Simbol = "Celsius"
else:
  print("Penamaan tidak dapat dikonversi")

print("The temperature in", Simbol, "is", Hasil, "degrees.")
```

```
The temperature in Celsius is 10 degrees.
```

2. **Buatlah suatu algoritma sebagai berikut:**
    1. **Buatlah suatu variabel, dengan nama List**
    2. **List akan berisi nilai sebagai berikut 32.78, 3+9j, False, Ini adala Text**
    3. **Gunakan *for loop* untuk mengeluarkan nilai variabel List, dan tipe data dari setiap nilai pada variabel List.**

Keluaran yang diinginkan

Tipe item 32.78 adalah <class 'float'>
Tipe item (3+9j) adalah <class 'complex'>
Tipe item False adalah <class 'bool'>
Tipe item Ini adala Text adalah <class 'str'>

Jawaban :

```
List = [32.78, 3+9j, False, "Ini adala Text"]
for item in List:
   print (item, type(item))
```

```
32.78 <class 'float'>
(3+9j) <class 'complex'>
False <class 'bool'>
Ini adala Text <class 'str'>
```

Modul 2
Data Wrangling

1. Penjelasan Singkat

Pada modul ini, kita akan menggunakan *library* Python bernama Pandas untuk melakukan analisa data pada modul percobaan. Pandas sangat berguna dalam proses menganalisa data, karena mampu untuk menggabungkan beberapa kolom yang sangat membantu dalam memfokuskan analisa pada data. Pandas juga akan kita gunakan untuk proses yang bernama *data wrangling*, dimana kita akan menelusuri nilai yang hilang, serta menggantinya dengan hasil *mean* rata-rata kolom data yang kehilangannya nilainya.

2. Langkah-Langkah Percobaan

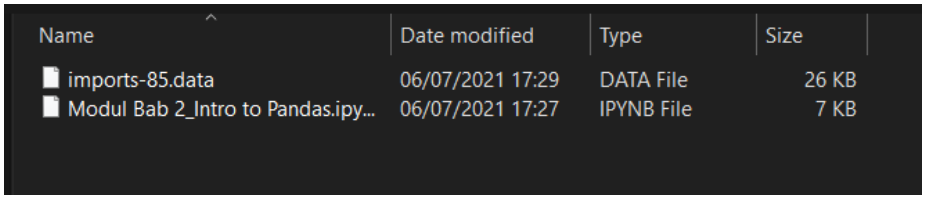
Data Acquisition

A. Terdapat banyak jenis dataset, seperti csv, .json, .xlsx etc yang bisa didapat dari *kaggle* atau berbagai sumber internet lainnya. Pada praktikum ini, kita akan mengambil Automobile Dataset dari internet, dataset ini berjenis CSV (Comma Separated Value).

***data source*:**

https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data

Unduh *dataset*, kemudian buatlah 1 file, pada file masukan *dataset* yang diunduh beserta notebook yang akan digunakan



B. Pandas adalah *library* Python yang digunakan untuk membaca data dari berbagai macam bentuk file, seperti csv, xlsx. Pandas juga mampu melakukan analisa dasar pada data; Jupyter Notebook sudah memiliki *library* Pandas, sehingga kita hanya perlu mengimpor Panda dengan perintah seperti dibawah.

Contoh Penulisan :

```
import pandas as pd
```

Jawaban :

```
# import pandas library
import pandas as pd
```

C. Untuk membuka data, tentunya ada beberapa format yang harus dikonversi terlebih dahulu untuk dapat dilakukan analisa. Pada Pandas kita dapat menggunakan fungsi *read*, seperti contoh gambar dibawah. Nama *file* adalah nama dari *dataset* yang di unduh, kemudian *headers* merupakan sub-nama setiap kolom data yang ditambahkan secara *manual* jika tidak ditemukan kehadiran sub-nama data.

Contoh Penulisan :

```
pandas.read_csv("nama file / alamat file", headers = none (jika tidak ada) )
```

Untuk file *dataset* selain csv, kita bisa baca dengan beberapa fungsi berikut

Contoh Penulisan :

Data Formate	Read
csv	pd.read_csv()
json	pd.read_json()
excel	pd.read_excel()
hdf	pd.read_hdf()
sql	pd.read_sql()

Gunakan fungsi read_csv pada *library* pandas, gunakan untuk membaca file *dataset*, kemudian simpan hasil pembacaan fungsi pada variabel df

Jawaban :

```
df = pd.read_csv("imports-85.data", header=None)
```

D. Setelah membaca dataset, terdapat dua cara utama pembacaan *dataset* bisa dilakukan.

   a. metode dataframe.head(n) untuk memeriksa n baris teratas dari dataframe.

      Contoh Penulisan :

```
df.head(3)
```

   b. Berlawanan dengan dataframe.head(n), dataframe.tail(n) akan memperlihatkan n baris terbawah dari dataframe.

      Contoh Penulisan :

```
df.tail(3)
```

Gunakan kedua metode, untuk membaca 5 baris teratas, dan 5 baris terbawah dari *dataset* yang disimpan pada variabel df.

Jawaban :

```
df.head(5)
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

```
df.tail(5)
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

E. Pada *dataset*, terlihat tidak mempunyai headers. Pandas untungnya dapat secara otomatis mengatur header pada *dataset*.

Sumber Header: https://archive.ics.uci.edu/ml/datasets/Automobile

Pada *dataset* ini, kita harus menambahkan header secara manual.

Sebutkan semua nama kolom secara berurutan dengan List. Kemudian, simpan pada variabel bernama headers. Gunakan fungsi pada gambar di bawah untuk mengisi kolom judul dengan nama pada variabel headers. Kemudian cek 5 data awal pada *dataset* apakah sudah ada penambahan judul

Contoh Penulisan :

```
pd.read_csv(filename, names = headers)
```

Jawaban :

```
headers = ["symboling","normalized-losses","make","fuel-type","aspiration", "num-of-doors",
           "body-style","drive-wheels","engine-location","wheel-base", "length","width","height",
           "curb-weight","engine-type","num-of-cylinders", "engine-size","fuel-system","bore",
           "stroke","compression-ratio","horsepower",
         "peak-rpm","city-mpg","highway-mpg","price"]
```

```
df = pd.read_csv("imports-85.data", names = headers)
df.head(10)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8
7	1	?	audi	gas	std	four	wagon	fwd	front	105.8
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5

