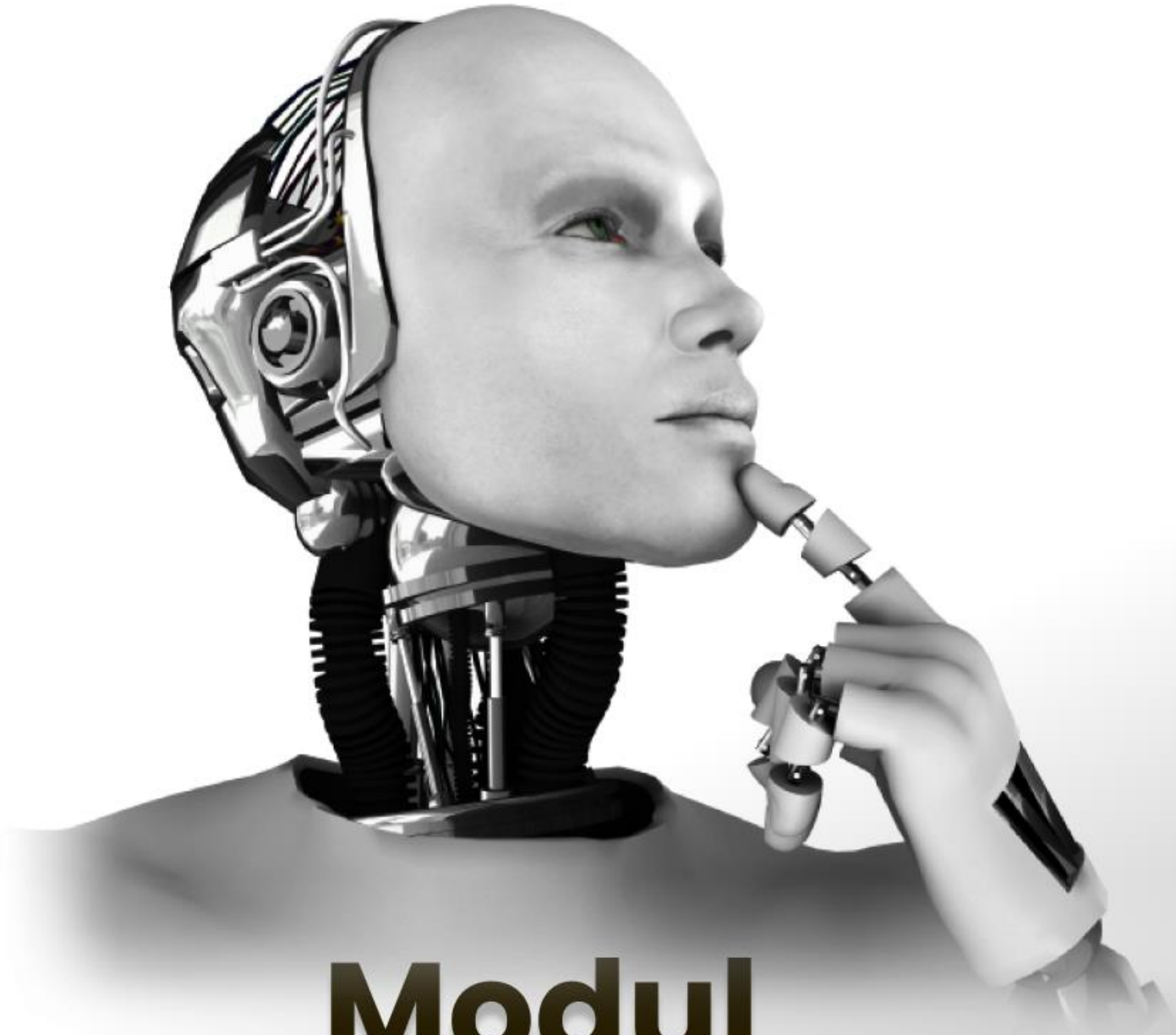


Modul Pelatihan

Machine Learning

ECS Laboratory

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
TAHAP PERSIAPAN

1.1 Istilah Umum

Module, modul di Python adalah satu file yang berisi kumpulan kode, mencakup fungsi, kelas, dan variabel. Ini bertindak sebagai blok bangunan mendasar untuk mengatur kode dan meningkatkan kemudahan pemeliharaan.

