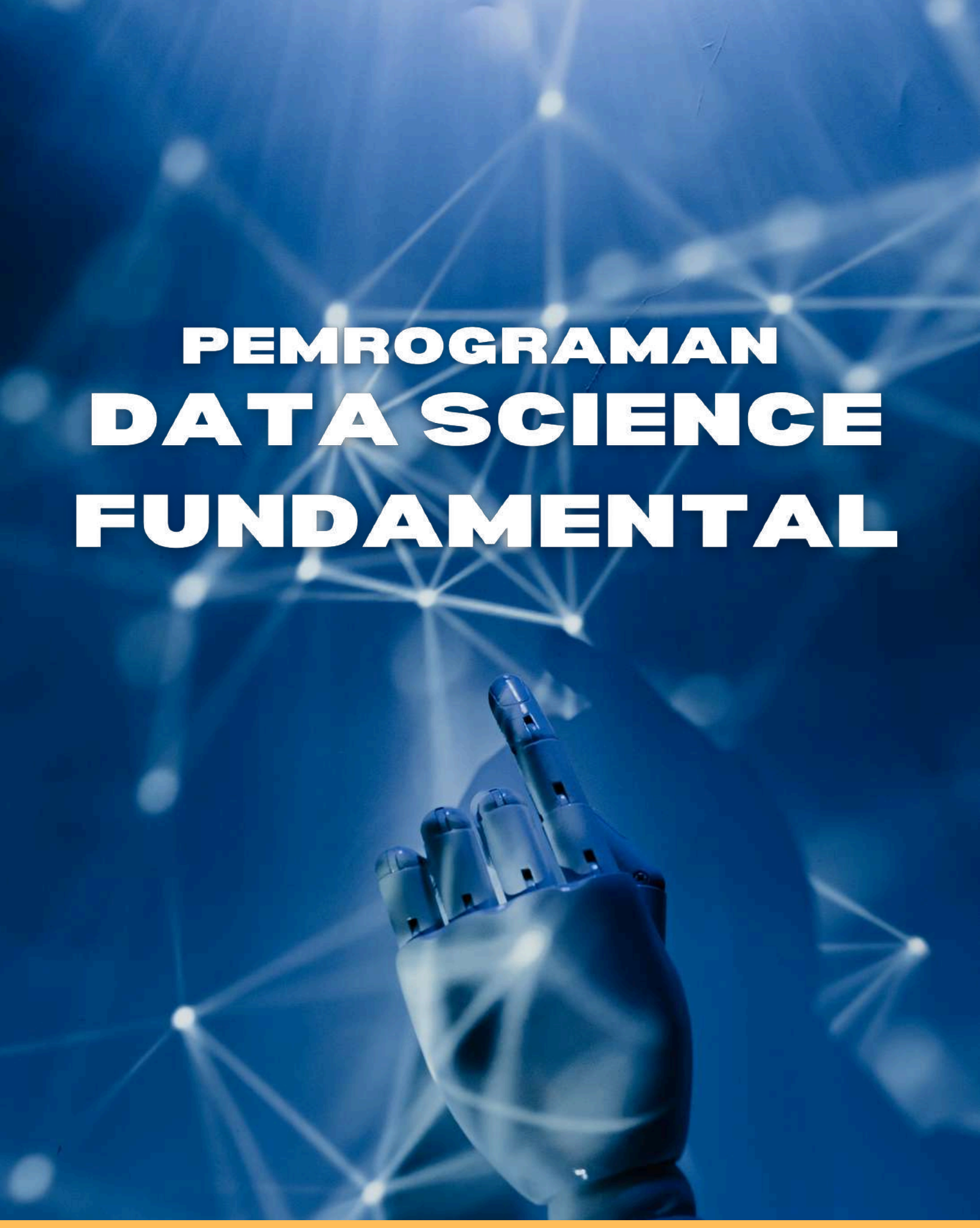


PEMROGRAMAN DATA SCIENCE FUNDAMENTAL

Amir Mahmud Husein, M.Kom

Dr.Abdi Dharma, M.Kom

PEMROGRAMAN DATA SCIENCE FUNDAMENTAL

PENULIS

Amir Mahmud Husein, M.Kom
Dr. Abdi Dharma, M.Kom
Mawaddah Harahap, M.Kom

EDITOR

Felix Thedora, S.Kom

PENERBIT

UNPRI PRESS

ANGGOTA IKAPI

UNIVERSITAS
PRIMA INDONESIA
UNPRI PRESS

PEMROGRAMAN DATA SCIENCE FUNDAMENTAL

Penulis : *Amir Mahmud Husein, M.Kom*
Dr. Abdi Dharma, M.Kom
Mawaddah Harahap, M.Kom

Editor : Felix Thedora, S.Kom

Desain Isi : *Amir Mahmud Husein, M.Kom*
Desain Cover : Felix Thedora, S.Kom

PENERBIT :
UNPRI PRESS
(ANGGOTA IKAPI)

Alamat Redaksi
Kampus 2
Jl. Sampul No. 4 Medan

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga buku monograf yang berjudul "Pemrograman Data Science Fundamental" ini dapat disusun dan diselesaikan dengan baik. Buku ini hadir sebagai bentuk kontribusi nyata dalam pengembangan ilmu pengetahuan, khususnya di bidang Data Science yang terus berkembang secara dinamis dan multidisipliner.

Data Science saat ini tidak lagi menjadi sekadar tren, melainkan telah menjadi fondasi penting dalam pengambilan keputusan berbasis data di berbagai sektor. Oleh karena itu, pemahaman terhadap dasar-dasar serta penerapan praktis dalam Data Science menjadi hal yang sangat esensial bagi akademisi, praktisi, maupun pengambil kebijakan.

Buku ini disusun untuk memberikan pemahaman konseptual yang kuat serta pendekatan praktis yang dapat diterapkan secara langsung. Pembahasan mencakup Tools yang digunakan dalam Data Science, Teknik Pengumpulan Data, Data Preparation and Analysis, Deep Learning Modeling, Marketing Analysis Model, hingga Predictive Analysis. Seluruh topik disajikan dengan bahasa yang ringkas dan ilustrasi kasus yang relevan, sehingga dapat digunakan sebagai referensi dalam kegiatan pembelajaran, penelitian, maupun pengembangan proyek berbasis data.

Penulis berharap kehadiran buku ini dapat memberikan manfaat yang luas dan menjadi salah satu referensi penting dalam memperkuat literasi Data Science di Indonesia. Terima kasih kepada semua pihak yang telah memberikan dukungan dan kontribusi dalam penyusunan buku ini. Semoga buku ini dapat menginspirasi dan mendorong lahirnya inovasi baru di bidang Data Science.

Medan, 10 September 2024

Penulis

DAFTAR ISI

KATA PENGANTAR **I**

DAFTAR ISI **II**

DAFTAR GAMBAR **V**

DAFTAR TABEL **VI**

BAB I TOOLS DATA SCIENCE **1**

A. PENDAHULUAN 1

B. INTI 1

1. Capaian Pembelajaran 1
2. Materi 2
3. Uraian Materi 2
4. Forum Diskusi 36

C. PENUTUP 37

1. Rangkuman 37
2. Tes Formatif 37

DAFTAR PUSTAKA 38

BAB II TEKNIK PENGUMPULAN DATA **39**

A. PENDAHULUAN 39

B. INTI 39

1. Capaian Pembelajaran 39
2. Materi 39
3. Uraian Materi 40
4. Forum Diskusi 66

C. PENUTUP 66

1. Rangkuman 66
2. Tes Formatif 67

DAFTAR PUSTAKA **67**

BAB III DATA PREPARATION AND ANALYSIS **69**

A. PENDAHULUAN 69

B. INTI 69

1. Capaian Pembelajaran 69
2. Materi 69
3. Uraian Materi 70
C. PENUTUP 98
1. Rangkuman 98
2. Tes Formatif 99
DAFTAR PUSTAKA 101
BAB IV SHALLOW LEARNING MODELING **102**
A. PENDAHULUAN 102
B. INTI 102
1. Capaian Pembelajaran 102
2. Materi 102
3. Uraian Materi 103
4. Forum Diskusi 116
B. PENUTUP 116
1. Rangkuman 116
2. Tes Formatif 116
DAFTAR PUSTAKA 118
BAB V MARKETING ANALYSIS MODEL **119**
A. PENDAHULUAN 119
B. INTI 120
1. Capaian Pembelajaran 120
2. Materi 120
3. Uraian Materi 120
4. Forum Diskusi 137
B. PENUTUP 137
1. Rangkuman 137
2. Tes Formatif 137
DAFTAR PUSTAKA 139
BAB VI PREDICTIVE ANALYTICS **140**
A. PENDAHULUAN 140

B. INTI 141
1. Capaian Pembelajaran 141
2. Materi 141
3. Uraian Materi 141
4. Forum Diskusi 147
B. PENUTUP 148
1. Rangkuman 148
2. Tes Formatif 148
DAFTAR PUSTAKA 150

DAFTAR GAMBAR

Gambar 1. Pilihan versi Anaconda 3
Gambar 2. Kotak dialog versi Anaconda 3
Gambar 3. Perjanjian Licensi 3
Gambar 4. Pilhan instal 4
Gambar 5. Menentukan lokasi instalasi 4
Gambar 6. Pilihan pendekatan yang disarankan 5
Gambar 7. Pilihan opsional untuk Microsoft VSCode 5
Gambar 8. Penambahan Key baru dengan nama command 6
Gambar 9. Hasil akhir setelah penambahan key baru 6
Gambar 10. Tampilan anaconda prompt 7
Gambar 11. Membuat folder baru di prompt 8
Gambar 12. Browser Default (rekomendasi Google Chrome) 9
Gambar 13. Membuat file python baru dengan jupyter 9
Gambar 14. Tampilan berhasil membuat Notebook Python3 Jupyter pertama 9
Gambar 15. Fungsi pada NumPy 46
Gambar 16. Tensor 0D, 1D, 2D 103
Gambar 17. Tensor 3D 104
Gambar 18. Tensor 4D 105
Gambar 19. Tensor 5D 105
Gambar 20. Pembagian dataset untuk training dan testing 114
Gambar 21. 5 – Fold Cross Validation 144

DAFTAR TABEL

Tabel 1. Tipe Data 7
Tabel 2. Icon tampilan Notebook Python3 9
Tabel 3. Operator Aritmatika 13
Tabel 4. Operator Perbandingan 15
Tabel 5. Operator Penugasan 15
Tabel 6. Prioritas Eksekusi Operator di Python 16
Tabel 7. Daftar operasi dasar pada list python 25
Tabel 8. Python Expression 25
Tabel 9. Python Function 26
Tabel 10. Python Method 26
Tabel 11. Operasi Dasar Pada Tuple Python 28
Tabel 12. Indexing, Slicing dan Matrix Pada Tuple Python 28
Tabel 13. Fungsi Build-in Pada Tuple Python 28
Tabel 14. Fungsi Build-in Pada Dictionary 30
Tabel 15. Method Build-in Pada Dictionary 30
Tabel 16. Objek tanggal ke string 32
Tabel 17. Fungsi waktu pada Python 33
Tabel 18. Atribut struktur time 34
Tabel 19. Reservasi 76

BAB I
TOOLS DATA SCIENCE

A. PENDAHULUAN

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya. Dengan kode yang simpel dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari syntax error.

Semua materi praktikum Data Science Fundamental menggunakan versi Anaconda. Download paket Anaconda di https://www.anaconda.com/products/individual#windows. Anaconda adalah manajer paket, manajer lingkungan, dan distribusi Python yang berisi kumpulan banyak paket sumber terbuka. Ketika akan mengerjakan proyek Data Science (ilmu data), kita akan memerlukan banyak paket yang berbeda (numpy, scikit-learn, scipy, panda untuk beberapa paket lain), yang telah diinstal sebelumnya dengan instalasi Anaconda. Jika Anda memerlukan paket tambahan setelah menginstal Anaconda, Anda dapat menggunakan manajer paket Anaconda, conda, atau pip untuk menginstal paket tersebut. Ini sangat menguntungkan karena kita tidak perlu mengelola dependensi antara beberapa paket sendiri. Conda bahkan memudahkan untuk beralih antara Python 2 dan 3 (New Environment). Bahkan, instalasi Anaconda juga merupakan cara yang disarankan untuk menginstal Notebook Jupyter yang dapat Anda pelajari lebih lanjut di sini Course Data Science Fundamental.

B. INTI

1. Capaian Pembelajaran
    a. Mampu memahami perintah dasar bahasa pemrograman python, memahami type data dan variabel dalam pemrograman python, membuat program sederhana dengan bahasa

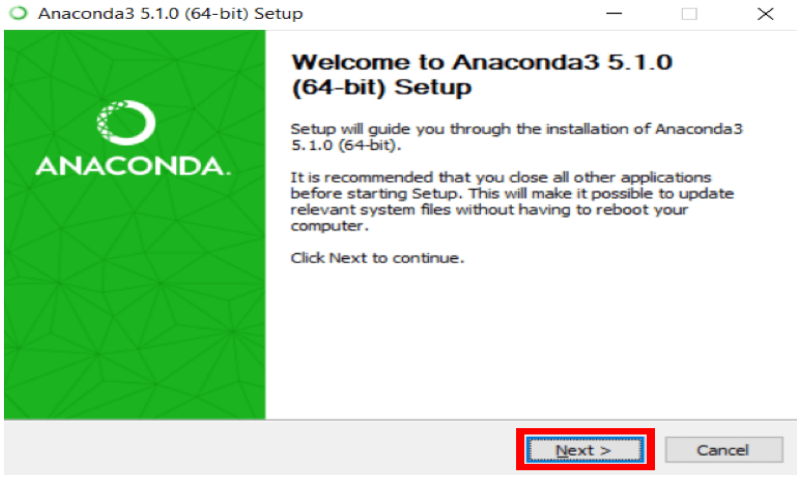
pemrogaman python.

b. Mampu menginstal Anaconda di Windows.
c. Mampu memahami dan menjelaskan fungsi tipe Data dan variable.
d. Mampu memahami dan membuat fungsi (function), operator, kondisi, perulangan, tipe list, type tuple, type directory, tanggal, fungsi waktu dan print dengan bahasa pemrograman Python.

2. Materi
   a. Menginstal Anaconda di Windows
   b. Type Data dan Variabel
   c. Fungsi (Function)
   d. Operator
   e. Kondisi
   f. Perulangan
   g. Tipe List
   h. Type Tuple
   i. Type Directory
   j. Fungsi Tanggal
   k. Fungsi Waktu
   l. Print
3. Uraian Materi
   a. Menginstal Anaconda di Windows
      1. Pastikan sudah mendownload paket Anaconda di SINI. Buka Situs Web Anaconda dan pilih penginstal grafis Python 3.x (A). Pilih versi Anaconda yang sesuai dengan OS anda, dalam praktikum ini menggunakan Windows 10 64bit, jika anda menggunakan versin Win 32 bit, silahkan pilih versi 32 bit. Default Anaconda akan menginstall paket Python 3.8.

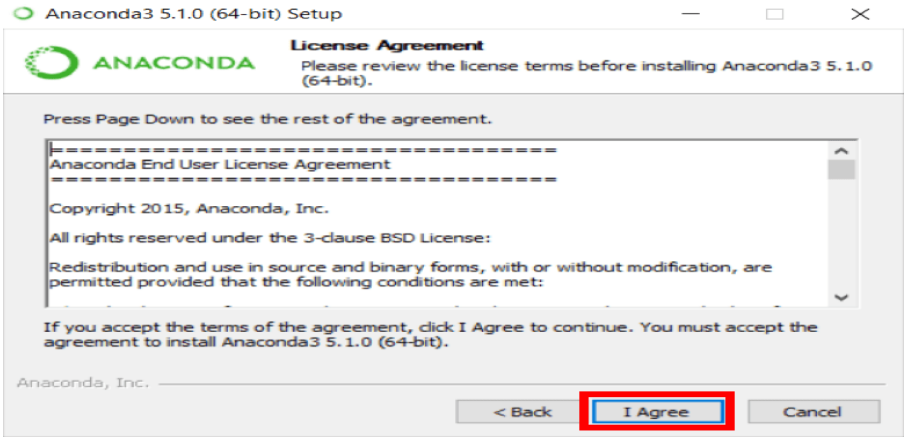
Gambar 1. Pilihan versi Anaconda

2. Buka kembali lokasi file Installer Anaconda sesuai lokasi folder penyimpanan anda, kemudian double klik file tersebut, maka akan muncul kotak dialog kemudian pilih NEXT.



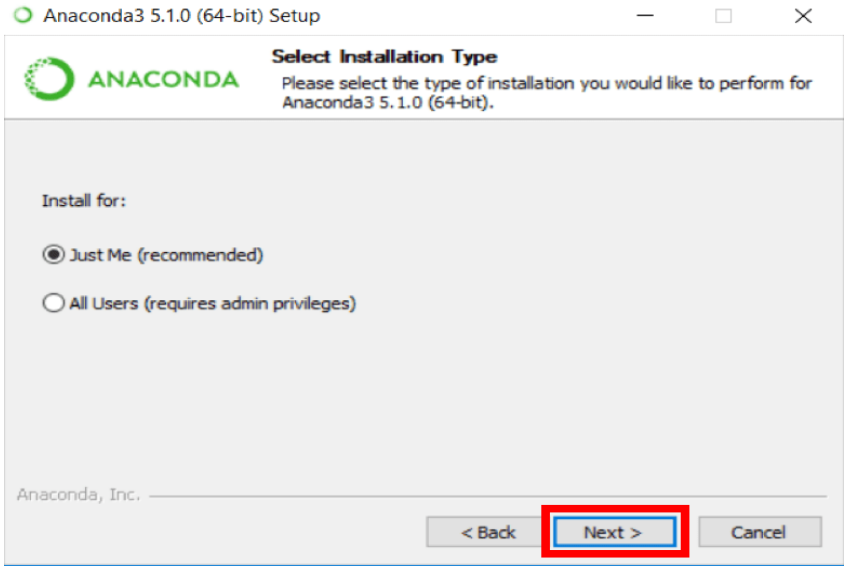
Gambar 2. Kotak dialog versi Anaconda

3. Perjanjian Licenci



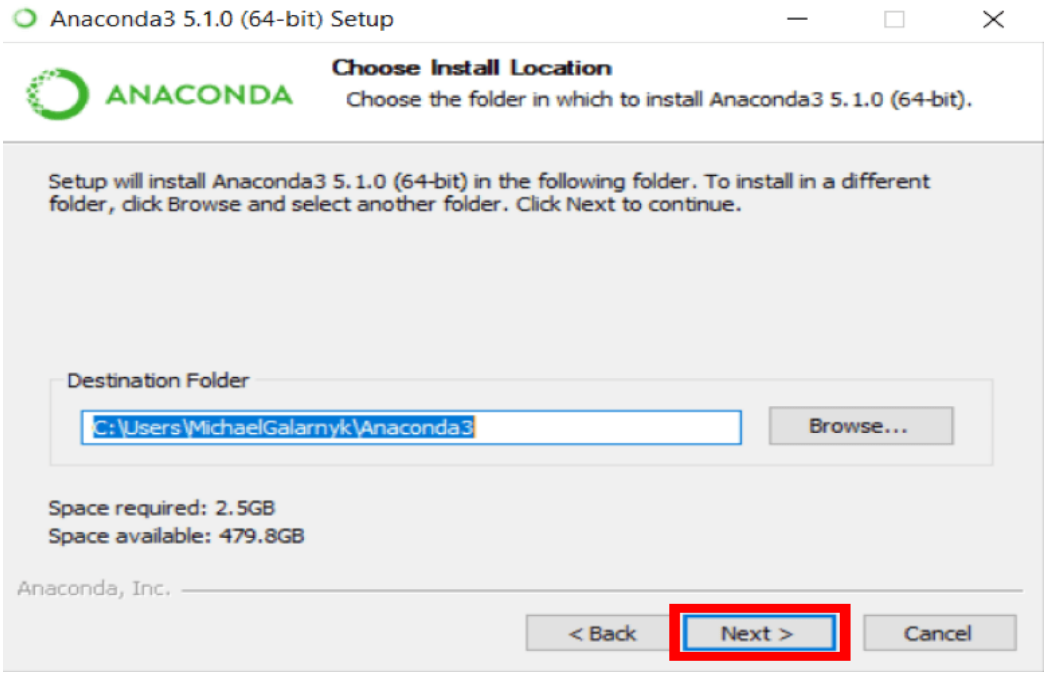
Gambar 3. Perjanjian Licensi

4. Pilih I Agree



Gambar 4. Pilhan instal

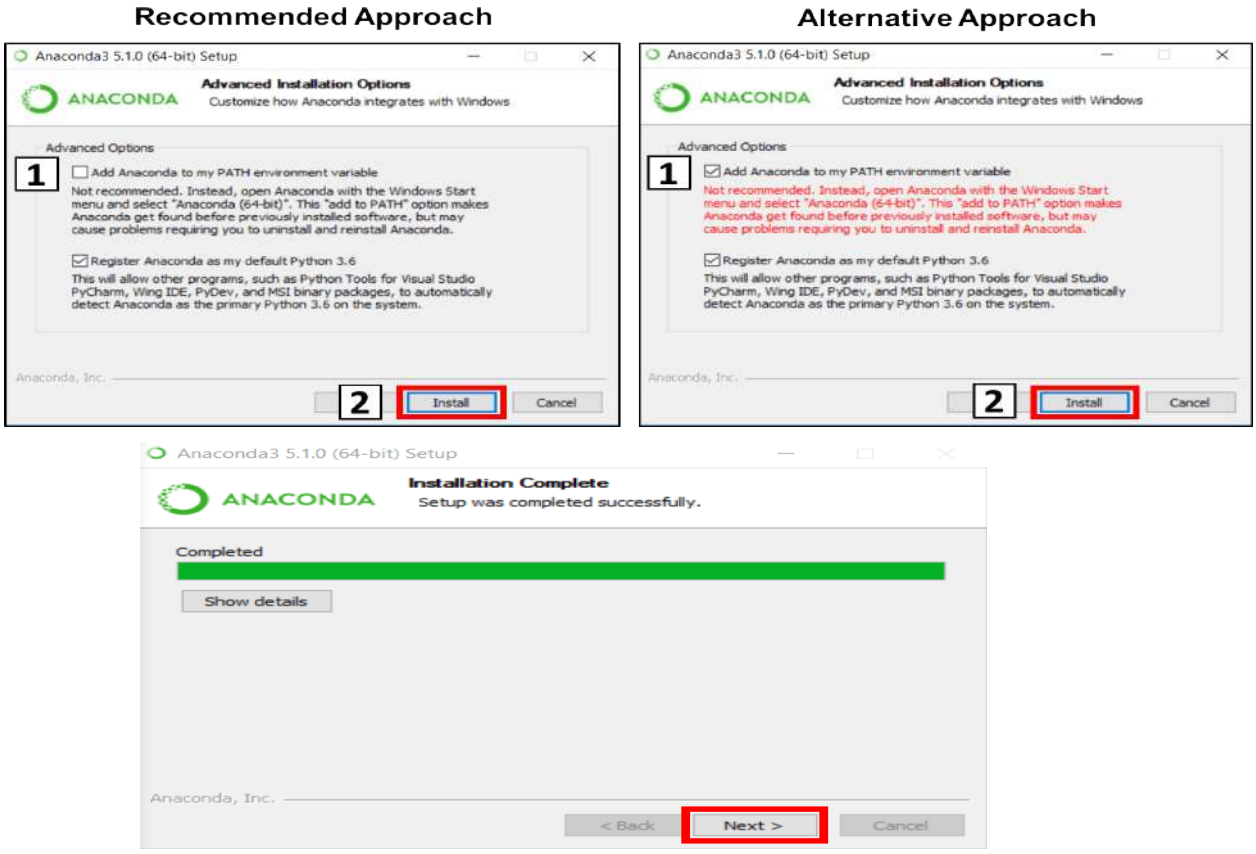
5. Pilih salah satu, kemudian Next
6. Tentukan lokasi instalasi; disarankan sesuaikan dengan folder default.



Gambar 5. Menentukan lokasi instalasi

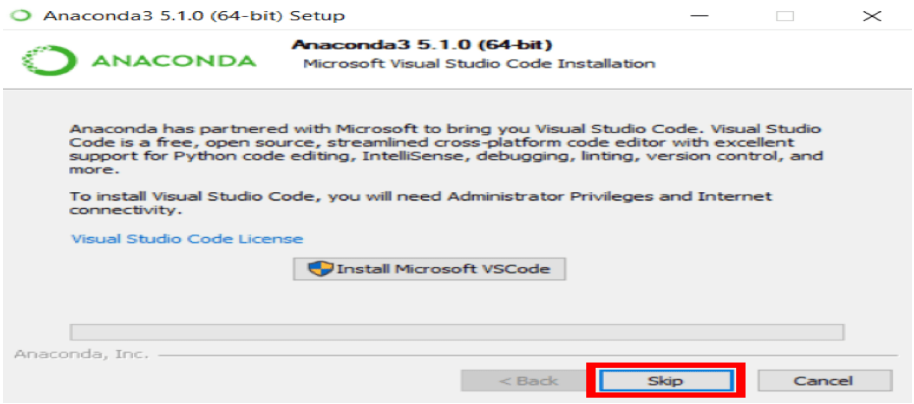
7. Ini adalah bagian penting dari proses instalasi. Pendekatan yang disarankan adalah tidak mencentang kotak untuk menambahkan Anaconda ke PATH Environment. Pilihan Ini berarti akan membuat kita harus menggunakan Anaconda Navigator

atau Anaconda Command Prompt (terletak di Start Menu di bawah "Anaconda") ketika Anda ingin menggunakan Anaconda (Anda selalu dapat menambahkan Anaconda ke PATH Anda nanti jika Anda tidak mencentang kotak). Jika Anda ingin dapat menggunakan Anaconda di command prompt Anda (atau git bash, cmder, PowerShell dll), silakan gunakan pendekatan alternatif dan centang kotak.



Gambar 6. Pilihan pendekatan yang disarankan

8. Kita dapat menginstal Microsoft VSCode jika diinginkan, tetapi ini opsional



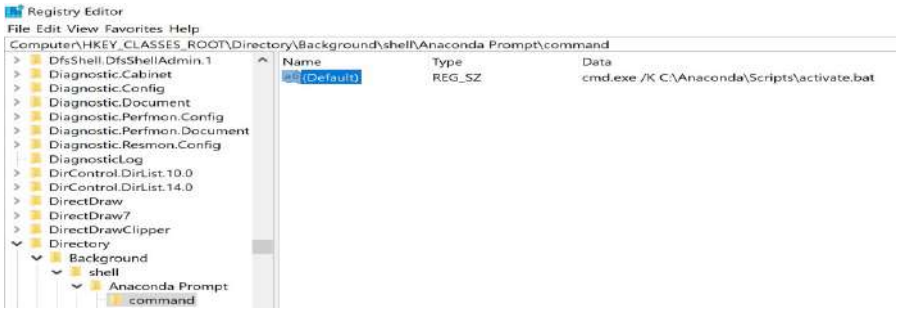
Gambar 7. Pilihan opsional untuk Microsoft VSCode

9. Selesai
10. Terakhir, menambahkan Anaconda Command Prompt ke konteks Klik Kanan.
    a. Klik Run (Windows R), ketikkan Regedit
    b. Buka HKEY_CLASSES_ROOT > Directory > Background > shell
    c. Klik Kanan pada folder Shell; New -> Key (masukkan nama Anaconda Prompt, atau dengan nama judul saat anda klik kanan nantinya)
    d. Kemudian tambahkan Key Baru pada dengan nama command kemudian atur nilainya (klik 2 x pada Default) dengan:

       cmd.exe /K C:\Anaconda\Scripts\activate.bat (Tulisan Bold disesuaikan dengan lokasi folder anaconda anda.

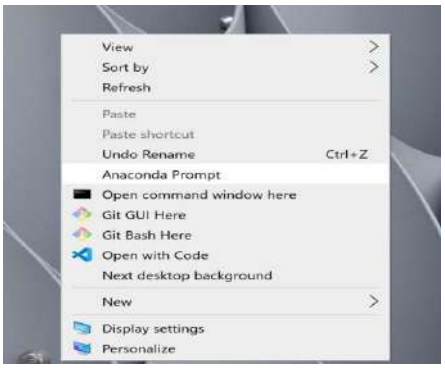
Gambar 8. Penambahan Key baru dengan nama command

    e. Hasil akhir seperti gambar dibawah ini



Gambar 9. Hasil akhir setelah penambahan key baru

    f. Contoh Penerapan; klik kanan di salah satu folder; maka akan muncul Anaconda Prompt



Gambar 10. Tampilan anaconda prompt

b. Type Data dan Variabel

Tipe data adalah suatu media atau memori pada komputer yang digunakan untuk menampung informasi. Python sendiri mempunyai tipe data yang cukup unik bila kita bandingkan dengan bahasa pemrograman yang lain.

Tabel 1. Tipe Data

Tipe Data	**Contoh**	**Penjelasan**
Boolean	True atau False	Menyatakan benar True yang bernilai 1 , atau salah False yang bernilai 0
String	"Ayo belajar Python" Menyatakan	karakter/kalimat bisa berupa huru angka, dll (diapit tanda " atau ') f
Integer	25 atau 1209	Menyatakan bilangan bulat
Float	3.14 atau 0.99	Menyatakan bilangan yang mempunyai koma
Hexadecimal	9a atau 1d3	Menyatakan bilangan dalam format heksa (bilangan berbasis 16)
Complex	1 + 5j	Menyatakan pasangan angka real dan imajiner
List	['xyz', 786, 2.23]	Data untaian yang menyimpan berbagai tipe data dan isinya bisa diubah-ubah
Tuple	('xyz', 768, 2.23)	Data untaian yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah
Dictionary	{'nama': 'adi','id':2}	Data untaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai
Set	{10,20,30}	Set (Himpunan) adalah tipe koleksi yang setiap elemennya bersifat unik

Variabel adalah lokasi memori yang dicadangkan untuk menyimpan nilai-nilai. Ini berarti bahwa ketika Anda membuat sebuah variabel Anda memesan beberapa ruang di memori. Variabel menyimpan data yang dilakukan selama program dieksekusi, yang nantinya isi dari variabel tersebut dapat diubah oleh operasi-operasi tertentu pada program yang menggunakan variabel.

Variabel dapat menyimpan berbagai macam tipe data. Di dalam pemrograman Python, variabel mempunyai sifat yang dinamis, artinya variabel Python tidak perlu didekralasikan tipe data tertentu dan variabel Python dapat diubah saat program dijalankan.

Beberapa aturan dalam membuat variabel dalam python, yaitu :

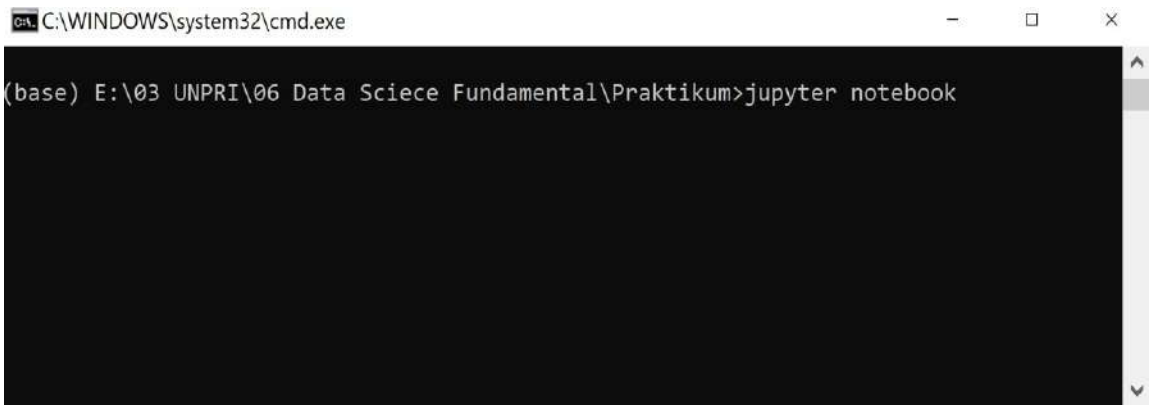
1. Karakter pertama harus berupa huruf atau garis bawah/underscore `_`
2. Karakter selanjutnya dapat berupa huruf, garis bawah/underscore `_` atau angka
3. Karakter pada nama variabel bersifat sensitif (case-sensitif). Artinya huruf kecil dan huruf besar dibedakan. Sebagai contoh, variabel `**namaDepan**` dan `**namadepan**` adalah variabel yang berbeda.

Untuk mulai membuat variabel di Python caranya sangat mudah, Anda cukup menuliskan variabel lalu mengisinya dengan suatu nilai dengan cara menambahkan tanda sama dengan `=` diikuti dengan nilai yang ingin dimasukan. Dalam membuat sebuah variabel tidak diperbolehkan menggunakan kata kunci berikut ini.

and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield

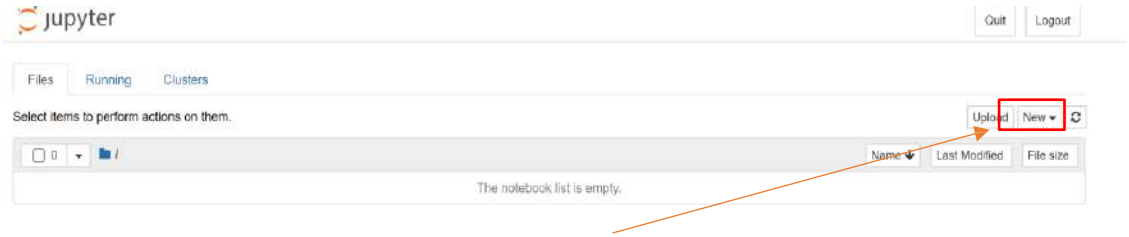
Praktik:

Buat folder baru di Drive D (lokasi penyimpanan modul praktikum); kemudian klik kanan→ Anaconda Prompt, maka akan muncul kotak dialog command prompt seperti berikut



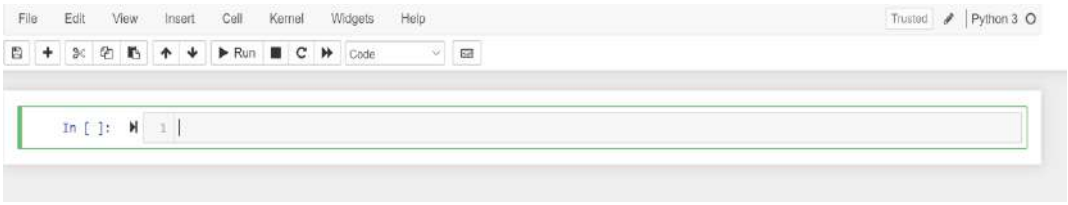
Gambar 11. Membuat folder baru di prompt

Ketikkan **jupyter notebook**, maka anda akan diarahkan ke browser Default (rekomendasi Google Chrome) seperti berikut:



Gambar 12. Browser Default (rekomendasi Google Chrome)

Untuk membuat file python baru dengan jupyter. Klik **New → Python 3**



Gambar 13. Membuat file python baru dengan jupyter

Selamat, anda sudah berhasil membuat Notebook Python3 Jupyter pertama. Secara default Notebook akan membuat nama file **Untitled.IPYNB**, kita dapat merubah nama dengan memilih menu File → Rename

Gambar 14. Tampilan berhasil membuat Notebook Python3 Jupyter pertama

Beberapa hal yang perlu di perhatikan pada icon tampilan Notebook Python3 mulai dari kiri.

Tabel 2. Icon tampilan Notebook Python3

Icon	Kegunaan
	Untuk menyimpan file
	Menambahkan Sel Kode Baru
	Memotong (menghapus) sel kode
	Mencopy sel kode atau perintah dalam sel kode
	Paste kode yang sudah di copy
	Menaikkan Sel Kode atau menurunkan sel kode
▶ Run	Menjalankan (eksekusi) perintah sel kode tertentu

Icon	Kegunaan
	Stop kode yang sedang berjalan
	Restart (Ulang) perintah sel kode
	Menjalankan semua perintah pada sel kode
	Code : memasukkan kode perintah python pada sel Markdown : digunakan untuk membuat judul, catatan, dokumentasi, dll. NBConvert : Jenis sel ini dapat digunakan untuk merender format kode yang berbeda ke dalam HTML atau LaTeX. Informasi ini disimpan dalam metadata buku catatan dan dikonversi dengan tepat

Type Data String

```
In [1]:
print("Data Science Fundamental, UNPRI") #print digunakan untuk menampilkan output (keluaran)

Data Science Fundamental, UNPRI

In [2]:
#tipe data String
strJudul="Data Science Fundamental"
print(strJudul)

print('Universitas Prima Indonesia')

#pemisah antara string \n
print(strJudul+ ' \nOleh' + ' \nUniversitas Prima Indonesia')
# Gabungan beberapa string menjadi satu
strJudul+ ' Oleh' + ' Universitas Prima Indonesia'

Data Science Fundamental
Universitas Prima Indonesia
Data Science Fundamental
Oleh
Universitas Prima Indonesia
Out[2]: 'Data Science Fundamental Oleh Universitas Prima Indonesia'
```

Boolean

```
In [3]:
# Boolean nilai antara True atau False
print(True)
print(5>3)
print(5<4)

True
True
False
```

Integer, Float, Hexadecimal dan Complex

```
In [4]:
#tipe data Integer
print(20)

#tipe data Float
print(3.14)

#tipe data Hexadecimal
print(hex(9))

number = 435
print(number, 'Nilai hex =', hex(number))

#tipe data Complex
print(5j)

20
3.14
0x9
435 Nilai hex = 0x1b3
5j
```

List dan Tuple

```
In [5]:
#tipe data List
print([1,2,3,4,5])
print(["satu", "dua", "tiga"])

#tipe data Tuple
print((1,2,3,4,5))
print(("satu", "dua", "tiga"))
```

```
[1, 2, 3, 4, 5]
['satu', 'dua', 'tiga']
(1, 2, 3, 4, 5)
('satu', 'dua', 'tiga')
```

Dictionary

```
In [6]:
#tipe data Dictionary
print({"nama":"Amir Mahmud", 'umur':39})
```

```
{'nama': 'Amir Mahmud', 'umur': 39}
```

```
In [7]:
#tipe data Dictionary dimasukan ke dalam variabel biodata
biodata = {"nama":"Amir Mahmud", 'umur':39, "Mata Kuliah" :"Data Science"} #proses inisialisasi variabel biodata

#proses pencetakan variabel biodata yang berisi tipe data Dictionary
print(biodata)

#fungsi untuk mengecek jenis tipe data. akan tampil <class 'dict'> yang berarti dict adalah tipe data dictionary
print(type(biodata))
```

```
{'nama': 'Amir Mahmud', 'umur': 39, 'Mata Kuliah': 'Data Science'}
<class 'dict'>
```

Set(Himpunan)

```
In [8]:
print(set ([10,20,30,40,50,60]))
```

```
{40, 10, 50, 20, 60, 30}
```

Variabel dengan Type Data

```
Variabel dan Type Data
```

```
In [9]:
number=12345
print('Variabel Integer :', number)
print(type(number))

nama='Data Science Fundamental'
print('Variabel String ', nama)
print(type(nama))

number=12.56
print('Variabel Float :', number)
print(type(number))

number=89e-4
print('Variabel Float :', number)
print(type(number))
```

```
Variabel Integer : 12345
<class 'int'>
Variabel String  Data Science Fundamental
<class 'str'>
Variabel Float : 12.56
<class 'float'>
Variabel Float : 0.0089
<class 'float'>
```

Konvresi type data digunakan untuk mengganti type data sebuah variabel dengan type data tertentu.

```
Konversi Type Data
```

```
# deklarasikan sebuah variabel type data String dengan nilai 12345
var='12345'
print("Type Data", var, " adalah ", type(var))

# konversi String menjadi type data integer
var=int(var)
print("Type Data", var, " adalah ", type(var))
```

```
Type Data 12345  adalah  <class 'str'>
Type Data 12345  adalah  <class 'int'>
```

```
# variabel dengan type float
number=12.5
print("Type Data", number, " adalah ", type(number))

number=int(number)
print("Type Data", number, " adalah ", type(number))
```

```
Type Data 12.5  adalah  <class 'float'>
Type Data 12  adalah  <class 'int'>
```

c. Fungsi (Function)

Fungsi adalah bagian atau blok program yang berisi satu tugas spesifik. Ketika dipanggil, fungsi ada yang menghasilkan atau mengembalikan nilai dan ada juga yang tidak. Nilai yang dihasilkan oleh fungsi disebut dengan istilah nilai balik (return value). Dalam beberapa bahasa pemrograman lain, fungsi dengan nilai balik disebut fungsi dan fungsi tanpa nilai balik disebut prosedur. fungsi hanya perlu didefinisikan satu kali, tapi dapat digunakan atau dipanggil berkali-kali. Dalam Phyton, fungsi didefinisikan menggunakan perintah def melalui bentuk umum berikut :

def nama_fungsi(parameter_1, parameter2, parameter_N...):

Fungsi global adalah fungsi yang didefinisikan di dalam suatu modul dan dapat dipanggil oleh fungsi lain, baik yang berada di dalam modul yang sama maupun modul lain.

```
# Fungsi Global
def jumlah(bil_1, bil_2):
    jum_bil=bil_1+bil_2 # penjumlahan variabel bil_1 dengan bil_2
    return jum_bil # pengembalian nilai ke baris pemanggil

```

```
# panggil fungsi global (jumlah)
a=10
b=50
hitung=jumlah(a,b)
print('Hasil Penjumlahan', a, '+', b, 'adalah :', hitung)
```

```
Hasil Penjumlahan 10 + 50 adalah : 60
```

```
def varFunc(name,*args):
        print("Ini Adalah Argumentasi Pertama "+str(name))
        #Menampilkan hasil argumentasi
        print(args)
```

```
print("Pertama:")
varFunc("Fungsi Pertama",2, 3, 4, 5)

print("Kedua:")
varFunc("Panggil ke-2","Data","Science","Fundamental")

```

```
Pertama:
Ini Adalah Argumentasi Pertama Fungsi Pertama
(2, 3, 4, 5)
Kedua:
Ini Adalah Argumentasi Pertama Panggil ke-2
('Data', 'Science', 'Fundamental')
```

Fungsi Lokal adalah fungsi yang didefinisikan di dalam fungsi lain. Fungsi lokal sering

disebut fungsi bersarang (nested function). berbeda dengan fungsi global, fungsi lokal hanya akan dikenal oleh fungsi luar tempat fungsi lokal tersebut didefinisikan.

```
# fungsi local
def lokal_func(var):
    def var_lokal():
        print("Hello, ", var)
    var_lokal()
```

```
lokal_func("Data Science Fundamental")
```

```
Hello,  Data Science Fundamental
```

Latihan:

Buatlah sebuah program yang menampilkan identitas anda dengan bentuk keluaran sebegai berikut

NIM :

Nama :

Email :

Universitas :

Fakultas :

Jurusan :

d. Operator

Operator adalah konstruksi yang dapat memanipulasi nilai dari operan. Sebagai contoh operasi 3 + 2 = 5. Disini 3 dan 2 adalah operan dan + adalah operator.