Basic Insight of Data-set

A. Terdapat berbagai macam jenis data. Jenis utama yang disimpan dalam kerangka data Pandas adalah objek, float, int, bool, dan datetime64. Pada pandas jenis data pada *dataset* dapat dibaca menggunakan fungsi berikut
Contoh Penulisan :

```
.dtypes
```

Gunakan fungsi tersebut, untuk membaca jenis data pada *dataset* yang kita miliki

Jawaban :

```
print(df.dtypes)
```

```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
```

B. Pandas juga memudahkan kita untuk mencari penjelasan statistik dari kumpulan data yang ingin dianalisis, untuk mengakses penjelasan statistik ini, kita bisa gunakan fungsi *describe*. Gunakan kedua cara seperti gambar dibawah, untuk mencari penjelasan statistik.

a. Metode ini akan memberikan berbagai statistik ringkasan, tidak termasuk nilai NaN
Contoh Penulisan :

```
.describe()
```

b. Metode ini akan memberikan berbagai statistik ringkasan, termasuk nilai NaN
Contoh Penulisan :

```
.describe(include = "all")
```

Jawaban :

```
df.describe()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000

```
df.describe(include = "all")
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system
count	205.000000	205	205	205	205	205	205	205	205	205.000000	...	205.000000	205
unique	NaN	52	22	2	2	3	5	3	2	NaN	...	NaN	8
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi
freq	NaN	41	32	185	168	114	96	120	202	NaN	...	NaN	94
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317	NaN
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693	NaN
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000	NaN
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000	NaN

C. Kita dapat memilih kolom dari bingkai data dengan menunjukkan nama setiap kolom, misalnya, kita dapat memilih tiga kolom sebagai berikut:
Contoh Penulisan :

```
dataframe[[' kolom 1 ',kolom 2', 'kolom 3']]
```

Di mana "kolom" adalah nama kolom, kita dapat menerapkan metode .describe() untuk mendapatkan statistik kolom tersebut sebagai berikut:
Contoh Penulisan :

```
dataframe[[' kolom 1 ',kolom 2', 'kolom 3']].describe()
```

Terapkan metode ke .describe() pada nama yang sesuai dengan *header* yaitu pada 'panjang' dan 'rasio kompresi'.

Jawaban :

```
df[['length', 'compression-ratio']].describe()
```

	length	compression-ratio
count	205.000000	205.000000
mean	174.049268	10.142537
std	12.337289	3.972040
min	141.100000	7.000000
25%	166.300000	8.600000
50%	173.200000	9.000000
75%	183.100000	9.400000
max	208.100000	23.000000

Replace Missing Value

A. Pada data yang dianalisis, terlihat bahwa data yang hilang diwakilkan dengan simbol "?". *Import* Numpy, kemudian gunakan fungsi replace, untuk mengganti "?" dengan "NaN" seperti contoh gambar dibawah. Numpy disini untuk menyesuaikan “NaN” menjadi bentuk yang bisa diterima oleh *dataframe*, oleh karena itu pada modul ini kita akan menggunakannya bersamaan dengan Pandas.
   Contoh Penulisan :

```
.replace(A, B, inplace = True) // Simbol A diganti menjadi B
```

Jawaban :

```
import numpy as np

# replace "?" to NaN
df.replace("?", np.nan, inplace = True)
df.head(5)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3

B. Gunakan fungsi Python, seperti contoh gambar dibawah.
Contoh Penulisan :

```
.isnull()
```

Fungsi ini akan memberikan keluaran *True* untuk data yang memiliki label NaN, dan *False* untuk data yang tidak memiliki label NaN. Gunakan fungsi tersebut, dengan variabel df. Masukan keduanya dalam variabel bernama missing_data.

Jawaban :

```
missing_data = df.isnull()

missing_data.head(5)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height
0	False	True	False	False	False	False	False	False	False	False	False	False	False
1	False	True	False	False	False	False	False	False	False	False	False	False	False
2	False	True	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False

C. Menggunakan *for loop* di Python, kita dapat dengan cepat mengetahui jumlah nilai yang hilang di setiap kolom. Seperti disebutkan di atas, *True* mewakili nilai yang hilang, *False* berarti nilai ada dalam kumpulan data. Gunakan kedua fungsi, bersamaan dengan *for loop* untuk menghitung *True* pada data

Fungsi berikut untuk mengubah kolom menjadi list, sehingga dapat di *looping*
Contoh Penulisan :

```
missing_data.columns.values.tolist()
```

Fungsi berikut untuk membaca jumlah *True* dan *False* pada kolom
Contoh Penulisan :

```
missing_data[column].value_counts()
```

Jawaban :

```
for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

```
symboling
False    205
Name: symboling, dtype: int64

normalized-losses
False    164
True      41
Name: normalized-losses, dtype: int64

make
False    205
Name: make, dtype: int64

fuel-type
False    205
Name: fuel-type, dtype: int64
```

D. Kita akan mengganti, nilai yang hilang dengan hasil mean pada setiap kolom.
Cth : normalized loss memiliki 41 data yang hilang, ini bisa digantikan dengan mencari *mean*, kemudian menggantikan data yang hilang dengan nilai *mean*
Gunakan fungsi pada gambar dibawah, untuk menghitung *mean* dari kolom "normalized-loss"
Contoh Penulisan :

```
df[data-label].astype("float").mean(axis=0)
```

Jawaban :

```
avg_norm_loss = df["normalized-losses"].astype("float").mean(axis=0)
print("Average of normalized-losses:", avg_norm_loss)
```

```
Average of normalized-losses: 122.0
```

Gunakan fungsi pada gambar dibawah, untuk menggantikan data yang hilang, dengan nilai *mean* dari kolom "normalized-loss"

```
df[data-label].replace(np.nan, avg_norm_loss, inplace=True)
```

Jawaban :

```
df["normalized-losses"].replace(np.nan, avg_norm_loss, inplace=True)
```

Gunakan fungsi *head* menggunakan Pandas, dan terlihat bahwa data yang tadinya NaN. Sekarang digantikan dengan hasil *Mean* dari kolom *normalized-loss*

Jawaban :

```
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height
0	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8
1	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3

E. Sekarang kita harus cek, apakah tipe data memiliki format yang tepat. Fungsi seperti gambar dibawah, digunakan untuk mengetahui tipe data.

Contoh Penulisan :

```
.dtype()
```

Fungsi seperti gambar dibawah digunakan untuk menggantikan tipe data lama, menjadi tipe data baru.