Package, paket adalah direktori yang berisi beberapa modul Python, sering kali disusun secara hierarki. Contoh hierarki di bawah ini.

```
my_package/
|-- __init__.py
|-- module1.py
|-- module2.py
```

Library: sebenarnya tidak ada definisi khusus secara teknikal mengenai library di dokumentasi python. Tapi penyebutan ini populer dan sangat sering merujuk pada hal yang sama dengan package, misal: numpy library atau numpy package. Pada banyak forum, definisi library adalah kumpulan kode yang ditulis untuk menyelesaikan masalah tertentu atau menyediakan fungsionalitas. Ini seperti "alat" yang siap pakai. Jadi lebih fokus pada fungsionalitas. Sedangkan package istilah yang lebih fokus pada struktur.

Framework: kerangka kerja, menentukan keseluruhan struktur dan aliran aplikasi. Mereka hadir dengan seperangkat aturan, konvensi, dan alat, menyederhanakan proses pengembangan dan memastikan pendekatan standar. Framework bisa dianalogikan seperti kerangka rumah yang sudah jadi, sehingga kamu tidak perlu membangun dari awal (*from scratch*).

1.2 Instalasi Python dan VSCode

Visual Studio Code adalah perangkat lunak penyunting kode-sumber atau text editor buatan Microsoft untuk Linux, macOS, dan Windows. VSCode adalah text editor paling populer sekarang.

Instalasi Python:

1. Cek terlebih dahulu apakah kamu sudah menginstal python sebelumnya dengan cara buka cmd dan ketik “python --version”
2. Jika muncul versi python maka anda sudah siap
3. Jika belum silahkan kunjungi situs resmi Python di https://www.python.org/downloads/
4. Pilih versi Python yang ingin Anda instal (disarankan untuk memilih versi terbaru yang stabil, seperti Python 3.11.6).
5. Klik tombol Download untuk mendapatkan file instalasi.
6. Setelah unduhan selesai, buka file .exe yang telah diunduh.
7. Di layar instalasi, pastikan untuk mencentang opsi "Add Python to PATH".
8. Pilih "Install Now" untuk instalasi standar atau "Customize installation" jika Anda ingin menyesuaikan pengaturan instalasi.
9. Cek lagi di cmd apakah python sudah terinstal dengan cara membuka cmd dan ketik “python --version”

Instalasi Visual Studio Code:

1. Unduh penginstal Visual Studio Code untuk Windows pada situs berikut https://go.microsoft.com/fwlink/?LinkID=534107
2. Setelah diunduh, jalankan penginstal (VSCodeUserSetup-{version}.exe). Ini hanya akan memakan waktu satu menit.
3. Secara default, VS Code diinstal di C:\Users\{Username}\AppData\Local\Programs\Microsoft VS Code.
4. Lihat dokumentasi VS Code https://code.visualstudio.com/docs untuk tata cara penggunaan yang lengkap

1.3 Virtual Environment Python

Pengetahuan akan environment adalah hal yang penting dalam membuat proyek *Machine learning*. Memiliki banyak proyek dengan kebutuhan library berbeda-beda bisa menjadi hal yang sangat rumit. Bagaimana jika satu proyek membutuhkan versi terbaru dari sebuah library, sementara proyek lainnya memerlukan versi lebih lama? Di sinilah mana Python Virtual Environment berfungsi. Melalui sebuah environment yang terisolasi

dan kustomisasi penuh, kamu bisa menjaga setiap proyek tetap dalam jalurnya sendiri, tanpa takut ada konflik antar dependensi.