Tabel 3. Operator Aritmatika

Operator	Contoh	Penjelasan
Penjumlahan +	1 + 3 = 4	Menjumlahkan nilai dari masing-masing operan atau bilangan
Pengurangan -	4 - 1 = 3	Mengurangi nilai operan di sebelah kiri menggunakan operan di sebelah kanan
Perkalian *	2 * 4 = 8	Mengalikan operan/bilangan
Pembagian /	10 / 5 = 2	Untuk membagi operan di sebelah kiri menggunakan operan di sebelah kanan
Sisa Bagi %	11 % 2 = 1	Mendapatkan sisa pembagian dari operan di sebelah kiri operator ketika dibagi oleh operan di sebelah kanan
Pangkat **	8 ** 2 = 64	Memangkatkan operan disebelah kiri operator dengan operan di sebelah kanan operator
Pembagian Bulat //	10 // 3 = 3	Sama seperti pembagian. Hanya saja angka dibelakang koma dihilangkan

Penjumlahan

```
#Penjumlahan
print('Hasil Penjumlahan 13+2 =', 13 + 2)
apel = 7
jeruk = 9
buah = apel + jeruk #
print('Sisa Buah', buah)
```

```
Hasil Penjumlahan 13+2 = 15
Sisa Buah 16
```

Pengurangan dan Perkalian

```
#Pengurangan
hutang = 10000
bayar = 5000
sisaHutang = hutang - bayar
print("Sisa hutang Anda adalah ", sisaHutang)
```

```
Sisa hutang Anda adalah  5000
```

```
#Perkalian
panjang = 15
lebar = 8
luas = panjang * lebar
print('Hasil Luas=', luas)
```

```
Hasil Luas= 120
```

Pambagian, Sisa Bagi (Modulus) dan Pangkat

```
#Pembagian
kue = 16
anak = 4
kuePerAnak = kue / anak
print("Setiap anak akan mendapatkan bagian kue sebanyak ", kuePerAnak)
```

```
Setiap anak akan mendapatkan bagian kue sebanyak  4.0
```

```
#Sisa Bagi / Modulus
bilangan1 = 14
bilangan2 = 5
hasil = bilangan1 % bilangan2
print("Sisa bagi dari bilangan ", bilangan1, " dan ", bilangan2, " adalah ", hasil)
```

```
Sisa bagi dari bilangan  14  dan  5  adalah  4
```

```
#Pangkat
bilangan3 = 8
bilangan4 = 2
hasilPangkat = bilangan3 ** bilangan4
print('Hasil Pangkat', hasilPangkat)
```

```
Hasil Pangkat 64
```

Pembagian Bilangan Bulat

```
#Pembagian Bulat
a=10
b=3
print('Hasil Pembalian 10/3 adalah', a/b)
print('Hasil Pembulatan 10/3 adalah', 10//3)
#10 dibagi 3 adalah 3.3333. Karena dibulatkan maka akan menghasilkan nilai 3
```

```
Hasil Pembalian 10/3 adalah 3.3333333333333335
Hasil Pembulatan 10/3 adalah 3
```

1. Operator Perbandingan

   Operator perbandingan (comparison operators) digunakan untuk membandingkan suatu nilai dari masing-masing operan.

Tabel 4. Operator Perbandingan

Operator	Contoh	Penjelasan
Sama dengan ==	1 == 1	bernilai True Jika masing-masing operan memiliki nilai yang sama, maka kondisi bernilai benar atau True.
Tidak sama dengan !=	2 != 2	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Tidak sama dengan <>	2 <> 2	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Lebih besar dari >	5 > 3	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisi menjadi benar.
Lebih kecil dari <	5 < 3	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, maka kondisi menjadi benar.
Lebih besar atau sama dengan >=	5 >= 3	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, atau sama, maka kondisi menjadi benar.
Lebih kecil atau sama dengan <=	5 <= 3	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, atau sama, maka kondisi menjadi benar.

2. Operator Penugasan

   Operator penugasan digunakan untuk memberikan atau memodifikasi nilai kedalam sebuah variabel.

Tabel 5. Operator Penugasan

Operator	Contoh	Penjelasan
Sama dengan =	a = 1	Memberikan nilai di kanan ke dalam variabel yang berada di sebelah kiri
Tambah sama dengan +=	a += 2	Memberikan nilai variabel dengan nilai variabel itu sendiri ditambah dengan nilai di sebelah kanan
Kurang sama dengan -=	a -= 2	Memberikan nilai variabel dengan nilai variabel itu sendiri dikurangi dengan nilai di sebelah kanan
Kali sama dengan *=	a *= 2	Memberikan nilai variabel dengan nilai variabel itu sendiri dikurangi dengan nilai di sebelah kanan
Bagi sama dengan /=	a /= 4	Memberikan nilai variabel dengan nilai variabel itu sendiri dikali dengan nilai di sebelah kanan
Sisa bagi sama dengan %=	a %= 3	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya.
Pangkat sama dengan **=	a **= 3	Memberikan nilai variabel dengan nilai variabel itu sendiri dipangkatkan dengan nilai di sebelah kanan
Pembagian bulat sama dengan //=	a //= 3	Membagi bulat operan sebelah kiri operator dengan operan sebelah kanan operator kemudian hasilnya diisikan ke operan sebelah kiri

3. Prioritas Eksekusi Operator di Python

   Dari semua operator diatas, masing-masing mempunyai urutan prioritas yang nantinya prioritas pertama akan dilakukan paling pertama, begitu seterusnya sampai dengan prioritas terakhir.

Tabel 6. Prioritas Eksekusi Operator di Python

Operator	**Keterangan**
**	Aritmatika
~, +, -	Bitwise
*, /, %, //	Aritmatika
+, -	Aritmatika
>>, <<	Bitwise
&	Bitwise
^, |	Bitwise
<=, <, >, >=	Perbandingan
<>, ==, !=	Perbandingan
=, %=, /=, //=, -=, +=, *=, **=	Penugasan
is, is not	Identitas
in, not in	Membership(keanggotaan)
not, or, and	Logika

e. Kondisi

Pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang terjadi saat jalanya program dan menentukan tindakan apa yang akan diambil sesuai dengan kondisi.

Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar True. Jika kondisi bernilai salah False maka statement/kondisi if tidak akan di-eksekusi.

Kondisi IF

```
#Kondisi if adalah kondisi yang akan dieksekusi oleh program jika bernilai benar atau TRUE

nilai = 75
#jika kondisi benar/TRUE maka program akan mengeksekusi perintah dibawahnya
if(nilai > 70):
    print("Selamat, Anda Lulus") # Kondisi Benar, Dieksekusi
```

```
Selamat, Anda Lulus
```

```
#jika kondisi salah/FALSE maka program tidak akan mengeksekusi perintah dibawahnya
if(nilai > 80):
    print("Selamat, Anda Lulus") # Kondisi Salah, Maka tidak tereksekusi
```

Perintah pertama akan menampilkan output “Selamat, Anda lulus” karena kondisi BENAR, dimana nilai 75 lebih besar dari 70; untuk program kedua, tidak menampilkan

output karena kondisi SALAH, dimana 75 Tidak lebih besar dari nilai 80.

Kondisi if-else

```
nilai = 75
#jika kondisi benar/TRUE maka program akan mengeksekusi perintah dibawahnya
if(nilai > 70):
    print("Selamat, Anda Lulus") # Kondisi Benar, Dieksekusi
else:
    print("Maaf, Anda Tidak Lulus") # Kondisi Salah
```

```
Selamat, Anda Lulus
```

```
nilai=input('Masukkan Nilai: ')
nilai=int(nilai)
if(nilai > 70):
    print("Selamat, Anda Lulus") # Kondisi Benar, Dieksekusi
else:
    print("Maaf, Anda Tidak Lulus") # Kondisi Salah
```

```
Masukkan Nilai: 80
Selamat, Anda Lulus
```

```
nilai=input('Masukkan Nilai: ')
nilai=int(nilai)
if(nilai > 70):
    print("Selamat, Anda Lulus") # Kondisi Benar, Dieksekusi
else:
    print("Maaf, Anda Tidak Lulus") # Kondisi Salah
```

```
Masukkan Nilai: 70
Maaf, Anda Tidak Lulus
```

Kondisi elif

```
n=input('Masukkan angka') #SImpan nilai ke variabel n
n=int(n)  #conversi nilai n menjadi Integer
#Periksa apakah nilai n diantar 1 an 10
if n>=1 and n<=10:
    print("too low");

# periksa apakah nilai di antara 11 dan 20
elif n>=11 and n<=20:
    print("medium");
# periksa apakah nilai di antara 21 dan 30
elif n>=21 and n<=30:
    print("large");

# jika n lebih besar dari 30
else:
    print("too large")
```

Jalankan perintah diatas dan perhatikan

Latihan

1. Buatlah sebuah program yang menampung 2 bilangan bulat, kemudian tampilkan bilangan terbesar dan terkecil dari kedua nilai tersebut.
2. Buatlah sebuah program dengan menggunakan fungsi (function) untuk memeriksa apakah nilai yang dimasukkan bilangan ganjil atau genap.
3. Buatlah sebuah program yang dapat menghitung nilai akhir mahasiswa dengan ketentuan, absensi 10%, tugas 10%, MID 35%, UAS 45%, kemudian nilai akhir dibawah 50 tidak lulus. Contoh keluaran seperti berikut

   Nama Mahasiswa :

Nilai Absensi :

Nilai Tugas :

Nilai MID :

Nilai UAS :

Nilai Akhir :

Keterangan :

4. Buatlah sebuah program yang dapat berfungsi sebagai operator dimana dua nilai dimasukkan dan memilih jenis operator.

f. Perulangan

Perulangan atau *Loop* adalah suatu blok atau kumpulan instruksi yang dilaksanakan secara berulang-ulang. Perulangan yang disebut juga *repetition* akan membuat efisiensi proses dibandingkan jika dioperasikan secara manual.

While loop

Perulangan while akan terus di eksekusi selama kondisi terpenuhi (TRUE). Sintaks dasar:

While kondisi:

Statement

.....

Statement

Contoh:

```
# Contoh While Loop : mencetak angka dari 1 hingga 4 di dalam loop dan 5 di luar loop
cnt=1 #this is the initial variable
while cnt < 5 :
        #Print Perulangan
        print (cnt,"Perulangan")
        cnt+=1
else :
        #this statement will be printed if cnt is equals to 5
        print (cnt, "Di Luar Perulangan While" )
```

```
1 Perulangan
2 Perulangan
3 Perulangan
4 Perulangan
5 Di Luar Perulangan While
```

```
# menampilkan perulangan di bawah angka 9
count = 0 #inisial variabel count
while (count < 9): #periksa apakan nilai variabel count lebih kecil dari angka 9
    print ("Angka Ke: ", count) #cetak nilai variabel count
    count = count + 1 #Lakukan iterasi untuk merubah nilai kondisi terpenuhi

print ("Selesai!")
```

```
Angka Ke:  0
Angka Ke:  1
Angka Ke:  2
Angka Ke:  3
Angka Ke:  4
Angka Ke:  5
Angka Ke:  6
Angka Ke:  7
Angka Ke:  8
Selesai!
```

Dalam perulangan while pastikan kondisi akan terpenuhi; apabila tidak maka program akan terus-menerus mengulang.

```
word="UNPRI"
pos=0 #initial
while pos < len(word) :
    print (word[pos])
    #menambahkan posisi setelah mencetak huruf posisi itu
    pos+=1
```

```
U
N
P
R
I
```

```
#Nested While (Perulangan bersarang)
line=1 # initial variable
while line <= 5 :
        pos = 1
        while pos < line:
            #Mencetak nilai variabel pos dan spasi (agar membuat jarak)
            print(pos, end=' ')
            #tambahkan nilai variabel pos (incement)
            pos += 1
        else:
            #Mencetak baris baru setelah mencetak nilai variabel pos
            print(pos)
        #increment
        line += 1
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

For Loop

Sintaks

```
for itarator_variable in sequence_name:
    Statements
    . . .
    Statements
```

1. Kata pertama dari pernyataan dimulai dengan kata kunci “for” yang menandakan awal dari perulangan for.
2. Kemudian kita memiliki variabel iterator yang mengulangi urutan dan dapat digunakan dalam loop untuk melakukan berbagai fungsi
3. Berikutnya adalah kata kunci "in" di Python yang memberi tahu variabel iterator untuk mengulang elemen dalam urutan
4. Dan akhirnya, kita memiliki variabel urutan yang dapat berupa daftar, tupel, atau jenis iterator lainnya.
5. Bagian pernyataan dari loop adalah tempat Anda dapat bermain-main dengan variabel iterator dan melakukan berbagai fungsi

Contoh

```
word="UNPRI"
for letter in word:
    print (letter)
```

```
U
N
P
R
I
```

```
angka = [1,2,3,4,5]
for x in angka:
    print(x)

```

```
1
2
3
4
5
```

```
data = ["Data", "Data Science", "Data Science Fundamental", "Bersama UNPRI"]
for belajar in data:
    print (belajar)
```

```
Data
Data Science
Data Science Fundamental
Bersama UNPRI
```

```
fam = [1.73, 1.68, 1.71, 1.89]
for index, height in enumerate(fam) :
    print("index ke-" + str(index) + ": " + str(height))
```

```
index ke-0: 1.73
index ke-1: 1.68
index ke-2: 1.71
index ke-3: 1.89
```

```
#membalikkan data urutan
items = ["apple", 1, 4, "exit", "321"] #inisialisasi
#sebelum cetak membalik urutan list
for item in reversed(items):
        # print the item
        print (item),
```

```
321
exit
4
1
apple
```

```
#mengurutkan data dalam perulangan
#initialize a list
items = [7, 1, 4, 9, 3]
for item in sorted(items):
        # print the item
        print (item),
```

```
1
3
4
7
9
```

```
#inisialisasi
items=10
for n in range(1,items,2): # urutan lompat ke e
     print ('Nilai Dari',n),

```

```
Nilai Dari 1
Nilai Dari 3
Nilai Dari 5
Nilai Dari 7
Nilai Dari 9
```

Menggunakan python untuk loop Anda dapat melintasi dua atau lebih urutan secara bersamaan. Misalnya, dalam satu urutan Anda memiliki daftar nama dan di urutan lain Anda memiliki daftar hobi orang-orang yang sesuai. Jadi Anda harus mencetak nama orang beserta hobinya

```
names = ['Alice', 'Bob', 'Trudy' ]
hobbies = ['melukis', 'menyanyi', 'membaca']
ages = [21, 17, 22]
for person,age, hobby in zip(names,ages,hobbies):
        print (person+" berusia "+str(age)+" tahun dan hobinya adalah "+hobby)
```

```
Alice berusia 21 tahun dan hobinya adalah melukis
Bob berusia 17 tahun dan hobinya adalah menyanyi
Trudy berusia 22 tahun dan hobinya adalah membaca
```

Nested For

```
#Contoh penggunaan Nested Loop
words= ["Data", "Science", "Fundamental", "UNPRI" ]
for word in words:
    #This loop is fetching word from the list
    print ("Cetak Setiap Huruf Dari: "+word)
    for letter in word:
        #This loop is fetching letter for the word
        print (letter)
```

```
Cetak Setiap Huruf Dari: Data
D
a
t
a
Cetak Setiap Huruf Dari: Science
S
c
i
e
n
c
e
```

```
#Catatan: Mencetak bilangan prima
i = 2
while(i < 50):
    j = 2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print(i, " Bil Prima")
    i = i + 1

print("Selesai!")
```

```
2  Bil Prima
3  Bil Prima
5  Bil Prima
7  Bil Prima
11  Bil Prima
13  Bil Prima
17  Bil Prima
19  Bil Prima
23  Bil Prima
29  Bil Prima
31  Bil Prima
37  Bil Prima
41  Bil Prima
43  Bil Prima
47  Bil Prima
Selesai!
```

Fungsi break

Fungsi break adalah stetement yang digunakan untuk menghentikan sebuah perulangan pada kondisi tertentu

```
#Fungsi Break
#catatan: perulangan akan berhenti apabila variabel nilai >50
nilai = 0
while True:
    angka = int(input("Masukkan Angka: "))
    nilai += angka
    if nilai > 50:
        break
print("Angka berhenti pada jumlah: ", b)
```

```
Masukkan Angka: 10
Masukkan Angka: 20
Masukkan Angka: 30
Angka berhenti pada jumlah:  150
```

```
#Fungsi Break perulangan While
words = ["Data", "Science", "Fundamental", "exit", "UNPRI"]
for word in words:
    #checking for the breaking condition
    if word == "exit" :
        #if the condition is true, then break the loop
        break;
        #Otherwise, print the word
    print (word)
```

```
Data
Science
Fundamental
```

```
nums = [1, 2, 3, 4, 5, 6]
n = 2
found = False
for num in nums:
    if n == num:
        found = True
        break

print(f'Nilai Dari {n}: {found}')
```

```
Nilai Dari 2: True
```

```python
def print_total_nilai(even_nums):
    total = 0
    for x in even_nums:
        if x % 2 != 0:
            break
        total += x
    else:
        print("Data -->",even_nums)
        print(f'Total Angka: {total}')
```

```python
print_total_nilai([2, 4, 6, 8])
```

```
Data --> [2, 4, 6, 8]
Total Angka: 20
```

Fungsi continue

Pernyataan lanjutan Python digunakan untuk melewati iterasi perulangan ketika suatu kondisi terpenuhi.

```python
numbers = range(1,11)
#fungsi range(a,b) membuat daftar nomor 1 hingga (b-1) Jadi, dalam hal ini akan menghasilkan angka dari 1 sampai 10
for number in numbers:
    #periksa kondisi yang di abaikan (skip)
    if number == 7:
        print("Number ", number," diabaikan") #tampil apabila nilai TRUE
        continue
        #this statement won't be executed
        print ("pernyataan ini tidak akan dieksekusi ")

    #Cetak perkalian 2 dari nilai iterasi (perulangan)
    #print (number*2),
    print ("Number: ", number),
```

```
Number:  1
Number:  2
Number:  3
Number:  4
Number:  5
Number:  6
Number  7  diabaikan
Number:  8
Number:  9
Number:  10
```

```python
numbers = [ 1, 2, 4, 3, 6, 5, 7, 10, 9 ]
pos = 0 #initial position
while pos < len(numbers):
    #memeriksa kondisi skipping jika angka habis dibagi dua, itu genap
    if numbers[pos] % 2 == 0:
        pos = pos + 1
        continue
    print (numbers[pos])
    pos = pos + 1
```

```
1
3
5
7
9
```

Latihan

1. Tuliskan kode program di bawah ini dan jelaskan hasil Analisis program berdasarkan output (lakukan beberapa eksperimen untuk lebih memahami kode program)

   latihan pertama

```python
x = "UNPRI"
while x:
    print(x, " ")
    x = x[1:]
```

Latihan kedua

```
is_fast = True
car = "Ferrari" if is_fast else "Sedan"
if is_fast:
    car = "Ferrari"
else:
    car = "Sedan"

print (car)

is_fast = False
car = ("Sedan", "Ferrari")[is_fast]
print(car)
```

Latihan ketiga

```
karakter ={
    'a' : 122,
    'b' : 123,
    'c' : 124,
    'd' : 125
}

kata = input('Masukkan sebuah karakter : ')

print('Hasilnya dari ',kata,' : ', karakter.get(kata, -1))
```

Latihan ke empat

```
def switch_func(value, x):
    return {
        'a': lambda x: x+122,
        'b': lambda x: x*2,
        'c': lambda x: x-123,
        'd': lambda x: x/2
    }.get(value)(x)

inp = input('Masukkan sebuah karakter : ')
print('Hasilnya : ', switch_func(inp, 2))
```

2. Buat sebuah program untuk memfilter bilangan genap dan ganjil dari suatu list kemudian menjumlahkan berapa jumlah bilangan genap & ganjil nya.
3. Buat sebuah program menampilkan nilai perkalian dari masukkan nilai (ex: nilai masukan 5, maka akan muncul perkalian 1-5)
4. Buat sebuah program yang menampilkan kata yang lebih dari 3 huruf.

Tugas

1. Buatlah program yang meminta masukan user sebuah bilangan bulat N dimana (N>0). Program kemudian menampilkan penjumlahan N bilangan genap positif pertama (bilangan genap ≥ 0). Contoh:
    a. Jika user memasukkan nilai N=2, maka output : 0 + 2 + 4 = 6
    b. Jika user memasukkan nilai N=5, maka output : 0 + 2 + 4 + 6 + 8 =20
2. Buat sebuah program yang dapat menampung sebanyak N, kemudian tampilkan total nilai, bilangan terbesar dan terkecil dari nilai yang di Input. Contoh:

Jumlah Perulangan : 10

Masukkan nilai ke – 1 : 2

....

Masukkan nilai ke-10 : 20

Output:

Total Nilai : (tampilkan total nilai)

Nilai Terbesar adalah: tampilkan nilai terbesar

Nilai Terkecil adalah : tampilkan nilai terkecil

g. Tipe List

List adalah tipe data yang paling serbaguna yang tersedia dalam bahasa Python, yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya. Penerapan list dalam data science akan sering digunakan untuk memodelkan dataset.

Membuat list sangat sederhana, tinggal memasukkan berbagai nilai yang dipisahkan koma diantara tanda kurung siku. Dibawah ini adalah contoh sederhana pembuatan list dalam bahasa Python.

```
#Contoh sederhana pembuatan list pada bahasa pemrograman python
list1 = ['sistem komputer', 'teknik informatika', 'Data Science Fundamental', 'Universitas Prima Indonesia']
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
print(list1)
print(list2)
print(list3)
```

```
['sistem komputer', 'teknik informatika', 'Data Science Fundamental', 'Universitas Prima Indonesia']
[1, 2, 3, 4, 5]
['a', 'b', 'c', 'd']
```

Akses nilai dalam list

```
print ("list1[2]: ", list1[2]) #tampilkan indek ke 4
print ("list2[3:4]: ", list2[2:4]) #tampilkan nilai antara 2 dan 4
```

```
list1[2]:  Data Science Fundamental
list2[3:4]:  [3, 4]
```

Menambah atau merubah nilai dalam list

```
print('Data Awal :',list1)
print ("Nilai ada pada index 2 : ", list1[2])
list1[2] = 'Sistem Informasi' # tambahkan nilai baru pada index ke 2
print ("Nilai baru ada pada index 2 : ", list1[2])
print('Data Akhir :',list1)
```

```
Data Awal : ['sistem komputer', 'teknik informatika', 'Sistem Informasi', 'Universitas Prima Indonesia']
Nilai ada pada index 2 :  Sistem Informasi
Nilai baru ada pada index 2 :  Sistem Informasi
Data Akhir : ['sistem komputer', 'teknik informatika', 'Sistem Informasi', 'Universitas Prima Indonesia']
```

Hapus data dalam list

```
#Contoh cara menghapus nilai pada list python
print (list1)
del list1[2]
print ("Setelah dihapus nilai pada index 2 : ", list1)
```

```
['sistem komputer', 'teknik informatika', 'Sistem Informasi', 'Universitas Prima Indonesia']
Setelah dihapus nilai pada index 2 :  ['sistem komputer', 'teknik informatika', 'Universitas Prima Indonesia']
```

1. Operasi Dasar Pada List Python

   List Python merespons operator + dan * seperti string; Itu artinya penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah list baru, bukan sebuah String. **Contoh**

```
#program contoh yang memanfaatkan list
num = [-1, 2, 53, 5, 50, 153, 91,87]
genap = [x for x in num if x%2==0]
print("list angka ",num)
print("angka genap pada list tersebut: ",genap)

#program contoh yang memanfaatkan list
num = [-1, 2, 53, 5, 50, 153, 91,87]
ganjil = [x for x in num if x%2!=0]
print("\nlist angka ",num)
print("angka ganjil pada list tersebut: ",ganjil)
```

```
list angka  [-1, 2, 53, 5, 50, 153, 91, 87]
angka genap pada list tersebut:  [2, 50]

list angka  [-1, 2, 53, 5, 50, 153, 91, 87]
angka ganjil pada list tersebut:  [-1, 53, 5, 153, 91, 87]
```

```
s = ['U', 'N', 'P', 'R', 'I'] #list
str1 = ''.join(s) #menggabungkan list
print(str1) #menampilkan str1
```

```
UNPRI
```

   Pada list merespons semua operasi urutan umum yang di gunakan pada String dibagian sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada list python.

Tabel 7. Daftar operasi dasar pada list python

Python Expression	**Hasil**	**Penjelasan**
len([1, 2, 3, 4])	4	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Halo!'] * 4	['Halo!', 'Halo!', 'Halo!', 'Halo!']	Repetition
2 in [1, 2, 3]	True	Membership
for x in [1,2,3] : print (x, end= ' ')	1 2 3	Iteration

2. Indexing, Slicing dan Matrix Pada List Python

   Karena list adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk list seperti yang mereka lakukan untuk String.

   Dengan asumsi input berikut :

   L = ['C++", 'Java', 'Python']

Tabel 8. Python Expression

Python Expression	**Hasil**	**Penjelasan**
L[2]	'Python'	Offset mulai dari nol
L[-2]	'Java'	Negatif: hitung dari kanan
[1:]	['Java', 'Python']	Slicing mengambil bagian

3. Method dan Fungsi Build-in Pada List Python Python menyertakan fungsi built-in sebagai berikut :

Tabel 9. Python Function

Python Function	**Penjelasan**
cmp(list1, list2) #	Tidak lagi tersedia dengan Python 3
len(list)	Memberikan total panjang list.
max(list)	Mengembalikan item dari list dengan nilai maks.
min(list)	Mengembalikan item dari list dengan nilai min.
list(seq)	Mengubah tuple menjadi list.

Python menyertakan methods built-in sebagai berikut:

Tabel 10. Python Method

Python Methods	**Penjelasan**
list.append(obj)	Menambahkan objek obj ke list
list.count(obj)	Jumlah pengembalian berapa kali obj terjadi dalam list
list.extend(seq)	Tambahkan isi seq ke list
list.index(obj)	Mengembalikan indeks terendah dalam list yang muncul obj
list.insert(index, obj)	Sisipkan objek obj ke dalam list di indeks offset
list.pop(obj = list[-1])	Menghapus dan mengembalikan objek atau obj terakhir dari list
list.remove(obj)	Removes object obj from list
list.reverse()	Membalik list objek di tempat
list.sort([func])	Urutkan objek list, gunakan compare func jika diberikan

h. Type Tuple

Seperti yang dijelaskan pada bagian sebelumnya sebuah tupel adalah urutan objek Python yang tidak berubah. Tupel adalah urutan, seperti daftar. Perbedaan utama antara tupel dan daftarnya adalah bahwa tupel tidak dapat diubah tidak seperti List Python. Tupel menggunakan tanda kurung, sedangkan List Python menggunakan tanda kurung siku.

Membuat tuple semudah memasukkan nilai-nilai yang dipisahkan koma. Secara opsional, Anda dapat memasukkan nilai-nilai yang dipisahkan koma ini di antara tanda kurung juga. Sebagai contoh :

```python
#Contoh sederhana pembuatan tuple pada bahasa pemrograman python
tup1 = ['sistem komputer', 'teknik informatika', 'Data Science Fundamental', 'Universitas Prima Indonesia']
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"

print(tup1)
print(tup2)
print(tup3)
```

```
['sistem komputer', 'teknik informatika', 'Data Science Fundamental', 'Universitas Prima Indonesia']
(1, 2, 3, 4, 5)
('a', 'b', 'c', 'd')
```

Tupel kosong ditulis sebagai dua tanda kurung yang tidak berisi apa-apa, contohnya : tup1=(); Untuk menulis tupel yang berisi satu nilai, Anda harus memasukkan koma, meskipun hanya ada satu nilai, contohnya : tup1 = (50,) Seperti indeks String, indeks tuple mulai dari 0, dan mereka dapat diiris, digabungkan, dan seterusnya.

Nilai Dalam Tuple Python

```python
#Cara mengakses nilai tuple
print ("tup1[0]: ", tup1[0])
print ("tup2[1:5]: ", tup2[1:5])
```

```
tup1[0]:  sistem komputer
tup2[1:5]:  (2, 3, 4, 5)
```

Update Nilai Dalam Tuple Python

Tuple tidak berubah, yang berarti Anda tidak dapat memperbarui atau mengubah nilai elemen tupel. Anda dapat mengambil bagian dari tupel yang ada untuk membuat tupel baru seperti ditunjukkan oleh contoh berikut

```python
print('Tuple 3: ',tup3)
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')
# Aksi seperti dibawah ini tidak bisa dilakukan pada tuple python
# Karena memang nilai pada tuple python tidak bisa diubah
# tup1[0] = 100;

# Jadi, buatlah tuple baru sebagai berikut
tup3 = tup1 + tup2
print ('Tuple 3:',tup3)
```

```
Tuple 3:  ('a', 'b', 'c', 'd')
Tuple 3: (12, 34.56, 'abc', 'xyz')
```

Hapus Nilai Dalam Tuple Python

```python
tup4= ('fisika', 'kimia', 1993, 2017)
print(tup4)
del tup4
print("Setelah menghapus tuple : ")
print(tup4)
```

```
('fisika', 'kimia', 1993, 2017)
Setelah menghapus tuple :

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-b6ed2d487d68> in <module>
      3 del tup4
      4 print("Setelah menghapus tuple : ")
----> 5 print(tup4)

NameError: name 'tup4' is not defined
```

Menghapus elemen tuple individual tidak mungkin dilakukan. Tentu saja, tidak ada yang salah dengan menggabungkan tuple lain dengan unsur-unsur yang tidak diinginkan dibuang. Untuk secara eksplisit menghapus keseluruhan tuple, cukup

gunakan del statement. Sebagai contoh kode baris ke 5 terjadi error karena variabel tuple sudah di hapus pada baris perintah 3

Operasi Dasar Pada Tuple Python

Tuple merespons operator + dan * sama seperti String; dalam hal ini proses penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah tupel baru, bukan string. Sebenarnya, Tuple merespons semua operasi urutan umum yang digunakan pada String. Dibawah ini adalah tabel daftar operasi dasar pada Tuple python.

Tabel 11. Operasi Dasar Pada Tuple Python

Python Expression	Hasil	Penjelasan
len((1, 2, 3))	3	Length
(1, 2, 3) +(4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
(‘Halo!’,) * 4	(‘Halo!’, ‘Halo!’, ‘Halo!’, ‘Halo!’)	Repetition
3 in(1, 2, 3)	True	Membership
for x in (1,2,3) : print (x, end = ‘ ‘)	1 2 3	Iteration

Indexing, Slicing dan Matrix Pada Tuple Python

Karena tupel adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk tupel seperti pada String, dengan asumsi masukan berikut

Dengan asumsi input berikut : T = ('C++', 'Java', 'Python').

Tabel 12. Indexing, Slicing dan Matrix Pada Tuple Python

Python Expression	Hasil	Penjelasan
T[2]	'Python'	Offset mulai dari nol
T[-2]	'Java'	Negatif: hitung dari kanan
T[1:]	('Java', 'Python')	Slicing mengambil bagian

Fungsi Build-in Pada Tuple Python

Python menyertakan fungsi built-in sebagai berikut

Tabel 13. Fungsi Build-in Pada Tuple Python

Python Function	Penjelasan
cmp(tuple1, tuple2)	# Tidak lagi tersedia dengan Python 3
len(tuple)	Memberikan total panjang tuple.
max(tuple)	Mengembalikan item dari tuple dengan nilai maks.
min(tuple)	Mengembalikan item dari tuple dengan nilai min.
tuple(seq)	Mengubah seq menjadi tuple.

i. Membuat Type Directory

Dictionary Python berbeda dengan List ataupun Tuple. Karena setiap urutanya berisi key dan value. Setiap key dipisahkan dari value-nya oleh titik dua (:), item dipisahkan oleh koma, dan semuanya tertutup dalam kurung kurawal. Di kosong tanpa barang ditulis hanya dengan dua kurung kurawal, seperti ini: {}. Nilai kamus bisa berupa tipe apa pun, namun key harus berupa tipe data yang tidak berubah seperti string, angka, atau tupel.

Akses Nilai Dalam Dictionary

Untuk mengakses elemen Dictionary, Anda dapat menggunakan tanda kurung siku yang sudah dikenal bersama dengan key untuk mendapatkan nilainya. Berikut adalah contoh sederhananya :

```
#Contoh cara membuat Dictionary pada Python
dict = {'Nama': 'Data Science Fundamental', 'Waktu': 8, 'Class': 'DSF B'}
print ("dict['Nama']: ", dict['Nama'])
print ("dict['Waktu']: ", dict['Waktu'])
```

```
dict['Nama']:  Data Science Fundamental
dict['Waktu']:  8
```

Update Nilai Dalam Dictionary

Anda dapat memperbarui Dictionary dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada seperti ditunjukkan pada contoh sederhana yang diberikan di bawah ini.

```
#Update dictionary python
dict['Waktu'] = 10; # Mengubah entri yang sudah ada
dict['Lokasi'] = "Universitas Prima Indonesia" # Menambah entri baru

print ("dict['Nama']: ", dict['Nama'])
print ("dict['Waktu']: ", dict['Waktu'])
print ("dict['Lokasi']: ", dict['Lokasi'])
```

```
dict['Nama']:  Data Science Fundamental
dict['Waktu']:  10
dict['Lokasi']:  Universitas Prima Indonesia
```

Hapus Elemen Dictionary

Anda dapat menghapus elemen Dictionary individual atau menghapus keseluruhan isi Dictionary. Anda juga dapat menghapus seluruh Dictionary dalam satu operasi.

Untuk menghapus seluruh Dictionary secara eksplisit, cukup gunakan del statement. Berikut adalah contoh sederhana :

```
#Contoh cara menghapus pada Dictionary Python

del dict['Nama'] # hapus entri dengan key 'Name'
dict.clear()     # hapus semua entri di dict
del dict         # hapus dictionary yang sudah ada

print ("dict['Waktu']: ", dict['Waktu'])
print ("dict['Lokasi']: ", dict['Lokasi'])
```

Pada saat anda menjalankan program di atas akan terjadi error, karena data dictionary sudah di hapus pada baris perintah ke 4 dan ke 5.

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-27-1e430f998dfb> in <module>
      1 #Contoh cara menghapus pada Dictionary Python
      2
----> 3 del dict['Nama'] # hapus entri dengan key 'Name'
      4 dict.clear()     # hapus semua entri di dict
      5 del dict         # hapus dictionary yang sudah ada
```

Fungsi Build-in Pada Dictionary

Tabel 14. Fungsi Build-in Pada Dictionary

Fungsi Python	**Penjelasan**
Cmp(dict1, dict2)	Membandingkan unsur keduanya.
len(dict)	Memberikan panjang total Dictionary. Ini sama dengan jumlah item dalam Dictionary.
Str(dict)	Menghasilkan representasi string yang dapat dicetak dari Dictionary
type(variable)	Mengembalikan tipe variabel yang lulus. Jika variabel yang dilewatkan adalah Dictionary, maka akan mengembalikan tipe Dictionary.

Method Build-in Pada Dictionary

Tabel 15. Method Build-in Pada Dictionary

Metode Python	**Penjelasan**
dict.clear()	Menghapus semua elemen Dictionary
dict.copy()	Mengembalikan salinan Dictionary
dict.fromkeys()	Buat Dictionary baru dengan kunci dari seq dan nilai yang disetel ke nilai.
dict.get(key, default=None)	For key, nilai pengembalian atau default jika tombol tidak ada dalam Dictionary
dict.has_key(key)	Mengembalikan true jika key dalam Dictionary, false sebaliknya
dict.items()	Mengembalikan daftar dari pasangan tuple dictionary (key, value)
dict.keys()	Mengembalikan daftar key dictionary
dict.setdefault(key, default=None)	Mirip dengan get (), tapi akan mengatur dict [key] = default jika kunci belum ada di dict
dict.update(dict2)	Menambahkan pasangan kunci kata kunci dict2 ke dict
dict.values()	Mengembalikan daftar nilai dictionary

```
#contoh penggunaan dictionary
print("\ncontoh 1")
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

```
contoh 1
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

```
print("\ncontoh 2")
d = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
cek_angka = 3
if cek_angka in d:
    print(cek_angka,' tersedia dalam dictionary')
else:
    print(cek_angkat,'tidak tersedia dalam dictionary')
```

```
contoh 2
3  tersedia dalam dictionary
```

```
print("\ncontoh 3")
d1 = {'a': 100, 'b': 200}
d2 = {'x': 300, 'y': 200}
d = d1.copy()
d.update(d2)
print(d)
```

```
contoh 3
{'a': 100, 'b': 200, 'x': 300, 'y': 200}
```

Latihan

1. Program yang menghapus key pada sebuah dictionary kemudian menampilkan perbedaan antara dictionary pertama kemudian yang sudah berubah
2. Program yang menampilkan penjumlahkan bilangan yang ada pada dictionary dan tampilkan bilangan terbesar dan terkecil.
3. Program yang menampilkan isi dari sebuah tuple berdasarkan kata yang dimasukkan oleh user.
4. Tuliskan kode program di bawah ini, analisis dan jelaskan hasilnya

```
#initial list
l = [1,2,3,4,5,6,7,8]

#print the initial list
print("Data Inisial Awal: ")
print(l)

#choose n
n = int(input('Enter Nilai ke N: '))

#increment all the values of the list by using map with help of lambda
l = list(map(lambda x: x+n , l))

#print the changed list
print("Perubahan Data adalah: ")
print(l)

```

```
list1 = [1,2,3,4,5,6,7,8]
print("List1:", end = ' ')
print(list1)

list2 = [2,3,5,9,12,14]
print("List2:", end = ' ')
print(list2)

list3 = list(filter(lambda x: x in list2 , list1))
print("List3 :", end=' ')
print(list3)
```

j. Fungsi Tanggal

Python dapat menangani tanggal dan waktu dengan beberapa cara. Konversi antara format tanggal adalah tugas umum untuk komputer. Modul waktu dan kalender Python melacak tanggal dan waktu.

Tanggal dalam Python bukanlah tipe datanya sendiri, tetapi kita dapat mengimpor modul bernama **datetime** untuk bekerja dengan tanggal sebagai objek tanggal.

```
import datetime

x = datetime.datetime.now()
print(x)
print(x.year)
print(x.strftime("%A"))
```

```
2021-08-16 16:21:06.291307
2021
Monday
```

Membuat Objek Tanggal

Untuk membuat tanggal, kita dapat menggunakan datetime()kelas (konstruktor) dari datetime modul. Fungsi kelas datetime() membutuhkan tiga parameter untuk membuat tanggal: tahun, bulan, hari.

```
x = datetime.datetime(2021, 8, 17)
print(x)
```

```
2021-08-17 00:00:00
```

Format objek tanggal ke string

```
#menampilkan Nama Bulan, Tahun, Bulan dan Tanggal
x = datetime.datetime(2021, 8, 17)
print(x.strftime("%B"))
print(x.strftime("%Y"))
print(x.strftime("%m"))
print(x.strftime("%d"))
```

```
August
2021
08
17
```

Tabel 16. Objek tanggal ke string

Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41

Directive	Description	Example
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00- 53	52
%W	Week number of year, Monday as the first day of week, 00- 53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%

k. Fungsi Waktu

Fungsi Tick

Interval waktu adalah bilangan floating-point dalam satuan detik. Instansi tertentu dalam waktu dinyatakan dalam hitungan detik sejak pukul 12:00 1 Januari 1970.

```
import time;  # Digunakan untuk meng-import modul time

ticks = time.time()
print ("Berjalan sejak 12:00am, January 1, 1970:", ticks)
```

```
Berjalan sejak 12:00am, January 1, 1970: 1629104031.839903
```

Banyak fungsi waktu Python menangani waktu sebagai tuple dari 9 nomor, seperti yang terdapat pada tabel di bawah ini.