Contoh Penulisan :

```
.astype()
```

Sekarang gunakan fungsi pandas, untuk mengetahui setiap tipe dari data yang terkandung pada *data-set*

Jawaban :

```
df.dtypes
```

```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
```

Seperti yang bisa kita lihat di atas, beberapa kolom memiliki tipe data yang salah.

1. Variabel numerik harus memiliki tipe 'float' atau 'int'
2. Variabel dengan string seperti kategori harus memiliki tipe 'objek'.
3. Variabel 'bore' dan 'stroke' adalah nilai numerik yang menggambarkan mesin, jadi kita harus mengharapkan variabel tersebut bertipe 'float' atau 'int'; namun, mereka ditampilkan sebagai tipe 'objek'

Gunakan fungsi "astype()", untuk melakukan konversi ke tipe data yang benar

Jawaban :

```
df[["bore", "stroke"]] = df[["bore", "stroke"]].astype("float")
df[["normalized-losses"]] = df[["normalized-losses"]].astype("int")
df[["price"]] = df[["price"]].astype("float")
df[["peak-rpm"]] = df[["peak-rpm"]].astype("float")
```

```
df[["bore", "stroke", "normalized-losses", "price", "peak-rpm" ]].dtypes
```

```
bore                 float64
stroke               float64
normalized-losses      int32
price                float64
peak-rpm             float64
dtype: object
```

Pertanyaan

1. **Bersihkan data yang tersisa, pada modul ini, data yang belum dibersihkan adalah stroke, horsepower, peak-rpm. Gunakan metode statistik, dengan menghitung *mean* dari setiap kolom sebagai nilai pengganti data yang hilang.**
   Print hasil rata-rata setiap data yang belum dibersihkan, kemudian tampilkan *data frame* data dengan memakai fungsi *head*.

   Jawaban :

```
avg_stroke_loss = df["stroke"].astype("float").mean(axis=0)
print("Average of normalized-losses:", avg_stroke_loss)

df["stroke"].replace(np.nan, avg_stroke_loss, inplace=True) # Stroke Column

avg_horsepower = df['horsepower'].astype('float').mean(axis=0)
print("Average horsepower:", avg_horsepower)

df['horsepower'].replace(np.nan, avg_horsepower, inplace=True) # horsepower

avg_peakrpm=df['peak-rpm'].astype('float').mean(axis=0)
print("Average peak rpm:", avg_peakrpm)

df['peak-rpm'].replace(np.nan, avg_peakrpm, inplace=True) #peak-rpm
```

```
Average of normalized-losses: 3.2554228855721337
Average horsepower: 104.25615763546799
Average peak rpm: 5125.369458128079
```

```
df[['stroke','horsepower','peak-rpm']].head()
```

	stroke	horsepower	peak-rpm
0	2.68	111	5000.0
1	2.68	111	5000.0
2	3.47	154	5000.0
3	3.40	102	5500.0
4	3.40	115	5500.0

2. **Tampilkan tipe data dari 'curb-weight' dan 'length', tanpa memasukan keseluruhan data pada dataset. Kemudian gunakan fungsi describe untuk mencari ringkasan statistik dari kolom data**

Jawaban :

```
print(df[['curb-weight', 'length']].dtypes)
```

```
curb-weight      int64
length         float64
dtype: object
```

```
df[['curb-weight', 'length']].describe()
```

	curb-weight	length
count	205.000000	205.000000
mean	2555.565854	174.049268
std	520.680204	12.337289
min	1488.000000	141.100000
25%	2145.000000	166.300000
50%	2414.000000	173.200000
75%	2935.000000	183.100000
max	4066.000000	208.100000

Exploratory Data Analysis

1. Penjelasan Singkat

Sebelum melatih *data-set* pada model *machine learning*, melakukan analisa pada *data-set* harus dilakukan untuk mencari pola dari fitur-fitur yang dianalisa pada *data-set*. Eksplorasi data membantu menciptakan tampilan kumpulan data sehingga memungkinkan analisis data yang lebih dalam, terutama mencari pola dan tren yang ingin diidentifikasi.

2. Langkah-Langkah Percobaan

Import Library

A. *Import library* bernama pandas sebagai pd, dan *import library* bernama numpy sebagai np.

Jawaban :

```
import pandas as pd
import numpy as np
```

B. *data source*: https://www.kaggle.com/statsakash/used-car-price-prediction

Unduh *data source* kemudian import dalam Python. Gunakan fungsi pandas sesuai gambar di bawah untuk membaca *dataset*, dan masukan kedalam variabel df.
Contoh Penulisan :

```
pd.read_csv(dataset_path, names)
```

Gunakan fungsi *head* pada pandas sesuai gambar di bawah, untuk membaca lima data pertama.

Jawaban :

```
path = r"automobileEDA.csv"

df = pd.read_csv(path)

df.head(5)
```

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	...
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	...
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	...

C. *Import* kedua *library* sebagai berikut, pertama uduh seaborn sebagai sns, dan uduh matplotlib.pyplot sebagai plt. Kemudian, tambahkan inline pada matplotlib seperti contoh dibawah, supaya jupyter notebook bisa melakukan *plotting* pada *notebook*.

Contoh Penulisan :

```
%matplotlib inline
```

Jawaban :

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Exploratory menggunakan regresi dan korelasi

A. Variabel numerik kontinu adalah variabel yang mungkin berisi nilai apa pun dalam rentang tertentu. Variabel numerik kontinu dapat memiliki tipe "int64" atau "float64". Cara yang bagus untuk memvisualisasikan variabel-variabel ini adalah dengan menggunakan *scatter plots* dengan garis pas

Kita akan mulai dengan memahami hubungan (linier) antara variabel *engine-size* dan *price*. Kita dapat melakukan ini dengan menggunakan *regplot*, yang mem-plot *scatter plot* ditambah garis regresi yang sesuai untuk data. Gunakan

fungsi dibawah untuk melakukan plot regresi dengan *regplot*, dan memvisualisasi hasil analisa yang dilakukan