Membuat virtual environment:

1. Virtual environment bisa kita buat dengan library yang sudah di sediakan python
2. Untuk membuatkan, pertama buka terminal cmd
3. Buka folder yang diinginkan dengan cara “cd direktori yang diinginkan”
4. Contoh “cd D:/projek/env”
5. Membuat environment dengan cara “python -m venv nama_env”
6. Mengaktifkan environment “nama_env\Scripts\activate”
7. Kita bisa menginstal library yang diperlukan dengan cara “pip install nama_library”
8. Jika semua library yang dibutuhkan sudah diinstall maka environment siap dipakai

1.4 Instalasi Miniconda Untuk Management Environment

Untuk memudahkan pembelajaran *machine learning*, tools yang memudahkan kita salah satunya adalah dengan menggunakan python package manager. Package manager berguna untuk mengatur package yang kita pakai pada komputer agar package proyek *machine learning* kita rapi dan teratur. Selain itu menggunakan package manager memudahkan kita untuk mengatur environment. Jika sebelumnya kita secara manual menambahkan environment pada folder projek kita. Pada miniconda environment akan disimpan dalam satu folder yang di manage oleh miniconda.

Seperti yang disebutkan sebelumnya. Platform yang sangat populer untuk python package manager ini adalah anaconda dan miniconda. Apa perbedaannya? Perbedaannya hanya pada ukuran dan jumlah package awal yang lebih lengkap pada anaconda dibandingkan miniconda. Miniconda adalah versi kecilnya anaconda, sehingga akan lebih cepat untuk diinstalasi dan juga tidak memakan banyak penyimpanan komputer anda. Mari kita install miniconda.

Instalasi Miniconda:

1. Kunjungi situs resmi Miniconda: https://docs.conda.io/en/latest/miniconda.html
2. Pilih installer yang sesuai dengan sistem operasi Anda (Windows, macOS, atau Linux) dan arsitektur (64-bit atau 32-bit).
3. Jalankan installer yang telah diunduh.
4. Pilih opsi "Install for me only (recommended)".
5. Pilih direktori tempat Miniconda akan diinstal (biasanya secara default di %USERPROFILE%\Miniconda3).
6. Aktifkan opsi **Add Miniconda3 to my PATH environment variable** agar Anda bisa menjalankan conda dari command prompt.
7. Selesaikan instalasi dengan mengikuti petunjuk di layar.

Membuat Environment dengan Miniconda:

1. Bila miniconda sudah terinstal, silahkan buka terminalnya
2. Kemudian cek installasi conda dengan mengetikkan “conda --version”
3. Buat environment baru dengan “conda create --name nama_env python=versi_python”
4. Sekarang environment sudah siap pakai dan tinggal diisi dengan library yang dibutuhkan
5. Untuk mengaktifkannya kita perlu ketik “conda activate _nama environment_”
6. Kita akan mengisi environment dengan python terlebih dahulu dengan cara “conda install _nama library_” contoh “conda install python=3.11”
7. Environment siap dipakai

1.5 Jupyter Notebook vs Google Colab

Jupyter Notebook dan Google Colab adalah dua alat populer yang digunakan untuk menjalankan dan berbagi kode Python, khususnya di bidang data science, machine learning, dan analisis data. Keduanya menyediakan antarmuka yang memudahkan penulisan kode, visualisasi hasil, dan penggabungan teks penjelasan dalam satu dokumen interaktif.