Tabel 17. Fungsi waktu pada Python

Index	Field	Value
0	4-digit year	2008
1	Bulan	1 sampai 12
2	Hari	1 sampai 31
3	Jam	0 sampai 23
4	Menit	0 sampai 59
5	Detik	0 sampai 61
6	Hari dalam Minggu	0 sampai 6 (0 adalah Senin)
7	Hari dalam Bulan	1 sampai 366
8	Daylight savings	-1, 0, 1, -1 means library determines DST

Tuple diatas setara dengan struktur struct_time. Struktur ini memiliki atribut berikut :

Tabel 18. Atribut struktur time

Index	Atribut	Value
0	tm_year	2008
1	tm_mon	1 sampai 12
2	tm_mday	1 sampai 31
3	tm_hour	0 sampai 23
4	tm_min	0 sampai 59
5	tm_sec	0 sampai 61
6	tm_wday	0 sampai 61(0 adalah senin)
7	tm_yday	1 sampai 366
8	tm_isdst	-1, 0, 1, -1 means library determines DST

Mendapatkan Waktu Saat Ini

Untuk menerjemahkan waktu instan dari satu detik sejak nilai floating-point ke waktu menjadi tupel waktu, lewati nilai floating-point ke fungsi (mis., Localtime) yang mengembalikan waktu tupel dengan semua sembilan item valid

```
localtime = time.localtime(time.time())
print ("Waktu lokal saat ini :", localtime)
```

```
Waktu lokal saat ini : time.struct_time(tm_year=2021, tm_mon=8, tm_mday=16, tm_hour=15, tm_min=55, tm_sec=5, tm_wday=0, tm_y
day=228, tm_isdst=0)
```

Mendapatkan Waktu yang berformat

Anda dapat memformat kapan saja sesuai kebutuhan Anda, namun metode sederhana untuk mendapatkan waktu dalam format yang mudah dibaca adalah asctime ()

```
localtime = time.asctime( time.localtime(time.time()) )
print ("Waktu lokal saat ini :", localtime)
```

```
Waktu lokal saat ini : Mon Aug 16 15:56:13 2021
```

Mendapatkan kalender dalam sebulan

Modul kalender memberikan berbagai macam metode untuk dimainkan dengan kalender tahunan dan bulanan. Di sini, kami mencetak kalender untuk bulan tertentu (Agustus 2021)

```
import calendar

cal = calendar.month(2021, 8)
print ("Dibawah ini adalah kalender:")
print (cal)
```

```
Dibawah ini adalah kalender:
    August 2021
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

l. Print

Python menyediakan beberapa cara untuk mempresentasikan keluaran suatu program; data dapat dicetak dalam bentuk yang dapat dibaca, atau ditulis ke berkas untuk digunakan di masa mendatang. Bab ini akan membahas beberapa kemungkinan.

formatted string literals

Menampilkan keluaranmenggunakan *formatted string literals*, mulailah string dengan f atau F sebelum tanda kutip pembuka atau tanda kutip tiga. Di dalam string ini, Anda bisa menulis ekspresi Python antara karakter { dan } yang dapat merujuk ke variabel atau nilai literal. Contoh seperti berikut ini.

```
year = 2021
event = 'Data Science Fundamental - UNPRI'
print(f'Hasil Keluaran : {year} {event}')
```

```
Hasil Keluaran : 2021 Data Science Fundamental - UNPRI
```

Modul string berisi kelas yang menawarkan cara lain untuk mengganti nilai menjadi string, menggunakan penampung seperti **$x** dan menggantinya dengan nilai-nilai dari dictionary, tetapi menawarkan kontrol format yang jauh lebih sedikit. Berikut contoh keluaran fungsi string.

```
print(str(event))
print(repr(event))
print(str(1/7))
```

```
Data Science Fundamental - UNPRI
'Data Science Fundamental - UNPRI'
0.14285714285714285
```

```
x = 10 * 3.25
y = 200 * 200
s = 'Nilai Variabel x adalah ' + repr(x) + ', and variabel y adalah ' + repr(y) + '...'
print(s)
```

```
Nilai Variabel x adalah 32.5, and variabel y adalah 40000...
```

Literal String Terformat

Formatted string literals (juga disebut f-string) memungkinkan Anda menyertakan nilai ekspresi Python di dalam string dengan mengawali string dengan f atau F dan menulis ekspresi sebagai {expression}.

Penentu format opsional dapat mengikuti ekspresi. Ini memungkinkan kontrol yang lebih besar atas nilai keluaran yang diformat. Contoh berikut ini pembulatan pi ke tiga tempat setelah desimal:

```
import math #panggil kelas modul math
print(f'Nilai dari Phi {math.pi:.3f}.')
```

```
Nilai dari Phi 3.142.
```

Melewatkan bilangan bulat setelah ':' akan menyebabkan field itu menjadi jumlah minimum lebar karakter. Ini berguna untuk membuat kolom berbaris.

```
table = {'Data': 80, 'Science': 90, 'Fundamental': 10}
for data, nilai in table.items():
    print(f'{data:15} ==> {nilai:10d}')
```

```
Data            ==>         80
Science         ==>         90
Fundamental     ==>         10
```

Metode String format

```
print('UNPRI {} Belajar DSF dengan: "{}!"'.format('The Best Choice', 'Python'))
```

```
UNPRI The Best Choice Belajar DSF dengan: "Python!"
```

Tanda kurung dan karakter di dalamnya (disebut fields format) diganti dengan objek yang diteruskan ke metode str.format(). Angka dalam tanda kurung dapat digunakan untuk merujuk ke posisi objek yang dilewatkan ke dalam metode str.format().

```
print('{0} and {1}'.format('Data Science', 'Fundamental'))
print('{1} and {0}'.format('Data Science', 'Fundamental'))
```

```
Data Science and Fundamental
Fundamental and Data Science
```

Jika argumen kata kunci keyword argument digunakan dalam metode str.format(), nilainya dirujuk dengan menggunakan nama argumen.

```
print('Belajar {judul} dengan {ket}.'.format(
    judul='Data Science', ket='UNPRI'))
```

```
Belajar Data Science dengan UNPRI.
```

Argumen posisi dan kata kunci dapat dikombinasikan secara bergantian:

```
print('Belajar {0} {1} bersama {other}.'.format('Data Science', 'Fundamental',
                                                other='UNPRI'))
```

```
Belajar Data Science Fundamental bersama UNPRI.
```

4. Forum Diskusi

   a. Tuliskan kode program di bawah ini, dan jelaskan hasilnya

```
for x in range(1, 11):
    print(repr(x).rjust(2), repr(x*x).rjust(3), end=' ')
    print(repr(x*x*x).rjust(4))
```

```
for x in range(1, 11):
    print(repr(x).rjust(2), repr(x*x).rjust(3), end=' ')
    print(repr(x*x*x).rjust(4))
```

```
print('{:+d}'.format(42))
stark = {'first': 'Ned', 'second': 'Brandon', 'third': 'Rob'}
print('{first} {third}'.format(**stark))

from datetime import datetime
print('{:%Y-%m-%d %H:%M}'.format(datetime(2017, 12, 7, 13, 22)))

myDate = datetime(2017, 12, 7, 13, 22)
print('{:{dfmt} {tfmt}}'.format(myDate, dfmt='%Y-%m-%d', tfmt='%H:%M'))

value = '{:{width}.{prec}f}'.format(3.1428, width=5, prec=2)
print(value)

name = '{:^30}'.format('Data Science Fundamental')
print(name)

course = "Data Science Fundamental"
best_place = "Universitas Prima Indonesia"
print("Tempat Belajar {1} di Kampus {0}".format(best_place, course))
```

b. Lakukan beberapa modifikasi untuk lebih memahami fungsi yang digunakan pada kedua kode program di atas, lakukan analisis dan buat laporan keluaran kedua program kemudian buat program sendiri dengan mengikuti beberapa contoh penggunaan string di atas seperti data identitas diri anda (biodata) atau company profile.

C. PENUTUP

1. Rangkuman

   Memasuki era Industri 4.0, istilah Data Science, Artificial Intelligence, dan Machine Learning, mungkin sudah familiar untuk kita. Hal ini diiringi dengan semakin banyaknya perusahaan yang mulai menyadari pentingnya menerapkan Data Science dan menggunakan Big Data untuk bisnis mereka. Profesi talenta digital di bidang tersebut pun semakin diminati di berbagai kalangan saat ini.Metodologi data science adalah langkah- langkah digunakan dalam proyek data science agar dapat menghasilkan hasil yang optimal yang dapat menjawab pertanyaan dari suatu masalah yang ingin diselesaikan. Tahapan dalam metodologi data science merupakan proses berulang, yang mana jika dalam suatu tahapan dirasa masih belum sesuai, bisa kembali ke tahap sebelumnya.

2. Tes Formatif

   Sebagai evaluasi pemahaman materi pada Bab 1, jawablah pertanyaan berikut ini.

   a. Pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang terjadi saat jalanya program dan menentukan tindakan apa yang akan diambil sesuai dengan kondisi. Pada python ada beberapa statement/kondisi diantaranya adalah :

      1. Kondisi IF
      2. Kondisi if-else

3. Kondisi elif
4. Kondisi if then

b. Operator adalah konstruksi yang dapat memanipulasi nilai dari operan. Berikut ini yang merupakan operator adalah
    1. Penjumlahan
    2. Pengurangan
    3. Pembagian
    4. Perkalian

c. Python sendiri mempunyai tipe data yang cukup unik bila kita bandingkan dengan bahasa pemrograman yang lain. Berikut ini yang merupakan tipe data pada Python adalah
    1. Boolean
    2. String
    3. Integer
    4. Float

d. Bagian atau blok program yang berisi satu tugas spesifik

e. Manajer paket, manajer lingkungan, dan distribusi Python yang berisi kumpulan banyak paket sumber terbuka. Hal tersebut disebut dengan apa?

f. Fungsi lokal fungsi yang didefinisikan di dalam suatu modul dan dapat dipanggil oleh fungsi lain, baik yang berada di dalam modul yang sama maupun modul lain
    1. True
    2. False

DAFTAR PUSTAKA

Data Science From Scratch first principal with Python, published by O'Reilly Media, Inc.,1005 gravenstein Highway North Sebatopol, CA

Data Scientist with Python, Datacamp, https://www.datacamp.com/users/sign_in?redirect=http%3A%2F%2Fapp.datacamp.com%2Flearn%2Fcareer-tracks%2Fdata-%2520scientist-with-python%3Fversion%3D5&version=5

E-Modul Data Science, IlmudataPy.com Tutorial Python, https://docs.python.org/id/3.8/tutorial/

BAB II
TEKNIK PENGUMPULAN DATA

A. PENDAHULUAN

Modul pada Bab 2 ini membahas tentang bagaimana Teknik pengumpulan dataset yang agar dapat dijadikan sebagai pertimbangan dalam mengambil keputusan berdasarkan hasil akurasi yang diperoleh dan mempelajari library-library untuk scientific computing serta bagaimana proses Managing Environment dan Packages dalam Anaconda Platform.

B. INTI

1. Capaian Pembelajaran
    a. Mampu memahami dan menjelaskan library - library untuk scientific computing dan bagaimana proses Managing Environment dan Packages dalam Anaconda Platform.
    b. Mampu memahami dan menjelaskan Mananging environment & packages dan Package Numpy
    c. Mampu memahami dan menjelaskan apa itu Indexing, Slicing & Filtering, Broadcasting dan package pandas
    d. Mampu memahami dan menjelaskan bagaimana selecting & filtering dataset
    e. Mampu memahami dan menjelaskan bagaimana group dataset by operation
    f. Mampu memahami dan menggunakan Notebook
2. Materi
    a. Mananging environment & packages
    b. Package Numpy
    c. Indexing
    d. Sicing & Filtering
    e. Broadcasting
    f. Package Pandas
    g. Selecting & Filtering
    h. Group by Operation
    i. Pengenalan Notebook

3. Uraian Materi

a. Mananging environment & packages

Sebelum memulai mempelajari library-library untuk scientific computing, kita akan mempelajari terlebih dahulu, bagaimana proses Managing Environment dan Packages dalam Anaconda Platform.

1. Managing Environments

Untuk bacaan lebih lengkap, dapat mengacu pada laman Managing Environments di situs dokumentasi Anaconda. Proses managing environments dapat dijalankan dengan menggunakan Anaconda Navigator dan Anaconda Prompt, dan environments sendiri mempunyai arti lingkungan kerja. Tujuan dari environments untuk mempartisi lingkungan kerja yang diisi dengan library - library khusus agar nantinya tidak terjadi bentrok, seperti environment untuk library yang berbeda versi, satu untuk python 3.6, satu untuk python 3.5, atau bisa juga berbeda versi untuk packagenya.

Adapun proses Managing Environments terdiri dari:

a. Proses List environments untuk melihat daftar environments;
b. Proses Create environment membuat environment baru;
c. Proses Activate environment untuk mengaktivasi environment;
d. Proses Deactivate untuk menonaktifkan environment dan kembali ke environment sebelumnya;
e. Proses Clone untuk menduplikasi environment;
f. Proses Remove untuk menghapus environment.

Proses-proses environments diatas akan kita coba dengan menggunakan Anaconda Prompt.

List Environments

conda env list

Perintah diatas akan menampilkan daftar environments yang ada, dan tanda bintang (*) mempunyai arti environment yang aktif saat ini.

Create Environment

conda create --name namaenv

Perintah diatas akan membuat sebuah environment baru dengan nama baru melalui parameter name, dan apabila dijalankan, console akan menampilkan tulisan Proceed ([y]/n)? dan perlu dilanjutkan dengan menekan tombol y di keyboard untuk melanjutkan pembuatan environment.

Activate Environment

conda activate namaenv

Perintah diatas akan mengaktivasi environment yang kita inginkan sesuai nama yang kita tuliskan setelah kata kunci activate.

Deactivate Environment

conda deactivate

Perintah diatas akan menonaktifkan environment aktif, dan akan kembali ke environment sebelumnya, dan proses ini bisa dilakukan terus sampai keluar dari environment base. Environment base adalah environment dasar yang diciptakan pada saat pertama kali kita melakukan instalasi Anaconda, dan ini merupakan environment bawaan (default).

Clone Environment

conda create --name namaenvbaru --clone namaenvlama

Perintah diatas akan menduplikasi sebuah environment dengan menggunakan parameter clone, dan apabila dijalankan, akan tercipta sebuah environment baru dengan nama yang mengacu pada parameter name, dan environment ini akan mempunyai packages yang sama dengan environment sebelumnya.

Remove Environmen

conda env remove --name namaenv

Perintah diatas akan menghapus environment, dan sangat disarankan untuk berhati-hati dalam menggunakan perintah, karena sebuah environment yang sudah dihapus, tidak akan bisa dikembalikan lagi.

Experimen Enviroment

Dan untuk praktek kita, mari kita ciptakan sebuah environment baru dengan nama dsf, dan lakukan aktivasi langsung ke environment ini. Perintahnya dapat dilakukan

dalam dua perintah berikut, yaitu:

a. conda create --name dsf
b. conda activate dsf

2. Managing Packages

Untuk bacaan lebih lengkap, dapat mengacu pada laman Managing Packages di situs dokumentasi Anaconda. Istilah **library** dan **package** mempunyai arti yang sama, yaitu kumpulan kelas dan fungsi yang terangkum menjadi satu, dan umumnya dua kata ini dapat dipertukarkan satu sama lainnya, dan dalam Anaconda Platform, lebih sering digunakan istilah package ketimbang library.

Adapun proses Managing Packages terdiri dari:

a. Poses Search package untuk melakukan pencarian package;
b. Proses List package untuk melihat daftar package;
c. Proses Install package untuk instalasi package baru;
d. Proses Update package untuk memperbaharui versi package;
e. Proses Remove package untuk menghapus package yang sudah tidak diinginkan.

Sebelum memulai proses Managing Packages, diharapkan untuk mengaktifkan environment dsf untuk eksperimen dengan membuka Anaconda Prompt dan ketikan perintah berikut:

conda activate dsf

Dan pastikan bahwa environment sudah aktif dengan melihat kata pertama dalam kurung pada anaconda prompt, dan bila terjadi pesan kegagalan, silakan mengacu kembali kepada eksperimen environment diatas.

Search Package

Sebelum instalasi, kita dapat melakukan proses **search** terhadap packages pada repository conda sendiri, yaitu dengan perintah **search** sebagai berikut:

a. Perintah: **conda search namapackage**-nama_package disesuaikan dengan nama package yang akan kita cari;
b. Contoh 1: **conda search**-mencari seluruh package yang tersedia;
c. Contoh 2: **conda search numpy**-mencari package numpy dan menampilkan seluruh versi yang tersedia;

d. Contoh 3: **conda search numpy=1.6**-mencari package numpy khusus untuk versi 1.6 dan versi minornya;
e. Contoh 4: **conda search sci***-mencari seluruh package dengan awalan sci beserta versinya masing-masing.

Perintah **search** hanya berfungsi untuk melihat daftar package yang tersedia, dan tidak melakukan perubahan apapun terhadap environment kita.

Lisy Packages

Proses **list** berbeda dengan proses **search**, dimana proses **list**, adalah proses untuk melihat package yang sudah terinstalasi didalam environment kita dan sudah siap untuk digunakan.

a. Perintah: **conda list namapackage** - nama_package disesuaikan dengan nama package yang akan kita lihat;
b. Contoh 1: **conda list** - melihat seluruh package yang terinstalasi di environment kita;
c. Contoh 2: **conda list numpy** - melihat package numpy dan menampilkan seluruh versi yang terinstalasi;
d. Contoh 4: **conda list sci*** - melihat seluruh package dengan awalan sci beserta versinya masing - masing.

Install Package

Proses **search** dapat digunakan untuk melihat package yang tersedia, dan apabila package yang kita inginkan tersedia, maka tahap selanjutnya adalah proses instalasi dengan perintah **install**, dan dapat dilihat pada contoh berikut:

a. Perintah: **conda install namapackage** - nama package disesuaikan dengan nama package yang akan kita install;
b. Contoh 1: **conda install numpy** - untuk instalasi package numpy dengan versi yang paling baru;
c. Contoh 2: **conda install --name namaenv numpy** - untuk instalasi package numpy pada environment tertentu;
d. Contoh 3: **conda install numpy=1.14.5** - untuk instalasi package numpy dengan versi tertentu;
e. contoh 4: **conda install numpy pandas** - untuk instalasi package numpy dan

pandas sekaligus.

Update Package

conda update namapackage

Perintah diatas akan melakukan proses untuk memperbaharui package sesuai dengan nama package yang kita tuliskan setelah kata kunci update.

Remove Packages

conda remove namapackage

Perintah diatas akan melakukan proses untuk menghapus package sesuai dengan nama package yang kita tuliskan setelah kata kunci remove.

3. Eksprimen Instalasi Jupyter Notebook

INSTALASI JUPYTER NOTEBOOK

Dan untuk mulai package pada environment dsf kita, akan dimulai dengan instalasi package paling penting, yaitu package Jupyter Notebook, dan ini bisa dilakukan dengan perintah berikut:

conda install jupyter

Apabila perintah diatas dijalankan, akan ditampilkan pesan Proceed ([y]/n)? yang meminta persetujuan kita untuk melanjutkan instalasi, dan silakan tekan tombol y di keyboard untuk lanjut instalasi Jupyter Notebook. Setelah selesai instalasi, dapat dilakukan uji coba menjalankan Jupyter Notebook dengan perintah berikut di anaconda prompt:

a. jupyter notebook

   Bila perintah diatas berhasil, maka anaconda prompt akan menginstruksikan untuk membuka browser, dan laman URL browser akan langsung diarahkan menuju http://localhost:8888/tree.

   Dan terkadang, apabila kita ingin menjalankan jupyter notebook dalam mode server, yang mempunyai arti komputer yang berada dalam jaringan yang sama dengan komputer server tersebut juga bisa menjalankan Jupyter Notebook,

maka perlu ditambahkan parameter ip yang menyatakan ip dari komputer server tersebut.

b. jupyter notebook --ip alamatip

Dan selain menjalankan perintah diatas, kita juga perlu membuat password, agar server kita aman terhadap pengguna yang tidak diinginkan, yaitu dengan perintah berikut:

jupyter notebook password

Setelah menjalankan perintah diatas, kita harus mengisi password baru dan verifikasinya, dan password ini akan digunakan setiap kita ingin mengakses Jupyter Notebook.

4. Eksprimen Instalasi lainnya

Contoh diatas adalah contoh untuk instalasi package jupyter notebook, dan untuk eksperimen lainnya silakan lanjutkan dengan instalasi package Numpy, Pandas, dan Matplotlib.

b. Package Numpy

Untuk instalasi package NumPy, lakukan perintah berikut pada Anaconda Prompt dan diakhiri dengan menjalankan Jupyter Notebook:

1. conda activate dsf
2. conda install numpy
3. jupyter notebook

Untuk mengecek versi dari setiap package yang telah diinstall, dapat menggunakan atribut **__version__**setelah kita mengimport package tersebut.

```
In [1]: 1 import numpy as np

In [2]: 1 type(np)
Out[2]: module

In [3]: 1 np.__version__
Out[3]: '1.20.3'
```

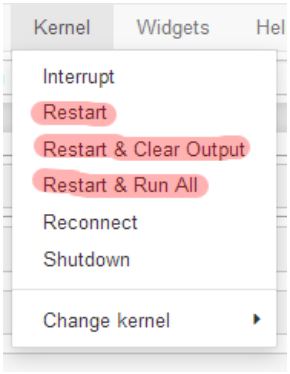
Penjelasan:

1. Perintah **import numpy as np**, mempunyai arti kita mengimport package **NumPy** dan disingkat dengan perintah as menjadi **np** untuk memudahkan pengetikan

kode;

2. Perintah kedua, menampilkan tipe dari objek **np**, dan dikarenakan np adalah objek yang diimport dari package **NumPy**, maka dipastikan objek ini berbentuk **module**;
3. Pada perintah ketiga, atribut **np.__version__**dapat digunakan untuk mengecek versi dari package NumPy.

Pada contoh diatas, versi NumPy yang digunakan adalah versi **1.20.3.** n ingat untuk selalu menggunakan fungsi **Restart** dan variannya untuk membersihkan memori kernel dan mengulang kembali proses eksekusi kode.



Gambar 15. Fungsi pada NumPy

c. Indexing

Package NumPy adalah sebuah package khusus untuk scientific computing dengan bahasa pemrograman Python, yang memang dirancang khusus untuk mempermudahan pengolahan data yang berbentuk numerik.

Sesuai tulisan di situsnya, package NumPy mempunyai kemampuan khusus, yaitu:

1. Kemampuan untuk mengelola N-dimensi objek array;
2. Fungsi - fungsi pendukung yang lengkap dalam mengolah data numerik;
3. Fungsi-fungsi pendukung operasi aljabar aritmatika yang sudah diimplementasikan;
4. Dan integrasi dengan bahasa pemrograman C/C++ dan juga Fortran.

Untuk menggunakan package NumPy, terlebih dahulu, pastikan subtopik Managing

Packages sudah dilakukan, dipelajari dan diimplementasi sampai proses instalasi NumPy sudah berhasil.

Sekilas, objek Numpy, mirip dengan objek List, seperti contoh berikut:

```
In [ ]: x_list = [i for i in range(10)]
In [ ]: type(x_list)
In [ ]: print(x_list)
In [ ]: x_list
In [ ]: import numpy as np
In [ ]: x_numpy = np.arange(10)
In [ ]: type(x_numpy)
In [ ]: print(x_numpy)
In [ ]: x_numpy
```

Penjelasan:

1. Metode **arange()** pada object **x_numpy**, berfungsi untuk membuat sebuah objek **NumPy** baru yang berisikan angka 0 sampai 9 yang sama seperti contoh pada **list** diatas**;**
2. Type dari **x_numpy** adalah **ndarray**, atau **N Dimension Array**.

Salah satu perbedaan **List** dan **NumPy** terletak pada proses pemetaan memori dan tipe datanya, dimana pada **List**, objek apapun bisa dimasukan sebagai elemen pada **List**, tetapi khusus untuk **NumPy**, elemen yang bisa dimasukan umumnya hanya satu tipe saja, dan umumnya tipe numerik, karena memang pada dasarnya package **NumPy** berfungsi untuk mengolah data numerik. Perhatikan contoh berikut:

```
In [ ]: x = [1, 2.0, 'tiga']
In [ ]: print('Tipe dari variabel x adalah: {}'.format(type(x)))
In [ ]: for idx, elt in enumerate(x):
            print('Tipe dari elemen ke - {} adalah elemen {}'.format(idx, type(elt)))
In [ ]: import numpy as np
In [ ]: x = np.arange(3)
In [ ]: print('Tipe dari variabel x adalah: {}'.format(type(x)))
In [ ]: print('Tipe data dari isi variabel x adalah: {}'.format(x.dtype))
In [ ]: for idx, elt in enumerate(x):
            print('Tipe dari elemen ke - {} adalah elemen {}'.format(idx, type(elt)))
```

Penjelasan:

1. Apa fungsi perintah **format()** dalam fungsi **print()** tersebut? Diskusikan dalam

Forum Diskusi;

2. Atribut **dtype**, dapat digunakan untuk mengetahui isi tipe data dari objek **NumPy**.

Indexing

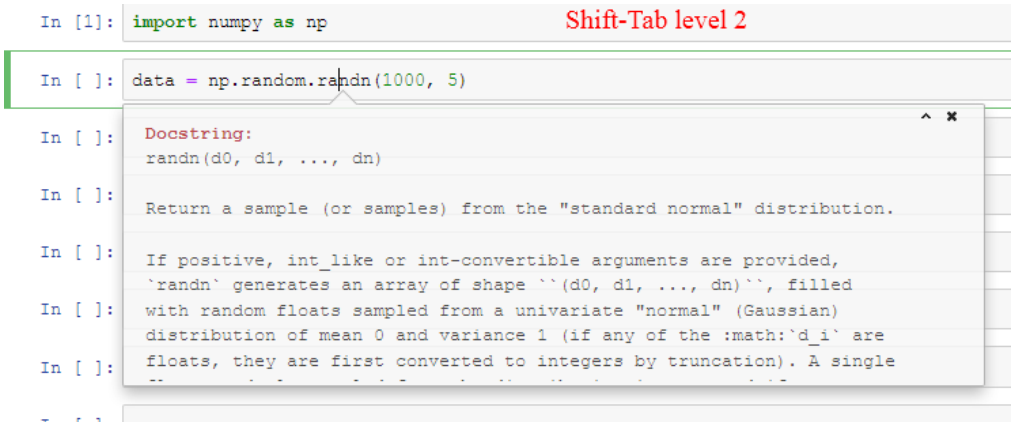
Objek array dalam NumPy mempunyai sistem indexing yang dimulai dari angka 0 (nol), dan proses pencarian data didalam objek NumPy bisa dengan menggunakan angka index sesuai dengan jumlah dimensi yang telah dibuat sebelumnya.

Salah satu keunggulan pada Jupyter Notebook, adalah proses untuk mendapatkan informasi mengenai fungsi, module, maupun attribute dengan cepat dan mudah, yaitu menggunakan tombol **Shift-Tab** dengan menyorot kepada nama bagian yang ingin kita lihat informasinya.

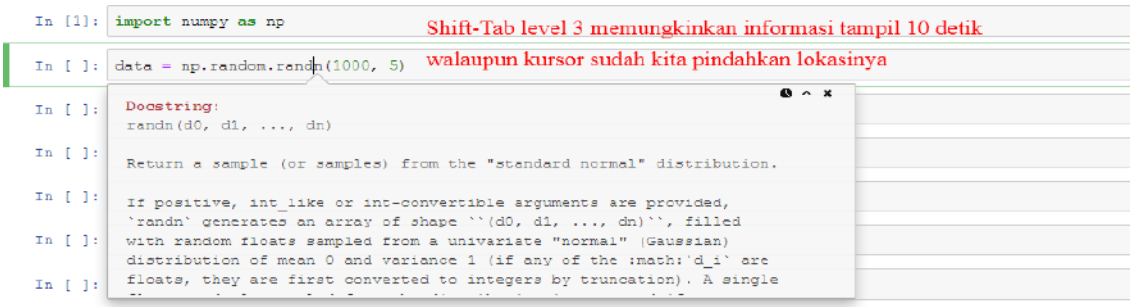
Tombol **Shift-Tab** bisa diproses sampai 4 (empat) level kedalaman informasi, tetapi harus dipastikan untuk mengimport terlebih dahulu packagenya sebelum mencoba tombol **Shift-Tab**.

Seperti pada contoh berikut:

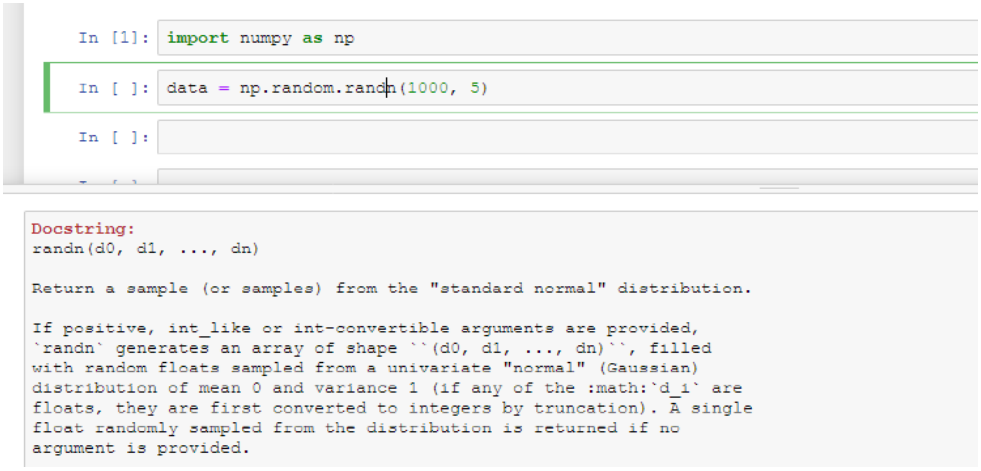
Shift-Tab level 2 (dua):



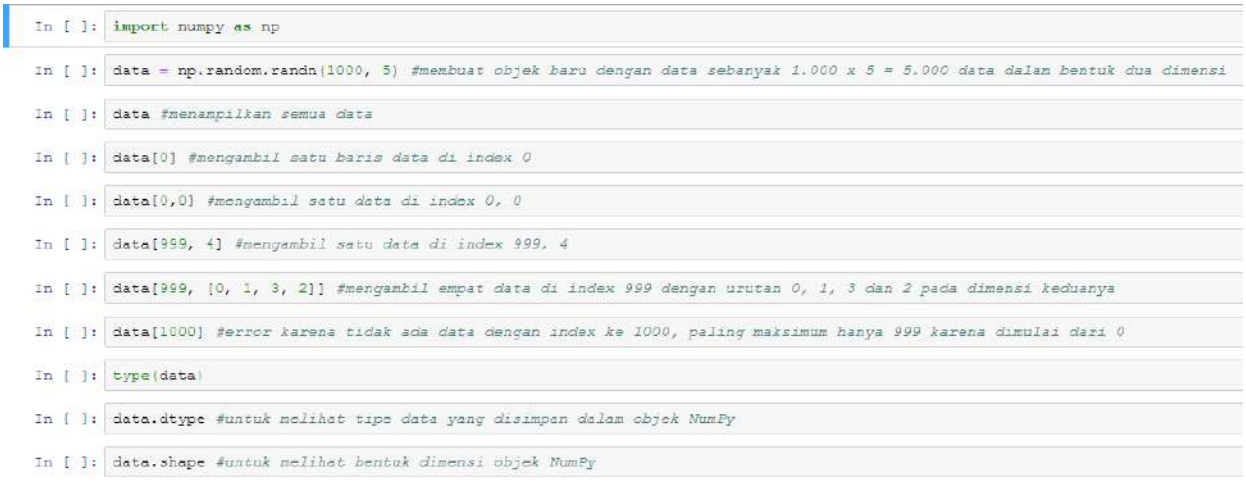
Shift-Tab level 3 (tiga):



Shift-Tab level 4 (empat):



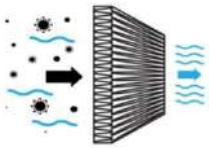
Berikut ini adalah contoh proses indexing dalam mengakses data yang tersedia didalam objek NumPy, dengan menggunakan bantuan fungsi **randn()** untuk menghasilkan data acak.



Proses indexing juga bisa dilakukan dengan menggunakan tanda minus ataupun bantuan dari atribut shape.

```
In [ ]: import numpy as np #perintah 4, 5 dan 9 akan menghasilkan hasil yang sama dengan cara yang berbeda
In [ ]: data = np.random.randn(1000, 5)
In [ ]: data
In [ ]: data[999, 4] #untuk mengakses data terakhir di index 999, 4
In [ ]: data[-1, -1] #untuk mengakses data terakhir di index 999, 4 dengan menggunakan tanda minus
In [ ]: data.shape #menghasilkan tuple
In [ ]: data.shape[0]
In [ ]: data.shape[1]
In [ ]: data[data.shape[0] - 1, data.shape[1] - 1] #untuk mengakses data terakhir dengan bantuan atribut shape dikurang 1
```

d. Slicing & Filtering



Import Data

Sebelum mempelajari proses **slicing** atau **filtering**, kita akan mempelajari terlebih dahulu bagaimana cara mengimport data dari file teks dengan format CSV yang sudah disiapkan pada file Data Uji Coba Obat Kepada Pasien **(peradangan.csv)**. File CSV merupakan singkatan dari comma-separated values, dan ini merupakan file teks biasa yang dituliskan dengan format khusus, dimana pembeda antar kolom menggunakan huruf koma, atau terkadang juga bisa menggunakan huruf titik koma. Silakan download terlebih dahulu file **peradangan.csv**, dan letakan pada posisi yang sama dengan file Jupyter Notebook yang akan digunakan, dan lihat isi dari file tersebut dengan menggunakan aplikasi Notepad.

Proses import bisa dilakukan dengan menggunakan fungsi **loadtxt()** dengan menyebutkan nama file dan juga karakter pemisah data (**delimiter**) didalam file tersebut, dan khusus untuk file dengan format CSV, delimiter yang umum digunakan adalah **tanda koma (,)**.

```
In [ ]: import numpy as np
In [ ]: data = np.loadtxt(fname='peradangan.csv', delimiter=',')
In [ ]: data
In [ ]: data.dtype
In [ ]: data.shape
```

Dalam memproses data, pada contoh diatas, selalu ingat untuk mengecek data tersebut

dengan menggunakan atribut **dtype** dan **shape**, agar kita lebih jelas dalam menghadapi data apa yang akan kita proses, dan pada contoh dibawah kita akan menggunakan package matplotlib.pyplot untuk menggambar grafik.

```
In [ ]: import matplotlib.pyplot as plt
In [ ]: %matplotlib inline
In [ ]: plt.matshow(data)
In [ ]: data[0]
```

Pada contoh diatas, kita akan mengimport library matplotlib, dan dengan dengan library ini, kita akan menggambarkan data peradangan.csv dalam grafik dua dimensi, dan pada gambar tersebut, nampak jelas, tingkat peradangan naik pada pertengahan uji coba pada hari ke 15 sampai 25.

1. Slicing

   Proses slicing pada Numpy sama dengan proses slicing pada, yaitu **penggunaan start:stop:stride.**

```
In [ ]: data[0] #mengambil data pasien pertama
In [ ]: data[0, 0:10] #mengambil data pasien pertama hari pertama sampai ke sepuluh
In [ ]: data[0, :10] #apabila komponen start dihilangkan, maka secara default dianggap dari data pertama
In [ ]: data[0, :] #sama dengan data[0]
In [ ]: data[0, ::2] #mengambil seluruh data pasien pertama mulai hari pertama, kelipatan 2
In [ ]: data[0, 14] #mengambil data pasien pertama, pada hari ke lima belas
In [ ]: data[::, 2] #mengambil data seluruh pasien pada hari ketiga
In [ ]: data[::, 2:5] #mengambil data seluruh pasien mulai hari ketiga sampai kelima
In [ ]: data[10:15, 39] #mengambil data pasien ke sebelas sampai lima belas untuk hari ke empat puluh
In [ ]: data[10:15, 35:40] #mengambil data pasien ke sebelas sampai lima belas untuk hari ke tiga puluh enam sampai empat puluh
In [ ]: data[::, 40:42] #mengambil data seluruh pasien pada hari ke empat puluh satu sampai empat puluh tiga, apakah bisa ?
In [ ]: plt.matshow(data[::, 15:25]) #melihat grafik seluruh pasien pada hari ke enam belas sampai dua puluh lima
```

   Perhatikan contoh diatas, dan pahami dengan baik perintah slicing, terutama perbedaan pada **start** dan **stop**, yang memang terkadang agak membingungkan.

   Dan untuk mempermudah pengamatan, kode berikut akan menampilkan grafik dari pasien pertama dengan data mulai dari hari pertama sampai hari terakhir dan juga variannya.

```
In [ ]: plt.plot(data[0, :])
In [ ]: plt.plot(data[0, 0:15])
In [ ]: plt.plot(data[0, 15:25])
```

EKSPERIMEN 1

Bagaimanakah caranya mengambil data pasien ke **tiga puluh satu** sampai **enam puluh** hanya untuk hari ke **tiga puluh** sampai **empat puluh** ?

EKSPERIMEN 2

Bagaimanakah caranya mengambil data **seluruh pasien** pada hari **pertama sampai ketiga**, digabung dengan hari ke **lima belas** sampai **delapan belas**?

Tips: gunakan fungsi **concatenate()** dalam NumPy.

2. Filtering

Filtering adalah proses untuk mengambil data berdasarkan kondisi filtering yang kita terapkan.

```
In [ ]: data > 0 #apabila data 0 maka False, dan sebaliknya True
In [ ]: data[data > 0] #akan menampilkan seluruh data peradangan yang skalanya lebih besar dari 0
In [ ]: pasien_pertama = data[0,:] #mengambil data pasien pertama
In [ ]: pasien_pertama[pasien_pertama > 10] #melihat hari dimana kondisi pasien dengan tingkat peradangan lebih dari skala 10
In [ ]: pasien_pertama[pasien_pertama > 10].shape #melihat jumlah hari pada tingkat peradangan lebih dari skala 10
In [ ]: pasien_pertama[pasien_pertama == 0].shape #melihat jumlah hari dengan tingkat peradangan skala 0
In [ ]: seluruh_pasien = data[:, 15] #mengambil data seluruh pasien pada hari ke enam belas
In [ ]: seluruh_pasien[seluruh_pasien > 14].shape #berapa jumlah pasien yang tingkat peradangan lebih dari skala 14
In [ ]: seluruh_pasien = data[:, 1] #mengambil data seluruh pasien pada hari kedua
In [ ]: seluruh_pasien[seluruh_pasien != 0].shape #jumlah pasien yang mulai mengalami gejala peradangan pada hari kedua
```

EKSPERIMEN 3

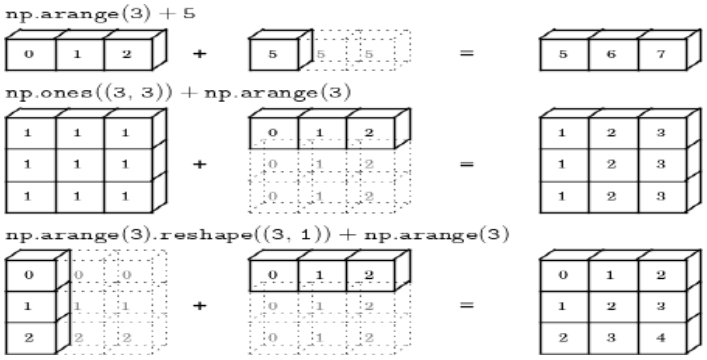
Berapa jumlah pasien yang sudah sembuh pada hari terakhir?

e. Broadcasting

Broadcasting adalah proses penggabungan dua objek NumPy, dimana proses penggabungan ini memungkinan dua objek NumPy dengan dimensi yang berbeda, dapat bergabung menjadi satu.

Broadcasting akan menerapkan tiga aturan berikut dalam proses penggabungannya, yaitu:

1. Jika dua objek berbeda dalam jumlah dimensinya, bentuk objek dengan dimensi yang lebih sedikit diisi dengan yang di sisi terdepan (kiri);
2. Jika bentuk dua objek tidak cocok dalam dimensi apa pun, objek dengan bentuk yang sama dengan 1 dalam dimensi tertentu direntangkan agar sesuai dengan bentuk dimensi lainnya;
3. Jika dalam dimensi apa pun ukurannya tidak sesuai dan tidak memiliki bentuk sama dengan 1, maka akan menimbulkan pesan kesalahan.



Amati gambar diatas, dan perhatikan bentuk dimensinya, terutama dimensi yang berbentuk bayangan, dimana bayangan tersebut adalah bentuk broadcasting objek NumPy sebelum operasi pertambahan dilakukan.

Untuk memperjelas gambar diatas, silakan dicoba rangkaian contoh berikut, fungsi **arange()** berfungsi untuk membuat objek **1 x n** dan untuk **n** merupakan argumen pada fungsi tersebut, bila **n = 3** maka akan dihasilkan objek dengan dimensi **1 x 3**.