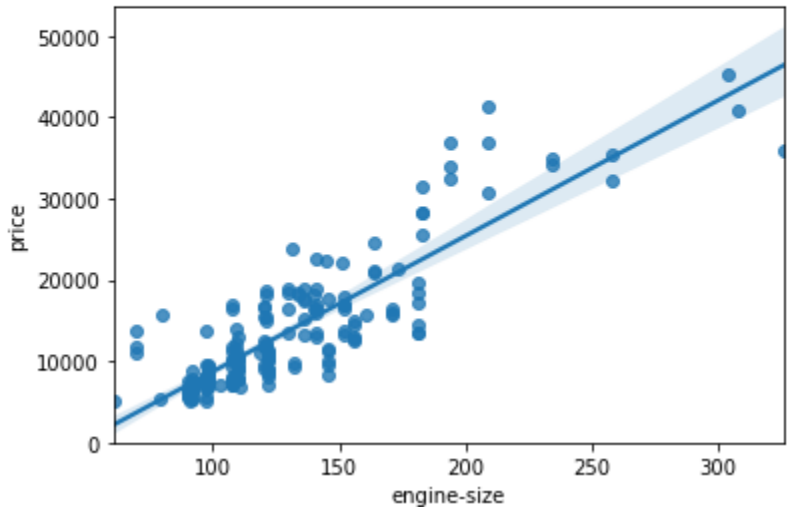
Contoh Penulisan :

```
sns.regplot(x="engine-size", y="price", data=df)
plt.ylim(0,)
```

Jawaban :

```
sns.regplot(x="engine-size", y="price", data=df)
plt.ylim(0,)
```

```
(0.0, 53582.19656338086)
```



B. Saat ukuran mesin naik (*engine-size*), harganya naik (*price*). Hal ini menunjukkan korelasi positif antara kedua variabel ini. Terlihat Ukuran mesin merupakan prediktor harga yang cukup bagus karena garis regresi hampir menyerupai garis diagonal yang sempurna.

   Kita dapat menggunakan pandas, untuk memverifikasi bahwa korelasi yang didapat adalah positif dengan cara seperti dibawah.

   Contoh Penulisan :

```
df[['engine-size', 'price']].corr()
```

Jawaban :

```
df[['engine-size', 'price']].corr()
```

	engine-size	price
engine-size	1.000000	0.872335
price	0.872335	1.000000

Kita dapat memeriksa korelasi antara *engine-size* dan *price* dan terlihat bahwa betul korelasi yang dihasilkan positif, dengan nilai 0,87

Exploratory menggunakan Boxplots

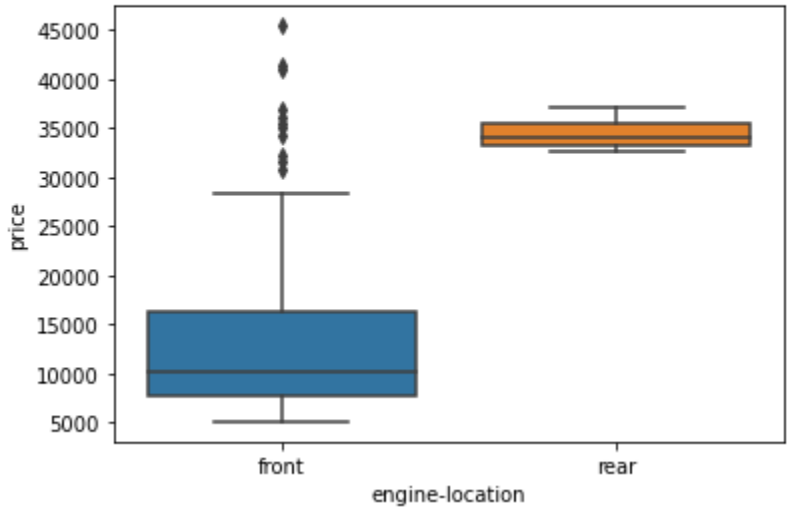
A. Variabel Kategori, adalah variabel yang menggambarkan 'karakteristik' dari unit data, dan dipilih dari sekelompok kecil kategori. Variabel kategori dapat memiliki tipe "objek" atau "int64". Cara yang baik untuk memvisualisasikan variabel kategori adalah dengan menggunakan boxplots

Gunakan fungsi dibawah untuk menghasilkan *plot* yang menjelaskan relasi antara *engine-location* dan *price*.

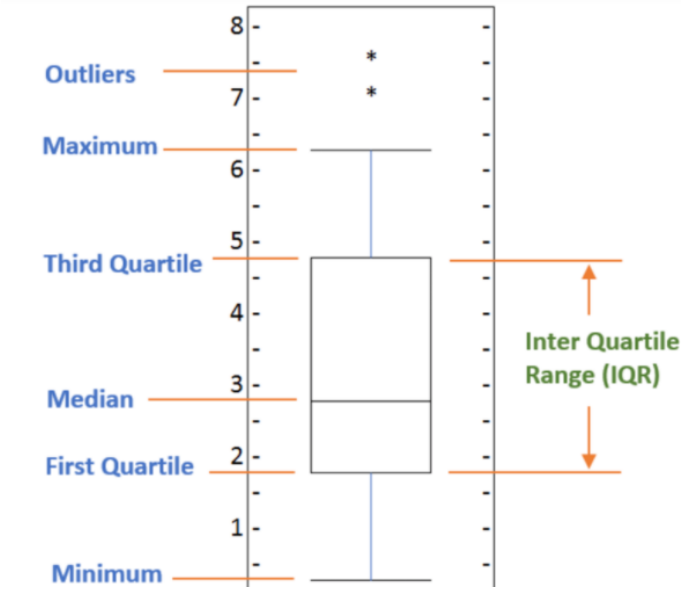
Jawaban :

```
sns.boxplot(x="engine-location", y="price", data=df)
```

```
<AxesSubplot:xlabel='engine-location', ylabel='price'>
```



Box Plot adalah cara untuk merepresentasikan distribusi data kedalam 5 dimensi utama seperti gambar dibawah.

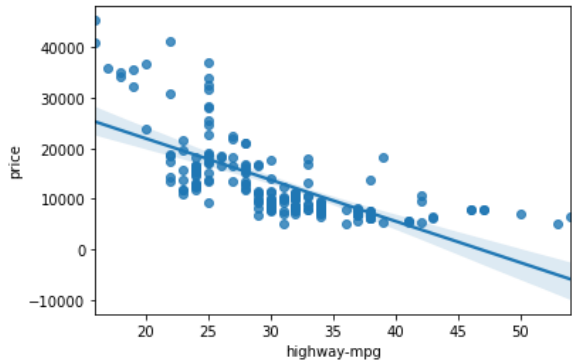


A. **Minimum:** Angka terkecil dalam kumpulan data.
B. **Kuartil pertama:** Angka tengah antara minimum dan median.
C. **Kuartil kedua (Median):** Angka tengah dari kumpulan data (diurutkan).
D. **Kuartil ketiga:** Angka tengah antara median dan maksimum.
E. **Maksimum:** Angka tertinggi dalam kumpulan data.