Jupyter Notebook adalah aplikasi open-source berbasis web yang memungkinkan pengguna untuk membuat dan berbagi dokumen yang berisi kode, teks, visualisasi, dan grafik. Notebook ini mendukung berbagai bahasa pemrograman seperti Python, R, dan Julia, namun Python adalah yang paling umum digunakan. Anda bisa menginstalnya di **VSCode.**

Fitur utama Jupyter Notebook:

- Antarmuka interaktif: Pengguna dapat menulis kode dan menjalankannya sel per sel, yang memungkinkan hasil ditampilkan segera setelah eksekusi. Ini sangat berguna untuk eksperimen data science yang membutuhkan banyak iterasi.
- Dokumen yang mudah dibagikan: File notebook disimpan dalam format .ipynb yang mudah dibagikan. Notebook ini dapat menampilkan kode, grafik, teks penjelas (menggunakan Markdown), serta matematika (dengan LaTeX).
- Visualisasi: Mendukung visualisasi data menggunakan pustaka Python seperti Matplotlib, Seaborn, dan Plotly. Grafik dan plot dapat langsung muncul di notebook.
- Ekstensi dan integrasi: Dapat diperluas dengan berbagai ekstensi untuk mendukung fitur tambahan seperti pengeditan kode lebih baik, pengaturan gaya, dan fitur debugging.

Jupyter Notebook bisa dijalankan di komputer lokal atau di server (misalnya, melalui JupyterHub atau Binder) untuk memungkinkan akses jarak jauh.

Kemudian Google Colab (Colaboratory) adalah layanan berbasis cloud yang disediakan oleh Google, memungkinkan pengguna untuk menjalankan Jupyter Notebook langsung di browser tanpa perlu menginstal perangkat lunak apa pun. Colab memiliki beberapa keunggulan karena integrasi dengan ekosistem Google dan kemampuannya untuk menggunakan perangkat keras yang lebih kuat secara gratis.

Fitur utama Google Colab:

- Berbasis Cloud: Colab sepenuhnya berbasis cloud, yang berarti Anda tidak perlu menginstal apa pun di komputer lokal. Notebook dapat diakses dan dijalankan dari mana saja selama ada koneksi internet.

- Akses ke GPU dan TPU: Salah satu fitur menonjol Colab adalah kemampuannya untuk menggunakan unit pemrosesan grafis (GPU) dan tensor processing units (TPU) secara gratis, yang mempercepat komputasi, terutama dalam aplikasi deep learning.
- Integrasi dengan Google Drive: Colab terintegrasi dengan Google Drive, sehingga notebook dapat disimpan dan dikelola dengan mudah. Ini juga memungkinkan pengguna menyimpan data atau hasil di Google Drive.
- Kompatibilitas Jupyter: Notebook Colab disimpan dalam format .ipynb, yang juga kompatibel dengan Jupyter Notebook. Anda dapat dengan mudah mengimpor atau mengekspor notebook dari/ke Jupyter.
- Kolaborasi: Sama seperti Google Docs, Colab memungkinkan beberapa pengguna bekerja bersama secara real-time dalam satu notebook.
- Prainstal Pustaka: Colab sudah dilengkapi dengan berbagai pustaka Python yang umum digunakan seperti TensorFlow, Keras, NumPy, Pandas, dan lainnya. Jadi, pengguna tidak perlu menginstal pustaka-pustaka ini secara manual.

Kapan menggunakan Jupyter Notebook atau Google Colab:

- Jupyter Notebook lebih cocok digunakan jika Anda ingin bekerja secara lokal, memiliki kendali penuh atas lingkungan kerja, atau ingin mengatur infrastruktur server sendiri.
- Google Colab lebih cocok untuk pengguna yang membutuhkan akses ke GPU/TPU, tidak ingin repot dengan instalasi lokal, atau ingin berbagi dan berkolaborasi secara mudah dengan rekan kerja.