```
In [ ]: #contoh pertama
In [ ]: a = np.arange(3)
In [ ]: a
In [ ]: a.shape #Objek a merupakan NumPy dengan dimensi 1x3
In [ ]: b = 5
In [ ]: b
In [ ]: c = a + b #Pada saat pertambahan, objek b secara otomatis akan mengembangkan diri menjadi dimensi 1x3
In [ ]: c
In [ ]: c.shape
```

Pada contoh kedua, fungsi **ones()** adalah fungsi untuk menciptakan objek NumPy baru yang secara otomatis diisi dengan angka satu.

```
In [ ]: #contoh kedua
In [ ]: a = np.ones((3,3))
In [ ]: a
In [ ]: a.shape #Objek a merupakan NumPy dengan dimensi 3x3
In [ ]: b = np.arange(3)
In [ ]: b
In [ ]: b.shape #Objek b merupakan NumPy dengan dimensi 1x3
In [ ]: c = a + b #Pada saat pertambahan, objek b secara otomatis akan mengembangkan diri menjadi dimensi 3x3
In [ ]: c
In [ ]: c.shape
```

Contoh ketiga, adalah contoh pertambahan objek dengan dimensi **3x1** dengan **1x3**, yang akan menghasilkan objek dengan dimensi **3x3**, dan untuk objek dengan dimensi **3x1** didapatkan dengan menggunakan fungsi **reshape()** dari dimensi **1x3**.

```
In [ ]: #contoh ketiga
In [ ]: a = np.arange(3)
In [ ]: a
In [ ]: a.shape #Objek a merupakan NumPy dengan dimensi 1x3
In [ ]: a = a.reshape((3,1)) #Diubah dimensinya dengan fungsi reshape() menjadi 3x1
In [ ]: a
In [ ]: a.shape  #Objek a merupakan NumPy dengan dimensi 3x1
In [ ]: b = np.arange(3)
In [ ]: b
In [ ]: b.shape #Objek b merupakan NumPy dengan dimensi 1x3
In [ ]: c = a + b #Pada saat pertambahan, objek a dan b secara otomatis akan mengembangkan diri menjadi dimensi 3x3
In [ ]: c
```

Salah satu keunggulan broadcasting dalam NumPy, adalah kemampuan untuk melakukan proses perkalian matriks dengan perintah sederhana, ikuti contoh berikut:

```
In [ ]: a = np.ones((2,3)) + 2
In [ ]: a
In [ ]: b = np.ones((3,2)) + 3
In [ ]: b
In [ ]: c = a * b #bisa jalan atau tidak ?
In [ ]: c = np.dot(a,b) #Fungsi dot() untuk perkalian matriks
In [ ]: c
In [ ]: c.shape
In [ ]: c = a @ b #Operator @ khusus untuk perkalian matriks dengan fungsi yang sama dengan fungsi dot()
In [ ]: c
In [ ]: c.shape
```

EKSPERIMEN

1. Apakah proses pertambahan bisa dilakukan untuk objek NumPy dengan dimensi 2x3 dan 3x2 ?
2. Apakah proses perkalian bisa dilakukan untuk objek NumPy dengan dimensi 2x3 dan 3x2 ?

f. Package Pandas

Package Pandas, adalah sebuah package yang ditujukan untuk pengelolaan data mirip seperti pada package NumPy, tetapi dengan pengembangan berupa penambahan label untuk baris dan kolom, bisa mengatur tipe data yang variatif, dan masih banyak lainnya. Apa alasan kita menggunakan Pandas dibandingkan dengan NumPy ? Apabila kita sering bekerja dengan data tabular atau data terstruktur, seperti data yang disimpan dalam dokumen Excel, Calc, dan sebagainya.

Umumnya package Pandas digunakan untuk:

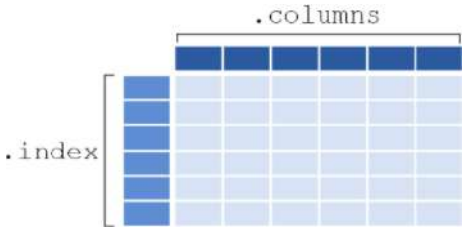
1. Impor data dari sumber data yang sudah ada;
2. Membersihkan data, bisa berupa data noise, dan sebagainya;
3. Melakukan ekslorasi terhadap data, untuk mendapatkan pemahaman terhadap data yang sedang kita kerjakan;

4. Mempersiapkan dan memproses data lebih lanjut untuk dianalisa;
5. Melakukan proses analisa data, dengan bantuan dari package lainnya.

Apabila kita bekerja dengan data yang sifatnya variatif, dan berbentuk tabular, maka gunakanlah package Pandas, tetapi apabila kita hanya bekerja dengan data tunggal, seperti numerik, gunakanlah package NumPy, jangan dipaksakan dengan package Pandas.

Dataframe

Sebuah **DataFrame** adalah sebuah struktur data tabular, yang terdiri dari baris dan kolom yang mempunyai label, mirip seperti contoh spreadsheet pada excel, tabel pada database, dan sebagainya.



Sebelum mencoba package Pandas, pastikan bahwa package ini sudah terinstalasi di komputer masing-masing dengan menggunakan perintah **conda install pandas.** Dan pastikan Anda sudah mendownload file **titanic.csv** sebelum memulai contoh dibawah ini.

```
In [ ]: import pandas as pd
In [ ]: df = pd.read_csv("titanic.csv")
In [ ]: df.head()
```

Fungsi **head()** diatas berfungsi untuk menampilkan beberapa baris data awal yang terkandung didalam file **titanic.csv**.

Dalam DataFrame package Pandas, dikenal atribut **index** untuk nama bagi baris data, dan atribut **columns** untuk nama bagi kolom data.

```
In [ ]: import pandas as pd
In [ ]: df = pd.read_csv("titanic.csv")
In [ ]: df.index #atribut untuk melihat statistik jumlah baris data
In [ ]: df.columns #atribut untuk melihat nama kolom data
In [ ]: df.dtypes #atribut untuk melihat nama kolom data beserta tipe datanya
In [ ]: df.info() #fungsi untuk melihat informasi statistik data
In [ ]: df.values #atribut untuk melihat data keseluruhan
```

Selain atribut **index** dan **columns**, juga ada beberapa atribut dasar lainnya, seperti **dtypes, values,** dan fungsi **info()**. Berikut ini adalah contoh data yang dimasukan secara manual dengan menggunakan **tanda kurung kurawal ({})**, **titik dua (:)**, dan **kurung siku ([])**.

```
In [ ]: data_kota = {'kota': ['DKI Jakarta', 'Surabaya', 'Medan', 'Bandung', 'Makassar'],
                    'populasi': [9992842, 2806306, 2467183, 2341097, 1652305],
                    'area': [664.01, 350.54, 265, 167.67, 199.26]
                   }

In [ ]: df_kota = pd.DataFrame(data_kota)

In [ ]: df_kota

In [ ]: df_kota.head()

In [ ]: df_kota.index

In [ ]: df_kota.columns

In [ ]: df_kota.dtypes

In [ ]: df_kota.values

In [ ]: df_kota.info()
```

Series

Salah satu keunggulan dari package Pandas, adalah manajemen data series atau data kolom dengan menggunakan label yang sesuai pada label di data.

```
In [ ]: df['Age'] #menggunakan nama kolom untuk mengambil data kolom Age

In [ ]: umur = df['Age'] #menyimpannya ke dalam sebuah variabel baru

In [ ]: umur.index #atribut index

In [ ]: umur.values #atribut values

In [ ]: umur.values[:10] #mengambil data umur penumpang pertama sampai ke sepuluh

In [ ]: umur * 100 #mengalikan seluruh data umur dengan angka seratus

In [ ]: umur.mean() #menghitung umur rata - rata

In [ ]: umur[ umur > 70 ] #mencari penumpang dengan umur diatas 70 tahun

In [ ]: df[96:97] #mencari kebenaran data penumpang ke 96
```

Fungsi - fungsi lain yang umum dipakai, dapat dilihat pada contoh berikut:

```
In [ ]: df['Embarked'].value_counts() #menghitung jumlah data di Embarked

In [ ]: df['Fare'].max() #mencari harga tiket paling mahal

In [ ]: df['Fare'].median() #mencari median dari harga tiket

In [ ]: df['Survived'].sum() / len(df['Survived']) #mencari rata - rata rasio penumpang yang selamat

In [ ]: df['Survived'].mean() #sama dengan contoh sebelumnya
```

g. Selecting & Filtering

Proses selecting dan filtering pada library Pandas, mirip dengan library NumPy, tetapi dengan perbedaan adanya tambahan series, sehingga proses selecting dan filtering bisa dilakukan dengan menggunakan label pada data.

```
In [ ]: import pandas as pd
In [ ]: df = pd.read_csv("titanic.csv")
In [ ]: df.head(3) #untuk melihat tiga data awal
In [ ]: df.tail() #untuk melihat sampel data akhir
In [ ]: df.describe() #melihat data statistik dasar untuk data yang bersifat numerik
In [ ]: df['Age'] #untuk mengambil data kolom umur
In [ ]: df[['Age', 'Fare']] #untuk mengambil data kolom umur dan harga
In [ ]: df[['Name','Age','Survived']] #untuk mengambil data kolom nama, umur, dan selamat
```

1. Selecting

   Atribut **loc** dan **iloc**, bisa digunakan apabila index pada data bukan menggunakan data numerik tetapi data teks. Pada contoh berikut, fungsi **set_index()** akan mengubah indexing dari awalnya numerik, menjadi indexing dengan nama penumpang.

```
In [ ]: import pandas as pd
In [ ]: df = pd.read_csv("titanic.csv")
In [ ]: df
In [ ]: df_baru = df.set_index('Name')
In [ ]: df_baru
```

   Setelah menjalankan fungsi **set_index()**, kita bisa melakukan operasi pencarian dengan berdasarkan pada nama penumpang, sesuai contoh berikut:

```
In [ ]: df_baru.loc['Bonnell, Miss. Elizabeth'] #mencari berdasarkan nama penumpang
In [ ]: df_baru.loc['Bonnell, Miss. Elizabeth', 'Fare'] #mencari harga dari nama penumpang tersebut
In [ ]: df_baru.loc['Bonnell, Miss. Elizabeth':'Andersson, Mr. Anders Johan', :] #mencari penumpang dengan sistem slicing
In [ ]: df_baru.iloc[0:2,1:3] #fungsi iloc() bisa digunakan untuk mencari berdasarkan nomor indeks
In [ ]: df_baru.loc['Braund, Mr. Owen Harris', 'Survived'] = 100 #kita bisa mengubah langsung data
In [ ]: df_baru
```

2. Filtering

   Silakan eksperimen kode berikut untuk memahami bagaimana cara kerja proses filtering pada Pandas dengan memanfaatkan series.

```
In [ ]: df['Fare'] > 50 #menampilkan data harga yang lebih besar dari 50
In [ ]: df[df['Fare'] > 50] #menampilkan data penumpang dengan harga lebih besar dari 50
In [ ]: df[df['Sex'] == 'male'] #menampilkan penumpang dengan jenis kelamin laki - laki
In [ ]: df.loc[df['Sex'] == 'male', 'Age'].mean() #menampilkan umur rata - rata penumpang laki - laki
In [ ]: df.loc[df['Sex'] == 'female', 'Age'].mean() #menampilkan umur rata - rata penumpang perempuan
In [ ]: len(df[df['Age'] > 70]) #menampilkan jumlah penumpang dengan umur lebih dari tujuh puluh tahun
In [ ]: (df['Age'] > 70).sum() #sama seperti contoh sebelumnya
```

EKSPERIMEN

1. Berapakah jumlah penumpang laki - laki yang selamat?
2. Berapakah jumlah penumpang laki - laki yang tidak selamat?
3. Berapakah jumlah penumpang perempuan yang selamat?
4. Berapakah jumlah penumpang perempuan yang tidak selamat?

h. Group by Operation

Sebelum mempelajari **group-by**, kita akan melihat contoh berikut terlebih dahulu:

```
In [ ]: import pandas as pd
In [ ]: df = pd.DataFrame({'key':['A','B','C','A','B','C','A','B','C'],
                           'data': [0, 5, 10, 5, 10, 15, 10, 15, 20]})
In [ ]: df
```

Apabila kita ingin menghitung jumlah keseluruhan data, maka dapat digunakan fungsi

sum(), akan tetapi bagaimana dengan jumlah keseluruhan data berasarkan kunci masing-masing?

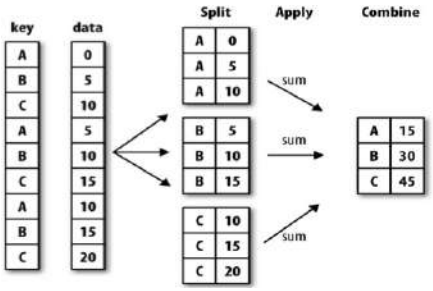
```
In [ ]: df['data'].sum() #mencari jumlah keseluruhan data
```

```
In [ ]: #mencari jumlah keseluruhan data berdasarkan kunci
        for key in ['A', 'B', 'C']:
            print(key, df[df['key'] == key]['data'].sum())
```

Ketika kita menganalisa data, sering kita jumpai proses statistika dasar, seperti penjumlahan, rata - rata, maksimum, minimum dan sebagainya. Akan tetapi apabila kita harus melakukan proses statistika dasar berdasarkan kunci, maka peran group-by sangat dibutuhkan untuk menjawab permasalahan tersebut.

Konsep group-by disini mempunyai arti, kita akan menerapkan fungsi - fungsi untuk subset dari DataFrame, berdasarkan pada kunci- kunci yang digunakan untuk membagi data tersebut. Operasi group-by biasanya diartikan dengan istilah **split-apply-combine**, yang terdiri dari tiga tahap, yaitu:

1. **Splitting**, proses untuk memisahkan data ke dalam grup - grup sesuai kriteria;
2. **Applying**, proses untuk memberikan fungsi kepada setiap grup independen;
3. **Combining**, proses untuk menggabungkan hasil ke dalam sebuah struktur data.



Bagi yang pernah mempelajari perintah Group By pada bahasa pemrograman SQL, maka konsep group-by disini sama dengan konsep group-by pada bahasa pemrograman tersebut.

```
In [ ]: df[df['key'] == "A"].sum()
```

```
In [ ]: df[df['key'] == "B"].sum()
```

```
In [ ]: df[df['key'] == "C"].sum()
```

```
In [ ]: df.groupby('key').sum() #fungsi groupby dapat menggantikan tiga perintah diatas
```

Contoh dibawah akan menghasilkan hasil yang sama, tapi dengan cara yang berbeda:

```
In [ ]: import numpy as np
```

```
In [ ]: #fungsi aggregate dapat digunakan untuk mengambil fungsi dari library lain
        df.groupby('key').aggregate(np.sum)
```

```
In [ ]: df.groupby('key')['data'].sum() #contoh lain dengan memanggil kolom data terlebih dahulu
```

Dan dengan melihat kembali contoh pada file titanic.csv, kita dapat melakukan eksperimen-eksperimen berikut:

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

```
In [ ]: df = pd.read_csv("titanic.csv")
```

```
In [ ]: df.head()
```

```
In [ ]: df.groupby('Sex')['Age'].mean() #kalkulasi umur rata - rata berdasarkan jenis kelamin
```

```
In [ ]: #karena survived hanya berisikan 1 dan 0, maka kalau disum, dapat diketahui jumlah yang selamat
        df.groupby('Sex')['Survived'].sum()
```

```
In [ ]: df.groupby(['Pclass', 'Sex']).size() #mengecek jumlah jenis kelamin berdasarkan kelas penumpang
```

```
In [ ]: df.groupby(['Sex', 'Survived']).size() #mengecek jumlah jenis kelamin yang selamat dan tidak selamat
```

Berikut ini adalah contoh fungsi dan **aggregate()** dan fungsi pembantu **mean()** yang ada:

```
In [ ]: #persentase selamat berdasarkan jenis kelamin dengan fungsi aggregate
        df.groupby('Sex')[['Survived']].aggregate(lambda x: x.sum() / len(x))
```

```
In [ ]: df.groupby('Sex')[['Survived']].mean() #hasil yang sama
```

```
In [ ]: #persentase selamat berdasarkan kelas penumpang dengan fungsi aggregate
        df.groupby('Pclass')['Survived'].aggregate(lambda x: x.sum() / len(x))
```

```
In [ ]: df.groupby('Pclass')['Survived'].mean() #hasil yang sama
```

Pandas dapat juga digabungkan dengan library MatPlotLib, seperti pada contoh berikut:

```
In [ ]: import matplotlib as plt

In [ ]: %matplotlib inline

In [ ]: import pandas as pd

In [ ]: df = pd.read_csv("data/titanic.csv")

In [ ]: df['Age'].hist()

In [ ]: df['Pclass'].hist()

In [ ]: df.groupby('Pclass').size().plot(kind='pie')

In [ ]: df.groupby('Sex').size().plot(kind='bar')

In [ ]: df.groupby('Pclass')['Survived'].mean().plot(kind='barh')
```

EKSPERIMEN

1. Berapa jumlah penumpang kelas 3 laki - laki yang selamat?
2. Berapa jumlah penumpang kelas 1 perempuan yang selamat?

i. Pengenalan Notebook

Data Acquisition

Ada berbagai format untuk dataset, .csv, .json, .xlsx dll. Dataset dapat disimpan di tempat yang berbeda, di perangkat lokal atau terkadang online. Di bagian ini, Anda akan mempelajari cara memuat kumpulan data ke dalam Jupyter Notebook.

Dalam kasus ini, Ada Kumpulan Data Mobil yang bersumber online, dan dalam format CSV (nilai yang dipisahkan koma). Mari kita gunakan dataset ini sebagai contoh untuk berlatih membaca data.

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

Data type: csv

Library Pandas adalah tool yang berguna yang memungkinkan kita membaca berbagai kumpulan data ke dalam data frame. Sebelum kita memulai ada baiknya menginstall library pandas dan numpy di environment kita.

```
[ ]: conda install pandas
     conda install numpy
```

Untuk mengimport library pandas dan numpy

```
[ ]: # import library
     import pandas as pd
     import numpy as np
```

Read Data

Kita menggunakan fungsi pandas.read_csv() untuk membaca file csv. Di dalam kurung, kami menempatkan jalur file bersama dengan tanda kutip, sehingga panda akan membaca file ke dalam data frame dari alamat itu. Jalur file dapat berupa URL atau alamat file lokal Anda.

Karena data tidak menyertakan header, kita dapat menambahkan argumen headers = None di dalam metode read_csv(), sehingga pandas tidak secara otomatis menetapkan baris pertama sebagai header. Anda juga dapat menetapkan kumpulan data ke variabel apa pun yang Anda buat.

```
[ ]: df = pd.read_csv(auto.csv, header=None)
```

Setelah membaca dataset, kita dapat menggunakan metode dataframe.head(n) untuk memeriksa n baris teratas dari dataframe; dimana n adalah bilangan bulat. Berlawanan dengan dataframe.head(n), dataframe.tail(n) akan menunjukkan n baris terbawah dari dataframe.

```
[ ]: # tampilkan 5 baris pertama menggunakan metode dataframe.head()
     print("5 baris pertama dari data frame")
     df.head(5)
```

Soal 1 :

Periksa 10 baris terbawah dari data frame "df".

Save Dataset

Sejalan dengan itu, Pandas memungkinkan kita untuk menyimpan dataset ke csv dengan menggunakan metode dataframe.to_csv(), Anda dapat menambahkan path file dan nama bersama dengan tanda kutip dalam tanda kurung.

Misalnya, jika Anda akan menyimpan dataframe df sebagai mobil.csv ke penyimpanan lokal Anda, Anda dapat menggunakan sintaks di bawah ini:

```
df.to_csv("automobile.csv", index=False)
```

Kita juga bisa membaca dan menyimpan format file lain, kita bisa menggunakan fungsi yang mirip dengan pd.read_csv() dan df.to_csv() untuk format data lainnya, fungsinya tercantum dalam tabel berikut:

Read/Save Format Data Lain

Data Formate	Read	Save
csv	pd.read_csv()	df.to_csv()
json	pd.read_json()	df.to_json()
excel	pd.read_excel()	df.to_excel()
hdf	pd.read_hdf()	df.to_hdf()
sql	pd.read_sql()	df.to_sql()
...	...	...

Add Headers

Lihatlah kumpulan data, pandas secara otomatis mengatur header dengan bilangan bulat dari 0. Jadi, kita harus menambahkan header secara manual.

Pertama, kita akan membuat daftar "headers" yang menyertakan semua nama kolom secara berurutan. Kemudian menggunakan dataframe.columns = headers untuk mengganti header dengan daftar yang telah ada

```
[ ]: # create headers list
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"]
print("headers\n", headers)
```

Kita akan mengganti header dan memeriksa ulang data frame kita.

```
[ ]: df.columns = headers
df.head(10)
```

Kita perlu mengganti "?" simbol dengan NaN sehingga dropna() dapat menghapus nilai yang hilang

```
[ ]: df1=df.replace('?',np.NaN)
```

kita dapat menghapus nilai yang hilang di sepanjang kolom "harga" sebagai berikut:

```
[ ]: df=df1.dropna(subset=["price"], axis=0)
df.head(20)
```

Sekarang, kita telah berhasil membaca dataset mentah dan menambahkan header

yang benar ke dalam data frame.

Soal 2 :

Temukan nama kolom di data frame

Basic Insight of Dataset

Setelah membaca data ke dalam data frame Panda, saatnya kita menjelajahi dataset. Ada beberapa cara untuk mendapatkan wawasan penting tentang data untuk membantu kami lebih memahami kumpulan data kami.

Tipe Data

Data memiliki berbagai jenis. Jenis utama yang disimpan dalam data frame Pandas adalah objek, float, int, bool, dan datetime64. Untuk mempelajari setiap atribut dengan lebih baik, selalu baik bagi kita untuk mengetahui tipe data setiap kolom. Di Pandas: dtypes mengembalikan Seri dengan tipe data setiap kolom.

```
[ ]: # check the data type of data frame "df" by .dtypes
     print(df.dtypes)
```

Akibatnya, seperti yang ditunjukkan di atas, jelas terlihat bahwa tipe data "symboling" dan "curb-weight" adalah int64, "normalized-losses" adalah objek, dan "wheel-base" adalah float64, dll.

Describe

Jika kita ingin mendapatkan ringkasan statistik dari setiap kolom, seperti hitungan, nilai rata-rata kolom, standar deviasi kolom, dll. Kami menggunakan metode describe:

```
dataframe.describe()
```

Metode ini akan memberikan berbagai statistik ringkasan, tidak termasuk nilai NaN (Not a Number).

```
[ ]: df.describe()
```

Ini menunjukkan ringkasan statistik dari semua kolom bertipe numerik (int, float). Misalnya, atribut "symboling" memiliki 205 hitungan, nilai mean kolom ini adalah 0,83, standard deviation adalah 1,25, nilai minimumnya adalah -2, persentil ke-25 adalah 0, persentil ke-50 adalah 1, persentil ke-75 adalah 2, dan nilai maksimum adalah 3. Namun, bagaimana jika kita juga ingin memeriksa semua kolom termasuk yang bertipe objek. Anda dapat menambahkan argumen include = "all" di dalam

kurung. Mari kita coba lagi.

```
[ ]: # describe all the columns in "df"
     df.describe(include = "all")
```

Sekarang, ini memberikan ringkasan statistik dari semua kolom, termasuk atribut yang diketik objek. Kita sekarang dapat melihat berapa banyak nilai unik, yang merupakan nilai teratas dan frekuensi nilai teratas di kolom yang diketik objek.

Beberapa nilai pada tabel di atas ditampilkan sebagai "NaN", hal ini karena angka tersebut tidak tersedia untuk jenis kolom tertentu.

Soal 3 :

Anda dapat memilih kolom data frame dengan menunjukkan nama setiap kolom, misalnya, Anda dapat memilih tiga kolom sebagai berikut:

dataframe[['column 1', 'column 2', 'column 3']]

Dimana "column" adalah nama kolom, Anda dapat menerapkan metode ".describe()" untuk mendapatkan statistik kolom tersebut sebagai berikut:

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

Terapkan metode ".describe()" ke kolom 'length' dan 'compression-ratio'.

4. Forum Diskusi
   a. Dengan tim kelompok diskusi anda, selesaikanlah semua latihan yang terdapat pada Bab 2, lalu jalankan program tersebut
   b. Buatlah laporan dari setiap program yang anda kerjakan berserta hasilnya

C. PENUTUP

1. Rangkuman

   Sebagai penutup, pada topik 2 ini, kita mempelajari mengenai praktek Teknik Pengumpulan Data yang dimulai dengan mempelajari environment dan package Anaconda, dan kita mempelajari dua package dasar, yaitu package NumPy dan package Pandas. Dua packages ini merupakan packages inti dalam data science, karena memang package ini mempunyai fungsi dasar untuk manipulasi data.

2. Tes Formatif

Sebagai evaluasi pemahaman materi pada Bab 2, jawablah pertanyaan berikut ini.

a. Sebutkan dan jelaskan proses proses Managing Environments

b. Berikut ini yang merupakan proses Managing Packages adalah

1. Proses Search package untuk melakukan pencarian package
2. Proses List package untuk melihat daftar package;
3. Proses List package untuk melihat daftar package;
4. Proses List package untuk melihat daftar package;

c. Berikut ini yang merupakan perintah untuk instalasi package NumPy pada Anaconda Prompt dan diakhiri dengan menjalankan Jupyter Notebook adalah:

1. Conda activate dsf
2. Conda install numpy
3. Jupyter notebook
4. Conda notebook jupiter

d. Berikut ini yang merupakan kemampuan khusus Package NumPy adalah:

1. kemampuan untuk mengelola N-dimensi objek array
2. fungsi-fungsi pendukung yang lengkap dalam mengolah data numerik
3. fungsi-fungsi pendukung operasi aljabar aritmatika yang sudah di implementasikan
4. integrase dengan Bahasa pemrograman C/C++ dan juga fortran

e. Apa yang dimaksud dengan slicing dan filtering?

DAFTAR PUSTAKA

Group by: split-apply-combine, Pandas, https://pandas.pydata.org/pandas-docs/stable/user_guide/groupby.html

Indexing and selecting data, Pandas, https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

Numpy, https://numpy.org/doc/stable/reference/arrays.indexing.html

Managing environments, Anaconda.Documentation, https://docs.anaconda.com/anaconda/navigator/tutorials/manage-environments/

Python for Excel : A Modern Environment for Automation and Data Analysis, Published by Felix Zumstein, 2021.

BAB III
DATA PREPARATION AND ANALYSIS

A. PENDAHULUAN

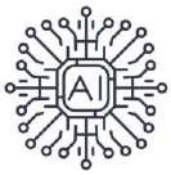
Materi ini menjelaskan tentang bagaimana mempersiapkan dataset dan kemudian melakukan analisa terhadap dataset tersebut untuk di training dan testing untuk mencari hasil akurasi yang baik.

B. INTI

1. Capaian Pembelajaran
    a. Mampu membuat dan menerapkan aplikasi yang dapat membantu dalam melakukan training dan testing terhadap dataset guna menghasilkan hasil akurasi yang baik dalam mendukung pengambila keputusan
    b. Mampu memahami dan menjelaskan apa itu Machine Learning
    c. Mampu menyelesaikan permasalahan regresi
    d. Mampu mengilustrasikan model dan mengimplementasilan model tersebut kedalam masalah sebagai alternate solusi
    e. Mampu memahami dan menjelaskan apa itu Bias dan mengimplementasikan bias tersebut
    f. Mampu memahami dan menjelaskan apa itu Gradien Descent dan mengimplementasikan Gradisen Descent tersebut.
2. Materi
    a. Machine Learning
    b. Permasalahan Regresi
    c. Ilustrasi Model
    d. Implementasi Model
    e. Bias
    f. Implementasi Bias
    g. Gradien Descent
    h. Implementasi Gradien Descent

3. Uraian Materi

a. Machine Learning



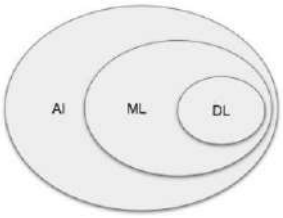
Pada beberapa topik sebelumnya, kita mempelajari dasar - dasar pemrograman bahasa Python dan penggunaan module Numpy, dan mulai topik ini kita akan memasuki pembahasan mengenai model machine learning.

Artificial Intelligence

Artificial Intelligence adalah sebuah ilmu yang memakai kemampuan komputer untuk meniru kemampuan berpikir manusia dalam pengambilan keputusan, pengolahan teks, dan persepsi visual. AI adalah satu bagian dari ilmu komputer yang mempunyai banyak subbidang, seperti Machine Learning, Robotics, Computer Vision, dan lain sebagainya.

Artificial Intelligence atau Kecerdasan Buatan dalam bahasa Indonesia, telah menjadi subjek yang banyak diperbincangkan, terutama pada mulai berkembangnya tagline istilah Industry 4.0 yang memasukan unsur AI kedalam dunia bisnis pada generasi keempat ini.

Istilah AI, Machine Learning (ML) dan Deep Learning (DL) telah banyak muncul di berbagai artikel - artikel dunia, mulai dari aplikasi chatbot yang bisa berkomunikasi dengan manusia, mobil yang bisa berjalan sendiri, dan kedepannya akan mengakibatkan pekerjaan manusia mulai diambil alih satu persatu, dan sistem perekonomian akan ditangani oleh robot atau agen–agen AI



Pada gambar diatas, dapat dilihat, bahwa AI menduduki bagian terbesar dari seluruh konsep kecerdasan buatan, dan diperkecil dengan subset ML, dan lebih diperkecil lagi untuk DL.

AI sendiri terbagi menjadi dua bidang besar, yaitu:

1. Logic Programming
2. Machine Learning

AI Logic Programming

AI Logic Programming adalah ilmu AI klasik yang mendefinisikan AI dengan menggunakan pendekatan pemrograman logik. Pendekatan pemrograman yang dimaksud disini adalah pendekatan dengan menggunakan semua kemungkinan (possibility) yang ada (if rules). Kita membuat program komputer untuk mencari instruksi apa yang harus dikerjakan, dan instruksi tersebut berasal dari kita sebagai programmer. Kita harus mendefisinikan setiap kondisi dan juga setiap aksi yang harus dilakukan.

AI Logic Programming paling umum yang kita temui, umumnya ada dalam sebuah permainan komputer. Sewaktu kita bermain, dan bertemu dengan musuh, untuk pertama kalinya, akan terasa sulit sekali menghadapi musuh tersebut, dan terasa bahwa musuh tersebut seperti mempunyai pikiran sendiri. Akan tetapi setelah beberapa kali permainan diulang, akan terasa bahwa kita sudah menghafal pola musuh tersebut, dan kita bisa mengakali pola tersebut untuk mencapai kemenangan.

Pada contoh kasus diatas, musuh utama tersebut mempunyai himpunan pola - pola yang sudah didefinisikan sebelumnya, dan musuh tersebut hanya menjalankan dan mengikuti pola tersebut, dan inilah yang disebut dengan AI Programming dengan menggunakan algoritma Rule-based.

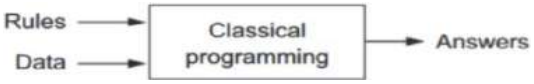
Dan disini kita mengenal istilah bahwa **program komputer tidak pernah salah**, komputer adalah pekerja keras yang bisa bekerja tanpa berhenti selama 24 jam dibandingkan dengan manusia, karena komputer hanya melakukan himpunan instruksi-instruksi yang sudah disiapkan sebelumnya oleh manusia yang memprogramnya.

AI Machine Learning

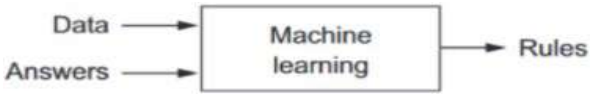
AI Machine Learning adalah subbidang AI yang memungkinkan komputer untuk melakukan proses - proses pembelajaran apabila diberikan tugas dan pengalaman mengenai cara mengerjakan tugas tersebut. Semua teknik ML, pasti diklasifikasikan sebagai bagian dari AI, tapi tidak semua bagian dari AI, bisa disebut ML, seperti

contoh teknik algoritma Rule-based, bisa dikategorikan sebagai AI, tapi tidak bisa dikategorikan sebagai ML, karena tidak ada proses pembelajaran pada teknik tersebut.

Pada pemrograman klasik, manusia memasukan aturan-aturan dalam bentuk program dan data akan diproses berdasarkan aturan tersebut, dan keluar sebuah hasil atau jawaban.



Dan sebaliknya untuk ML, manusia akan memasukan data dan jawabannya, dan dihasilkan aturan-aturan, dan aturan yang baru dihasilkan tersebut bisa digunakan untuk menghasilkan jawaban baru pada data yang baru.



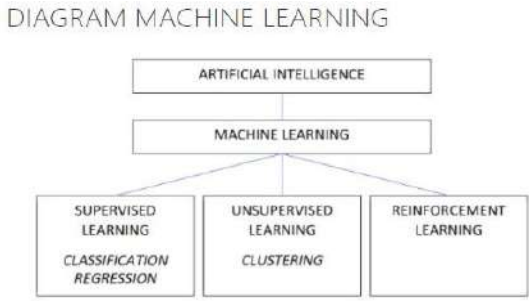
Sebuah ML bukan dihasilkan melalui program, tetapi dihasilkan melalui proses pembelajaran/Training. Sebagai contoh, apabila kita ingin membuat sebuah sistem yang bisa secara otomatis menandai gambar-gambar rekreasi kita, maka kita perlu membuat terlebih dahulu ML tersebut berdasarkan contoh-contoh gambar yang sudah ditandai sebelumnya, sehingga ML tersebut akan belajar dan melakukan proses-proses pengingatan berdasarkan data yang disediakan untuk membuat aturan-aturan baru.

Contoh sederhana yang bisa menjelaskan ML dengan lebih sederhana, adalah contoh seorang bayi yang baru lahir. Bayi yang baru lahir pastinya tidak akan bisa membedakan jenis kelamin bukan? Pada umur dua sampai tiga tahun, mulai bisa mengenal Bapak dan Ibunya, dan seiring berjalannya waktu menuju kedewasaan dengan banyak melihat dan berpikir, kita bisa membedakan laki-laki dan perempuan, cukup dengan hanya melihat wajah manusia, seperti fitur rambutnya panjang, mata lebih lentik, dan sebagainya. Akan tetapi disini terkadang kita juga bisa melakukan kesalahan, karena seorang manusia yang rambutnya panjang, bukan berarti berjenis kelamin perempuan, dan juga contoh lainnya, apabila kita ke negara Thailand, kita akan susah membedakan mana laki-laki dan mana perempuan.

Dan contoh diatas adalah contoh AI Machine Learning, ada data, ada pembelajaran, ada pengambilan keputusan, tetapi terkadang keputusan tersebut bisa salah

dikarenakan data yang digunakan untuk belajar juga salah, maka AI Machine Learning ini juga boleh disebut **program komputer yang bisa salah**.

Diagram Machine Learning



ML sendiri mempunyai tiga subbagian, yaitu:

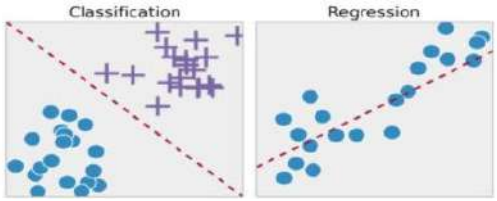
1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Supervised Learning

Supervised Learning adalah proses pembelajaran dari ML yang menggunakan **data** yang mempunyai **label**, sebagai contoh: apabila diberikan 50 gambar pasfoto laki - laki, dan diberi label L, dan 50 gambar pasfoto perempuan, dan diberi label P, dan 100 gambar ini, akan dilihat oleh komputer, dan dilakukan proses pembelajaran untuk mengingat gambar tersebut. Harapannya, apabila diberikan gambar ke 101, komputer bisa secara otomatis memprediksi dan memberikan label L atau P, sesuai dengan memori atau ingatan yang sudah dilakukan pada proses pembelajaran mengingat 100 gambar awal.

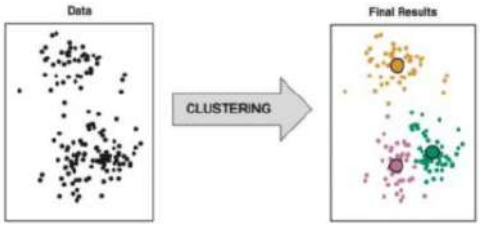
Contoh kasus diatas, adalah proses pembelajaran Supervised Learning, dimana untuk belajar, komputer akan dihadapkan pada sekelompok data yang mempunyai label. Proses pembelajaran pada Supervised Learning umumnya jatuh kepada dua bagian besar, yaitu:

1. **Classification**, proses untuk mengklasifikasi sebuah data input;
2. **Regression**, proses untuk menentukan nilai sebuah data input.



Unsupervised Learning

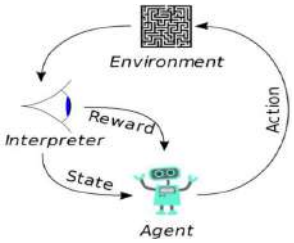
Unsupervised Learning adalah proses pembelajaran tanpa adanya label, dan bertujuan untuk membentuk sendiri label - label berdasarkan pola - pola data input yang diberikan. Sering juga disebut dengan istilah Clustering, atau klasterisasi, yaitu proses pembentukan klaster data.



Reinforcement Learning

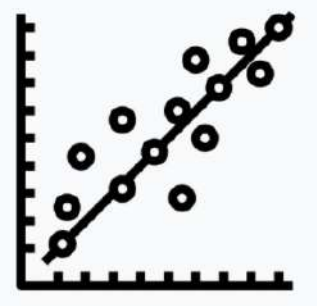
Reinforcement Learning(RL) adalah bagian ML yang berkaitan dengan bagaimana komputer, yang dalam hal ini, diistilahkan dengan istilah agen, harus mengambil tindakan atau keputusan dalam sebuah lingkungan dimana agen tersebut berada, dan tindakan atau keputusan tersebut, akan menghasilkan nilai, yang bisa berupa nilai positif (penghargaan) atau negatif (sanksi), dan agen tersebut, akan berusaha sendiri mencari jalan di lingkungan tersebut, yang bisa memaksimalkan nilai positif yang bisa diterima.

Proses RL, boleh diibaratkan sama seperti manusia yang belajar memainkan game komputer Role Playing Game (RPG), dimana pemain tersebut, akan belajar memainkan game tersebut di lingkungan game itu dengan tujuan meningkatkan kemampuannya.



Dan untuk subtopik selanjutnya, kita akan fokus membahas masalah Supervised Learning, yaitu masalah Regression dengan menggunakan contoh kasus yang sederhana.

b. Permasalahan Regresi



Simple Linear Regression

Pada topik sebelumnya kita mempelajari teori-teori dasar dari Artificial Intelligence sampai dengan Machine Learning dengan tujuan untuk memprediksi masalah-masalah yang berhubungan dengan Regression dan Classification. Machine Learning sendiri mempunyai banyak arsitektur model yang bisa digunakan untuk memecahkan masalah tersebut, seperti K Nearest Neighbour, Support Vector Machine, dan lain sebagainya.

Deep learning sendiri adalah bagian dari Machine Learning, dan hanyalah sebuah istilah yang mendefinisikan kumpulan arsitektur yang menggunakan arsitektur model berbentuk neural network yang meniru bentuk jaringan syaraf manusia, dan akan sangat susah memahami neural network secara langsung, karena neural network sendiri adalah kumpulan dari beberapa model dalam statistics, seperti linear regression, sigmoid, dan sebagainya.

Kenapa judul dalam topik ini mengambil judul Simple Linear Regression? Karena memang dasar dari neural network dimulai dengan model ini, dan ini adalah model yang sangat sederhana dan siapapun harusnya sudah mempelajari model ini pada bangku sekolah menengah.

Studi Kasus Bahan Baku Pada Restoran

Untuk memulai pembelajaran deep learning ini, akan disajikan terlebih dahulu sebuah permasalahan pada sebuah restoran. Pemilik restoran ini sebelum memulai usahanya setiap hari, selalu mengecek terlebih dahulu jumlah meja yang sudah direservasi, dan juga jumlah bahan baku yang tersedia untuk hari tersebut. Terlalu banyak bahan yang disediakan bisa menimbulkan kerugian kalau tidak terpakai dan menjadi basi, terlalu sedikit bahan yang disediakan tidak bisa memenuhi permintaan pesanan sehingga menimbulkan kerugian juga. Selain meja yang direservasi, terkadang banyak juga pelanggan yang datang sendiri tanpa reservasi. *Sekarang pertanyaannya adalah, dengan berdasarkan jumlah data reservasi, bisakah kita membuat sebuah program yang bisa memprediksi jumlah pelanggan berdasarkan data tersebut? Bila jumlah pelanggan bisa diprediksi sebelumnya, maka persediaan bahan juga bisa ditetapkan dengan baik sehingga tidak menimbulkan kerugian.*

Jawaban dari pertanyaan diatas dapat dijawab dengan sebuah model statistik yang diberi nama Simple Linear Regression, dan kasus diatas adalah kasus dari Supervised Learning masalah regresi, yaitu masalah yang bisa memprediksi sebuah nilai dari data yang sudah ada sebelumnya.

Untuk mempersingkat, segala sesuatu selalu dimulai dengan data, mari kita minta pada pemilik restoran tersebut untuk mencatat data reservasi beserta bahan baku yang disiapkan per harinya, dan data tersebut diakumulasi selama 30 hari dan disimpan dalam sebuah file text, dapatkan file text tersebut di subtopik Data Reservasi 1, dan simpan dengan format *data-reservasi-1.txt* pada folder yang sama dengan file jupyter notebook yang akan kita buat nantinya.

Tabel 19. Reservasi

Reservasi	Bahan
9	27
20	39
27	42
16	18
14	24
15	18
13	22
17	31

Gambar diatas adalah sampel data dari file tersebut, pada hari pertama, ada 9 reservasi, dan ada 27 bahan baku yang diperlukan untuk melayani 27 meja tersebut,

pada hari kedua, terdapat 20 reservasi dan ada 19 yang datang tanpa reservasi, dan seterusnya.

Dan cukup berteori ria, mari kita mulai projek sederhana kita dalam menyelesaikan masalah diatas, silakan buka sebuah jupyter notebook masing - masing, dan mari kita mulai dengan perintah sederhana untuk melihat isi file tersebut dan simpan dalam bentuk object array Numpy.