Pertanyaan

1. Carilah korelasi, serta regresi dari *highway-mg* dan *price* pada *dataset* yang digunakan pada praktikum. Gambarkan hasil regresi dari kedua variabel dengan seaborn, dan tampilkan hasil korelasinya menggunakan Pandas.

```
In [17]: sns.regplot(x="highway-mpg", y="price", data=df)
Out[17]: <AxesSubplot:xlabel='highway-mpg', ylabel='price'>
```

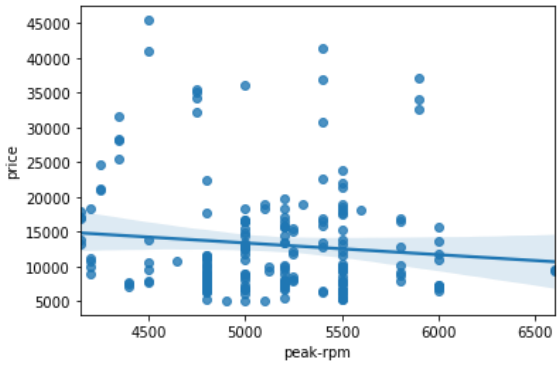



```
df[['highway-mpg', 'price']].corr()
```

	highway-mpg	price
highway-mpg	1.000000	-0.704692
price	-0.704692	1.000000

2. Carilah korelasi dan gambarkan regresi dari *peak-rpm* dan *price* pada *dataset* yang digunakan pada praktikum. Gambarkan hasil regresi dari kedua variabel dengan seaborn, dan tampilkan hasil korelasinya

```
sns.regplot(x="peak-rpm", y="price", data=df)
<AxesSubplot:xlabel='peak-rpm', ylabel='price'>
```



```
df[['peak-rpm','price']].corr()
```

	peak-rpm	price
peak-rpm	1.000000	-0.101616
price	-0.101616	1.000000

3. Tuliskan analisa apa yang bisa didapat dari 2 nomor sebelumnya, serta tuliskan manakah yang merupakan *predictor* harga terbaik, apakah *peak-rpm* atau *highway-mpg* ?
    a. **Pada nomor 1. Terlihat bahwa Saat *highway-mpg* naik, harganya turun. Ini menunjukan hubungan yang berbanding terbalik antara kedua variabel ini.**
    b. **Peak rpm bukan prediktor harga yang baik, hal ini terlihat dari *graph* bahwa garis regresinya mendekati horizontal.**
    c. **Highway mpg berpotensi menjadi penentu kenaikan harga.**

Data Modelling

1. Penjelasan Singkat

Ketika melakukan analisa pada *data-set*, kita akan mengembangkan model, model ini dipakai untuk membantu dalam memprediksi pengamatan di masa mendatang dari data yang kita miliki. Sebuah model akan membantu kita memahami hubungan yang tepat antara variabel yang berbeda dan bagaimana variabel ini digunakan untuk memprediksi hasilnya.

2. Langkah-Langkah Percobaan

Linear Regression

A. Salah satu contoh model data yang akan kita gunakan adalah regresi linier. Regresi linier adalah metode untuk membantu kita memahami hubungan antara dua variabel.

1. Variabel prediktor/independen (X)
2. Respon/variabel dependen (yang ingin kita prediksi)(Y)

Hasil dari Regresi Linier adalah fungsi linier yang memprediksi variabel respon (dependen) sebagai fungsi dari variabel prediktor (independen).

Regresi Linier, dapat dituliskan dalam bentuk fungsi sebagai berikut

$$Y = a + bX$$

1. a mengacu pada intersep dari garis regresi, yang memiliki arti bahwa nilai Y ketika X adalah 0
2. b mengacu pada kemiringan garis regresi, yang memiliki arti bahwa nilai Y berubah ketika X bertambah sebesar 1 unit

B. Pada modul ini kita akan menggunakan *library* pada *sci-kit* untuk memprediksi *dataset* menggunakan model regresi linear. Gunakan fungsi dibawah untuk *import library* regresi linear dari *sci-kit.*

Contoh Penulisan :

```
from sklearn.linear_model import LinearRegressione
```

Jawaban :

```
from sklearn.linear_model import LinearRegression
```

C. Buatlah suatu variabel bernama lm, dan masukan fungsi dari *linear regression* yang sudah kita *import* sebelumnya

Jawaban :

```
lm = LinearRegression()
lm
```

```
LinearRegression()
```

D. Pada contoh ini, kita ingin melihat bagaimana *highway-mpg* dapat membantu kita dalam memprediksi harga mobil. Dengan menggunakan regresi linier sederhana, kita akan membuat fungsi linier dengan "highway-mpg" sebagai variabel prediktor dan "harga" sebagai variabel respon.

Gunakan fungsi *data-frame* pada Pandas untuk memasukan *highway-mpg* sebagai variabel independen, dan *price* sebagai variabel dependen. Simpan *highway-mpg* yang diubah oleh Pandas menjadi Pandas *dataframe* kepada variabel X, dan *price* yang diubah oleh Pandas menjadi Pandas *series* kepada variabel Y

Jawaban :

```
X = df[['highway-mpg']]
Y = df['price']
```

E. Kemudian gunakan fungsi *fit* untuk memuat data pada model linier. Gunakan fungsi dibawah berbarengan dengan *library* regresi linear, dan variabel X dan Y

```
fit(X,y)
```

Jawaban :

```
lm.fit(X,Y)
```

```
LinearRegression()
```

F. Gunakan fungsi *predict* pada gambar dibawah, masukan kedalam variabel Y. Pada fungsi *predict* masukan nilai X, X adalah nilai *highway-mpg* karena pada ingin dicari korelasi antara *price* dan *high-way mpg*, *highway-mpg* adalah *independen variabel* yang berperan sebagai input yang menentukan perubahan harga

```
.predict()
```

Jawaban :

```
Y=lm.predict(X)
Y[0:5]
```

```
array([16236.50464347, 16236.50464347, 17058.23802179, 13771.3045085 ,
       20345.17153508])
```

G. Gunakan fungsi di bawah untuk mencari Y intercept dari regresi linear yang dihasilkan

```
.intercept_
```

Jawaban :

```
lm.intercept_
```

```
38423.305858157386
```

H. Setelah melakukan beberapa *run* pada program-program diatas, tuliskan fungsi regresi linear yang dihasilkan dari program diatas