BAB 2
INTRO TO PYTHON

2.1 Pengantar Python

Bahasa pemrograman Pyhton merupakan salah satu pemrograman yang paling banyak digunakan, khususnya pada bidang *Artificial Intelegent, Machine Learning*, atau pengembangan website. Python memiliki sintaks yang mudah dipahami seperti bahasa Inggris yang membuatnya banyak digunakan. Python memiliki struktur yang lebih sederhana dibandingkan dengan beberapa bahasa lainnya.

```
main.py

x = 5
y = 10
print(x + y)
```

```
main.java

public class Main {
public  static  void  main(String[]
args) {
        int x = 5;
        int y = 10;
        System.out.println(x + y);
    }
}
```

Dapat dilihat pada contoh kode diatas, menggunakan bahasa Python jelas lebih sederhana dan mudah dipahami dengan struktur kode yang lebih sederhana. Python yang merupakan *interpreted language* dimana kode Python tidak di-*compile* menjadi kode mesin secara langsung sebelum dijalankan dan dapat dieksekusi secara langsung. Dengan menggunakan *notebook* seperti Jupyter atau GoogleColab sintak Python dapat menjalankan kode perbaris tanpa perlu menjalankan keseluruhan kode.

Penggunaan Python dalam *machine learning* didukung dengan berbagai pustaka yang mendukung dalam pengolahan data dan model *machine learning*. Beberapa Pustaka yang biasa digunakan, yaitu NumPy, Pandas, Matplotlib dan Seaborn yang berguna dalam pengolahan data, serta Scikit-learn, TensorFlow dan PyTorch yang digunakan untuk membangun model *machine learning.* Dengan pustaka yang ada pada Python dapat digunakan untuk membangun proyek *machine learning* dari awal hingga akhir, termasuk eksplorasi data, pemodelan, serta *deployment* model yang telah dibuat.

2.2 Tipe Data

2.2.1 Variabel

Variabel merupakan Lokasi memori yang disiapkan untuk menyimpan nilai-nilai yang akan dimasukkan. Variabel akan menyimpan data selama program dijalankan yang isi dari variabel dapat diubah oleh operasi-operasi tertentu pada program. Variabel dapat menyimpan beerbagai macam tipe data dan memiliki sifat dinamis, dimana dalam bahasa Python tidak perlu mendeklarasikan tipe data tertentu dalam penulisan variabel.

Dalam penulisan variabel, terdapat beberapa aturan umum seperti memulai penamaan dengan huruf atau garis bawah (_) dan tidak dimulai dengan angka. Penulisan nama variabel juga tidak memperbolehkan menggunakan kata kunci Pyhton seperti *if, else, while, for* untuk menghindari eror dalam pemrograman. Python juga merupakan bahasa yang *case-sensitive* yang membedakan huruf besar dan huruf kecil dalam penamaan variabel. Berikut merupakan contoh penulisan variabel dalam bahasa Python.

```
x = 56
name = "Speed"

print(x)
print (name)
```

2.2.2 Standard Data Types

Seperti bahasa pemrograman yang lain, Python memiliki beberapa tipe data umum seperti tipe data angka, string, list, tuple, dan dictionary. Untuk tipe data angka dibagi menjadi tiga tipe yaitu integer, float yang merupakan bilangan pecahan, dan bilangan kompleks yang memiliki nilai real dan imaginer. Berikut merupakan contoh penulisan tipe data angka.

```
a = 5
b = 6.946
c = 5 + 3j
```

Selanjutnya tipe data string yang merupakan kumpulan dari beberapa huruf/karakter. Penulisan untuk tipe data string dapat menggunakan tanda petik satu atau tanda petik dua seperti berikut.

```
sentences = 'Hello World'
kalimat = "Halo Dunia"
```

2.3 Operators

Dalam bahasa Python, operator merupakan symbol yang digunakan dalam operasi pada variabel dan nilai. Dengan menggunakan operator kita dapat memanipulasi data dan variabel sesuai yang kita inginkan. Python menyediakan beberapa kategori operator untuk menangani berbagai jenis operasi, mulai dari perhitungan matematika hingga perbandingan, dan logika.

2.3.1 Operator Aritmatika

Operator jenis ini digunakan untuk melakukan operasi matematika dasar. Operator ini akan berguna ketika bekerja dengan nilai numerik.