```
In [ ]:
1 import numpy as np

In [ ]:
1 X, Y = np.loadtxt(
2     "data-reservasi-1.txt",
3     skiprows= 1,
4     unpack= True
5 )
```

Baris pertama, perintah untuk import module numpy dan buat sebagai alias np, bentuk import dan alias ini sudah sangat umum dipakai oleh programmer Python, dan dilanjutkan dengan perintah kedua dengan fungsi loadtxt dari numpy, yang berfungsi untuk melihat isi sebuah file dengan ekstensi .txt dan disimpan dalam objek ***X*** *untuk data reservasi dan* ***Y*** *untuk data bahan*.

Kenapa fungsi loadtxt bisa langsung tahu bahwa ada dua kolom data di file tersebut? Karena penggunaan parameter *unpack= True*, yang artinya menyuruh fungsi ini untuk melihat spasi sebagai pemisah kolom, dan karena data kita ada dua kolom, maka secara otomatis output dari fungsi ini adalah dua object array numpy. Dan pada baris pertama data kita adalah merupakan judul dari kolom, dan seharusnya judul ini tidak perlu kita ambil, oleh karena itu dipakai parameter *skiprows= 1*, yang artinya kita mengabaikan data pada baris pertama.

Dan kenapa data tersebut kita simpan pada variabel X dan Y? bukan reservasi dan bahan? karena untuk mempermudah proses pembentukan grafiknya nanti, karena kita akan beranggapan bahwa data reservasi adalah data sumbu X dan data bahan adalah data sumbu Y. Dan ini juga merupakan aturan umum dalam dunia machine learning, dimana X bisa disebut juga dengan independent variable atau independent feature, yang berisikan data - data yang akan dipakai untuk membuat model machine learning, dan Y adalah dependent variable atau dependent feature yang merupakan target prediksi kita.

Apabila kita berhasil menjalankan dua perintah diatas tanpa ada kesalahan, maka bisa kita lanjutkan dengan mencoba melihat isi data tersebut dengan fungsi slicing

pada numpy.

```
In [3]: 1 X[0:8]
Out[3]: array([ 9., 20., 27., 16., 14., 15., 13., 17.])

In [4]: 1 Y[0:8]
Out[4]: array([27., 39., 42., 18., 24., 18., 22., 31.])
```

Dengan menampilkan data diatas menggunakan slicing 0:8, dapat dilihat pada gambar diatas, bahwa data yang ditampilkan sama dengan data pada gambar sampel diatas. Dan selain sampel data, kita juga harus bisa memastikan berapa jumlah data keseluruhan yang ada? Bisa dicoba dengan menggunakan fungsi shape pada numpy.

```
In [5]: 1 X.shape
Out[5]: (30,)

In [6]: 1 Y.shape
Out[6]: (30,)
```

Nampak bahwa data yang disimpan dalam objek X dan Y merupakan data 1 dimensi dengan total 30 data. Dan untuk benar - benar memastikan bahwa objek X dan Y adalah memang benar objek array numpy, bisa kita pastikan dengan menggunakan fungsi type.

```
In [7]: 1 type(X)
Out[7]: numpy.ndarray

In [8]: 1 type(Y)
Out[8]: numpy.ndarray
```

Dan memang benar, fungsi loadtxt benar - benar mengembalikan objek dengan tipe *numpy.ndarray*, dimana ndarray adalah singkatan dari *n dimension array*.

Plotting Grafik

Mata manusia pada dasarnya memang susah untuk melihat data dalam bentuk tabular seperti diatas, oleh karena itu mari kita sajikan data diatas dalam bentuk yang mudah dilihat, yaitu bentuk grafik, dan bagi Python hal ini sangat mudah karena sudah ada module-module khusus yang bisa mengubah data menjadi grafik dalam beberapa perintah sederhana. Sebelum memulai grafik, kita memerlukan

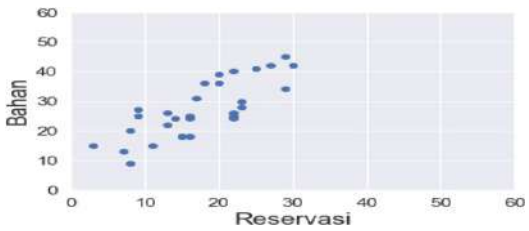
module untuk membuat grafik dengan nama matplotlib dan seaborn, jadi silakan install terlebih dahulu dua module tersebut dalam anaconda prompt.

```
In [ ]:
1 import matplotlib.pyplot as plt
2 import seaborn as sns
```

Apabila sudah berhasil instalasi module matplotlib dan seaborn, maka pemanggilan import terhadap module tersebut dapat dilakukan dengan dua perintah diatas. Dan untuk selanjutnya, perintah berikut adalah perintah untuk menampilkan grafik data data reservasi kita.

```
In [ ]:
1 sns.set()
2 plt.axis([0, 60, 0, 60])
3 plt.xticks(fontsize= 15)
4 plt.yticks(fontsize= 15)
5 plt.xlabel("Reservasi", fontsize= 20)
6 plt.ylabel("Bahan", fontsize= 20)
7 plt.plot(X, Y, "bo")
8 plt.show()
```

Fungsi-fungsi pada perintah diatas sangat sederhana, fungsi axis mengatur ukuran grafik kita, fungsi xticks dan yticks mengatur ukuran font, fungxi xlabel dan ylabel mengatur tampilan tulisan batangnya, dan fungsi utamanya adalah fungsi plot dengan 3 parameter, dimana parameter pertama dan kedua diisikan data - data kita, objek X dan Y, serta parameter ketiga bo, yang mempunyai arti sajikan grafik dengan warna b= blue, dan bentuk o bulat.



Seluruh data selama 30 hari tersebut dapat terlihat pada grafik diatas, setiap satu titik biru menggambarkan data 1 hari dan total ada 30 titik biru, Kesan yang kita dapat dengan melihat grafik diatas, data tersebut hampir berbentuk garis diagonal, dan dengan melihat bentuk datanya, maka solusi sederhana yang bisa kita lakukan untuk melakukan proses prediksi, adalah dengan menggunakan Simple Linear Regression yang akan kita bahas pada subtopik selanjutnya.

c. Ilustrasi Model

Linear Regression adalah sebuah model arsitektur dalam ilmu statistika yang berfungsi untuk membuat prediksi dengan menggunakan sebuah garis linear, dan

fungsi garis linear yang paling sederhana dapat digambarkan dengan fungsi berikut:

```
In [ ]:  1  y = x * w
```

Y dan X pada fungsi diatas sama dengan X dan Y pada objek array numpy yang kita buat sebelumnya, sehingga hanya objek w saja yang belum ada, dan dapat kita anggap objek w adalah objek penentu dari fungsi tersebut, apabila kita tahu nilai w, dengan data X, kita dapat memprediksi nilai Y.

Dengan melihat contoh grafik sebelumnya, dan kita mengambil asumsi:

```
In [ ]:  1  w = 1.5
```

Dengan adanya objek w yang bernilai 1.5, maka kita dapat memprediksi nilai y dari dua nilai berikut:

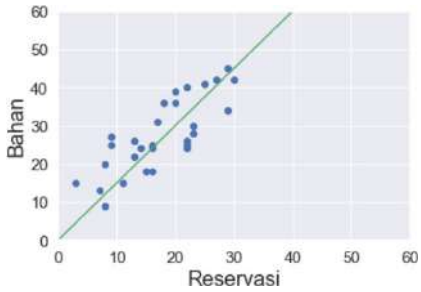
```
In [ ]:  1  x1 = 0
         2  x2 = 40
         3  y1 = x1 * w
         4  y2 = x2 * w
```

Dengan asumsi x1 = 0, dan x2 = 40, maka dapat dihitung y1 dan y2, sehingga apabila diplot grafiknya dengan perintah berikut:

Semua perintah yang ada masih sama dengan perintah pada grafik sebelumnya, dengan penambahan fungsi plot baru yang khusus untuk menggambar garis dengan warna hijau:

```
8  plt.plot([x1,x2], [y1,y2], "g")
```

Dan didapat hasil grafik sebagai berikut:

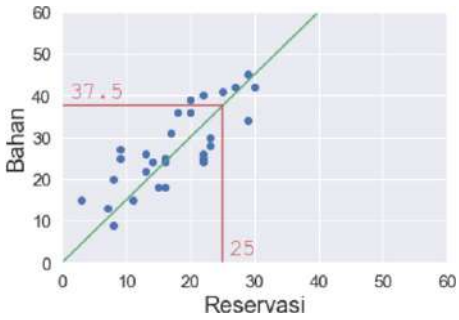


Dari gambar diatas, kita mendapatkan sebuah garis dengan asumsi w, sehingga proses prediksi bisa dilakukan, apabila ada reservasi sebanyak 25 meja, kira - kira berapa bahan yang diperlukan untuk persiapan pada hari tersebut?

```
In [ ]:  1  x3 = 25
         2  y3 = x3 * w
```

Maka dengan menggunakan data x3 dan y3 diatas, kita gambarkan grafik dengan perintah berikut:

In [ ]:
```
sns.set()
plt.axis([0, 60, 0, 60])
plt.xticks(fontsize= 15)
plt.yticks(fontsize= 15)
plt.xlabel("Reservasi", fontsize= 20)
plt.ylabel("Bahan", fontsize= 20)
plt.plot(X, Y, "bo")
plt.plot([x1,x2], [y1,y2], "g")
plt.plot([x3,0], [y3,y3], "r")
plt.plot([x3,x3], [0,y3], "r")
plt.text(26, 2, x3, fontsize= 20, color= "r", family= "Courier New")
plt.text(1, 39.5, y3, fontsize= 20, color= "r", family= "Courier New")
plt.show()
```

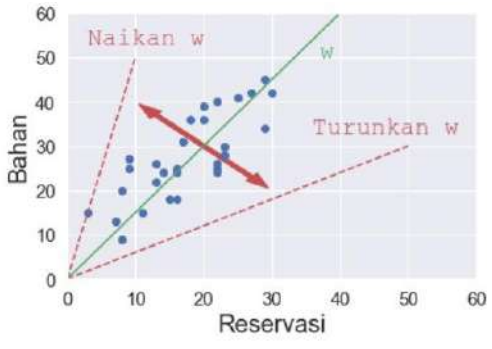


Dan output sebagai berikut:

Dengan melihat gambar diatas, didapat prediksi bahwa apabila ada reservasi 25 meja, maka kita perlu menyiapkan 37.5 bahan baku atau digenapkan menjadi angka 38 meja.

Kesimpulannya, tugas kita yang paling penting saat ini adalah mencari nilai w, apakah nilai w = 1.5 diatas sudah mencukupi untuk masalah kita? Dan bagaimana cara mencari nilai 1.5 itu? Karena nilai w hanya sebuah nilai constan biasa, dan grafik berikut menggambarkan pergerakan nilai w. Dan untuk pemecahan masalah kita, akan digunakan algoritma optimasi untuk menghitung nilai Loss / Kerugian.

```
In [18]:
sns.set()
plt.axis([0, 60, 0, 60])
plt.xticks(fontsize= 15)
plt.yticks(fontsize= 15)
plt.xlabel("Reservasi", fontsize= 20)
plt.ylabel("Bahan", fontsize= 20)
plt.plot(X, Y, "bo")
plt.plot([x1,x2], [y1,y2], "g")
plt.plot([0,50], [0,30], "r--")
plt.plot([0,10], [0,50], "r--")
plt.text(37, 50, "w", fontsize= 20, color= "g", family= "Courier New")
plt.text(36, 33, "Turunkan w", fontsize= 20, color= "r", family= "Courier New")
plt.text(3, 53, "Naikan w", fontsize= 20, color= "r", family= "Courier New")
plt.arrow(20, 30, 7, -7, width = 0.8, color= "r")
plt.arrow(20, 30, -7, 7, width = 0.8, color= "r")
plt.show()
```



d. Implementasi Model

Pada subtopik ini, kita akan mulai bangun sebuah arsitektur linear regression, dengan menggunakan pendekatan optimasi terhadap nilai target dan prediksi untuk mendapatkan. Sebelum lebih lanjut, kita harus mempelajari dulu apa yang disebut Loss function.

Proses pembentukan arsitektur linear regression ini terdiri dari dua tahapan, yaitu:

1. Tahapan Prediction/Prediksi
2. Tahapan Training/Pelatihan

Tahapan Prediction sangat sederhana, karena pada tahapan ini hanya perlu melakukan proses perhitungan matematika, yaitu menghitung rumus y = x * w. Sedangkan tahapan Training yang lebih kompleks, disinilah proses optimasi dilakukan dengan menggunakan Loss function.

Loss function adalah sebuah fungsi yang bertujuan untuk menghitung selisih antara target dan prediksi dari dependent variable.

Seperti pada contoh kasus kita, loss function disini berfungsi untuk menghitung selisih antara bahan baku target dengan bahan baku prediksi, nilai loss yang kecil mempunyai arti selisih antara target dan prediksi kecil sehingga arsitektur yang kita

buat bisa dianggap bagus dan bisa dipercaya untuk digunakan, apabila nilai loss besar, maka selisih target dan prediksi juga besar sehingga prediksi yang dilakukan akan dianggap tidak valid juga.

Salah satu loss function yang bisa dipakai untuk kasus kita adalah Mean Squared Error (MSE).

Mean Squared Error

Perhitungan MSE sangat sederhana dan dillihat pada contoh berikut, misalkan diketahui pasangan prediksi dan target sebagai berikut:

prediksi, target

3, 2

15, 17

8, 5

Langkah pertama yang harus dilakukan adalah membuat pengurangan antara prediksi dengan target, dan kita beri nama selisih.

prediksi, target, selisih

3, 2, 1

15, 17, -2

8, 5, 3

Langkah kedua, kalau kita lihat contoh diatas, hasil selisih mengandung minus, dengan adanya minus ini akan susah dilakukan penghitungan selisih yang benar, oleh karena itu hasil dari selisih ini akan dikuadratkan, dan diberi nama kuadrat.

prediksi, target, selisih, kuadrat

3, 2, 1, 1

15, 17, -2, 4

8, 5, 3, 9

Langkah ketiga, untuk menghitung MSE dari kasus diatas, cukup dengan menjumlahkan seluruh hasil kuadrat dan dibagi dengan banyaknya data yang tersedia, dan dalam kasus ini, data tersedia adalah 3 sampel data.

$$MSE = (1 + 4 + 9) / 3 = 4.67$$

Angka 4.67 diatas mungkin tidak mempunyai arti sama sekali, tapi apabila disandingkan dengan contoh kedua berikut, yaitu 3 sampel data, tapi dengan data

prediksi yang mirip dengan target, akan didapatkan nilai MSE yang lebih kecil.

prediksi, target, selisih, kuadrat

4, 2, 2, 4

17, 17, 0, 0

4, 5, -1, 1

MSE = (4 + 0 + 1) / 3 = 1.67

Pada contoh kedua sampel data diatas, dapat dilihat bahwa prediksi hampir mendekati target, dan nilai MSE = 1.67, lebih kecil daripada contoh pada sampel pertama.

Bagaimana kalau seandainya prediksi sama dengan target? Jawabannya adalah nilai MSE = 0, dan dapat dilihat pada contoh sampel ketiga dibawah

prediksi, target, selisih, kuadrat

2, 2, 0, 0

17, 17, 0, 0

5, 5, 0, 0

MSE = (0 + 0 + 0) / 3 = 0

Selain loss function, proses training disini juga melibatkan iterasi untuk mencari w yang mendekati 0, makin dekat dengan nilai 0, maka makin baik arsitektur yang kita hasilkan. Dan untuk rangkumannya, algoritma MSE diatas dapat dirangkum dalam sebuah model matematika sebagai berikut:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(target_i - target_i)^2$$

Dan cukup dengan teori, mari kita langsung masuk ke pemrograman python, dan untuk mewakili setiap fungsi, maka akan dibuatkan terlebih dahulu tiga fungsi utama, yaitu fungsi:

1. Predict
2. Loss
3. Train

Fungsi Predict

Fungsi predict adalah sebuah fungsi untuk memprediksi nilai berdasarkan parameter X dan w yang ada, dengan penulisan sebagai berikut:

```
In [ ]:
1 import numpy as np
```

```
In [ ]:
1 def predict(X, w):
2     return X * w
```

Baris pertama, kita wajib mengimport terlebih dahulu module numpy, dan berdasarkan fungsi prediksi kita yang sudah kita pelajari pada subtopik sebelumnya, yaitu **y = x * w**, serta kemampuan sistem broadcasting dari numpy, maka fungsi predict dapat dituliskan dengan sederhana seperti pada contoh diatas, dimana **X** diharapkan adalah data array numpy 1 dimensi, sedangkan **w** adalah sebuah nilai constant.

Fungsi Loss

Fungsi loss juga sama sederhananya seperti fungsi train, cukup memasukan model matematika MSE dan kemampuan broadcasting array, fungsi ini bisa dituliskan sebagai berikut:

```
In [ ]:
1 def loss(X, Y, w):
2     return np.average((predict(X, w) - Y) ** 2)
```

Fungsi loss dituliskan dengan menggunakan tiga parameter, yaitu X, Y, dan w, dimana X dan Y adalah objek array numpy dimensi 1, dan w adalah constant, sesuai dengan rumus MSE, diambil fungsi predict dikurangkan dengan target Y dan dikuadratkan, dan hasil keseluruhan digunakan fungsi rata - rata dari numpy, yaitu np.average untuk menghitung rata - rata dari nilai MSE.

Fungsi Train

Fungsi train adalah fungsi utama dari arsitektur kita, dimana fungsi inilah menjadi fungsi yang membangunkan nilai w dengan loss yang bagus berdasarkan iterasi yang berulang - ulang, sebagai berikut:

```
In [ ]:
1  def train(X, Y, iterations, lr):
2      w = 0
3      for i in range(iterations):
4          current_loss = loss(X, Y, w)
5          print("Iterasi %4d -> Loss: %.6f" % (i, current_loss))
6
7          if loss(X, Y, w + lr) < current_loss:
8              w += lr
9          elif loss(X, Y, w - lr) < current_loss:
10             w -= lr
11         else:
12             return w
13     raise Exception("Terjadi kesalahan pada iterasi ke %d" % iterations)
```

Parameter yang diperlukan dalam fungsi ini ada 4, yaitu parameter X sebagai data, dan Y sebagai target, serta jumlah iterations, dan lr. Apa itu lr? merupakan singkatan dari Learning Rate.

Learning rate (lr) adalah sebuah parameter yang umum digunakan dalam arsitektur model Machine Learning sebagai sebuah nilai untuk menyetel kemampuan model kita seberapa cepat dalam pembelajaran dan adaptasi terhadap loss function. Sampai sekarang memang tidak ada standarisasi apapun mengenai nilai lr yang bisa digunakan, tapi pada umumnya orang - orang statistik menggunakan model nilai desimal seperti 0.01, 0.001, 0.0001, dan juga kelipatannya

Berdasarkan penulisan fungsi train diatas dalam bahasa pemrograman python, dapat dilihat bahwa inisialisasi awal untuk objek w adalah 0, yang kemudian dilanjutkan dengan iterasi dari nilai i = 0 sampai dengan jumlah nilai iterasi yang kita masukan dalam objek iterations. Untuk setiap iterasi, kita hitung terlebih dahulu current_loss dengan menggunakan loss function.

current_loss = loss(X, Y, w)

Setelah itu, kita menggunakan perbandingan, apabila nilai w ditambah dengan lr apabila hasilnya lebih kecil dari current_loss, maka nilai w akan diperbaharui dengan ditambah nilai lr, dan juga sebaliknya, bila nilai loss dengan w dikurang lr lebih kecil dari current_loss, maka nilai w diperbaharui dengan menggunakan w dikurang dengan lr.

```
if loss(X, Y, w + lr) < current_loss:
w += lr
elif loss(X, Y, w - lr) < current_loss:
w -= lr
else:
return w
```

Bila dua pilihan diatas tidak tersedia, dan nilai loss lebih besar current_loss, maka dikembalikan nilai w, dan nilai w yang dihasilkan harusnya adalah nilai w yang paling optimal.

Pada baris akhir perintah, dilakukan proses Exception, yang mempunyai arti, ada kesalahan apapun, akan ditampilkan letak kesalahan tersebut pada iterasi ke berapa.

Tahapan Training

Dan dengan lengkapnya tiga fungsi diatas, maka kita bisa mulai membentuk arsitektur linear regression yang kita inginkan dengan pertama kali mengambil data

yang akan diproses:

```
In [ ]:
X, Y = np.loadtxt(
    "data-reservasi-1.txt",
    skiprows= 1,
    unpack= True
)
```

Lalu dilanjutkan dengan proses train dengan memanggil fungsi train yang akan dilakukan sebanyak 10000 iterasi dengan lr 0.01:

```
In [ ]:
w = train(
    X,
    Y,
    iterations= 10000,
    lr= 0.01
)
```

Hasil training diatas akan mencapai nilai w yang sempurna pada iterasi ke 150 dengan nilai loss = 42.66

```
Iterasi  141 -> Loss: 45.584143
Iterasi  142 -> Loss: 44.981640
Iterasi  143 -> Loss: 44.448490
Iterasi  144 -> Loss: 43.984693
Iterasi  145 -> Loss: 43.590250
Iterasi  146 -> Loss: 43.265160
Iterasi  147 -> Loss: 43.009423
Iterasi  148 -> Loss: 42.823040
Iterasi  149 -> Loss: 42.706010
Iterasi  150 -> Loss: 42.658333
```

Dan selanjutkan kita tampilkan nilai w optimal yang kita dapatkan melalui iterasi diatas dengan perintah berikut:

```
In [ ]:
print("w= %.3f" % w)
```

Dan nilai w optimal yang dicetak adalah:

```
In [7]:
print("w= %.3f" % w)

w= 1.500
```

Maka dengan berakhirnya proses training ini, kita sudah mendapatkan model arsitektur linear regression yang kita inginkan, yaitu:

$$y = x * w = x * 1.5$$

Tahapan Prediction

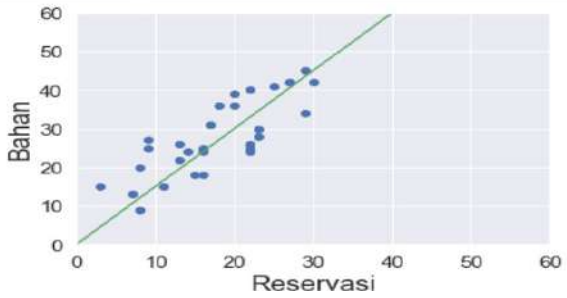
Dengan menggunakan objek w yang telah didapat pada training diatas, maka kita sudah bisa memprediksi berdasarkan data reservasi yang diinginkan, dengan asumsi apabila ada 24 meja yang direservasi, berapakah bahan baku yang perlu disiapkan? Jawabannya dapat langsung dilihat pada penulisan program berikut ini:

```
In [12]: 1 reservasi = 24

In [13]: 1 print("Prediksi: bila reservasi = %d, maka bahan baku yang perlu disiapkan = %.2f" %(reservasi, predict(reservasi, w)))

Prediksi: bila reservasi = 24, maka bahan baku yang perlu disiapkan = 36.00
```

Plotting Grafik

Dan sebagai penutup, kita akan mencoba menyajikan arsitektur yang kita dapatkan dalam bentuk grafik dengan perintah berikut:

```
In [15]:
1 import matplotlib.pyplot as plt
2 import seaborn as sns

In [16]:
1 x1 = 0
2 x2 = 50
3 y1 = x1 * w
4 y2 = x2 * w

In [17]:
1 sns.set()
2 plt.axis([0, 60, 0, 60])
3 plt.xticks(fontsize= 15)
4 plt.yticks(fontsize= 15)
5 plt.xlabel("Reservasi", fontsize= 20)
6 plt.ylabel("Bahan", fontsize= 20)
7 plt.plot(X, Y, "bo")
8 plt.plot([x1,x2], [y1,y2], "g")
9 plt.show()
```

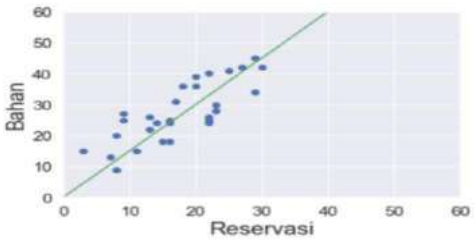


Perintah untuk menampilkan grafik diatas mirip seperti perintah pada subtopik sebelumnya, dimana kita mengambil asumsi garis yang kita gambar, adalah garis untuk titik X pada posisi 0 dan 50, dan prediksi Y yang dihasilkan masing - masing. Dan dengan ini selamat, kita telah berhasil menciptakan sebuah arsitektur linear regression yang sederhana dengan menggunakan kode yang sederhana menggunakan bahasa pemrograman python.

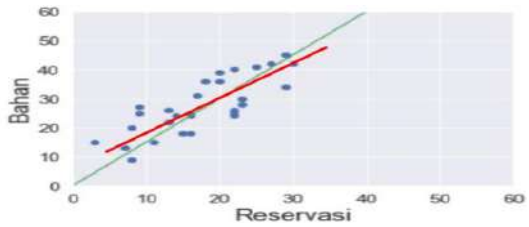
e. Bias

Apa itu bias? Bias hanyalah sebuah variabel tambahan, dan tujuan dari variabel bias ini adalah untuk menambahkan elevasi dari grafik linear regression kita. Pertanyaannya sekarang, apakah garis grafik pada kasus terakhir kita sudah

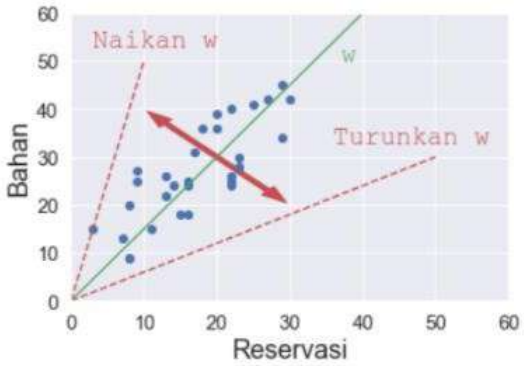
merupakan garis yang maksimal?



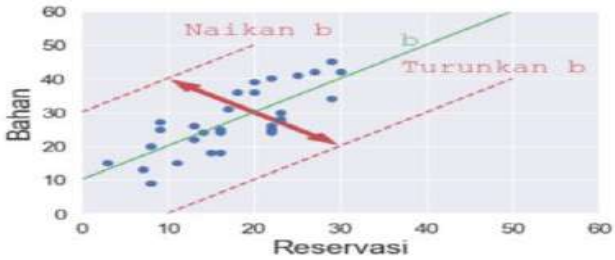
Bila dengan mata manusia, maka kalau digambarkan, bukankah gambar berikut lebih pas dengan pola garis tersebut?



Bila melihat gambar diatas, dapat terlihat, alangkah baiknya apabila garisnya bisa dinaikan sedikit agar lebih bisa mendekati pola dari data - data kita. Dikarenakan kita hanya menggunakan satu variabel w, maka garis kita akan tetap lengket pada titik nol untuk vertikalnya sesuai gambar berikut:



Maka satu-satunya cara untuk menambah elevasi dari garis kita, adalah dengan menambahkan variabel tambahan, dan umumnya variabel tambahan ini diberi nama variabel **bias**, dan bisa digambarkan sebagai berikut:



Jadi variabel w (weight) digunakan untuk memutar garis kita, apakah berputar searah jarum jam? atau berlawanan arah jarum jam? Sedangkan variabel b (bias) digunakan untuk menaikan elevasi dari garis kita, apakah naik ke atas? atau turun kebawah?

Dengan menggunakan dua variabel tersebut maka diharapkan kita bisa membuat sebuah garis linear regression yang memang benar - benar bisa pas dengan pola dari data kita.

Rumus Bias

Pada pembahasan sebelumnya, kita telah mempelajari rumus dasar dari Linear Regression, yaitu:

```
In [ ]: 1 y = x * w
```

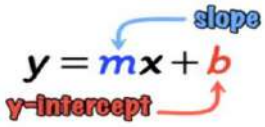
Dengan menambahkan bias, maka rumus kita akan berubah menjadi:

```
In [ ]: 1 y = x * w + b
```

Dan sepertinya rumus diatas tidak asing bukan? Rumus diatas adalah rumus dasar geometri yang pasti dipelajari oleh siapapun baik jurusan IPA maupun IPS, yaitu rumus Linear Equation/Persamaan Linier, dan memang yang kita lakukan sekarang, topik Linier Regression sebenarnya memang merupakan rumus tersebut, dan biasanya rumus tersebut sering dituliskan dalam bentuk:

```
In [ ]: 1 y = m * x + b
```

Perbedaan hanya terletak pada penamaan saja, biasanya m disebut dengan lereng, dan b disebut dengan pemotong sumbu y, akan tetapi dalam topik machine learning, m diganti menjadi weight (w) dan b menjadi bias (b). Bagi yang ingin belajar lebih lanjut mengenai rumus tersebut, dapat dilihat pada laman Slope-Intercept Form of a Line (y = mx + by=mx+b).



f. Implementasi Bias

Setelah mengetahui fungsi bias, maka langkah selanjutnya adalah implementasi, dan tetap menggunakan data pada kasus restoran kita, mari kita mulai implementasinya dengan menggunakan kode sumber sebelumnya, dan diperbaharui bagian yang berhubungan dengan penambahan bias.

Fungsi predict

Fokus utama penambahan, tetap terhadap tiga fungsi utama kita, yaitu fungsi predict, loss, dan train. Mari kita mulai dengan fungsi predict terlebih dahulu, dimana sebelumnya fungsi predict digambarkan sebagai berikut:

```
In [ ]:
def predict(X, w):
    return X * w
```

Dan rumus diatas merupakan rumus dasar kita bukan? dan untuk menambah bias, cukup tambahkan saja objek **b**, sehingga menjadi:

```
In [ ]:
def predict(X, w, b):
    return X * w + b
```

Fungsi loss

Dan yang kedua, habis predict, maka dilanjutkan dengan loss, dan juga sangat sederhana, cukup ditambahkan saja objek b, dari:

```
In [ ]:
def loss(X, Y, w):
    return np.average((predict(X, w) - Y) ** 2)
```

Dengan menambah b menjadi:

```
In [ ]:
def loss(X, Y, w, b):
    return np.average((predict(X, w, b) - Y) ** 2)
```

Fungsi train

Dan datanglah pemain utama kita, yaitu fungsi train, yang berfungsi untuk mempelajari pola - pola data kita, dan perubahan paling banyak akan terjadi di fungsi ini, karena kita perlu melibatkan proses untuk melatih objek b, dari yang semula:

```
In [ ]:
def train(X, Y, iterations, lr):
    w = 0
    for i in range(iterations):
        current_loss = loss(X, Y, w)
        print("Iterasi %4d -> Loss: %.6f" % (i, current_loss))

        if loss(X, Y, w + lr) < current_loss:
            w += lr
        elif loss(X, Y, w - lr) < current_loss:
            w -= lr
        else:
            return w
    raise Exception("Terjadi kesalahan pada iterasi ke %d" % iterations)
```

Menjadi gambar dibawah, dimana proses pelatihan variabel b menggunakan konsep kemungkinan-kemungkinan, dari dua percabangan menjadi empat percabangan sebagai berikut:

In [ ]:
```
def train(X, Y, iterations, lr):
    w = b = 0
    for i in range(iterations):
        current_loss = loss(X, Y, w, b)
        print("Iteration %4d -> Loss: %.6f" % (i, current_loss))

        if loss(X, Y, w + lr, b) < current_loss:
            w += lr
        elif loss(X, Y, w - lr, b) < current_loss:
            w -= lr
        elif loss(X, Y, w, b + lr) < current_loss:
            b += lr
        elif loss(X, Y, w, b - lr) < current_loss:
            b -= lr
        else:
            return w, b
    raise Exception("Couldn't converge within %d iterations" % iterations)
```

In [ ]:
```
def train(X, Y, iterations, lr):
    w = b = 0
    for i in range(iterations):
        current_loss = loss(X, Y, w, b)
        print("Iteration %4d -> Loss: %.6f" % (i, current_loss))

        if loss(X, Y, w + lr, b) < current_loss:
            w += lr
        elif loss(X, Y, w - lr, b) < current_loss:
            w -= lr
        elif loss(X, Y, w, b + lr) < current_loss:
            b += lr
        elif loss(X, Y, w, b - lr) < current_loss:
            b -= lr
        else:
            return w, b
    raise Exception("Couldn't converge within %d iterations" % iterations)
```

Implementasi Model

Dengan lengkapnya tiga fungsi diatas, maka untuk pembentukan model, yang kode sumber asal sebagai berikut:

In [ ]:
```
X, Y = np.loadtxt(
    "data-reservasi-1.txt",
    skiprows= 1,
    unpack= True
)
```

In [ ]:
```
w = train(
    X,
    Y,
    iterations= 10000,
    lr= 0.01
)
```

In [ ]:
```
print("w= %.3f" % w)
```

In [ ]:
```
reservasi = 24
```

In [ ]:
```
print("Prediksi: bila reservasi = %d, maka bahan baku yang perlu disiapkan = %.2f" %(reservasi, predict(reservasi, w)))
```

Akan kita ganti dengan penambahan objek b, menjadi:

In [ ]:
```
X, Y = np.loadtxt(
    "data-reservasi-1.txt",
    skiprows= 1,
    unpack= True
)
```

In [ ]:
```
w, b = train(
    X,
    Y,
    iterations= 10000,
    lr= 0.001
)
```

In [ ]:
```
print("w= %.3f, b= %.3f" % (w,b))
```

In [ ]:
```
reservasi = 24
```

In [ ]:
```
print("Prediction: x= %d -> y= %.2f" %(reservasi, predict(reservasi, w, b)))
```

Menggambar Garis Prediksi

Dan kita gambar hasil akhir garis prediksi kita dengan menggunakan matplotlib dan sns sebagai berikut:

```
In [ ]:
x1 = 0
x2 = 50
y1 = x1 * w + b
y2 = x2 * w + b
```

```
In [ ]:
sns.set()
plt.axis([0, 60, 0, 60])
plt.xticks(fontsize= 15)
plt.yticks(fontsize= 15)
plt.xlabel("Reservasi", fontsize= 20)
plt.ylabel("Bahan", fontsize= 20)
plt.plot(X, Y, "bo")
plt.plot([x1,x2], [y1,y2], "g")
plt.show()
```

Pembuktian

Mari kita jalankan implementasi model kita, dan dengan angka reservasi sebanyak 24 meja, maka akan didapatkan data bahan sebesar:

```
In [9]:
print("Prediction: x= %d -> y= %.2f" %(reservasi, predict(reservasi, w, b)))
```

```
Prediction: x= 24 -> y= 34.58
```

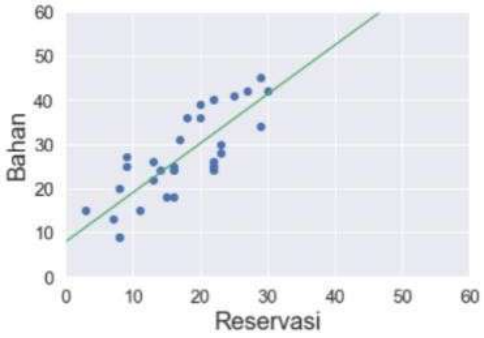
Semua proses membutuhkan **sebanyak 926 kali** pembelajaran dengan **w = 1.140**, dan **b = 7.400**

```
Iterasi  920 -> Loss: 33.359827
Iterasi  921 -> Loss: 33.358793
Iterasi  922 -> Loss: 33.357960
Iterasi  923 -> Loss: 33.357327
Iterasi  924 -> Loss: 33.356893
Iterasi  925 -> Loss: 33.356660
Iterasi  926 -> Loss: 33.356627
```

```
In [37]:
print("w= %.3f, b=%.3f" % (w,b))
```

```
w= 1.140, b=7.400
```

Pada model sebelumnya, kita memprediksikan 36 bahan baku, sedangkan pada model ini, kita memprediksi sekitar 35 bahan baku, jadi disini terbukti bahwa dengan penambahan bias, bisa menambah akurasi data kita, sehingga lebih menghemat persediaan bahan baku. Bagaimana dengan tampilan garisnya, terlebih jelas bahwa garis naik dari posisi vertikal keatas, dan terasa sekarang lebih mendekat terhadap pola-pola data kita.

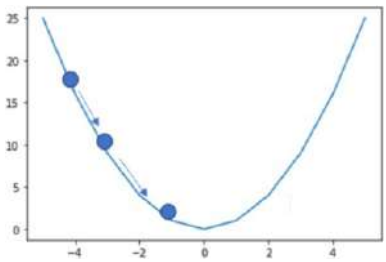


Selamat, kita telah berhasil mempelajari sebuah model simple Linear Regression yang lengkap.

g. Gradien Descent

Pada subtopik sebelumnya, kita sudah berhasil membuat sebuah model Simple Linear Regression yang bisa menebak dengan baik sebuah data tunggal, yaitu data reservasi pada kasus restoran kita. Dan model **hanya bisa** untuk data tunggal saja, karena bisa dibayangkan, dengan data tunggal saja, kita membutuhkan empat percabangan, bagaimana dengan data biner atau data dua kolom? bagaimana dengan tiga kolom? dan seterusnya? akan terjadi masalah yang dinamakan Combinatorial Explosion. Karena memang pada dunia nyata, kita tidak hanya berurusan dengan data tunggal, data yang harus diproses bisa terdiri dari data yang ratusan kolom, bahkan mungkin ribuan, jutaan dan sebagainya, dan ini tetap merupakan tantangan yang harus kita hadapi untuk bisa menciptakan sebuah model machine learning.

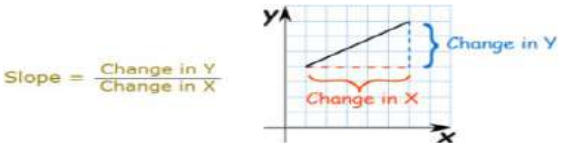
Gradient Descent



Gradient Descent adalah sebuah algoritma yang berfungsi untuk menghitung

turunan dari sebuah slope (slope), dan digambarkan oleh gambar diatas, diasumsikan ada sebuah lembah (kurva), dan diletakan sebuah bola pada sisi atas lembah tersebut, maka akibat adanya gravitasi, maka secara otomatis bola tersebut pasti akan bergerak turun mengikuti gravitasi sampai ke dasar lembah. Dengan ilustrasi diatas, kita asumsikan bola tersebut adalah nilai loss kita, maka kalau mengikuti algoritma gradient descent, kita mengharapkan loss kita harus turun seturun-turunnya sampai mendekati angka minimal nol, karena dengan loss yang kecil akan mempertinggi akurasi bukan? Dan dalam beberapa literatur, ada juga yang menyebutkan Gradient Ascent, apakah sama? Sama, dan hanya berbeda pada proses gravitasinya, kalau descent, berarti menurun, sedangkan ascent, berarti menaik dengan kurva yang terbalik, kalau descent berarti menimimalkan loss function, maka ascent mempunyai arti memaksimalkan loss function.

Slope



Jadi inti dari algoritmanya yang paling penting adalah, bagaimana cara kita mengukur slope dari lembah tersebut? Sesuai dengan namanya, kita bisa mengukur slope tersebut dengan menggunakan fungsi gradient dalam matematika. Untuk memperjelas apa itu gradient, silakan belajar lebih lanjut pada laman Gradient (Slope) of a Straight Line.

Derivatives

Untuk memahami mengenai gradient dan bagaimana cara menghitungnya, perlu dipahami juga mengenai kalkulus terutama pada kasus differensial, bagi jurusan IPA pasti pernah mengalami ketakutan dengan ilmu ini, tapi sebenarnya tidak susah kalau memang dipelajari dengan baik, dan salah satu sumber yang bagus untuk mempelajari mengenai turunan, dapat dilihat pada laman Introduction to Derivatives.

Apabila diketahui rumus MSEMSE sebagai berikut:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(prediksi_i - target_i)^2$$

Dan kita gantikan prediksiprediksi dan targettarget dengan yy dan y^y^ sebagai berikut:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$$

Kemudian kita gantikan yy dengan rumus y=wx+by=wx+b sebagai berikut:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}((wx_i + b) - \hat{y})^2$$

Maka kita bisa menghitung turunan MSEMSE terhadap ww sebagai berikut:

$$\frac{\partial MSE}{\partial w} = \frac{2}{n}\sum_{i=1}^{n}x_i((wx_i + b) - \hat{y}$$

Dan turunan MSEMSE terhadap bb sebagai berikut:

$$\frac{\partial MSE}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}x_i((wx_i + b) - \hat{y}$$

h. Implementasi Gradien Descent

Saatnya kembali ke Jupyter Notebook, dan kali ini kita akan kembali dengan menggunakan senjata baru, yaitu algoritma Gradient Descent untuk fokus menyelesaikan kasus restoran kita yang masih memakai data tunggal.

Fungsi gradient

Perubahan yang perlu kita lakukan hanyalah menambah fungsi gradient, serta mengubah proses di fungsi training kita, untuk fungsi gradient, bisa digambarkan sebagai berikut:

In [ ]:
```
def gradient(X, Y, w, b):
        w_gradient = 2 * np.average(X * (predict(X, w, b) - Y))
        b_gradient = 2 * np.average(predict(X, w, b) - Y)
        return (w_gradient, b_gradient)
```

Sebentar, bagaimana bisa muncul rumus untuk objek w_gradient, b_gradient? Darimana datangnya? jawabannya sederhana, rumus turunan dari diferensial loss terhadap w dan diferensial loss terhadap b.

Mengubah fungsi train

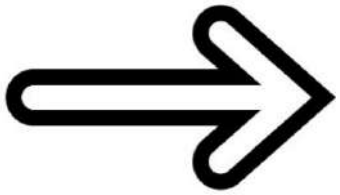
Setelah adanya fungsi gradient diatas, tugas kita selanjutnya hanya tinggal

mengubah seluruh proses percabangan pada fungsi train menjadi fungsi gradient.

In [ ]:
```
def train(X, Y, iterations, lr):
    w = b = 0
    for i in range(iterations):
        current_loss = loss(X, Y, w, b)
        print("Iterasi %4d -> Loss: %.6f" % (i, current_loss))

        w_gradient, b_gradient = gradient(X, Y, w, b)
        w -= w_gradient * lr
        b -= b_gradient * lr

    return w, b
```

Terkejut bukan? Proses percabangan yang pada kode sumber sebelumnya sebanyak 10 baris bisa digantikan oleh tiga baris tersebut, sebagai bahan perbandingan, lihat gambar dibawah:

```
if loss(X, Y, w + lr, b) < current_loss:
    w += lr
elif loss(X, Y, w - lr, b) < current_loss:
    w -= lr
elif loss(X, Y, w, b + lr) < current_loss:
    b += lr
elif loss(X, Y, w, b - lr) < current_loss:
    b -= lr
else:
    return w, b
```



```
w_gradient, b_gradient = gradient(X, Y, w, b)
w -= w_gradient * lr
b -= b_gradient * lr
```

Itulah istimewanya algoritma gradient, mengganti proses percabangan dengan menggunakan perhitungan matematika.

Finalisasi

Sekarang mari kita coba untuk implementasi training dengan kode berikut, tapi dilakukan perubahan terhadap learning rate dan iterations, karena memang akan membutuhkan iterasi yang sangat banyak untuk mencapai loss seperti yang sama dengan model terakhir kita.