Jawaban :

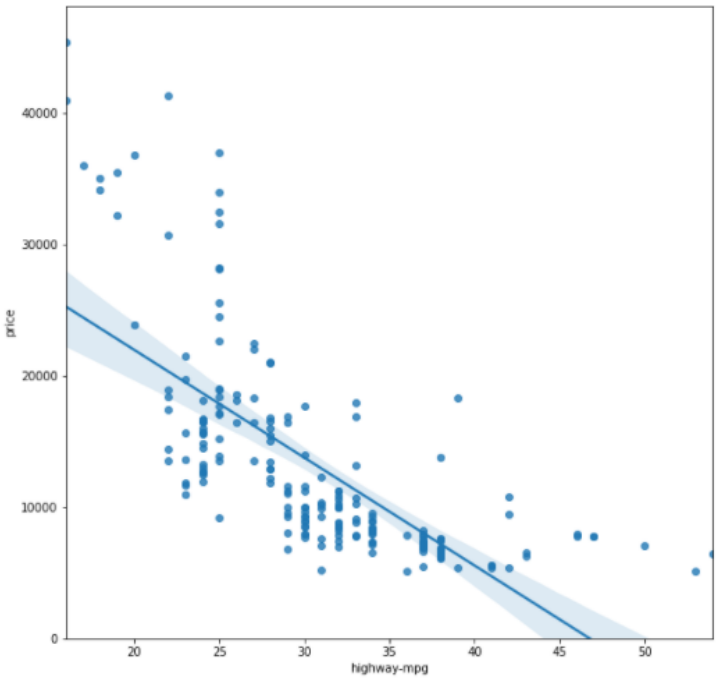
price = 38423.31 - 821.73 x **highway-mpg**

I. Untuk memudahkan visualisasi atas regresi linear yang dihasilkan, gunakan *library* seaborn untuk memvisualisasikan regresi linear yang dihasilkan

Jawaban :

```
import seaborn as sns
%matplotlib inline

width = 10
height = 10
plt.figure(figsize=(width, height))
sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```



Pertanyaan

1. Carilah regresi linear menggunakan 'ukuran mesin' sebagai x-axis dan 'harga' sebagai y-axis. Kemudian gunakan fungsi *linear regression*, untuk *fit* kedua data dan lakukan prediksi. Gunakan hasil prediksi, untuk mencari *coefficient*, dan y-*intercept*.

```
linearRegression = LinearRegression()
y = df["price"]
x = df[["engine-size"]]
linearRegression.fit(x,y)
predict=linearRegression.predict(X)
predict
```

```
array([-3458.11848261, -3458.11848261, -3624.9784983 , -2957.53843554,
       -4292.41856107, -3791.838514  , -3791.838514  , -3791.838514  ,
       -4626.13859245, -3124.39845123, -3124.39845123, -3291.25846692,
       -3291.25846692, -3791.838514  , -4292.41856107, -4292.41856107,
       -4626.13859245,   880.24192536,  -788.35823155,  -788.35823155,
       -1122.07826293, -1622.65831001, -2957.53843554, -1622.65831001,
       -1622.65831001, -1622.65831001, -2957.53843554, -2957.53843554,
       -3958.69852969,  1047.10194106, -1622.65831001,  -955.21824724,
       -2290.09837277, -2290.09837277, -2290.09837277, -2290.09837277,
       -2456.95838846, -2456.95838846, -2456.95838846, -2456.95838846,
       -3291.25846692, -2790.67841985, -3124.39845123, -3124.39845123,
       -4792.99860814, -4792.99860814, -5126.71863953, -2790.67841985,
       -1622.65831001, -1622.65831001, -1622.65831001, -1622.65831001,
       -4125.55854538, -4125.55854538, -4125.55854538, -4125.55854538,
       -2623.81840416, -2623.81840416, -2623.81840416, -2623.81840416,
        -955.21824724, -2623.81840416, -3458.11848261, -1455.79829432,
       -3791.838514  , -3791.838514  , -3791.838514  , -3791.838514  ,
       -4959.85862384, -4959.85862384, -5293.57865522, -5293.57865522,
       -3958.69852969, -1122.07826293, -1622.65831001, -1622.65831001,
       -2957.53843554, -2957.53843554, -2623.81840416, -3958.69852969,
       -3958.69852969, -3958.69852969, -2623.81840416, -2623.81840416,
       -2957.53843554, -2957.53843554, -1789.5183257 ,   379.66187829,
       -1789.5183257 , -1789.5183257 , -1789.5183257 , -1789.5183257 ,
       -1789.5183257 , -1789.5183257 , -1789.5183257 , -1789.5183257 ,
       -2290.09837277, -2290.09837277, -4292.41856107, -4292.41856107,
       -3791.838514  , -3791.838514  , -4125.55854538, -3791.838514  ,
       -3958.69852969, -2456.95838846, -3958.69852969, -3791.838514  ,
       -3958.69852969, -2456.95838846, -3958.69852969, -3791.838514  ,
       -3958.69852969, -2456.95838846, -3958.69852969, -1122.07826293,
       -2957.53843554, -1622.65831001, -1622.65831001, -1622.65831001,
       -2957.53843554, -3958.69852969, -3458.11848261, -3791.838514  ,
       -3791.838514  , -3791.838514  , -2790.67841985, -2790.67841985,
       -3291.25846692, -3291.25846692, -3291.25846692, -3291.25846692,
       -3624.9784983 , -3624.9784983 , -1956.37834139, -2790.67841985,
```