Tabel 1 Operator Aritmatika

Operator	Deskripsi	Contoh
+	Penjumlahan	7+2 = 9
-	Pengurangan	7–2 = 5
*	Perkalian	7*2 = 14
/	Pembagian (float)	7/2 = 3.5
//	Pembagian Bulat	7//2 = 3
%	Modulus (sisa bagi)	7%2 = 2
**	Eksponensial	7**2 = 49

Berikut contoh penulisan operator aritmatika dalam Python.

```
a = 7
b = 2

print(a+b)

print (a-b)

print(a * b)

print(a/b)

print(a//b)

print(a%b)

print(a**b)
```

2.3.2 Operator Perbandingan

Operator ini digunakan untuk membandingkan dua nilai atau lebih. Operator perbandingan akan mengembalikan nilai boolean (True atau False) berdasarkan hasil perbandingan.

Tabel 2 Operator Perbandingan

Operator	Deskripsi	Contoh
==	Sama dengan	7 == 2 adalah False
!=	Tidak sama dengan	7 != 2 adalah True
>	Lebih besar	7 > 2 adalah True
<	Lebih kecil	7 < 2 adalah False
>=	Lebih besar atau sama	7 >=7 adalah True
<=	Lebih kecil atau sama	7 <= -10 adalah False

Berikut contoh penggunaan operator perbandingan di Python.

```
print(3==2)   #False

print(2!=2)   #True

print(4>3)    #True

print(2<2)    #False

print(7>=7)   #True

print(4<=-10) #False
```

2.3.3 Operator Penugasan

Operator penugasan digunakan untuk menetapkan nilai pada variabel. Selain penugasan dasar =, Python menyediakan operator yang dapat melakukan operasi dan kemudian menetapkan hasilnya ke variabel.

Tabel 3 Operator Penugasan

Operator	Deskripsi	Contoh
=	Menetapkan nilai	x = 7
+=	Menambah dan menetapkan	x += 5 (sama seperti x = x + 5)

-=	Mengurangi dan menetapkan	x -= 5
*=	Mengalikan dan menetapkan	x *= 5
/=	Membagi dan menetapkan	x /= 5
%=	Modulus dan menetapkan	x %= 5
**=	Eksponensial dan menetapkan	x **= 5
//=	Pembagian bulat dan menetapkan	x //=5

Berikut contoh penulisan pada Python.

```
b = 10+3
print(b)

b += 3
print(b)

b -= 4
print(b)

b*=2
print(b)

b /= 2
print(b)

b %= 5
print(b)

b **= 2
print(b)

b //= 5
print(b)
```

2.3.4 Operator Logika

Operator logika digunakan untuk menggabungkan pernyataan kondisi. Hasil dari operasi logika akan berupa nilai True atau False. Operator logika juga bisa menggunakan symbol dari operator bitwise.

Tabel 4 Operator Logika

Operator	Deskripsi	Contoh
and (& dengan bitwise)	True jika keduanya benar	(7>5 and 2<10)

or (| dengan bitwise)	True jika salah satu benar	(2>5 or 4<10)
not	Membalikkan hasil	not(4>5)

Untuk mengetahui hasil ketika menggunakan operator logika, dapat dengan mengikuti tabel kebenaran seperti berikut.

Tabel 5 Tabel Kebenaran

AND			OR			NOT	
Input		Output	Input		Output	Input	Output
A	B	Y	A	B	Y	X	Y
0	0	0	0	0	0	0	1
1	0	0	1	0	1	1	0
0	1	0	0	1	1	-	-
1	1	1	1	1	1	-	-

Berikut contoh penggunaan operator logika dalam Pyhton.

```
a = True
b = False

print (a and b) #False
print (a&b)    #using bitwise

print(a or b)   #True
print (a|b)    #using bitwise

print(not a)    #False
```

2.3.5 Operator Keanggotaan

Dalam bahasa Python, terdapat syntax yang dapat digunakan untuk mencari apakah sebuah nilai terdapat di dalam data atau deret tertentu. Untuk melakukan ini dapat menggunakan operator keanggotaan yang akan memberikan keluaran bernilai True atau False.