```
In [ ]:
X, Y = np.loadtxt(
    "data-reservasi-1.txt",
    skiprows= 1,
    unpack= True
)
```

```
In [ ]:
w, b = train(
    X,
    Y,
    iterations= 20000,
    lr= 0.001
)
```

```
In [ ]:
print("w= %.3f, b=%.3f" % (w,b))
```

```
In [ ]:
reservasi = 24
```

```
In [ ]:
print("Prediksi: bila reservasi = %d, maka bahan baku yang perlu disiapkan = %.2f" %(reservasi, predict(reservasi, w, b)))
```

Apabila dijalankan kode sumber diatas, akan didapatkan nilai sebagai berikut:

```
Iterasi 19995 -> Loss: 33.314032
Iterasi 19996 -> Loss: 33.314031
Iterasi 19997 -> Loss: 33.314031
Iterasi 19998 -> Loss: 33.314031
Iterasi 19999 -> Loss: 33.314031
```

```
In [8]:
print("w= %.3f, b=%.3f" % (w,b))
w= 1.112, b=7.868
```

```
In [9]:
reservasi = 24
```

```
In [10]:
print("Prediksi: bila reservasi = %d, maka bahan baku yang perlu disiapkan = %.2f" %(reservasi, predict(reservasi, w, b)))
Prediksi: bila reservasi = 24, maka bahan baku yang perlu disiapkan = 34.56
```

Kesimpulan: kita mendapatkan nilai akurasi yang mirip dengan model percabangan sebelumnya, tapi dengan kesempatan yang sangat besar untuk menambah jumlah data kita, dan proses penjelasan mengenai Gradient Descent dan penambahan jumlah data (hyperspace) merupakan bahan untuk topik kita selanjutnya, dan memang gradient descent is the

4. Forum Diskusi
   a. Bersama dengan tim anda, hitunglah MSE dan fumgsi-fungsi pada sebuah model dari dataset yang terdapat pada Bab sebelumnya.
   b. Lakukanlah training/prediksi dari dataset tersebut

C. PENUTUP

1. Rangkuman

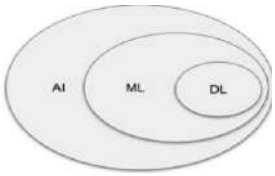
   Kembali kepada masalah kita, apa hubungannya dengan gradient descent? Hubungannya adalah untuk menghindari ledakan kombinatorial diatas, karena apabila kita mentraining model dan menghitung loss dengan menggunakan kemungkinan -

kemungkinan percabangan, maka ledakan kombinatorial tidak bisa kita hindari, akan tetapi kalau kita mentraining dan menghitung loss dengan gradient descent, maka mau sebanyak apapun data yang kita proses, maka yang kita hadapi tetap hanya perhitungan matematika saja, bukan percabangan - percabangan yang banyak. Dan seperti kita ketahui, kemampuan paling utama dari sebuah komputer, adalah kecepatan berhitung, dan seperti kita ketahui untuk setiap GHz dari sebuah Central Processing Unit (CPU), bisa dilakukan operasi perhitungan sebanyak 1.000.000 instruksi per detik, belum lagi kalau menggunakan Graphical Processing Unit (GPU) yang bisa puluhan kali lebih cepat dari CPU. Dengan melihat pernyataan diatas, maka kita akan coba mengubah kode sumber kita dari menggunakan percabangan, menjadi sebuah rumus matematika menggunakan algoritma gradient descent.

2. Tes Formatif

Sebagai evaluasi pemahaman materi pada Bab 3, jawablah pertanyaan berikut ini.

a. Pada gambar berikut ini, dapat dilihat, bahwa AI menduduki bagian terbesar dari seluruh konsep kecerdasan buatan, dan diperkecil dengan subset ML, dan lebih diperkecil lagi untuk DL.



b. Contoh sederhana yang bisa menjelaskan ML dengan lebih sederhana, adalah contoh seorang bayi yang baru lahir. Bayi yang baru lahir pastinya tidak akan bisa membedakan jenis kelamin bukan? Pada umur dua sampai tiga tahun, mulai bisa mengenal Bapak dan Ibunya, dan seiring berjalannya waktu menuju kedewasaan dengan banyak melihat dan berpikir, kita bisa membedakan laki-laki dan perempuan, cukup dengan hanya melihat wajah manusia, seperti fitur rambutnya panjang, mata lebih lentik, dan sebagainya. Akan tetapi disini terkadang kita juga bisa melakukan kesalahan, karena seorang manusia yang rambutnya panjang, bukan berarti berjenis kelamin perempuan, dan juga contoh lainnya, apabila kita ke negara Thailand, kita akan susah membedakan mana laki-laki dan mana perempuan.

c. Berikut ini yang merupakan sub bagian dari Machine Learning adalah:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning
4. Classification

d. Sebuah ilmu yang memakai kemampuan komputer untuk meniru kemampuan berpikir manusia dalam pengambilan keputusan, pengolahan teks, dan persepsi visual. AI adalah satu bagian dari ilmu komputer yang mempunyai banyak subbidang, seperti Machine Learning, Robotics, Computer Vision, dan lain sebagainya, Hal tersebut merupakan defenisi dari model arsitektur dalam ilmu statistika yang berfungsi untuk membuat prediksi dengan menggunakan sebuah garis linear, dan fungsi garis linear yang paling sederhana, disebut dengan apa?
e. Sebuah parameter yang umum digunakan dalam arsitektur model Machine Learning sebagai sebuah nilai untuk menyetel kemampuan model kita seberapa cepat dalam pembelajaran dan adaptasi terhadap loss function. Hal tersebut disebut dengan apa?
f. Proses pembelajaran tanpa adanya label, dan bertujuan untuk membentuk sendiri label-label berdasarkan pola-pola data input yang diberikan. Sering juga disebut dengan istilah Clustering, atau klasterisasi, yaitu proses pembentukan klaster data. Hal tersebut disebut dengan apa?
g. Sebuah algoritma yang berfungsi untuk menghitung turunan dari sebuah slope(slope), diasumsikan ada sebuah lembah(kurva), dan diletakan sebuah bola pada sisi atas lembah tersebut, maka akibat adanya gravitasi, maka secara otomatis bola tersebut pasti akan bergerak turun mengikuti gravitasi sampai ke dasar lembah. Hal tersebut disebut dengan apa?
h. Pada pemrograman klasik, manusia akan memasukan data dan jawabannya, dan dihasilkan aturan - aturan, dan aturan yang baru dihasilkan tersebut bisa digunakan untuk menghasilkan jawaban baru pada data yang baru.
    1. True
    2. False
i. Tahapan Prediction sangat sederhana, karena pada tahapan ini hanya perlu melakukan proses perhitungan matematika, yaitu menghitung rumus $y = x + w$.

Sedangkan tahapan Training yang lebih kompleks, disinilah proses optimasi dilakukan dengan menggunakan Loss function.

1. True
2. False

j. Sebuah Machine Learning dihasilkan melalui proses pembelajaran/Training

1. True
2. False

k. Proses Reinforcement Learning, boleh diibaratkan sama seperti manusia yang belajar memainkan game komputer Role Playing Game (RPG), dimana pemain tersebut akan belajar memainkan game tersebut di lingkungan game itu dengan tujuan meningkatkan kemampuannya.

1. True
2. False

DAFTAR PUSTAKA

Belajar Machine Learning : Simple Linear Regression di Python, Adipta Martulandi, https://medium.com/@adiptamartulandi/belajar-machine-learning-simple-linear-regression-di-python-e82972695eaf

Gradient Descent–Everything You Need To Know With Implementation In Python, https://analyticsindiamag.com/gradient-descent-everything-you-need-to-know-with-implementation-in-python/

Python Data Science Handbook, by Jake VanderPlas, Published by O'REILLY

Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, By Wes McKinney, Published by O'Reilly

BAB IV
SHALLOW LEARNING MODELING

A. PENDAHULUAN

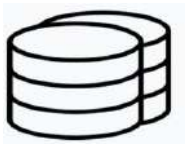
Materi pada Bab 4 membahas tentang hal-hal yang dilakukan dalam membentuk model pada Deep Learning berdasarkan dataset yang tersedia dalam mencari solusi berdasarkan nilai akurasi yang diperoleh, lalu melakukan evaluasi terhadap hasil yang diperoleh sebagai pertimbangan dalam memilih alternative solusi.

B. INTI

1. Capaian Pembelajaran
    a. Mampu memahami dan menjelaskan hal-hal perhitungan yang dapat dilakukan dalam bidang Deep Learning dalam membentuk model dan mengevalusi model tersebut
    b. Mampu memahami dan menjelaskan apa itu tensor
    c. Mampu memahami dan memahami jenis-jenis klasifikasi pada deep learning
    d. Mampu memahami dan menjelaskan apa itu Logististic Regression, Irisan pada dataset, overfitting, Hold out, dan cross validation
2. Materi
    a. Tensor
    b. Classification
    c. Logististic Regression
    d. Iris dataset
    e. Dummy Dataset
    f. Overfitting
    g. Hold out
    h. Cross validation

3. Uraian Materi

a. Tensor

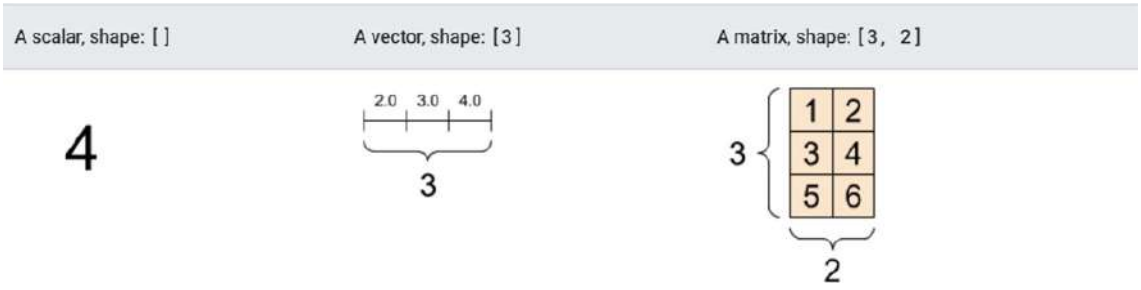


Dalam data science, sering muncul istilah tensor, apa itu tensor ? Istilah ini bahkan muncul dalam salah satu modul paling terkenal di deep learning, yaitu Tensorflow, sebuah modul yang diciptakan oleh Google untuk keperluan produksi dan penelitian pada machine learning dan artificial intelligence. Secara sederhana, tensor adalah struktur data dimensional yang membedakan jenis data. Vektor adalah struktur data satu dimensi dan matriks adalah struktur data dua dimensi. Tensor dapat berbentuk dimensi mulai dari nol hingga n dimensi.

Tujuan tensor adalah untuk membedakan jenis dataset, karena setiap dataset pasti mempunyai bentok tensor, dan umumnya tensor yang sering dipakai adalah sebagai berikut:

1. Scalar (0D tensor)
2. Vector (1D tensor)
3. Matrices (2D tensor)
4. Timeseries (3D tensor)
5. Images (4D tensor)
6. Videos (5D tensor)

Tensor 0D, 1D, 2D

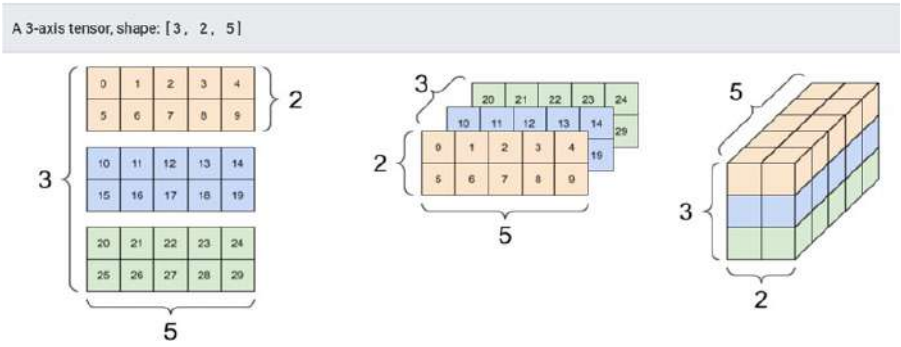


Gambar 16. Tensor 0D, 1D, 2D

Perhatikan gambar diatas, pada bentuk paling dasarnya, tensor scalar (0D), dapat diibaratkan sebuah data tunggal, dan kumpulan atau barisan angka, disebut juga

dengan tensor vector (1D), dan apabila barisan angka tersebut ditambah kolomnya, sehingga data kita berbentuk data tabel, disebut juga dengan tensor matrices (2D). Tensor matrices merupakan tensor yang paling banyak dipakai karena pada dasarnya data-data dibentuk dengan tensor ini, seperti contoh pada aplikasi Excel, dimana pengisian data-data dalam sheet yang berbentuk tabel.

Tensor 3D



Gambar 17. Tensor 3D

Jika diibaratkan tensor matrices adalah data dalam bentuk segiempat, maka untuk tensor selanjutnya, yaitu tensor timelines, adalah tensor yang berbentuk kubus, seperti dapat dilihat pada gambar diatas, dan untuk membantu memahami, kita akan menggunakan istilah axis (sumbu), atau level juga bisa, dimana axis pertama untuk tanggal, axis kedua untuk nama perusahaan, dan axis ketiga untuk data perusahaan tersebut.

Tahun 2018

1. Perusahaan A : Laba bersih 1 milyar, Jumlah karyawan 100
2. Perusahaan B : Laba bersih 2 milyar, Jumlah Karyawan 85
3. Perusahaan C : Laba bersih 1.5 milyar, Jumlah karyawan 125

Tahun 2019

1. Perusahaan A : Laba bersih 1.1 milyar, Jumlah karyawan 110
2. Perusahaan B : Laba bersih 1.8 milyar, Jumlah karyawan 90
3. Perusahaan C : Laba bersih 2 milyar, Jumlah karyawan 125

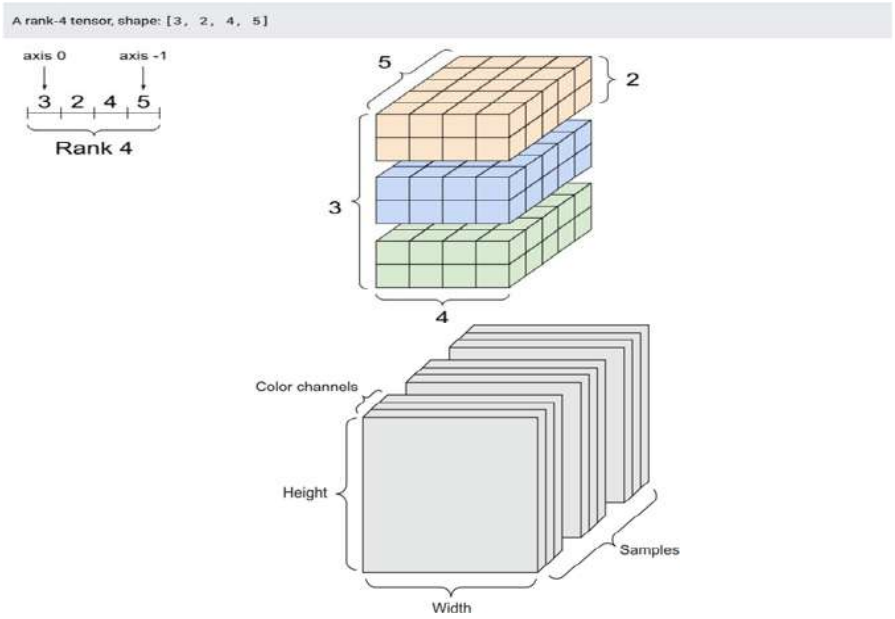
Tahun 2020

1. Perusahaan A : Laba bersih 800 juta, Jumlah karyawan 80
2. Perusahaan B : Laba bersih 1.5 milyar, Jumlah karyawan 100

3. Perusahaan C : Laba bersih 1.8 milyar, Jumlah karyawan 130

Contoh diatas adalah contoh dataset nilai perusahaan dari tahun 2018 sampai dengan tahun 2020, dimana data tahun sebagai axis pertama, nama perusahaan sebagai axis kedua, dan data perusahaan sebagai axis ketiga.

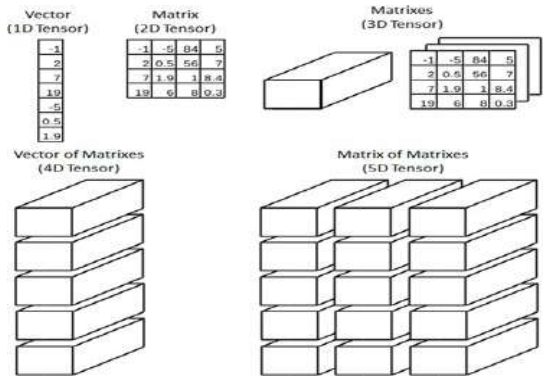
Tensor 4D



Gambar 18. Tensor 4D

Jika diibaratkan tensor sebelumnya merupakan tensor kubus, maka untuk tensor images (4D), adalah tensor kumpulan kubus, karena diibaratkan satu gambar sama dengan satu kubus, maka untuk menyimpan tensor gambar, apabila ada 100 gambar, maka diperlukan 100 kubus, dengan axis berupa Sample, Width, Height, dan Channel warna.

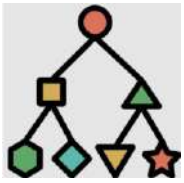
Tensor 5D



Gambar 19. Tensor 5D

Kumpulan kubus pada tensor images, apabila disusun dalam bentuk tabel, maka terbentuklah tensor videos (5D), seperti dapat dilihat pada gambar diatas, transformasi dimensi tensor dari 0D sampai dengan 5D, karena video sendiri merupakan gambar bergerak, dan umumnya setiap detik video, disebut juga dengan istilah frame, sehingga axis dari tensor 5D berbentuk Sample, Frame, Width, Height, dan Channel warna.

b. Classification



Classification adalah bagian dari supervised learning, yaitu sistem pembelajaran yang diawasi dengan penggunaan label, pada topik 6, kita sudah mempelajari mengenai contoh regresi, maka pada topik ini, kita akan mempelajari kasus klasifikasi.

Kasus-kasus klasifikasi dapat terbagi lagi menjadi:

1. Binary classification;
2. Multiclass classification;
    a. Single-label multiclass classification;
    b. Multi-label multiclass classification.

Binary Classification

Binary classification yaitu classification yang hanya bersifat ganda, yaitu ya atau tidak, 0 atau 1, true atau false, contoh paling sederhana untuk classification ini adalah kasus spam email, dimana aplikasi email berusaha mengklasifikasikan sebuah email yang masuk adalah email yang spam atau tidak.

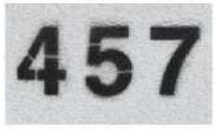
Single-label Multiclass Classification

Multiclass Classification sendiri adalah klasifikasi yang lebih dari dua, seperti contoh kita mengklasifikasi 1 digit gambar angka, dimana angka sendiri terdiri dari

angka 0, 1, 2, sampai 9, dan kita berusaha mengklasifikasi setiap gambar angka, apakah dia angka 0, 1 dan seterusnya.

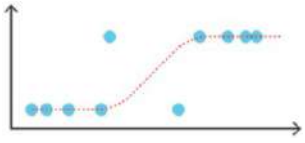
Multi-label Multiclass Classification

Contoh menebak gambar angka diatas adalah contoh untuk single-label multiclass classification, bagaimana kalau kita ingin mengklasifikasi kumpulan gambar angka ? seperti contoh mengklasifikasi gambar angka 852, dimana gambar ini terdiri dari 3 angka (label), dengan class berjumlah 1.000 class, yaitu dari angka 0 sampai dengan 999.



Contoh lain dari multilabel multiclass classification adalah mengklasifikasi sebuah gambar ruang tamu, dimana umumnya sebuah gambar ruang tamu, pasti berisikan meja tamu, kursi tamu, pot bunga, karpet lantai, dan lain sebagainya.

c. Logistic Regression



Dalam statistika, model logistik (atau model logit) digunakan untuk memodelkan probabilitas kelas atau peristiwa tertentu yang ada seperti lulus/gagal, menang/kalah, hidup/mati atau sehat/sakit. Ini dapat diperluas untuk memodelkan beberapa kelas kejadian seperti menentukan apakah suatu gambar berisi kucing, anjing, singa, dll. Setiap objek yang terdeteksi dalam gambar akan diberi probabilitas antara 0 dan 1, dengan jumlah satu.

Logistic regression bukanlah sebuah model regresi, walaupun mengandung kata regression, dan ini merupakan kesalahan paling umum yang sering dilakukan oleh pemula. Untuk kasus regresi, namanya Linear Regression, dan untuk kasus classification, namanya Logistic Regression. Jadi kenapa harus menggunakan istilah regression ? Karena memang pada dasarnya rumus dari model Logistic Regression adalah pengembangan dari Linear Regression dengan hanya menambah fungsi sigmoid.

Rumus Logistic Regression

Rumus Linear Regression $= wx + b$

Rumus Sigmoid Function $\frac{1}{1=e^{x}}$

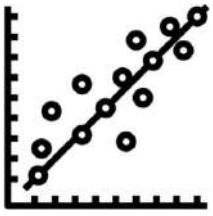
Dan masukan rumus linear regression ke dalam rumus sigmoid function, maka didapat rumus Logistic Regression sebagai berikut: $\frac{1}{1=e^{wx+b}}$

Fungsi sigmoid adalah fungsi yang menghasilkan nilai dengan ambang batas antara 0 sampai 1, maka dengan melakukan proses pembulatan terhadap rumus Logistic Regression, didapatlah model binary classification dengan prediksi 0 sampai 0,5 untuk false dan 0,5 sampai 1 untuk true.

$$LogisticRegression = round(\frac{1}{1 = e^{wx+b}})$$

Beda antara linear regression dan logistic regression dapat dilihat pada gambar berikut:

Linear Regression



Logistic Regression

Scikit Learn

Model Logistic Regression dapat dengan mudah kita pakai menggunakan modul sklearn dari Scikit Learn, dan pastikan untuk terlebih dahulu melakukan instalasi modul scikit-learn.

```
In [ ]: 1 from sklearn.linear_model import LogisticRegression
In [ ]: 1 model = LogisticRegression()
```

d. Iris dataset

Iris dataset adalah dataset populer untuk belajar logistic, dan dataset ini berasal dari UCI Machine Learning Repository, dan dataset ini merupakan bagian dari modul scikit-learn, yang dapat diimport langsung untuk dipakai.

```
In [ ]: 1 from sklearn.datasets import load_iris
In [ ]: 1 X, y = load_iris(return_X_y= True)
In [ ]: 1 X.shape
In [ ]: 1 y.shape
```

Jumlah sampel iris dataset ini berjumlah 150 sampel, dengan 4 feature, dan 1 target 3 class. Untuk mempermudah melihat datanya, kita akan import data numpy ini ke dalam dataframe Pandas.

```
In [ ]: 1 import pandas as pd

In [ ]: 1 import numpy as np

In [ ]: 1 df = pd.DataFrame(
        2     np.column_stack((X,y)),
        3     columns = ['sepal_length','sepal_width','petal_length','petal_width','class']
        4 )

In [ ]: 1 df.head()

In [ ]: 1 df.describe()

In [ ]: 1 df.info()

In [ ]: 1 df['class'].value_counts()

In [ ]: 1 df.groupby('class').mean()
```

Setelah memahami bentuk dataset iris, maka kita bisa mulai membuat model untuk memprediksi klasifikasi bunga iris dengan menggunakan model Logistic Regression.

```
In [ ]: 1 from sklearn.linear_model import LogisticRegression

In [ ]: 1 model = LogisticRegression()

In [ ]: 1 model.fit(X, y)

In [ ]: 1 model.predict(X[0:30, :])

In [ ]: 1 score = model.score(X, y)

In [ ]: 1 print("Tingkat akurasi: %.3f" % score)
```

Fungsi fit() adalah fungsi untuk melakukan pembentukan model dan fungsi predict() adalah fungsi untuk memprediksi klasifikasi, sedangkan fungsi score() berfungsi untuk melihat tingkat akurasi prediksi.

e. Dummy Dataset

Pada subtopik sebelumnya, kita bermain dengan dataset iris, dengan sampel sebanyak 150 sampel, dan dengan menggunakan model Logistic Regression, kita berhasil mencapai tingkat akurasi yang tinggi. Untuk praktek selanjutnya, mari kita ciptakan dummy dataset, yaitu dataset yang dibuat secara artificial, dengan menggunakan modul scikit-learn.

```
In [ ]:
from sklearn.datasets import make_classification
```

```
In [ ]:
X, y = make_classification(
    n_samples= 10000,
    n_features= 5,
    random_state= 1
)
```

Dan kita bedah dataset kita dengan menggunakan Pandas.

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

```
In [ ]:
import numpy as np
```

```
In [ ]:
df = pd.DataFrame(
    np.column_stack((X,y)),
    columns = ['feature 1','feature 2','feature 3','feature 4','feature 5','class']
)
```

```
In [ ]:
df.head()
```

```
In [ ]:
df.describe()
```

```
In [ ]:
df.info()
```

```
In [ ]:
df['class'].value_counts()
```

```
In [ ]:
df.groupby('class').mean()
```

Dan dengan menggunakan model Logistic Regression dengan kode yang sama seperti pada dataset iris, kita menciptakan model yang bisa mengklasifikasi dummy dataset ini.

```
In [ ]:
from sklearn.linear_model import LogisticRegression
```

```
In [ ]:
model = LogisticRegression()
```

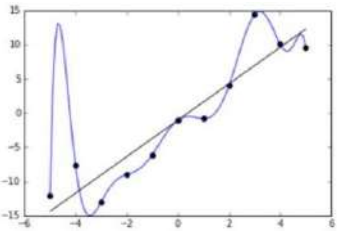
```
In [ ]:
model.fit(X, y)
```

```
In [ ]:
model.predict(X[0:30, :])
```

```
In [ ]:
score = model.score(X, y)
```

```
In [ ]:
print("Tingkat akurasi: %.3f" % score)
```

f. Overfitting



Overfitting adalah sebuah kondisi dimana model yang kita ciptakan tidak sesuai

dengan harapan kita, dan untuk memahami overfitting, dapat dilihat dengan contoh kasus berikut.

Contoh Overfitting

Seandainya kita mengajar matematika dasar kepada anak - anak sekolah dasar, dan sudah disiapkan 60 soal pilihan berganda kuis perkalian. Dan kuis ini akan disiapkan untuk kuis pekerjaan rumah, dan kuis ujian;

Jadi ada dua kelompok kuis, yaitu pekerjaan rumah, dan ujian, dan dari dua kelompok ini, ada dua opsi yang bisa dilakukan:

1. Apakah 60 soal kuis akan diberikan sebagai pekerjaan rumah, dan kemudian menggunakan 10 soal dari 60 soal tersebut sebagai ujian ?
2. Atau membagi kuis tersebut, dimana 50 soal akan diberikan sebagai pekerjaan rumah, dan sisa 10 soal sebagai ujian ?

Opsi yang bagus untuk dipilih, adalah opsi kedua;

Dengan alasan supaya peserta bukan menghafal perkaliannya, tapi peserta sendiri bisa harus memahami dan menerapkan konsep perkalian. Sebagai pengajar, kita pasti lebih menginginkan peserta kita untuk bisa menjawab pertanyaan yang belum pernah mereka lihat sebelumnya, daripada peserta menghafal mati jawaban dari soal kuis tersebut. Contoh diatas merupakan merupakan falsafah yang harus kita terapkan dalam pembuatan model Machine Learning,

Supervised Learning

Proses pembelajaran umumnya terbagi dua kategori, yaitu menghafal, atau memahami, dan yang kita harapkan dalam pembelajaran bukan menghafal, karena dengan menghafal, kita hanya bisa menyelesaikan soal-soal yang telah kita hafal sebelumnya, sehingga apabila kita disuguhkan soal yang belum pernah kita lihat sebelumnya, kita tidak akan bisa menjawabnya. Akan tetapi apabila kita memahami soal tersebut, maka setiap soal-soal baru yang kita terima, pasti bisa kita selesaikan dengan baik, dengan tingkat akurasi yang berbeda-beda, karena terkadang menghadapi soal yang benar-benar baru, kita bisa salah menjawabnya.

Hal ini berlaku juga dengan model machine learning, model yang kita harapkan itu adalah model yang bisa memahami masalah, bukan menghafal masalah, dan istilah menghafal masalah ini sama dengan istilah Overfitting, dan kebalikan dari

Overfitting adalah Underfitting, yaitu sebuah kondisi dimana model tidak belajar sama sekali.

Model Evaluation

Maka untuk mencegah overfitting, diperlukan sebuah evaluasi, dan evaluasi model dasar yang umum dilakukan dalam dunia machine learning ada dua, yaitu:

1. Hold out evaluation
2. Cross validation evaluation

g. Hold out

Evaluasi model dengan menggunakan konsep hold out adalah proses evaluasi dengan membagi data menjadi dua atau tiga bagian. Bila dataset dibagi menjadi dua bagian, maka bagian tersebut disebut dengan:

1. Train set;
2. Test set

Dan apabila dibagi menjadi tiga bagian, dapat disebut:

1. Train set;
2. Validation set;
3. Test set.

Bagaimana dengan angka persentasi pembagiannya? Tidak ada aturan baku yang menyebutkan angka yang baik itu seperti apa, tapi umumnya angka yang digunakan adalah perbandingan 80% dan 20%, dimana 80% dataset digunakan untuk membuat model, dan 20% digunakan mencoba model tersebut, atau bisa juga dengan model 60%, 20% dan 20%, yaitu 60% untuk model raining, 20% untuk validation, dan sisa 20% untuk model testing.

Gambar 20. Pembagian dataset untuk training dan testing

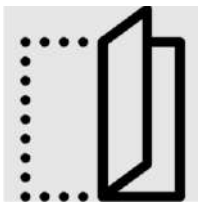
Proses hold out bisa dapat kita implementasi dengan menggunakan fungsi train_test_split pada modul scikit-learn.

```
In [ ]: from sklearn.model_selection import train_test_split

In [ ]: X_train, X_test, y_train, y_test = train_test_split(
            X,
            y,
            test_size= 0.2,
            random_state= 1
        )

In [ ]: X_train.shape

In [ ]: from sklearn.linear_model import LogisticRegression

In [ ]: model = LogisticRegression()

In [ ]: model.fit(X_train, y_train)

In [ ]: model.predict(X[0:30, :])

In [ ]: score = model.score(X_train, y_train)

In [ ]: print("Tingkat akurasi data training: %.3f" % score)

In [ ]: prediksi = model.predict(X_test)

In [ ]: benar = (prediksi == y_test).sum()

In [ ]: akurasi = benar / len(y_test)

In [ ]: print("Tingkat akurasi data test: %.3f" % akurasi)
```

Eksperimen

Bagaimana kalau untuk hold out tiga bagian? Silakan bereksperimen sendiri berdasarkan contoh diatas.

h. Cross validation



Cross validation adalah salah satu model evaluation yang digunakan apabila kita

mempunyai sampel yang sedikit, sehingga tidak memungkinkan untuk dilakukan hold out. Salah satu varian cross validation yang sering dipakai adalah K-Fold cross validation.

K-Fold Cross Validation

K-Fold cross validation adalah bentuk validasi yang akan memisahkan dataset menjadi dua bagian, yaitu training dan test dataset, dan akan diulang sebanyak K kali. Sebagai contoh dapat dilihat pada contoh berikut:

Seandainya kita mempunyai dataset: [1, 3, 7, 2, 5, 8, 0, 4, 9, 6], dan apabila dilakukan 3 Fold cross validation dan pembagian 80% dan 20%, maka akan dilakukan proses evaluasi sebanyak 3 putaran sebagai berikut:

Putaran 1: train [1, 3, 7, 2, 5, 8, 0, 4] test [9, 6]

Putaran 2: train [1, 3, 7, 2, 5, 8, 9, 6] test [0, 4]

Putaran 3: train [1, 3, 7, 2, 0, 4, 9, 6] test [5, 8]

Dengan bentuk putaran diatas, untuk setiap putaran, akan dihasilkan akurasi, dan akurasi - akurasi tersebut akan dirata - ratakan, dan nilai inilah yang akan menjadi acuan keberhasilan modelnya.