```
linearRegression.coef_
```

```
array([166.86001569])
```

```
linearRegression.intercept_
```

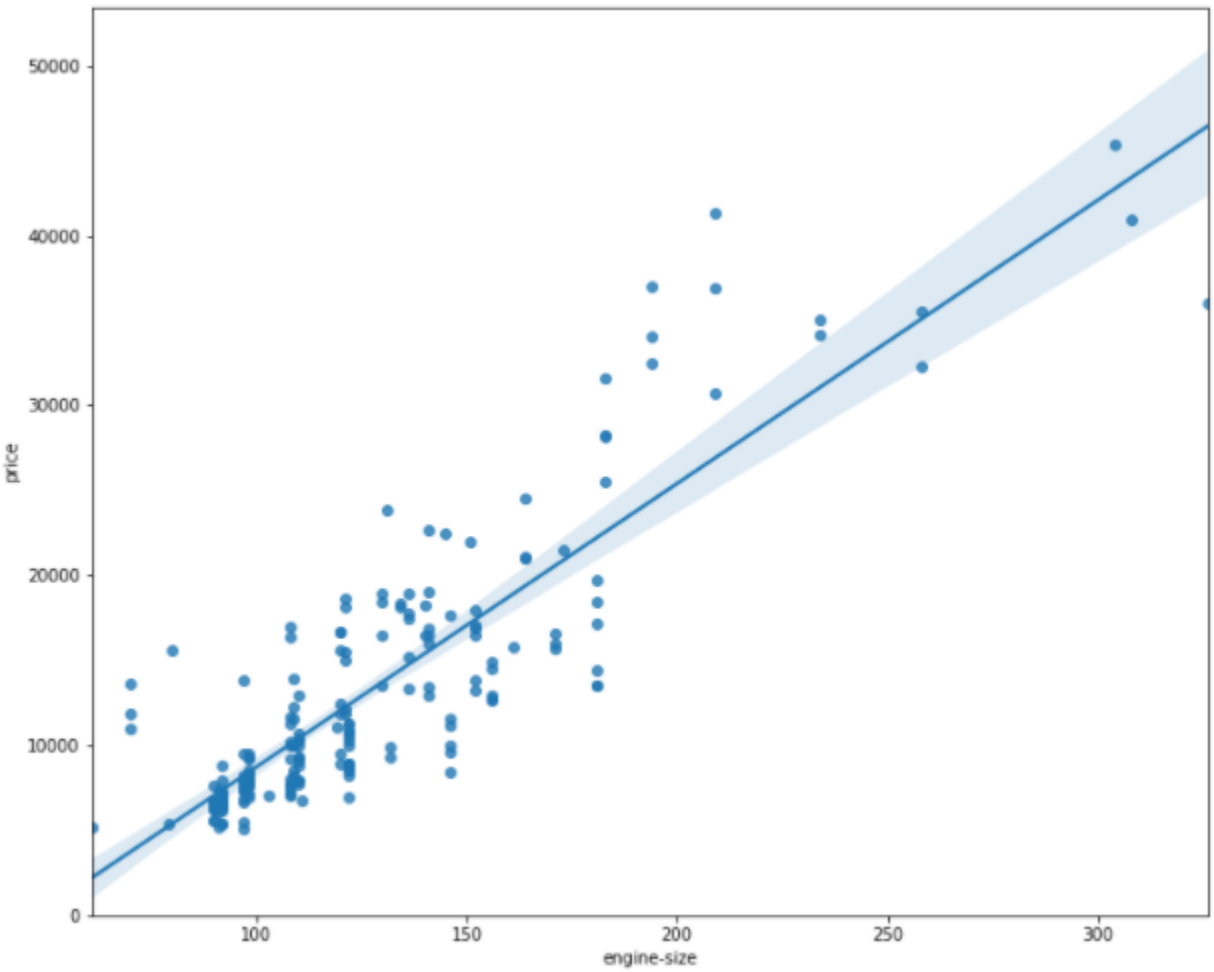
```
-7963.338906281049
```

2. Gambarkan visualisasi regresi linear dari kedua variabel yang diberikan

```
import seaborn as sns
%matplotlib inline

width = 12
height = 10
plt.figure(figsize=(width, height))
sns.regplot(x="engine-size", y="price", data=df)
plt.ylim(0,)
```

```
(0.0, 53392.807465897786)
```



Highest Symptom in Diabetic Patients using Logistic Regression

1. Penjelasan Singkat

Kita akan menggunakan *dataset* PIMA diabetes India, yang didapat dari *kaggle*. Menggunakan informasi yang didapat kita akan mencoba untuk menghasilkan *coefficient* dari hasil pelatihan dengan *logistic regression*, untuk melihat gejala apa yang terlihat tertinggi pada pasien yang terdiagnosa diabetes.

2. Langkah-Langkah Percobaan

Setup

A. *Import library* yang diperlukan untuk projek pada modul ini. Pada modul ini kita akan menggunakan *library* sebagai berikut Pandas, Numpy, Seaborn, Matplotlib. Kemudian kita akan menggunakan Scikit, untuk melakukan *import* pada model *logistic regression* seperti contoh gambar dibawah

Contoh Gambar :

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.linear_model import LogisticRegression
```

B. Unduh dataset yang akan kita gunakan dari link berikut : https://www.kaggle.com/uciml/pima-indians-diabetes-database/data, kemudian jadikan satu folder dengan *notebook* yang kita gunakan. Gunakan fungsi Pandas untuk membaca dataset, dan simpan hasil pembacaan di variabel diabetesDF.

Contoh Penulisan :

```
diabetesDF = pd.read_csv('diabetes.csv')
```

Jawaban :

```
diabetesDF.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

C. Kita akan mencari nilai yang hilang, dengan melakukan looping untuk mencari simbol tanda tanya, kemudian kita akan menggantinya dengan nilai null. False akan melambambangkan tidak ada null, dan True akan melambangkan null yang berarti data tersebut tidak ada nilainya

Jawaban :

```
diabetesDF.replace("?", np.nan, inplace = True)
```

```
missing_data = diabetesDF.isnull()
```

```
for column in missing_data.columns.values.tolist():
    print(column)
    print(missing_data[column].value_counts())
    print("")
```

```
Pregnancies
False    768
Name: Pregnancies, dtype: int64

Glucose
False    768
Name: Glucose, dtype: int64

BloodPressure
False    768
Name: BloodPressure, dtype: int64

SkinThickness
False    768
Name: SkinThickness, dtype: int64

Insulin
False    768
Name: Insulin, dtype: int64

BMI
False    768
Name: BMI, dtype: int64

DiabetesPedigreeFunction
False    768
Name: DiabetesPedigreeFunction, dtype: int64

Age
False    768
Name: Age, dtype: int64

Outcome
False    768
Name: Outcome, dtype: int64
```

Persiapan Analisa

A. Kita akan melakukan split pada *dataset* yang kita unduh, dataset yang dituduh memiliki 768 data. Kita akan bagi masing-masing data, dengan persentase 80% akan dialokasikan pada train data, dan 20 % akan dialokasikan pada test data.
   Pada variabel diabetesDF, set elemen hingga 613 dengan tanda kurung [ ], dan masukan pada variabel dfTrain. Pada variabel diabetesDF, set elemen dari 614 hingga 767.
   Jawaban :