Tabel 6 Operator Keanggotaan

Operator	Deskripsi	Contoh
in	Mengetahui apakah a merupakan bagian dari b	a in b

Operator	Deskripsi	Contoh
not in	Mengetahui apakah a bukan merupakan bagian dari b	a not in b

Berikut contoh penggunaan dalam bahasa Python.

```
a_list = [1,3,4,5,6]
print(3 in a_list)
print(10 not in a_list)
```

2.3.6 Operator Identitas

Operator identitas digunakan untuk membandingkan dua buah nilai atau objek untuk mengetahui apakah keduanya adalah objek yang sama atau menggunakan lokasi memori yang sama.

Tabel 7 Operator Identitas

Operator	Deskripsi	Contoh
is	Mengetahui apakah a merupakan objek yang sama dengan b	a is b
is not	Mengetahui apakah a merupakan objek yang berbeda dengan b	a is not b

Berikut contoh penggunaan operator identitas dalam bahasa Python.

```
print(10 is not 3)
print(2 is 2)

a_list = [1,2,3]
print([1,2,3] is [1,2,3])
print(a_list is a_list)
```

2.4 Pengondisian

Pengondisian atau biasa disebut *conditional statement* merupakan salah satu konsep dasar dalam pemrograman yang digunakan untuk membuat Keputusan berdasarkan kondisi tertentu. Dalam bahasa Python, untuk melakukan pengondisian dapat menggunakan pernyataan “if” yang memungkinkan program untuk menjalankan sebuah blok kode ketika suatu kondisi terpenuhi. Pernyataan “if” bekerja dengan memeriksa suatu kondisi, dan ketika kondisi tersebut bernilai benar, maka akan menjalankan blok kode yang sudah diberikan, namun ketika kondisi tersebut tidak terpenuhi, blok kode tersebut akan dilewatkan dan melanjutkan ke bagian selanjutnya. Berikut beberapa bentuk dasar dari pernyataan “if”.

2.4.1 Pernyataan If

Penggunaan pernyataan “if” perlu digunakan bersamaan dengan sebuah kondisi yang diberikan. Untuk memberikan kondisi dapat menggunakan operator perbandingan yang telah dibahas sebelumnya. Ketika sebuah kondisi terpenuhi, maka blok kode yang telah ditulis setelahnya akan dijalankan.

```
a = 10
b = 25

if b > a:
    print("b is greater than a")
```

Contoh diatas merupakan penggunaan dasar pernyataan “if” dengan kondisi yang diberikan adalah b memiliki nilai lebih besar dari a. Dengan variabel a dan b yang telah diberikan, pernyataan “if” akan memeriksa kondisi tersebut, ketika kondisi tersebut bernilai benar, maka program akan menuliskan “b is greater than a”.

2.4.2 Pernyataan Else

Ketika ingin menjalankan sebuah kode saat kondisi yang diberikan bernilai salah, pernyataan “else” dapat digunakan. Pernyataan ini akan berfungsi untuk menjalankan sebuah blok kode ketika kondisi dari pernyataan “if” bernilai salah atau tidak terpenuhi.

```
a = 24
b = 26

if a > b:
    print("a is greater than b")
else:
    print("b is greater than a")
```

Program diatas merupakan contoh penggunaan pernyataan “else”, dimana hasil dari pernyataan “if” yang bernilai salah dengan kondisi yang telah diberikan. Maka program akan menjalankan kode yang berada pada pernyataan “else”.