```
In [ ]:
from sklearn.datasets import make_classification
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
```

```
In [ ]:
X, y = make_classification(
    n_samples= 10000,
    n_features=5,
    random_state=1
)
```

```
In [ ]:
model = LogisticRegression()
```

```
In [ ]:
cv = KFold(
    n_splits= 10,
    random_state= 1,
    shuffle= True
)
```

```
In [ ]:
scores = cross_val_score(
    model,
    X,
    y,
    scoring= 'accuracy',
    cv= cv
)
```

```
In [ ]:
scores.mean()
```

4. Forum Diskusi

   Sebagai bahan eksperimen, ada dataset Bank Marketing dari UCI Machine Learning yang bisa digunakan untuk bereksperimen model

   a. Logistic Regression
   b. Iris dataset
   c. Overfitting

B. PENUTUP

1. Rangkuman

   Pada Bab ini, kita mempelajari bagaimana membangun model dengan menggunakan model Logistic Regression yang berasal dari Scikit-Learn, dan kita juga mempelajari konsep dasar Model Evaluation menggunakan Hold Out dan K-Fold Cross Validation

2. Tes Formatif

   Sebagai evaluasi pemahaman materi pada Bab ini, jawablah pertanyaan berikut ini.

   a. Dengan menggunakan model Logistic Regression, fungsu yang digunakan untuk melihat tingkat akurasi prediksi adalah
      1. Fungsi fit()
      2. Fungsi predict()
      3. Fungsi score ()
      4. Fungsi iris()

   b. Klasifikasi yang lebih dari dua, seperti contoh kita mengklasifikasi 1 digit gambar angka, dimana angka sendiri terdiri dari angka 0, 1, 2, sampai 9, dan kita berusaha mengklasifikasi setiap gambar angka, apakah dia angka 0, 1 dan seterusnya. Hal tersebut disebut dengan apa?
      1. Multiclass classification
      2. Multi-label multiclass classification
      3. Binary classification;
      4. Single-label multiclass classification;

   c. Untuk mencegah overfitting, diperlukan sebuah evaluasi, dan evaluasi model dasar yang umum dilakukan dalam dunia machine learning ada dua, yaitu:
      1. Hold out evaluation
      2. Cross validation evaluation

3. Hold out validation
4. Cross hold validation

d. Tujuan tensor adalah untuk membedakan jenis dataset, karena setiap dataset pasti mempunyai bentok tensor, dan umumnya tensor yang sering dipakai adalah sebagai berikut:
    1. Scalar (0D tensor)
    2. Vector (1D tensor)
    3. Matrices (2D tensor)
    4. Videos (3D tensor)
e. Classification yang hanya bersifat ganda, yaitu ya atau tidak, 0 atau 1, true atau false, contoh paling sederhana untuk classification ini adalah kasus spam email, dimana aplikasi email berusaha mengklasifikasikan sebuah email yang masuk adalah email yang spam atau tidak. Hal tersebut disebut dengan apa?
f. Fungsi yang menghasilkan nilai dengan ambang batas antara 0 sampai 1, maka dengan melakukan proses pembulatan terhadap rumus Logistic Regression, didapatlah model binary classification dengan prediksi 0 sampai 0,5 untuk false dan 0,5 sampai 1 untuk true. Disebut dengan fungsi apa?
g. Sebuah modul yang diciptakan oleh Google untuk keperluan produksi dan penelitian pada machine learning dan artificial intelligence, disebut dengan apa?
h. Kumpulan kubus pada tensor images, apabila disusun dalam bentuk tabel, maka terbentuklah tensor videos (4D)
    1. True
    2. False
i. Tensor adalah struktur data satu dimensi dan matriks adalah struktur data dua dimensi.
    1. True
    2. False
j. Jika diibaratkan tensor matrices adalah data dalam bentuk segiempat, maka untuk tensor selanjutnya, yaitu tensor timelines, adalah tensor yang berbentuk kubus
    1. True

2. False

DAFTAR PUSTAKA

Cross-validation: evaluating estimator performance, scikit-learn 0.24.2, https://scikit-learn.org/stable/modules/cross_validation.html

How to Identify Overfitting Machine Learning Models in Scikit-Learn, by Jason Brownlee on November 11, 2020 in Python Machine Learning,, https://machinelearningmastery.com/overfitting-machine-learning-models/

Logistic Regression in Python, by Mirko Stojiljković https://realpython.com/logistic-regression-python/

Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning, By Chris Albon, Published by O'Reilly

Metrics and scoring: quantifying the quality of predictions, scikit-learn 0.24.2, https://scikit-learn.org/stable/modules/model_evaluation.html

scikit-learn: machine learning in Python, Gael Varoquaux, https://scipy- lectures.org/packages/scikit-learn/index.html

Validating Machine Learning Modelswith scikit-learn, Deepika Singh, https://www.pluralsight.com/guides/validating-machine-learning-models-scikit-learn

BAB V
MARKETING ANALYSIS MODEL

A. PENDAHULUAN

Marketing Analyst atau analisis pemasaran adalah profesi yang bertanggung jawab mempelajari kondisi pasar dengan tujuan menilai potensi untuk penjualan produk dan layanan. Tujuan utama proses ini adalah menentukan produk apa yang diminati, di mana pasarnya, serta berapa harga yang pas untuk ditawarkan. Dengan demikian, peran dari profesi ini terbilang sangat penting bagi perusahaan. Oleh karena itu, dalam praktikum Data Science Fundamental ini kita akan membahas bagaimana membangun sebuah model yang dapat digunakan oleh seorang analisis marketing dalam pengambilan keputusan. Umumnya proses segmentasi pelanggan merupakan salah satu pendekatan yang digunakan dengan menerapkan analisis model Recency (R), frequency (F) dan monetary (M).

Analisis RFM merupakan teknik segmentasi populer yang mempelajari perilaku pelanggan berdasarkan data. Skor Recency (R), frequency (F), dan monetary (M) ditetapkan untuk setiap pelanggan secara terpisah. Skor ini mendapat skor dari 1 hingga 5 pada masing-masing kategori RFM, untuk mendapatkan skor RFM kita akan menggabungkan skor RFM berdampingan sebagai skor tunggal. Namun, interpretasi penilaian ini dapat bervariasi sesuai dengan bidang kegiatan perusahaan. Misalnya, jika bisnis Anda adalah bisnis mobil, Anda mengharapkan pelanggan Anda melakukan pembelian dengan harga tinggi, tetapi frekuensi pembelian dan waktu inovasi akan rendah. Oleh karena itu, menafsirkan skor individu dalam kondisi ini memberikan hasil yang lebih akurat.

Dalam lanskap pemasaran modern, analitik yang akurat lebih penting dari sebelumnya. Konsumen menjadi sangat selektif dalam memilih media bermerek yang mereka gunakan dan media yang mereka abaikan. Jika merek ingin menarik perhatian pembeli yang ideal, mereka harus mengandalkan analitik untuk membuat iklan pribadi yang ditargetkan berdasarkan minat individu, bukan asosiasi demografis yang lebih luas. Ini akan memungkinkan tim pemasaran untuk menayangkan iklan yang tepat, pada waktu yang tepat, di saluran yang tepat untuk menggerakkan konsumen ke saluran penjualan.

B. INTI

1. Capaian Pembelajaran
   a. Memahami penerapan model RFM untuk segmentasi pelanggan
   b. Memahami penerapan pustaka Scikit-Lear dalam membangun model pada Python
   c. Mampu membuat aplikasi segmentasi pelanggan dengan algoritma K-Means Clustering
2. Materi
   a. Studi Kasus – Segmentasi Pelanggan Retail
   b. Analisis RFM
   c. RFM Analisis Menggunakan K-Means Clustering
3. Uraian Materi
   a. Studi Kasus–Segmentasi Pelanggan Retail

   Pada bagian ini kita akan mencoba melakukan segmentasi pelanggan pada dataset terbukan Retail Online. Dataset ini merupakan kumpulan data Ritel Online yang memuat semua transaksi yang terjadi untuk ritel online non-toko yang berbasis di Inggris dan terdaftar antara 01/12/2009 dan 09/12/2011. Perusahaan ini menjual perlengkapan hadiah unik untuk semua kesempatan. Banyak pelanggan perusahaan tersebut adalah grosir.

Persiapan

```
In [ ]:
# Import Pustaka yang digunakan
import datetime as dt
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.float_format', lambda x: '%.5f' % x)
```

```
In [ ]:
# Load Dataset (Sesuaikan dengan lokasi/folder data)
df = pd.read_csv("../dataset/online_retail.csv")
print(df.shape)
```

Analisis Deskriptif Data

```
In [ ]:
#memeriksa statistik deskriptif dari dataset
df.head()
```

```
In [ ]:
df.describe()
```

```
In [ ]:
df.describe().T
```

```
In [ ]:
df.columns
```

```
In [ ]:
# Cek Type Data
df.dtypes
```

```
In [ ]:
df.info()
```

Deskripsi Data:

1. Invoice: Nomor faktur. Nominal, nomor integral 6 digit yang ditetapkan secara unik untuk setiap transaksi. Jika Nomor dimulai dengan huruf 'c', maka transaksi tersebut batal.
2. StockCode: Kode produk (barang). Nominal, bilangan integral 5 digit yang ditetapkan secara unik untuk setiap produk yang berbeda.
3. Description: Nama produk (barang). Nominal.
4. Quantity: Jumlah setiap produk (item) per transaksi. (numerik).
5. InvoiceDate: Tanggal dan waktu Transaksi. Numerik, hari dan waktu saat setiap transaksi dibuat.
6. Price: Harga satuan. Numerik, Harga produk per unit dalam sterling.
7. Customer ID: Nomor pelanggan. Nominal, nomor integral 5 digit yang ditetapkan secara unik untuk setiap pelanggan.
8. Country: Nama negara. Nominal, nama negara setiap pelanggan.

Data Cleaning (Pembersihan Data)

Pembersihan data adalah proses mendeteksi dan mengoreksi (atau menghapus) catatan yang rusak atau tidak akurat dari kumpulan catatan, tabel, atau basis data dan mengacu pada pengidentifikasian bagian data yang tidak lengkap, tidak benar, tidak akurat, atau tidak relevan dan kemudian mengganti, memodifikasi, atau menghapus data tersebut.

Handling Missing Values (Menangani Data yang hilang)

In [ ]:
```
# Periksa Data yang hilang
#checking Missing Values
plt.figure(figsize=(5, 5))
df.isnull().mean(axis=0).plot.barh()
plt.title("Rasio nilai yang hilang per kolom")
plt.show()
print('Detail Jumlah Nilai yang Hilang per kolom: ')
df.isna().sum().sort_values(ascending=False)
```

In [ ]:
```
# Cek Jumlah Data yang hilang tiap variabel
df.isnull().sum()
```

In [ ]:
```
# Hapus Data Null pada variabel CustomerID
df = df.dropna(subset=['Customer ID'])
```

In [ ]:
```
# Cek Data
df.info()
```

In [ ]:
```
#Memeriksa Nilai yang Hilang setelah Proses Cleaning
plt.figure(figsize=(5, 5))
df.isnull().mean(axis=0).plot.barh()
plt.title("Rasio nilai yang hilang per kolom")
plt.show()
df.isnull().sum().sort_values(ascending=False)
```

In [ ]:
```
df.describe()
```

In [ ]:
```
# Periksa data Duplikat
df.duplicated().sum()
```

In [ ]:
```
print("Jumlah transaksi sebelum penghapusan duplikat : %d " % df.shape[0])
# Dropping the duplicated transactions
df = df.drop(index=df[df.duplicated()].index)
print("Jumlah transaksi Sesudah penghapusan duplikat  : %d " % df.shape[0])
```

In [ ]:
```
# Periksa transaksi yang batal
df[df['Invoice'].astype(str).str[0] == 'C'].tail()
```

In [ ]:
```
# Periksa Quantity yang bernilai negative berdasarkan Kode Barang
df[(df.Quantity<1)].sort_values(by='StockCode').tail()
```

In [ ]:
```
# Konversi Format Tanggal
print("Format Variabel InvoiceDate adalah: ", df['InvoiceDate'].dtype)
print("*"*100)

df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
print("Format Variabel InvoiceDate : ", df['InvoiceDate'].dtype)
df.head()
```

Feature Engginering (Menambahkan Fitur Baru)

In [ ]:
```
# Menambah Variabel Baru (Years, Months, Days, Hours, dan Date)
df['Years'] = df['InvoiceDate'].dt.year
df['Months'] = df['InvoiceDate'].dt.month
df['Days'] = df['InvoiceDate'].dt.dayofweek+1
df['Hours'] = df['InvoiceDate'].dt.hour
df['Date'] = df['InvoiceDate'].dt.date
df['DayOfMonth'] = df['InvoiceDate'].dt.day

#Pengaturan Ulang DataFrame Dataset
df = df[['Invoice',  'InvoiceDate', 'Date', 'Years', 'Months',
            'Days','Hours','DayOfMonth','StockCode', 'Description',
            'Quantity', 'Price', 'Customer ID', 'Country']]

df.head()
```

In [ ]:
```
# Menghitung nilai Moneter (Quantity*Price)
df.insert(loc = 14, column = 'TotalCost', value = df['Quantity']*df['Price'])
df.head()
```

Simpan Data Hasil Data Cleaning

```
In [ ]: # Simpan Data Hasil Pembersihan
        df.to_csv("retail_data_clean.csv", index=False)
```

```
In [ ]: #Periksa data yang sudah di simpan
        cek_data = pd.read_csv("retail_data_clean.csv")
        cek_data.head(3)
```

Exploratory Data Analysis

Exploratory Data Analysis (EDA) adalah proses eksplorasi data yang bertujuan untuk memahami isi dan komponen penyusun data.Dalam kaitannya dengan data science, EDA dilakukan setelah tahap Preprocessing dan sebelum melakukan proses feature engineering dan modeling.

Dalam Praktikum ini, kita akan coba membahas beberapa pertanyaan berikut ini

1. Bulan Berapa Nilai Transaksi Tertinggi?
2. Hari Apa dalam seminggu yang memiliki Order dan Pendapatan tertinggi?
3. Tanggal berapa transaksi yang paling banyak dilakukan oleh pelanggan?
4. Negara mana pelanggan yang paling banyak melakukan transaksi, dan seperti apa polanya?
5. Bagaimana pengaruh diskon terhadap nilai bisnis Perusahaan?

Persiapan

```
In [ ]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [ ]: # Load data hasil pembersihan
        data = pd.read_csv("retail_data_clean.csv")
        data.head()
```

```
In [ ]: # Tampilkan 10 Produk yang paling laris
        data.groupby("StockCode").agg({"Quantity":"sum"}).sort_values(by="Quantity",ascending=False).head(10)
```

```
In [ ]: # Nama Barang 10 Terlaris
        # Tampilkan 10 Produk yang paling laris
        data.groupby("Description").agg({"Quantity":"sum"}).sort_values(by="Quantity",ascending=False).head(10)
```

```
In [ ]: # Tampilkan 10 Negara yang Transaksi terbesar
        data['Country'].value_counts().head(10)
```

Bulan Berapa Nilai Transaksi Tertinggi?

In [ ]:
```
# Cek Data Transaksi
print("Data Transaksi Dari {} s/d {}".format(data['Date'].unique()[0], data['Date'].unique()[-1]))
```

In [ ]:
```
ord_rev_month = pd.DataFrame({'Order': data.groupby('Invoice')['Months'].unique().value_counts().sort_index(),
                              'Revenue' : data.groupby('Months')['TotalCost'].sum()})
ord_rev_month
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = ord_rev_month['Order'].plot(kind='bar', color='y', label='Orders')
ax2 = ord_rev_month['Revenue'].plot(kind='line', marker='d', secondary_y=True, label = 'Revenue')
ax1.set_xlabel('Month',fontsize=15)
ax1.set_ylabel('Orders',fontsize=15)
ax2.set_ylabel('Revenue',fontsize=15)
ax1.set_title('Pendapatan dari Pesanan Pelanggan Tiap Bulan Berbeda (1 Dec 2010 - 9 Dec 2011)',fontsize=15)
ax1.set_xticklabels(('Dec_10','Jan_11','Feb_11','Mar_11','Apr_11','May_11','Jun_11',
                     'July_11','Aug_11','Sep_11','Oct_11','Nov_11'), rotation='horizontal', fontsize=13)
fig.legend(loc="upper left", bbox_to_anchor=(0.02,0.96), bbox_transform=ax1.transAxes)
plt.show()
```

In [ ]:
```
ord_rev_month.corr()
```

Hari Apa dalam seminggu yang memiliki Order dan Pendapatan tertinggi?

In [ ]:
```
data['Days'].describe()
```

In [ ]:
```
ord_rev_day = pd.DataFrame({'Order': data.groupby('Invoice')['Days'].unique().value_counts().sort_index(),
                            'Revenue' : data.groupby('Days')['TotalCost'].sum()})
ord_rev_day
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = ord_rev_day['Order'].plot(kind='bar', color='y', label='Orders')
ax2 = ord_rev_day['Revenue'].plot(kind='line', marker='d', secondary_y=True, label = 'Revenue')
ax1.set_xlabel('Hari',fontsize=15)
ax1.set_ylabel('Orders',fontsize=15)
ax2.set_ylabel('Revenue',fontsize=15)
ax1.set_title('Pendapatan Berdasarkan Pesanan Pelanggan Dalam Mingguan (1 Dec 2010 - 9 Dec 2011)',fontsize=15)
ax1.set_xticklabels(('Senin','Selasa','Rabu','Kamis','Jumat','Sabtu','Minggu'),
                    rotation='horizontal', fontsize=13)
fig.legend(loc="upper left", bbox_to_anchor=(0.02,0.96), bbox_transform=ax1.transAxes)
plt.show()
```

Tanggal berapa transaksi yang paling banyak dilakukan oleh pelanggan?

In [ ]:
```
ord_rev_date = pd.DataFrame({'Order': data.groupby('Invoice')['DayOfMonth'].unique().value_counts().sort_index(),
                             'Revenue' : data.groupby('DayOfMonth')['TotalCost'].sum()})
ord_rev_date
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = ord_rev_date['Order'].plot(kind='bar', color='y', label='Orders')
ax2 = ord_rev_date['Revenue'].plot(kind='line', marker='d', secondary_y=True, label = 'Revenue')
ax1.set_xlabel('Date',fontsize=15)
ax1.set_ylabel('Orders',fontsize=15)
ax2.set_ylabel('Revenue',fontsize=15)
ax1.set_title('Revenue and Orders For Each Date(1st Dec 2010 - 9th Dec 2011)',fontsize=15)
fig.legend(loc="upper left", bbox_to_anchor=(0.02,0.96), bbox_transform=ax1.transAxes)
plt.show()
```

Negara mana pelanggan yang paling banyak melakukan transaksi, dan seperti apa pola nya?

In [ ]:
```
print("Jumlah Negara : ", data['Country'].nunique())
print("List Negara : ")
n = 1
for i in data['Country'].unique():
    print("{}. {}".format(n, i))
    n += 1
```

In [ ]:
```
df_country = pd.DataFrame({'Order': data.groupby('Invoice')['Country'].unique().value_counts().sort_index(),
                           'Revenue': data.groupby('Country')['TotalCost'].sum()})
df_country.sort_values(by='Order', ascending=False).head()
```

In [ ]:
```
df_country['Percentage'] = (df_country['Revenue']/df_country['Revenue'].sum())*100
df_country.sort_values(by='Percentage', ascending=False).head(10)
```

In [ ]:
```
top_ten = df_country.sort_values(by='Percentage', ascending=False).head(10)

fig, ax1 = plt.subplots(figsize=(10, 6))
top_ten['Percentage'].plot(kind='bar', color='y', label='Orders')
ax1.set_ylabel('Percentage',fontsize=15)
ax1.set_xlabel('Country',fontsize=15)
plt.xticks(rotation=(365-45))
ax1.set_title('10 Negara Teratas Dengan Persentase Pendapatan Tertinggi',fontsize=15)
fig.legend(loc="upper right", bbox_transform=ax1.transAxes)
plt.show()
```

Bagaimana pengaruh diskon terhadap nilai bisnis perusahaan?

In [ ]:
```
months = ['january', 'february', 'march', 'april', 'may', 'june',
          'july', 'august', 'september', 'october', 'november', 'desember']

mon_rev = []
disc_count = []

for i in range(1,13):
    mon_rev.append(data[data['Months'] == i]['TotalCost'].sum())
    disc_count.append(data[(data['StockCode']=='D') &(data['Months'] == i)]['StockCode'].count())

disc_rev = pd.DataFrame({'monthly_revenue': mon_rev, 'num_of_disc': disc_count}, index=months)
disc_rev
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = disc_rev['num_of_disc'].plot(kind='bar', color='y', label='Number of Discounts')
ax2 = disc_rev['monthly_revenue'].plot(kind='line', marker='d', secondary_y=True, label = 'Monthly Revenue')
ax1.set_xlabel('Months',fontsize=15)
ax1.set_ylabel('Number of Discounts',fontsize=15)
ax2.set_ylabel('Monthly Revenue',fontsize=15)
ax1.set_title('Diskon Bulanan vs Pendapatan',fontsize=15)
fig.legend(loc="upper left", bbox_to_anchor=(0.02,0.96), bbox_transform=ax1.transAxes)
plt.show()
```

In [ ]:
```
plt.figure(figsize=(6,6))
sns.lmplot(data = disc_rev, x = 'num_of_disc', y = 'monthly_revenue')
plt.title('Korelasi Antara Jumlah Diskon dan Pendapatan Setiap Bulan')
plt.show()
```

In [ ]:
```
disc_rev.insert(loc=2, column='num_of_order',
                value=pd.Series(data.groupby('Invoice')['Months'].unique().value_counts().sort_index()).values)
disc_rev
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = disc_rev['num_of_disc'].plot(kind='bar', color='y', label='Number of Discounts')
ax2 = disc_rev['num_of_order'].plot(kind='line', marker='d', secondary_y=True, label = 'num_of_order')
ax1.set_xlabel('Months',fontsize=15)
ax1.set_ylabel('Number of Discounts',fontsize=15)
ax2.set_ylabel('num_of_order',fontsize=15)
ax1.set_title('Pengaruh Diskon terhadap Order setiap Bulan',fontsize=15)
fig.legend(loc="upper left", bbox_to_anchor=(0.02,0.96), bbox_transform=ax1.transAxes)
plt.show()
```

In [ ]:
```
plt.figure(figsize=(6,6))
sns.lmplot(data = disc_rev, x = 'num_of_disc', y = 'num_of_order')
plt.title('Korelasi Antara Jumlah Diskon dan Jumlah Pesanan Setiap Bulan')
plt.show()
```

In [ ]:
```
sns.heatmap(disc_rev.corr(method='spearman'), annot=True)
```

In [ ]:
```
disc_rev.corr(method='spearman')
```

Analisis Model RFM

Analisis RFM membantu menjawab pertanyaan berikut: Siapa pelanggan terakhir kami? Berapa kali dia membeli barang dari toko kami? Dan berapa nilai total perdagangannya? Semua informasi ini sangat penting untuk memahami seberapa baik atau buruk pelanggan bagi perusahaan.

Setelah mendapatkan nilai RFM, praktik umum adalah membuat 'kuartil' pada setiap metrik dan menetapkan urutan yang diperlukan. Misalnya, kita membagi setiap metrik menjadi 4 potongan. Untuk metrik keterkinian, nilai tertinggi, 4, akan diberikan kepada pelanggan dengan nilai kebaruan paling rendah (karena mereka adalah pelanggan terbaru). Untuk metrik frekuensi dan moneter, nilai tertinggi, 4, akan diberikan kepada masing-masing pelanggan dengan frekuensi 25% teratas dan nilai moneter. Setelah membagi metrik menjadi kuartil, kita dapat menyusun metrik ke dalam satu kolom (seperti string karakter { seperti '213' }) untuk membuat kelas nilai RFM bagi pelanggan. Selanjutnya kita dapat membagi metrik RFM menjadi lebih sedikit atau lebih banyak tergantung pada kebutuhan.

Untuk lebih memahami analisis RFM pada studi kasus ini, berikut adalah langkah-langkah utama yang kita terapkan contoh Kasus Segmentasi Data Pelanggan Retail Online:

Persiapan Data

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
# Load data
data = pd.read_csv("retail_data_clean.csv")
data.head()
```

In [ ]:
```
# Cek variabel StockCode yang mengandung karakter
data[data['StockCode'].str.contains('^[a-zA-Z]+', regex=True)]['StockCode'].value_counts()
```

In [ ]:
```
data[data['StockCode']=='M']
```

Membuat Recency DataFrames

In [ ]:
```
#kelompokkan pelanggan dan periksa tanggal terakhir pembelian
recency_df = data.groupby(by='Customer ID', as_index=False)['Date'].max()
recency_df.columns = ['Customer ID','LastPurchaseDate']
recency_df.head()
```

In [ ]:
```
now = data['Date'].max()
now = pd.to_datetime(now)
recency_df['LastPurchaseDate'] = pd.to_datetime(recency_df['LastPurchaseDate'])
```

In [ ]:
```
#Hitung Nilai recency (R)
recency_df['Recency'] = recency_df['LastPurchaseDate'].apply(lambda x: (now - x).days)
recency_df.head()
```

In [ ]:
```
recency_df['LastPurchaseDate'].apply(lambda x: (now - x).days)
```

In [ ]:
```
# Hapus variabel LastPurchaseDate karena kita tidak membutuhkan lagi
recency_df = recency_df.drop('LastPurchaseDate', axis=1)
recency_df.head()
```

Membuat Frekuensi DataFrames

```
In [ ]:
# Hapus Data duplicates
data_copy = data
data_copy.drop_duplicates(subset=['Invoice', 'Customer ID'], keep="first", inplace=True)

#Hitung nilai frequency pemesanan
frequency_df = data_copy.groupby(by=['Customer ID'], as_index=False)['Invoice'].count()
frequency_df.columns = ['Customer ID','Frequency']
frequency_df.head()
```

Membuat Moneter DataFrames

```
In [ ]:
monetary_df = data.groupby(by='Customer ID',as_index=False).agg({'TotalCost': 'sum'})
monetary_df.columns = ['Customer ID','Monetary']
monetary_df.head()
```

```
In [ ]:
data[data['Customer ID']==12346]
```

Membuat RFM DataFrames

```
In [ ]:
#gabungkan kerangka data Recency dengan kerangka data Frequency
temp_df = recency_df.merge(frequency_df,on='Customer ID')
temp_df.head()
```

```
In [ ]:
#gabungkan kerangka data Monetery menjadi 3 kolom
rfm_df = temp_df.merge(monetary_df,on='Customer ID')
#Jadikan Customer ID sebagai Indeks
rfm_df.set_index('Customer ID',inplace=True)
rfm_df.head()
```

b. Analisis RFM

Untuk keterkinian (Recency), pelanggan dengan nilai Recency yang lebih rendah lebih berharga dibandingkan dengan Frequency yang lebih tinggi. Misalnya, seseorang yang berbelanja 7 hari yang lalu jauh lebih mungkin menjadi pelanggan daripada seseorang yang terakhir dilihat 100 hari yang lalu. Akibatnya, kami menilai ini secara berbeda dengan label frekuensi dan moneter.

Menghitung Kuantil RFM

```
In [ ]:
quantiles = rfm_df.quantile(q=[0.25,0.5,0.75])
quantiles
```

```
In [ ]:
quantiles.to_dict()
```

Membuat Segmen menggunakan Kuantil

```
In [ ]:
#Membuat Segmentasi Pelanggan Quantiles Method
def RScore(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1
# Arguments (x = value, p = recency, monetary_value, frequency, k = quartiles dict)
def FMScore(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4
```

Membuat Tabel Segmentasi RFM

```
In [ ]:
#Buat Model segmentasi RFM dalam Tabel
rfm_segmentation = rfm_df
rfm_segmentation['R_Quartile'] = rfm_segmentation['Recency'].apply(RScore, args=('Recency',quantiles,))
rfm_segmentation['F_Quartile'] = rfm_segmentation['Frequency'].apply(FMScore, args=('Frequency',quantiles,))
rfm_segmentation['M_Quartile'] = rfm_segmentation['Monetary'].apply(FMScore, args=('Monetary',quantiles,))
```

```
In [ ]:
rfm_segmentation.head()
```

```
In [ ]:
rfm_segmentation['RFMScore'] = rfm_segmentation.R_Quartile.map(str) \
                             + rfm_segmentation.F_Quartile.map(str) \
                             + rfm_segmentation.M_Quartile.map(str)
rfm_segmentation.head()
```

```
In [ ]:
rfm_segmentation[rfm_segmentation['RFMScore']=='444'].sort_values('Monetary', ascending=False).head(10)
```

Menghitung RFMScore dan Menghasilkan Cluster

```
In [ ]:
print("Pelanggan Terbaik: ",len(rfm_segmentation[rfm_segmentation['RFMScore']=='444']))
print('Pelanggan Setia: ',len(rfm_segmentation[rfm_segmentation['F_Quartile']==4]))
print("Pembelanja Tertinggi: ",len(rfm_segmentation[rfm_segmentation['M_Quartile']==4]))
print('Pelanggan Hampir Hilang: ', len(rfm_segmentation[rfm_segmentation['RFMScore']=='244']))
print('Pelanggan Yang Hilang: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='144']))
print('Pelanggan Terendah Hilang: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='111']))
```

```
In [ ]:
rfm_segmentation['RFMScore'].describe()
```

```
In [ ]:
rfm_segmentation['RFMScore'].unique()
```

```
In [ ]:
rfm_segmentation
```

```
In [ ]:
# Simpan Hasil Segmentasi
rfm_segmentation.to_csv("RFM_Quantiles_Method.csv", index=False)
```

c. RFM Analisis Menggunakan K-Means Clustering

```
In [ ]:
# Gunakan Pustaka SkLearn
from sklearn.cluster import KMeans
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
```

Pra-pemrosesan data untuk Clustering

Pada bagian selanjutnya, kita akan menyiapkan data untuk pengelompokan Kmeans pada data Skor RFM. Untuk melakukan ini, kita perlu melakukan praproses data agar dapat memenuhi asumsi kunci dari algoritma Kmeans, yaitu:

1. Variabel harus didistribusikan secara simetris
2. Variabel harus memiliki nilai rata-rata yang serupa
3. Variabel harus memiliki nilai standar deviasi yang sama

In [ ]:
```
rfm_segment_positive = rfm_segmentation[rfm_segmentation['Monetary']>0]
```

In [ ]:
```
# Memeriksa distribusi variabel Recency, Frequency dan Monetary .
plt.figure(figsize=(12,10))

# Plot distribusi Recency
plt.subplot(3, 1, 1); sns.distplot(rfm_segment_positive['Recency'])

# Plot distribusi Frequency
plt.subplot(3, 1, 2); sns.distplot(rfm_segment_positive['Frequency'])

# Plot distribusi Monetary
plt.subplot(3, 1, 3); sns.distplot(rfm_segment_positive['Monetary'])
```

In [ ]:
```
# Memeriksa mean dan varians konstan.
rfm_segment_positive[['Recency','Frequency','Monetary']].describe()
```

Dari uraian di atas, kita dapat melihat bahwa Monetary minimum untuk customer ID tertentu adalah <0. Oleh karena itu, transaksi ini tidak masuk akal dan perlu dihapus.

In [ ]:
```
rfm_segment_positive[rfm_segment_positive['Monetary'] == 0]
```

Fitting Model untuk meningkatkan kualitas prediksi

In [ ]:
```
# Fitting Model
model_KM = KMeans(n_clusters = 4, max_iter = 10000) #tingkatkan iterasi untuk meningkatkan kualitas prediksi
model_KM.fit(rfm_segment_positive[['Frequency', 'Monetary']])
```

In [ ]:
```
model_KM.labels_
```

In [ ]:
```
#Predict Cluster
rfm_segment_positive['cluster'] = model_KM.predict(rfm_segment_positive[['Frequency', 'Monetary']])
rfm_segment_positive.head()
```

In [ ]:
```
rfm_segment_positive['cluster'].describe()
```

Visualisasi Cluster

In [ ]:
```
plt.plot(rfm_segment_positive['Frequency'][rfm_segment_positive['cluster'] == 0],
         rfm_segment_positive['Monetary'][rfm_segment_positive['cluster'] == 0], 'r.')
plt.plot(rfm_segment_positive['Frequency'][rfm_segment_positive['cluster'] == 1],
         rfm_segment_positive['Monetary'][rfm_segment_positive['cluster'] == 1], 'y.')
plt.plot(rfm_segment_positive['Frequency'][rfm_segment_positive['cluster'] == 2],
         rfm_segment_positive['Monetary'][rfm_segment_positive['cluster'] == 2], 'b.')
plt.plot(rfm_segment_positive['Frequency'][rfm_segment_positive['cluster'] == 3],
         rfm_segment_positive['Monetary'][rfm_segment_positive['cluster'] == 3], 'g.')
plt.xlabel('Frequency')
plt.ylabel('Monetary')
```

Penanganan Masalah Outliers Pada Dataset

In [ ]:
```
rfm_segmentation = rfm_segmentation[rfm_segmentation['Monetary'] > 0]
rfm_segmentation.reset_index(drop=True,inplace=True)
```

In [ ]:
```
rfm_segmentation.describe()
```

Mendeteksi Outliers dengan Plot Distribusi

In [ ]:
```
plt.style.use("fivethirtyeight")

plt.figure(1, figsize=(15,6))
n = 0
for x in ['Recency', 'Frequency', 'Monetary']:
    n += 1
    plt.subplot(1, 3, n)
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.distplot(rfm_segmentation[x], bins = 20)
    plt.title('Distplot of {}'.format(x))
plt.show()
```

Distribusi Plot menggunakan boxplot

In [ ]:
```
plt.style.use("fivethirtyeight")

plt.figure(1, figsize=(15,6))
n = 0
for x in ['Recency', 'Frequency', 'Monetary']:
    n += 1
    plt.subplot(3, 1, n)
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.boxplot(rfm_segmentation[x])
    plt.title('Distplot 0f {}'.format(x))
plt.tight_layout()
plt.show()
```

In [ ]:
```
plt.style.use("fivethirtyeight")

plt.figure(1, figsize=(15,6))
n = 0
for x in ['Recency', 'Frequency', 'Monetary']:
    n += 1
    plt.subplot(3, 1, n)
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.boxplot(rfm_segment_positive[x])
    plt.title('Distplot 0f {}'.format(x))
plt.tight_layout()
plt.show()
```

Mendeteksi Outliers Menggunakan Data Rescaling

In [ ]:
```
# Buat DataFrame tanpa Transaksi Minus(-) untuk menghindari Infinite numerik saat menghitung nilai Log
rfm_positive = rfm_segment_positive[rfm_segment_positive['Recency']>0]
```

In [ ]:
```
rfm_positive
```

In [ ]:
```
#Generate Fitur Log pada Nilai Recency, Frequency, and Monetary
cols = ['Recency', 'Frequency', 'Monetary']
for i in cols:
    rfm_positive['Log_of_{}'.format(i)] = [np.log(x) for x in rfm_positive[i]]
```

In [ ]:
```
rfm_positive.info()
```

In [ ]:
```
rfm_positive.isna().sum()
```

Plot Distribusi Recency, Frequency, dan Monetary setelah Penanganan Outliers dengan Data Rescalling

In [ ]:
```
plt.style.use("fivethirtyeight")

plt.figure(1, figsize=(15,6))
n = 0
for x in ['Log_of_Recency', 'Log_of_Frequency', 'Log_of_Monetary']:
    n += 1
    plt.subplot(1, 3, n)
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.distplot(rfm_positive[x], bins = 20)
    plt.title('Distplot 0f {}'.format(x))
plt.show()
```

In [ ]:
```
plt.style.use("fivethirtyeight")

plt.figure(1, figsize=(15,6))
n = 0
for x in ['Log_of_Recency', 'Log_of_Frequency', 'Log_of_Monetary']:
    n += 1
    plt.subplot(3, 1, n)
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.boxplot(rfm_positive[x])
    plt.title('Boxplot 0f {}'.format(x))
plt.tight_layout()
plt.show()
```

Cluster dengan K-Means

In [ ]:
```
k = range(1, 15)
inertia = []
inertia_2 = []
inertia_3 = []

for i in k:
    model = KMeans(n_clusters = i)
    model_2 = KMeans(n_clusters = i)
    model_3 = KMeans(n_clusters = i)
    model.fit(rfm_positive[['Log_of_Recency', 'Log_of_Frequency']])
    model_2.fit(rfm_positive[['Log_of_Recency', 'Log_of_Monetary']])
    model_3.fit(rfm_positive[['Log_of_Monetary', 'Log_of_Frequency']])
    inertia.append(model.inertia_)
    inertia_2.append(model_2.inertia_)
    inertia_3.append(model_3.inertia_)

print(k)
print(inertia)
print(inertia_2)
print(inertia_3)
```

In [ ]:
```
plt.figure(figsize=(8,6))
plt.plot(k, inertia)
plt.xlabel('k value')
plt.ylabel('inertia sum squared error')
plt.plot(k[3], inertia[3], 'ro')
plt.annotate('Best k value = 4', xy=(k[3], inertia[3]),
    xytext = (4, 50), arrowprops = dict(facecolor='black', shrink=0.1)
)
```

In [ ]:
```
plt.figure(figsize=(8,6))
plt.plot(k, inertia)
plt.xlabel('k value')
plt.ylabel('inertia sum squared error')
plt.plot(k[3], inertia_2[8], 'ro')
plt.annotate('Best k value = 3', xy=(k[3], inertia_2[8]),
    xytext = (4, 50), arrowprops = dict(facecolor='black', shrink=0.1)
)
```

In [ ]:
```
plt.figure(figsize=(8,6))
plt.plot(k, inertia)
plt.xlabel('k value')
plt.ylabel('inertia sum squared error')
plt.plot(k[2], inertia_3[2], 'ro')
plt.annotate('Best k value = 3', xy=(k[2], inertia_3[4]),
    xytext = (3, 50), arrowprops = dict(facecolor='black', shrink=0.1)
)
```

In [ ]:
```
df_new = rfm_positive[['Log_of_Recency', 'Log_of_Frequency', 'Log_of_Monetary']]
df_new.head()
```

Tentukan K dengan model Silhouette Score

In [ ]:
```
from sklearn.metrics import silhouette_score

silhouette_scores = []

for n_cluster in range(3,8):
    silhouette_scores.append(
    silhouette_score(df_new, KMeans(n_clusters = n_cluster).fit_predict(df_new)))

#Plotting the silhouette score
k = [3,4,5,6,7]
plt.bar(k, silhouette_scores)
plt.xlabel("Number of Clusters", fontsize=10)
plt.ylabel("Silhouette Score", fontsize=10)
plt.show()
```

In [ ]:
```
# Fitting Model
model_KM = KMeans(n_clusters = 4, max_iter = 10000)#tingkatkan iterasi untuk meningkatkan kualitas prediksi
model_KM_3 = KMeans(n_clusters = 3, max_iter = 10000)

model_KM.fit(rfm_positive[['Log_of_Recency', 'Log_of_Frequency']])
model_KM.fit(rfm_positive[['Log_of_Recency', 'Log_of_Monetary']])
model_KM_3.fit(rfm_positive[['Log_of_Monetary', 'Log_of_Frequency']])
```

In [ ]:
```
#Predict Cluster

rfm_positive['K-Means_RF'] = model_KM.predict(rfm_positive[['Log_of_Recency', 'Log_of_Frequency']])
rfm_positive['K-Means_RM'] = model_KM.predict(rfm_positive[['Log_of_Recency', 'Log_of_Monetary']])
rfm_positive['K-Means_MF'] = model_KM_3.predict(rfm_positive[['Log_of_Monetary', 'Log_of_Frequency']])
rfm_positive.head()
```

In [ ]:
```
rfm_positive[rfm_positive['K-Means_RF']==3]
```

Evaluasi Cluster

Clustering 1 : Monetary vs Recency

```
In [ ]: plt.style.use("fivethirtyeight")
```

```
In [ ]: plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RM'] == 0],
                rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_RM'] == 0], 'g.')
        plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RM'] == 1],
                rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_RM'] == 1], 'b.')
        plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RM'] == 2],
                rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_RM'] == 2], 'r.')
        plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RM'] == 3],
                rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_RM'] == 3], 'y.')
        plt.title("Monetary VS Recency")
        plt.xlabel('Recency')
        plt.ylabel('Monetary')

        #Saving Plot for dashboards
        plt.savefig(r"monetary vs recency.png", bbox_inches='tight', dpi=150)
```

Clustering 2 : Frequency vs Recency

```
In [ ]: plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RF'] == 0],
                rfm_positive['Log_of_Frequency'][rfm_positive['K-Means_RF'] == 0], 'y.')
        plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RF'] == 1],
                rfm_positive['Log_of_Frequency'][rfm_positive['K-Means_RF'] == 1], 'g.')
        plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RF'] == 2],
                rfm_positive['Log_of_Frequency'][rfm_positive['K-Means_RF'] == 2], 'y.')
        plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RF'] == 3],
                rfm_positive['Log_of_Frequency'][rfm_positive['K-Means_RF'] == 3], 'r.')
        plt.title("Frequency VS Recency")
        plt.xlabel('Recency')
        plt.ylabel('Frequency')

        #Saving Plot for dashboards
        plt.savefig(r"frequency vs recency.png",bbox_inches='tight', dpi=150)
```

Clustering 3 : Monetary vs Frequency

```
In [ ]: plt.plot(rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_MF'] == 0],
                rfm_positive['Log_of_Frequency'][rfm_positive['K-Means_MF'] == 0], 'g.')
        plt.plot(rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_MF'] == 1],
                rfm_positive['Log_of_Frequency'][rfm_positive['K-Means_MF'] == 1], 'r.')
        plt.plot(rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_MF'] == 2],
                rfm_positive['Log_of_Frequency'][rfm_positive['K-Means_MF'] == 2], 'b.')
        plt.title("Frequency VS Monetary")
        plt.xlabel('Monetary')
        plt.ylabel('Frequency')

        #Saving Plot for dashboards
        plt.savefig(r"frequency vs monetary.png", bbox_inches='tight', dpi=150)
```

Eksplorasi Cluster

Berapa Pelanggan untuk setiap grup/cluster?

```
In [ ]: cluster_member = pd.DataFrame({"Group 1" : [len(rfm_positive[rfm_positive["K-Means_RM"]==2])],
                    "Group 2" : len(rfm_positive[rfm_positive["K-Means_RM"]==1]),
                    "Group 3" : len(rfm_positive[rfm_positive["K-Means_RM"]==3]),
                    "Group 4" : len(rfm_positive[rfm_positive["K-Means_RM"]==0])}).T
        cluster_member.columns = ['number_of_member']
        cluster_member.sort_values('number_of_member')
```

```
In [ ]: plt.style.use('seaborn-white')

        fig, ax1 = plt.subplots(figsize=(15, 6))
        ax1 = cluster_member.sort_values('number_of_member')['number_of_member'].plot(kind='bar', color='y')
        ax1.set_ylabel('Jumlah Member',fontsize=15)
        ax1.set_title('Jumlah Anggota Cluster untuk Setiap grup',fontsize=15)
        plt.xticks(rotation=(0))
        plt.legend()

        #Saving Plot for dashboards
        plt.savefig(r"group pelanggan.png", bbox_inches='tight', dpi=150)
        plt.show()
```

Bagaimana kontribusi masing-masing cluster/grup terhadap pendapatan perusahaan?

```
In [ ]: rfm_positive.head()
```

```
In [ ]: group_1 = rfm_positive[rfm_positive["K-Means_RM"]==2].index
        group_2 = rfm_positive[rfm_positive["K-Means_RM"]==1].index
        group_3 = rfm_positive[rfm_positive["K-Means_RM"]==3].index
        group_4 = rfm_positive[rfm_positive["K-Means_RM"]==0].index
```

```
In [ ]: list_cluster = []
        for i in data['Customer ID']:
            if i in list(group_1) :
                list_cluster.append('Group 1')
            elif i in list(group_2) :
                list_cluster.append('Group 2')
            elif i in list(group_3) :
                list_cluster.append('Group 3')
            elif i in list(group_4) :
                list_cluster.append('Group 4')
            else:
                list_cluster.append('Outlier')
```

```
In [ ]: data['K-Means_Cluster'] = list_cluster
        data.head()
```

```
In [ ]: data.isna().sum()
```

```
In [ ]: ord_rev_cluster = pd.DataFrame({"Revenue" : data.groupby('K-Means_Cluster')['TotalCost'].sum(),
                                        "Order" :data.groupby('K-Means_Cluster')['Quantity'].sum()})
        ord_rev_cluster.drop(['Outlier'], inplace=True)
        ord_rev_cluster
```

```
In [ ]: # pie plot
        import plotly.express as px

        fig = px.pie(ord_rev_cluster['Revenue'],
                     values=ord_rev_cluster['Revenue'].values,
                     names=ord_rev_cluster['Revenue'].index,
                     title="Clusters Contribution by Revenue",
                     template="seaborn")
        fig.update_traces(rotation=-90, textinfo="percent+label")
        #Saving Plot for dashboards

        fig.show()
```

Berapa stok biasa yang dipesan di setiap grup

```
In [ ]: top_5_1 = data[data['K-Means_Cluster']=='Group 1'].sort_values(by='Quantity', ascending=False).head()['Description']
        top_5_2 = data[data['K-Means_Cluster']=='Group 2'].sort_values(by='Quantity', ascending=False).head()['Description']
        top_5_3 = data[data['K-Means_Cluster']=='Group 3'].sort_values(by='Quantity', ascending=False).head()['Description']
        top_5_4 = data[data['K-Means_Cluster']=='Group 4'].sort_values(by='Quantity', ascending=False).head()['Description']

        top_price_1 = data[data['K-Means_Cluster']=='Group 1'].sort_values(by='Quantity', ascending=False).head(10)['Price']
        top_price_2 = data[data['K-Means_Cluster']=='Group 2'].sort_values(by='Quantity', ascending=False).head(10)['Price']
        top_price_3 = data[data['K-Means_Cluster']=='Group 3'].sort_values(by='Quantity', ascending=False).head(10)['Price']
        top_price_4 = data[data['K-Means_Cluster']=='Group 4'].sort_values(by='Quantity', ascending=False).head(10)['Price']
```

```
In [ ]: temp_stock = pd.DataFrame()
        temp_stock.insert(loc=0, value=top_5_1.values, column='Group 1')
        temp_stock.insert(loc=1, value=top_5_2.values, column='Group 2')
        temp_stock.insert(loc=2, value=top_5_3.values, column='Group 3')
        temp_stock.insert(loc=3, value=top_5_4.values, column='Group 4')
        temp_stock
```

```
In [ ]: price_cluster = pd.DataFrame()
        price_cluster.insert(loc=0, value=top_price_1.values, column='Group 1')
        price_cluster.insert(loc=1, value=top_price_2.values, column='Group 2')
        price_cluster.insert(loc=2, value=top_price_3.values, column='Group 3')
        price_cluster.insert(loc=3, value=top_price_4.values, column='Group 4')
        price_cluster
```

```
In [ ]: plt.style.use('seaborn-white')

        fig, ax1 = plt.subplots(figsize=(12, 8))
        ax1 = price_cluster['Group 3'].plot(kind='bar', color='y')
        ax2 = price_cluster['Group 4'].plot(kind='bar', color='g')
        ax3 = price_cluster['Group 1'].plot(kind='bar', color='b')
        ax4 = price_cluster['Group 2'].plot(kind='bar', color='r')

        ax1.set_ylabel('Nilai Stok',fontsize=15)
        ax1.set_title('10 Nilai Stok Pembelian Teratas untuk setiap Cluster',fontsize=15)

        plt.xticks(rotation=(0))
        plt.legend()
        #Saving Plot for dashboards
        plt.savefig(r"Top 10 Pembelian.png",
                    bbox_inches='tight', dpi=150)

        plt.show()
```

Kapan setiap cluster biasanya melakukan pemesanan?