```
dfTrain = diabetesDF[:613]
dfTest = diabetesDF[614:767]
```

B. Kita akan memisahkan label, dan fitur data pada bagian yang digunakan untuk Train dan Test. Kemudian kita akan konversikan menjadi bentuk Numpy Array agar bisa diterima model, label akan berisi outcome berdasarkan gejala-gejala yang terdapat di gejala pasien, sementara data akan berisi nilai dari gejala-gejala pada setiap pasien. Liat contoh gambar dibawah, untuk menjalankan perintahnya

Contoh Gambar:

```
trainLabel = np.asarray(dfTrain['Outcome'])
trainData = np.asarray(dfTrain.drop('Outcome',1))
testLabel = np.asarray(dfTest['Outcome'])
testData = np.asarray(dfTest.drop('Outcome',1))
```

C. Sekarang kita akan melakukan normalisasi pada train dan test data. Ini juga akan memudahkan kita untuk memahami pentingnya setiap fitur yang dianalisa, dan akan terlihat saat kita akan melihat bobot model. Kita akan menormalkan data sedemikian rupa sehingga setiap variabel memiliki rata-rata 0 dan standar deviasi 1. Jalankan perintah, seperti contoh gambar di bawah.

Contoh Gambar :

```
means = np.mean(trainData, axis=0)
stds = np.std(trainData, axis=0)

trainData = (trainData - means)/stds
testData = (testData - means)/stds
```

D. Sekarang kita akan melakukan *train model* dengan *Logistic Regression*. Jika pada modul sebelumnya, kita menggunakan *Linear Regression*, pada modul ini kita menggunakan *Logistic Regression*.
Logistic Regression adalah metode statistik untuk memprediksi kelas biner. Variabel yang dihasilkan bersifat biner, dimana hanya terdapat dua kelas. Sifat

dari *logistic regression* ini yang membuat algoritma ini banyak digunakan untuk menghitung probabilitas terjadinya suatu peristiwa.

Pada modul ini kita akan mencari nilai *coefficient*, kenapa nilai *coefficient* dapat menghitung probabilitas gejala tertinggi adalah sebagai berikut:

1. Ini adalah fungsi *sigmoid*

$$p = 1/1 + e^{-y}$$

2. Mengingat pada modul sebelumnya, kita menggunakan *linear regression*. *Logistic regression*, menggunakan nilai y pada *linear regression* untuk dimasukan kepada fungsi *sigmoid*

$$y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots\ldots + \beta nXn$$

3. Sehingga terbentuk fungsi sebagai berikut, disini nilai *coefficient* dilambangkan sebagai beta yang akan kita cari

$$p = 1/1 + e^{-(\beta 0 + \beta 1X1 + \beta 2X2 \ldots\ldots \beta nXn)}$$

Jalankan perintah, seperti contoh gambar di bawah.

Jawaban :

```
diabetesCheck = LogisticRegression()
diabetesCheck.fit(trainData,trainLabel)
accuracy = diabetesCheck.score(testData,testLabel)
print("accuracy = ",accuracy * 100,"%")
```

E. Kemudian kita akan mengambil koefisien-koefisien yang dihasilkan oleh pelatihan dengan dataset dengan *logistic regression*, yang dilakukan sebelumnya. Nilai koefisien yang dihasilkan, akan kita simpan. Koefisien dengan nilai menuju positif, akan memberikan kesimpulan bahwa data tersebut mempengaruhi dalam kemungkinan orang terdiagnosa diabetes. Sementara koefisien dengan nilai menuju negatif, akan memberikan kesimpulan, bahwa data tersebut tidak mempengaruhi dalam kemungkinan orang terkena diabetes.

Jawaban :

```
coeff = list(diabetesCheck.coef_[0])
coeff
```

```
[0.4033431541928464,
 1.0569388276314948,
 -0.1985620869109347,
 -0.039043993148244034,
 -0.09599367455555358,
 0.8008181382199369,
 0.3568119130224063,
 0.11133034047275302]
```

F. Kemudian kita akan menghilangkan *row* untuk bagian *outcome*, karena disini kita ingin melihat gejala yang mempengaruhi kemungkinan pasien diabetes. *Outcome* tidak diperlukan

Jawaban :

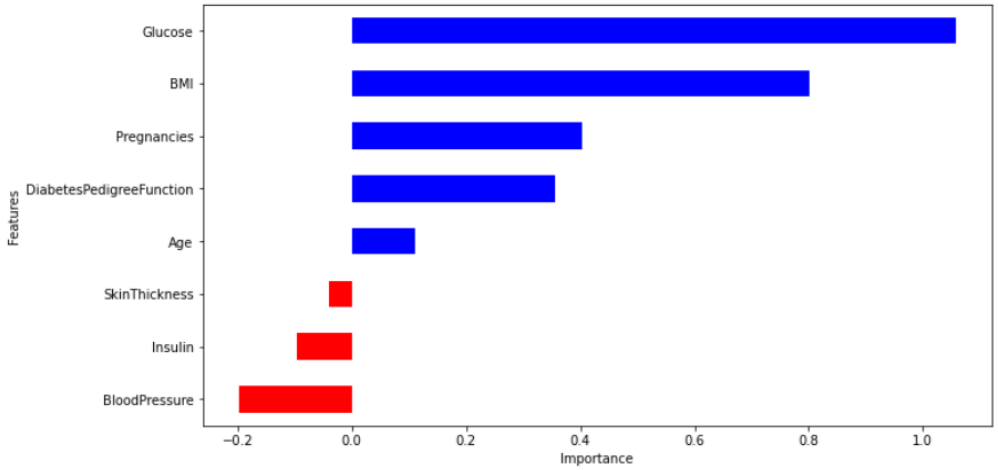
```
labels = list(dfTrain.drop('Outcome',1).columns)
labels
```

```
['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age']
```

G. Kita akan melakukan plot dengan bar graph, menggunakan nilai-nilai koefisien yang dihasilkan. Setiap nilai koefisien, akan menentukan pengaruhnya pada pasien yang terkena diabetes.

Jawaban :

```
features = pd.DataFrame()
features['Features'] = labels
features['importance'] = coeff
features.sort_values(by=['importance'], ascending=True, inplace=True)
features['positive'] = features['importance'] > 0
features.set_index('Features', inplace=True)
features.importance.plot(kind='barh', figsize=(11, 6),color =
                         features.positive.map({True: 'Blue', False: 'red'}))
plt.xlabel('Importance')
```


Glucose
BMI
Pregnancies
DiabetesPedigreeFunction
Age
SkinThickness
Insulin
BloodPressure
Features
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
Importance