In [ ]:
```
data[data['K-Means_Cluster']=='Group 1']
```

In [ ]:
```
hours_group = pd.DataFrame()
hours_group1 = pd.DataFrame(data[data['K-Means_Cluster']=='Group 1'].groupby('Hours').sum()['Quantity'])
hours_group2 = pd.DataFrame(data[data['K-Means_Cluster']=='Group 2'].groupby('Hours').sum()['Quantity'])
hours_group3 = pd.DataFrame(data[data['K-Means_Cluster']=='Group 3'].groupby('Hours').sum()['Quantity'])
hours_group4 = pd.DataFrame(data[data['K-Means_Cluster']=='Group 4'].groupby('Hours').sum()['Quantity'])

hours_group1.insert(loc=1, value=hours_group2['Quantity'], column='Group 2')
hours_group1.insert(loc=2, value=hours_group3['Quantity'], column='Group 3')
hours_group1.insert(loc=3, value=hours_group4['Quantity'], column='Group 4')
hours_group1
```

In [ ]:
```
fig = px.pie(hours_group1['Quantity'],
             values=hours_group1['Quantity'].values,
             names=hours_group1['Quantity'].index,
             title="Distribusi Jam Pesanan Dibuat oleh Grup 1 ",
             template="seaborn")
fig.update_traces(rotation=90, textinfo="percent+label")
fig.show()
```

In [ ]:
```
fig = px.pie(hours_group1['Group 2'],
             values=hours_group1['Group 2'].values,
             names=hours_group1['Group 2'].index,
             title="Distribusi Jam Pesanan Dibuat oleh Grup 2",
             template="seaborn")
fig.update_traces(rotation=90, textinfo="percent+label")
fig.show()
```

In [ ]:
```
fig = px.pie(hours_group1['Group 3'],
             values=hours_group1['Group 3'].values,
             names=hours_group1['Group 3'].index,
             title="HDistribusi Jam Pesanan Dibuat oleh Group 3",
             template="seaborn")
fig.update_traces(rotation=90, textinfo="percent+label")
fig.show()
```

In [ ]:
```
fig = px.pie(hours_group1['Group 4'],
             values=hours_group1['Group 4'].values,
             names=hours_group1['Group 4'].index,
             title="Distribusi Jam Pesanan Dibuat oleh Group 4",
             template="seaborn")
fig.update_traces(rotation=90, textinfo="percent+label")
fig.show()
```

In [ ]:
```
date_group1 = pd.DataFrame(data[data['K-Means_Cluster']=='Group 1'].groupby('DayOfMonth').sum()['Quantity'])
date_group2 = pd.DataFrame(data[data['K-Means_Cluster']=='Group 2'].groupby('DayOfMonth').sum()['Quantity'])
date_group3 = pd.DataFrame(data[data['K-Means_Cluster']=='Group 3'].groupby('DayOfMonth').sum()['Quantity'])
date_group4 = pd.DataFrame(data[data['K-Means_Cluster']=='Group 4'].groupby('DayOfMonth').sum()['Quantity'])

date_group1.insert(loc=1, value=date_group2['Quantity'], column='Group 2')
date_group1.insert(loc=2, value=date_group3['Quantity'], column='Group 3')
date_group1.insert(loc=3, value=date_group4['Quantity'], column='Group 4')
date_group1
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = date_group1['Group 3'].plot(kind='line', color='y', label='Group 3')
ax2 = date_group1['Group 4'].plot(kind='line', color='g', label='Group 4')
ax3 = date_group1['Quantity'].plot(kind='line', color='maroon', label='Group 1')
ax4 = date_group1['Group 2'].plot(kind='line', color='lightcoral', label='Group 2')

ax1.set_xlabel('Date',fontsize=15)
ax1.set_ylabel('Orders',fontsize=15)
ax1.set_title('Pesanan Dibuat semua Group Untuk Setiap Tanggal (1 Des 2010 - 9 Des 2011) ',fontsize=15)
plt.legend()
#Saving Plot for dashboards
plt.savefig(r"order all.png",
            bbox_inches='tight', dpi=150)
plt.show()
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))

ax3 = date_group1['Quantity'].plot(kind='line', color='maroon', label='Group 1')
ax4 = date_group1['Group 2'].plot(kind='line', color='lightcoral', label='Group 2')

ax1.set_xlabel('Date',fontsize=15)
ax1.set_ylabel('Orders',fontsize=15)
ax1.set_title('Pesanan Dibuat oleh Grup 1 dan 2 Untuk Setiap Tanggal (1 Des 2010 - 9 Des 2011) ',fontsize=15)
plt.legend()

#Saving Plot for dashboards
plt.savefig(r"order group 1 2.png",
          bbox_inches='tight', dpi=150)
plt.show()
```

Bagaimana setiap cluster bereaksi terhadap diskon?

In [ ]:
```
months = ['january', 'february', 'march', 'april', 'may', 'june',
          'july', 'august', 'september', 'october', 'november', 'desember']

disc_count = []
mon_ord = []
mon_ord_1 = []
mon_ord_2 = []
mon_ord_3 = []
mon_ord_4 = []

for i in range(1,13):
    mon_ord.append(data[data['Months'] == i]['Quantity'].sum())
    mon_ord_1.append(data[(data['Months'] == i)&(data['K-Means_Cluster']=='Group 1')]['Quantity'].sum())
    mon_ord_2.append(data[(data['Months'] == i)&(data['K-Means_Cluster']=='Group 2')]['Quantity'].sum())
    mon_ord_3.append(data[(data['Months'] == i)&(data['K-Means_Cluster']=='Group 3')]['Quantity'].sum())
    mon_ord_4.append(data[(data['Months'] == i)&(data['K-Means_Cluster']=='Group 4')]['Quantity'].sum())
    disc_count.append(data[(data['StockCode']=='D') &(data['Months'] == i)]['StockCode'].count())

disc_ord = pd.DataFrame({'num_of_disc': disc_count, "Group 1": mon_ord_1, "Group 2": mon_ord_2,
                         "Group 3": mon_ord_3, "Group 4": mon_ord_4}, index=months)
disc_ord
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = disc_ord['num_of_disc'].plot(kind='bar', color='y', label='Number of Discounts')
ax2 = disc_ord['Group 1'].plot(kind='line', marker='d', secondary_y=True, label = 'Order Made By Group 1')
ax3 = disc_ord['Group 2'].plot(kind='line', marker='+', secondary_y=True, label = 'Order Made By Group 2')
ax4 = disc_ord['Group 3'].plot(kind='line', marker='x', secondary_y=True, label = 'Order Made By Group 3')
ax5 = disc_ord['Group 4'].plot(kind='line', marker='X', secondary_y=True, label = 'Order Made By Group 4')

ax1.set_xlabel('Months',fontsize=15)
ax1.set_ylabel('Number of Discounts',fontsize=15)
ax2.set_ylabel('Order Made by Group 1',fontsize=15)
ax1.set_title('Diskon Bulanan vs semua Pesanan Grup',fontsize=15)
fig.legend(loc="upper left", bbox_to_anchor=(0.02,0.96), bbox_transform=ax1.transAxes)

#Saving Plot for dashboards
plt.savefig(r"monthly discount.png",
            bbox_inches='tight', dpi=150)
plt.show()
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = disc_ord['num_of_disc'].plot(kind='bar', color='y', label='Number of Discounts')
ax2 = disc_ord['Group 1'].plot(kind='line', marker='+', secondary_y=True, label = 'Order Made By Group 3')
ax3 = disc_ord['Group 2'].plot(kind='line', marker='x', secondary_y=True, label = 'Order Made By Group 4')


ax1.set_xlabel('Months',fontsize=15)
ax1.set_ylabel('Number of Discounts',fontsize=15)
ax2.set_ylabel('Order Made by Group 1',fontsize=15)
ax1.set_title('Diskon Bulanan vs Grup 3 dan 4',fontsize=15)
fig.legend(loc="upper left", bbox_to_anchor=(0.02,0.96), bbox_transform=ax1.transAxes)

#Saving Plot for dashboards
plt.savefig(r"monthly discounts 3&4.png",
            bbox_inches='tight', dpi=150)
plt.show()
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = disc_ord['num_of_disc'].plot(kind='bar', color='y', label='Number of Discounts')
ax2 = disc_ord['Group 3'].plot(kind='line', marker='+', secondary_y=True, label = 'Order Made By Group 2')
ax3 = disc_ord['Group 4'].plot(kind='line', marker='x', secondary_y=True, label = 'Order Made By Group 1')


ax1.set_xlabel('Months',fontsize=15)
ax1.set_ylabel('Number of Discounts',fontsize=15)
ax2.set_ylabel('Order Made',fontsize=15)
ax1.set_title('Diskon Bulanan vs Pesanan Grup 1 dan 2',fontsize=15)
fig.legend(loc="upper left", bbox_to_anchor=(0.02,0.96), bbox_transform=ax1.transAxes)

#Saving Plot for dashboards
plt.savefig(r"order made 1&2.png",
            bbox_inches='tight', dpi=150)
plt.show()
```

In [ ]:
```
disc_corr = disc_ord.corr()
disc_corr = pd.DataFrame(disc_corr.drop('num_of_disc')['num_of_disc'])
disc_corr.sort_values(by='num_of_disc', ascending=False)
```

In [ ]:
```
disc_corr = pd.DataFrame(disc_ord.corr('spearman')['num_of_disc'].sort_values(ascending=False))
disc_corr.drop(['num_of_disc'], inplace=True)

fig, ax1 = plt.subplots(figsize=(15, 6))
ax1 = disc_corr['num_of_disc'].plot(kind='bar', color='y', label='Discount Sentiment Each Cluster')
ax1.set_xlabel('Clusters',fontsize=15)
ax1.set_ylabel('Number of Discounts',fontsize=15)
ax2.set_ylabel('Order Made by Group 1',fontsize=15)
ax1.set_title('Sentimen Diskon Setiap Cluster',fontsize=15)
plt.xticks(rotation=(0))
plt.legend()

#Saving Plot for dashboards
plt.savefig(r"discount sentiment.png",
            bbox_inches='tight', dpi=150)
plt.show()
```

In [ ]:
```
plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RM'] == 0],
         rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_RM'] == 0], 'g.')
plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RM'] == 1],
         rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_RM'] == 1], 'b.')
plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RM'] == 2],
         rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_RM'] == 2], 'r.')
plt.plot(rfm_positive['Log_of_Recency'][rfm_positive['K-Means_RM'] == 3],
         rfm_positive['Log_of_Monetary'][rfm_positive['K-Means_RM'] == 3], 'y.')
plt.title("Monetary VS Recency")
plt.xlabel('Recency')
plt.ylabel('Monetary')
plt.show()
```

Simpan Hasil akhir

```
In [ ]:  data.to_csv("segmentation_cust.csv", index=False)
```

4. Forum Diskusi

   Lanjutan dari pertanyaan yang terdapat pada materi tersebut, hitunglah :

   a. Produk yang paling laris di negara tertentu dan kapan mereka melakukan pesanan?
   b. Produk apa yang paling sering terjadi pembatalan? dari negara mana?
   c. Bulan brp produk harga murah paling banyak di pesan pelanggan?
   d. Bulan brp transaksi pembatalan yang sering dilakukan oleh pelanggan? dan negara mana?

B. PENUTUP

1. Rangkuman

   Secara khusus analisis pemasaran adalah studi tentang data yang dikumpulkan melalui kampanye pemasaran untuk membedakan pola antara hal-hal seperti bagaimana kampanye berkontribusi pada konversi, perilaku konsumen, preferensi regional, preferensi kreatif, dan banyak lagi. Tujuan analitik pemasaran sebagai praktik adalah menggunakan pola dan temuan ini untuk mengoptimalkan kampanye masa depan berdasarkan apa yang berhasil.

2. Tes Formatif

   Sebagai evaluasi pemahaman materi, jawablah pertanyaan berikut ini.

   a. Jelaskan yang dimaksud dengan analisis analisis model Recency (R), frequency (F) dan monetary (M).
   b. Berikut ini merupakan hal-hal yang dilakukan dalam segmentasi pelangggan adalah:
      1. Persiapan
      2. Analisis deskriptif data
      3. Data cleaning
      4. Exploratory data analysis
   c. Beberapa hal yang dilakukan dalam melakukan deskripsi data adalah
      1. Invoice
      2. StockCode
      3. Description

4. Quantity

d. Dalam melakukan data cleaning, beberapa hal yang dapat dilakukan adalah:
    1. Handling Missing Values (Menangani Data yang hilang)
    2. Feature Engginering
    3. Simpan Data Hasil Data Cleaning
    4. Menambahkan Fitur Baru
e. Apa yang dimaksud dengan Exploratory Data Analysis (EDA)?
f. Dalam melakukan evaluasi cluster, berikut hal-hal yang terdapat pada evaluasi tersebut adalah:
    1. Clustering 1 : Monetary vs Recency
    2. Clustering 2 : Frequency vs Recency
    3. Clustering 3 : Monetary vs Frequency
    4. Clustering 3 : Monetary vs Recency
g. Untuk menyiapkan data untuk pengelompokan Kmeans pada data Skor RFM, kita perlu melakukan praproses data agar dapat memenuhi asumsi kunci dari algoritma Kmeans, yaitu:
    1. Variabel harus didistribusikan secara simetris
    2. Variabel harus memiliki nilai rata-rata yang serupa
    3. harus memiliki nilai standar deviasi yang sama
    4. Variabel harus memiliki nilai standar deviasi yang beda
h. Proses mendeteksi dan mengoreksi (atau menghapus) catatan yang rusak atau tidak akurat dari kumpulan catatan, tabel, atau basis data dan mengacu pada pengidentifikasian bagian data yang tidak lengkap, tidak benar, tidak akurat, atau tidak relevan dan kemudian mengganti, memodifikasi, atau menghapus data tersebut, hal tersebut disebut dengan apa?
i. Umumnya proses segmentasi pelanggan merupakan salah satu pendekatan yang digunakan dengan menerapkan analisis model Recency (R), frequency (F) dan monetary (M).
    1. True
    2. False
j. Untuk keterkinian (Recency), pelanggan dengan nilai Recency yang lebih rendah

lebih berharga dibandingkan dengan Frequency yang lebih tinggi.

1. True
2. False

DAFTAR PUSTAKA

Belajar Machine Learning Dengan Library Python : Scikit-Learn, DOLab Data Sceince, https://www.dqlab.id/belajar-machine-learning-dengan-library-python-scikit-learn

Data Science For Marketing Analytics : Achieve Your Marketing Goals With The Data Analytics Power Of Phyton, by Tommy Blanchard

Exploratory data analysis in Python, towards data science, https://towardsdatascience.com/exploratory-data-analysis-in-python-c9a77dfa39ce

Gustriansyah, R., Suhandi, N. and Antony, F., 2020. Clustering optimization in RFM analysis based on k-means. Indones. J. Electr. Eng. Comput. Sci, 18(1), pp.470-477.

Huang, Y., Zhang, M. and He, Y., 2020, June. Research on improved RFM customer segmentation model based on K-Means algorithm. In 2020 5th International Conference on Computational Intelligence and Applications (ICCIA) (pp. 24-27). IEEE.

Kabaskal, İ., 2020. Customer Segmentation Based On Recency Frequency Monetary Model: A Case Study in E-Retailing. International Journal of InformaticsTechnologies, 13(1).

Marketing Data Science: Modeling Techniques in Predictive Analytics with R and Python (FT Press Analytics), by Thomas Miller

Miglautsch, J., 2002. Application of RFM principles: What to do with 1–1–1 customers?. Journal of Database Marketing & Customer Strategy Management, 9(4), pp.319-324.

Unsupervised Learning : K-means Clustering using Python (Case : Online Retail Dataset), https://nzlul.medium.com/unsupervised-learning-k-means-clustering-using-python-case-online-retail-dataset-df7d18599a52

BAB VI
PREDICTIVE ANALYTICS

A. PENDAHULUAN

Predictive Analytics atau analitik prediktif adala bentuk analisis lanjutan yang menggunakan data baru dan data historis untuk memperkirakan aktivitas, perilaku, dan tren. Pengertian lain predictive analytics adalah metode statistik yang menggunakan algoritma dan machine learning untuk mengidentifikasi trend dalam data dan memprediksi perilaku di masa depan. Model ini merupakan cabang dari advanced analytics yang pada proses kerjanya menggunakan prediksi guna mengetahui kejadian di masa yang akan datang. Predictive analytics juga menggunakan banyak teknik mulai dari data mining, statistik, machine learning untuk menganalisis data sekarang dan membuat prediksi nya di masa mendatang.

Beberapa manfaat analitik prediktif dalam bisnis diantaranya mengoptimalkan strategi marketing dari perusahaan. Bahkan metode ini dapat digunakan untuk menebak perilaku pelanggan dan juga dapat mempromosikan produk kepada pelanggan yang membutuhkan. Banyak juga perusahaan yang menggunakan metode ini untuk sekadar ingin meningkatkan pengoperasian serta mengorganisir ketersediaan bahan baku. Industri medis yang akhir-akhir ini juga sedang menghadapi masalah tentang biaya operasi. Dalam hal ini, predictive analytics berfungsi untuk membantu mendiagnosis kondisi pasien secara benar, bahkan menentukan kemungkinan hasil perawatan yang tepat untuk orang dengan kondisi keuangan maupun kondisi kesehatan tertentu.

Sebagai contoh, prediksi tentang kesehatan pasien ini juga dapat membantu manager rumah sakit untuk membuat shift kerja bagi karyawan, agar nantinya saat kunjungan pasien membludak jumlah karyawan yang ada juga memadai. Sehingga diharapkan tidak akan terburu-buru dalam menangani pasien lainnya.

B. INTI

1. Capaian Pembelajaran
   a. Memahami manfaat analitik prediktif dalam dunia bisnis
   b. Memahami proses penerapan model analitik prediktif
   c. Mampu membangun model analitik preditif dengan python
2. Materi
   a. Persiapan Data
   b. Analisis Data Awal
   c. Pemodelan Machine Learning
   d. Peningkatan (Tuning) Machine Learning
3. Uraian Materi
   a. Persiapan Data

      Tahapan awal yang dilakukan adalam memasukkan pustaka untuk memodelkan data; kemudian menentukan dataframe yang akan di prediksi dengan menggunakan pustaka pandas. Untuk pemodelan data menggunakan pustaka numpy, sedangkan visualisasi data kita menggunakan matplotlib dan pustaka seaborn.

```
In [ ]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
%matplotlib inline
```

```
In [ ]:
df = pd.read_csv('retail_data_clean.csv')
```

   b. Analisis Data Awal

      Pada tahapan ini kita perlu memeriksa data yang digunakan termasuk evaluasi data yang hilang ataupun duplikat.

```
In [ ]:
df.head()
```

```
In [ ]:
df.shape
```

```
In [ ]:
# Periksa data
df.isnull().any()
df.isnull().sum()
df.isna().sum()
```

   c. Pemodelan Machine Learning

```
In [ ]:
#Import Model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

Pada studi kasus ini kita akan menggunakan 4 (empat) model

Pembagian Data

Pembagian data sangat sering ditemukan dalam pembelajaran mesin. Dengan melakukan pembagian data, kita dapat menemukan hyperparameter terbaik untuk model yang digunakan. Dengan kata lain, pembagian data dapat membantu mengurangi risiko underfitting maupun overfitting yang seringkali ditemui oleh nilai hyperparameter yang kurang baik. Selain itu, pembagian data juga dapat digunakan sebagai indikator baik atau buruknya kemampuan model prediktif untuk menyelesaikan suatu masalah tertentu. Secara umum data dapat dibagi menjadi 3 bagian, yaitu data latih (train data), data validasi (validation data), dan data uji (test data):

1. Data latih atau biasa disebut dengan training data merupakan bagian data yang digunakan saat melatih model.
2. Data validasi (validation data) adalah data yang digunakan untuk mengevaluasi model yang telah dilatih dengan training data. Dengan menguji model yang telah dilatih menggunakan validation data, performa model-model dengan hyperparameter berbeda akan dibandingkan dan diketahui hyperparameter mana yang menyebabkan underfitting, overfitting, dan yang menghasilkan fit model terbaik.
3. Data uji atau yang biasa disebut dengan testing data merupakan data yang digunakan untuk menguji performa suatu algoritme pembelajaran mesin satu dengan yang lainnya (misalkan membandingkan performa algoritme Logistic Regression dengan algoritme

Support Vector Machine dalam klasifikasi digit MNIST). Namun dalam beberapa literatur, data validasi sering dianggap sama dengan data uji.

Penggunaan pembagian data menjadi kewajiban dalam menangani data yang jumlahnya besar. Hal ini dikarenakan jika data yang digunakan sedikit, model akan dilatih pada training data yang lebih sedikit juga dan menyebabkan tidak terlatihnya model dengan baik. Selain itu validation data yang sedikit juga menyebabkan generalisasi model menjadi terlalu “optimis” atau dikatakan memiliki bias yang tinggi. Secara umum, persentase pembagian data untuk training data:validation data

yang sering digunakan adalah 66.6%:33.3%, 75%:25%, dan 80%:20%.

```
In [ ]:  #X, y, and train_test_split
         X = df[['Price','TotalCost','Customer ID']] # feature
         y = df['Quantity']

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.70)
```

```
In [ ]:  #Average quantity
         df['Quantity'].mean()
```

Dimana **X** merupakan fitur input, variabel **y** merupakan label output, dan **train_size** merupakan berapa persen dari dataset yang akan digunakan menjadi training data. Dalam studi ini kita memasukkan nilai **0.7** yang artinya **70%** data digunakan untuk training dan sisanya **30%** untuk testing. **X_train** dan **y_train** merupakan pasangan fitur input dan label output dari training data, sementara **X_test** dan **y_test** merupakan pasangan fitur input dan label output dari validation data / testing data.

Linear Regression

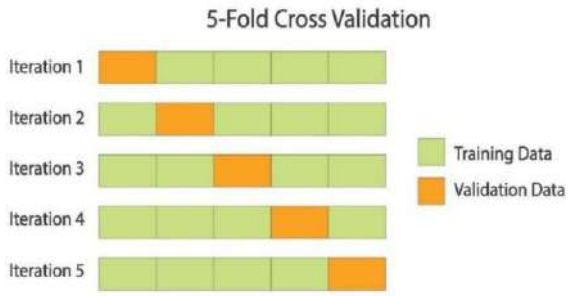
Logistic Regression adalah algoritma regresi yang digunakan untuk menyelesaikan masalah klasifikasi. Algoritma merupakan pengembangan dari algoritma Linear Regression dan bekerja dengan mencari Logistic Function (fungsi logistik) yang dapat membagi data menjadi 2 kelas dengan baik.

```
In [ ]:  linreg = LinearRegression()

         linreg.fit(X_train, y_train)
         pred_reg = linreg.predict(X_test)
         score_linreg_per = linreg.score(X_test, y_test) * 100
         print(f'Linear Regression Model Score: {score_linreg_per}%')
```

MAE (Mean Absolute Error) adalah rata-rata selisih mutlak nilai sebenarnya (aktual) dengan nilai prediksi (peramalan).MAE digunakan untuk mengukur keakuratan suatu model statistik dalam melakukan prediksi atau peramalan.

```
In [ ]:  # Mean Absolute Error (MAE)
         val_mae_linreg = mean_absolute_error(pred_reg, y_test)
         print(f'The quantity is off by: {val_mae_linreg} (MAE)') # Berapa jumlah prediksi
```

Cross Validation atau disebut juga K-Fold Cross Validation merupakan salah satu model yang digunakan untuk melatih dan mengevaluasi model pada sejumlah K kombinasi training data dan testing data yang berbeda. Dengan demikian, model dilatih dengan data yang tidak terlalu besar akan memiliki kemampuan generalisasi yang lebih baik dibandingkan dengan menggunakan Train-Validation Split (Ingat bahwa Train-Validation Split tidak baik jika digunakan pada data yang tidak terlalu besar). Ilustrasi sederhana dari K-Fold Cross Validation disajikan pada Gambar dibawah ini.



Gambar 21. 5 – Fold Cross Validation

```
In [ ]:
#Cross validation
cv_results_linreg = cross_val_score(linreg, X, y, cv=5)
cv_results_linreg
np.mean(cv_results_linreg)
```

Pada potongan kode di atas, X adalah fitur input, y adalah output label, cv adalah nilai K untuk K-Fold Cross Validation, dan model yang digunakan adalah Logistic Regression. Dengan menggunakan potongan kode di atas, proses pembagian data untuk setiap iterasi K dilakukan otomatis, sehingga tidak perlu repot untuk membagi data dengan index tertentu. Hasil akurasi untuk setiap iterasi (dalam potongan kode sebanyak 5 iterasi) disimpan di dalam list scores. Fungsi **np.mean(cv_results_linreg)** dipanggil untuk mengetahui rerata akurasi dari setiap iterasi.

Decission Tree Regressor

Decision Tree (Pohon Keputusan) merupakan salah satu algoritma pembelajaran mesin yang mengklasifikasi dengan mengambil suatu keputusan antara benar atau tidaknya suatu aturan. Kelebihan dari algoritma Decision Tree adalah input yang digunakan boleh berupa tipe data apapun (String, Integer, Float, Boolean, dll).

```
In [ ]:
def get_mae(max_leaf_nodes, X_train, X_test, y_train, y_test):
    model = DecisionTreeRegressor(max_leaf_nodes=max_leaf_nodes, random_state=0)
    model.fit(X_train, y_train)
    preds_val = model.predict(X_test)
    mae = mean_absolute_error(y_test, preds_val)
    return(mae)

candidate_max_leaf_nodes = [5, 25, 50, 100, 250, 500]

for max_leaf_nodes in candidate_max_leaf_nodes:
    my_mae = get_mae(max_leaf_nodes, X_train, X_test, y_train, y_test)
    print("Max leaf nodes: %d  \t\t Mean Absolute Error:  %d" %(max_leaf_nodes, my_mae))
```

```
In [ ]:
#Edit sesuai berdasarkan simpul terbaik

dtr = DecisionTreeRegressor(max_leaf_nodes=500) # <- Jumlah default tidak terbatas
dtr.fit(X_train, y_train)
pred_dtr = dtr.predict(X_test)
score_dtr_per = dtr.score(X_test, y_test) * 100
print(f'Decision Tree Regressor Model Score: {score_dtr_per}%')
```

```
In [ ]:
# Periksa hasil MEA
val_mae_dtr = mean_absolute_error(pred_dtr, y_test)
print(f'The quantity is off by: {val_mae_dtr} (MAE)')
```

```
In [ ]:
#Cross validation
cv_results_dtr = cross_val_score(dtr, X, y, cv=5)
cv_results_dtr
np.mean(cv_results_dtr)
```

Random Forest Regressor

Random forest (RF) adalah suatu algoritma yang digunakan pada klasifikasi data dalam jumlah yang besar. Klasifikasi random forest dilakukan melalui penggabungan pohon (tree) dengan melakukan training pada sampel data yang dimiliki. Penggunaan pohon (tree) yang semakin banyak akan mempengaruhi akurasi yang akan didapatkan menjadi lebih baik. Penentuan klasifikasi dengan random forest diambil berdasarkan hasil voting dari tree yang terbentuk. Pemenang dari tree yang terbentuk ditentukan dengan vote terbanyak.

Pembangunan pohon (tree) pada random forest sampai dengan mencapai ukuran maksimum dari pohon data. Akan tetapi,pembangunan pohon random forest tidak dilakukan pemangkasan (pruning) yang merupakan sebuah metode untuk mengurangi kompleksitas ruang. Pembangunan dilakukan dengan penerapan metode random feature

Selection untuk meminimalisir kesalahan. Pembentukan pohon (tree) dengan sample data menggunakan variable yang diambil secara acak dan menjalankan klasifikasi pada semua tree yang terbentuk. Random forest menggunakan Decision Tree untuk melakukan proses seleksi. Pohon yang dibangun dibagi secara rekursif dari data pada kelas yang sama.

Random forest merupakan salah satu cara penerapan dari pendekatan diskriminasi stokastik pada klasifikasi. Proses Klasifikasi akan berjalan jika semua tree telah terbentuk.Pada saat proses klasifikasi selesai dilakukan, inisialisasi dilakukan dengan sebanyak data berdasarkan nilai akurasinya. Keuntungan penggunaan random forest yaitu mampu mengklasifiksi data yang memiliki atribut yang tidak lengkap,dapat digunakan untuk klasifikasi dan regresi akan tetapi tidak terlalu bagus untuk regresi, lebih cocok untuk pengklasifikasian data serta dapat digunakan untuk menangani data sampel yang banyak. Proses klasifikasi pada random forest berawal dari memecah data sampel yang ada kedalam decision tree secara acak. Setelah pohon terbentuk,maka akan dilakukan voting pada setiap kelas dari data sampel. Kemudian, mengkombinasikan vote dari setiap kelas kemudian diambil vote yang paling banyak.Dengan menggunakan random forest pada klasifikasi data maka, akan menghasilkan vote yang paling baik.

```
In [ ]: rfg = RandomForestRegressor()

        rfg.fit(X_train, y_train)
        pred_rfg = rfg.predict(X_test)
        score_rfg_per = rfg.score(X_test, y_test) * 100
        print(f'Random Forest Regressor Model Score: {score_rfg_per}%')
```

```
In [ ]: #MAE: Random Forest
        val_mae_rfg = mean_absolute_error(pred_rfg, y_test)
        print(f'The quantity is off by: {val_mae_rfg} (MAE)')
```

```
In [ ]: #Cross validation: Random Forest
        cv_results_rfg = cross_val_score(rfg, X, y, cv=5)
        cv_results_rfg
        np.mean(cv_results_rfg)
```

Boosting Algorithm

Boosting Algorithm merupakan salah satu strategi dari ensemble untuk mengubah suatu weak model menjadi strong model. Konsep dari Boosting Algorithm adalah mengkombinasikan hasil prediksi dari setiap weak model untuk menjadi strong model dengan metode Average/Weighted Average dan Higher voting. Gradient boosting adalah algoritma machine learning yang menggunakan ensamble dari decision tree untuk memprediksi nilai. Gradient boosting mampu menangani complex pattern dan data ketika linear model tidak dapat menangani.

```
In [ ]: GBR = GradientBoostingRegressor(n_estimators= 400, max_depth=4, min_samples_split=2,
                learning_rate=0.1, loss='ls')

        GBR.fit(X_train, y_train)
        yPredict = pd.Series(GBR.predict(X_test))
        GBR.score(X_test, y_test)
```

```
In [ ]: # MEA
        err = mean_absolute_error(y_test, yPredict)
        print('{:.3f}'.format(err))
```

```
In [ ]: #Cross validation: Gradient Boosting Regressor
        cv_results_GBR = cross_val_score(GBR, X, y, cv=5)
        cv_results_GBR
        np.mean(cv_results_GBR)
```

d. Peningkatan (Tuning) Machine Learning

Pada bagian ini kita akan membangun model machinle learning dengan menggunakan proses tuning atau sering juga disebut optimasi model machine learning.

GridSearchCV

Grid Search adalah metode yang efektif untuk menyesuaikan parameter dalam supervised learning dan meningkatkan performa generalisasi model. Dengan Pencarian Kisi, kita akan mencoba semua kemungkinan kombinasi parameter yang diinginkan dan menemukan yang terbaik.

```
In [ ]: from sklearn.metrics import r2_score
        from sklearn.metrics import mean_squared_error
        from sklearn.model_selection import KFold
        from sklearn.model_selection import GridSearchCV
```

Untuk mengimplementasikan algoritma Grid Search kita perlu mengimpor GridSearchCVclass dari sklearn.model_selectionlibrary. Selanjutnya kita membangun

model machine learning, perhatikan kode berikut ini.

In [ ]:
```
# Linear Regression
fit_intercepts = [True, False]
param_grid_linear = dict(fit_intercept=fit_intercepts)
linear_model = LinearRegression()
```

In [ ]:
```
# Decision Tree
min_tree_splits = range(2,3)
min_tree_leaves = range(2,3)
param_grid_tree = dict(min_samples_split=min_tree_splits,
                       min_samples_leaf=min_tree_leaves)
tree_model = DecisionTreeRegressor()
```

In [ ]:
```
# Random Forest
estimators_space = [100]
min_sample_splits = range(2,4)
min_sample_leaves = range(2,3)
param_grid_forest = dict(min_samples_split=min_sample_splits,
                         min_samples_leaf=min_sample_leaves,
                         n_estimators=estimators_space)
forest_model = RandomForestRegressor()
```

In [ ]:
```
cv = 5
models_to_test = ['LinearRegression','DecisionTreeRegressor','RandomForest']
regression_dict = dict(LinearRegression=linear_model,
                       DecisionTreeRegressor=tree_model,
                       RandomForest=forest_model)
param_grid_dict = dict(LinearRegression=param_grid_linear,
                       DecisionTreeRegressor=param_grid_tree,
                       RandomForest=param_grid_forest)
```

In [ ]:
```
score_dict = {}
params_dict = {}
mae_dict = {}
mse_dict = {}
r2_dict = {}
best_est_dict = {}
```

In [ ]:
```
for model in models_to_test:
    regressor = GridSearchCV(regression_dict[model], param_grid_dict[model], cv=cv, n_jobs=-1)
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)

  # Cetak hasil tuned dan nilai akurasi
    print(" === Mulai Prediksi Model {} ===".format(model))
    score_dict[model] = regressor.best_score_
    print("Tuned Parameters: {}".format(regressor.best_params_))
    params_dict = regressor.best_params_
    print("Best score is {}".format(regressor.best_score_))

  # Evaluasi Hasil
    mae_dict[model] = mean_absolute_error(y_test, y_pred)
    print("MAE for {}".format(model))
    print(mean_absolute_error(y_test, y_pred))
    mse_dict[model] = mean_squared_error(y_test, y_pred)
    print("MSE for {}".format(model))
    print(mean_squared_error(y_test, y_pred))
    r2_dict[model] = r2_score(y_test, y_pred)
    print("R2 score for {}".format(model))
    print(r2_score(y_test, y_pred))
    print(" === Akhir Prediksi Model {} === \n".format(model))

  # Add best estimator to the dict
    best_est_dict[model] = regressor.best_estimator_
```

In [ ]:
```
# Tampilkan hasil dalam bentuk grafik
summary_cols = ['Best Score']
summary = pd.DataFrame.from_dict(r2_dict, orient='index')
summary.index.name = 'Regressor'
summary.columns = summary_cols
summary = summary.reset_index()

# Visualizing results
plt.figure(figsize=(12,4))
plt.xlabel('Best score')
plt.title('Regressor Comparison')

sns.barplot(x='Best Score', y='Regressor', data=summary)
```

4. Forum Diskusi

   Berdasarkan praktikum yang dipaparkan, terdapat diskusi lanjutan sebagai berikut.

   1. Lakukanlah Analisis Outlier
   2. Eksperimen Variabel yang digunakan sebagai input dan output

3. Tambahkan visualisasi terhadap hasil akurasi metode
4. Kombinasi dengan praktikum analisis marketing

B. PENUTUP

1. Rangkuman

   Beberapa manfaat analitik prediktif dalam bisnis diantaranya mengoptimalkan strategi marketing dari perusahaan. Bahkan metode ini dapat digunakan untuk menebak perilaku pelanggan dan juga dapat mempromosikan produk kepada pelanggan yang membutuhkan. Banyak juga perusahaan yang menggunakan metode ini untuk sekadar ingin meningkatkan pengoperasian serta mengorganisir ketersediaan bahan baku. Industri medis yang akhir- akhir ini juga sedang menghadapi masalah tentang biaya operasi. Dalam hal ini, predictive analytics berfungsi untuk membantu mendiagnosis kondisi pasien secara benar, bahkan menentukan kemungkinan hasil perawatan yang tepat untuk orang dengan kondisi keuangan maupun kondisi kesehatan tertentu.

2. Tes Formatif

   Sebagai evaluasi pemahaman materi, jawablah pertanyaan berikut ini.

   a. Pada model prediksi data, secara umum data dapat dibagi menjadi 3 bagian, yaitu :
      1. data latih (train data)
      2. data validasi (validation data)
      3. data uji (test data)
      4. data normal

   b. Data yang digunakan untuk mengevaluasi model yang telah dilatih dengan training data, disebut dengan apa?
      1. data latih (train data)
      2. data validasi (validation data)
      3. data uji (test data)
      4. data normal

   c. Pada tahap analisis data awal kita perlu memeriksa data yang digunakan termasuk ..... yang hilang ataupun duplikat.
      1. evaluasi data
      2. data duplikat

3. training data
4. testing data

d. Rata-rata selisih mutlak nilai sebenarnya (aktual) dengan nilai prediksi (peramalan), disebut dengan apa?
    1. MAE (Mean Absolute Error)
    2. Logistic Regression
    3. Cross Validation
    4. K-Fold Cross Validation

e. Tahapan awal yang dilakukan adalam memasukkan pustaka untuk memodelkan data; kemudian menentukan dataframe yang akan di prediksi dengan menggunakan pustaka pandas. Untuk pemodelan data menggunakan apa?
    1. Pustaka numpy
    2. Matplotlib
    3. Visualisasi data
    4. Pustaka seaborn

f. Algoritma regresi yang digunakan untuk menyelesaikan masalah klasifikasi, disebut dengan apa?

g. Dengan menguji model yang telah dilatih menggunakan validation data, performa model- model dengan hyperparameter yang sama akan dibandingkan dan diketahui hyperparameter mana yang menyebabkan underfitting, overfitting, dan yang menghasilkan fit model terbaik.
    1. True
    2. False

h. Decision Tree (Pohon Keputusan) merupakan salah satu algoritma pembelajaran mesin yang mengklasifikasi dengan mengambil suatu keputusan antara benar atau tidaknya suatu aturan.
    1. True
    2. False

i. Pembentukan pohon (tree) dengan sample data menggunakan variable yang diambil secara acak dan menjalankan klasifikasi pada semua tree yang

terbentuk.

1. True
2. False

j. Predictive Analytics atau analitik prediktif adala bentuk analisis lanjutan yang menggunakan data lama dan data historis untuk memperkirakan aktivitas, perilaku, dan tren.

1. True
2. False

DAFTAR PUSTAKA

Hands-On Predictive Analytics with Python, 2018. By Alvaro Fuentes. Learning Predictive Analytics with Python, 2016. by Ashish Kumar.

Marketing Data Science : Modeling Techniques in Predictive Analytics with R and Python, 2015.

By Thomas Miller.

Mastering Predictive Analytics with Python, 2016. By Joseph Babcock.

Modeling Techniques in Predictive Analytics with Python and R : A Guide To Data Science, 2014.

By Thomas W. Miller.