2.4.3 Pernyataan Elif (else if)

Pernyataan “elif” digunakan ketika ingin memberikan beberapa kondisi yang berbeda. Penggunaan “elif” akan bekerja dengan mengevaluasi beberapa kondisi secara berurutan.

```
grade = 65

if grade >= 85:
    print("A")
elif grade >= 70:
    print("B")
elif grade >= 60:
    print("C")
else:
    print("D")
```

2.4.4 Pernyataan Nested-If

Pernyataan “if” dapat dituliskan di dalam pernyataan “if” lain yang biasa disebut *nested if statement*. Pernyataan ini dapat digunakan ketika ingin membuat beberapa kondisi yang bergantung satu sama lain.

```
age = 20
gender = "man"

if age > 18:
    if gender == "man":
        print("He is an adult man")
    else:
        print("She is an adult woman")
```

Pada contoh diatas, kondisi pertama yang memeriksa umur apakah lebih besar dari 18. Jika kondisi tersebut benar, maka kondisi yang terdapat didalamnya akan diperiksa.

2.5 Loops

Loops atau pengulangan merupakan struktur program yang memungkinkan untuk menjalankan blok kode berulang-ulang berdasarkan kondisi tertentu. Dalam bahasa Python terdapat dua jenis loop, yaitu While loops dan For loops.

2.5.1 While Loops

Pengulangan menggunakan While akan terus berjalan selama kondisi yang diberikan bernilai True.

```
i = 1

while i <= 5:
    print(i)
    i += 1
```

Kondisi akan terus dievaluasi sebelum pengulangan, selama kondisi tersebut bernilai benar maka program didalamnya akan dijalankan. Jika kondisi berubah menjadi salah maka loop akan berhenti. Dalam contoh program diatas, kondisi yang diberikan adalah "i" lebih kecil dari 5, sehingga selama nilai "i" lebih kecil dari lima, program akan memberikan output nilai "i" dan menambahkan nilai "i". Program tersebut akan berhenti ketika "i" bernilai 5, dan nilai terakhir yang dikeluarkan adalah 4.

Dalam penggunaan struktur pengulangan, terdapat syntax "break" yang berfungsi untuk menghentikan pengulangan secara langsung. Penggunaan "break" memerlukan sebuah kondisi yang dapat ditemui dalam pengulangan, dan akan menghentikan pengulangan tersebut ketika kondisi tersebut terpenuhi.

```
i = 1

while i <= 6:
    print(i)
    if i == 3: break
    i += 1
```

Pada contoh program diatas, program akan berhenti ketika "i" bernilai 3 yang merupakan kondisi untuk menjalankan perintah "break". Program tersebut akan memberikan output terakhir yang bernilai 3 walaupun nilai tersebut masih dalam kondisi while yang dideklarasikan sebelumnya (nilai dibawah/sama dengan 6).

Selain "break", terdapat syntax "continue" yang dapat digunakan dalam struktur pengulangan. Syntax ini berfungsi untuk melewati iterasi atau pengualan yang berjalan dan melanjutkan ke iterasi berikutnya. Syntax "continue" akan berguna ketika ingin melewati nilai atau data tertentu dalam pengulangan tanpa menghentikan seluruh pengulangan.

```
i = 0
while i < 7:
    i += 1
    if i == 4: continue
    print(i)
```

Pada program diatas, syntax “continue” akan bekerja ketika nilai “i” sama dengan 4. Program tersebut akan berjalan seperti pengulangan biasa yang menuliskan nilai “i”. Namun ketika “i” bernilai 4, program tidak memberikan output, melainkan melanjutkan iterasi dan memberikan nilai “i” yang selanjutnya, yaitu 5.

2.5.2 For Loops

Pengulangan menggunakan For digunakan untuk mengulangi blok kode untuk setiap elemen dalam sebuah urutan seperti dalam list, tuple, string atau rentang angka yang diberikan. Pengulangan menggunakan For akan mengambil setiap elemen secara berurutan, menjalankan blok kode di dalamnya, dan berlanjut ke elemen berikutnya hingga semua elemen telah digunakan.

```
food = ["Telur", "Nasi", "Sosis", "Sayur"]

for x in food:
    print(x)
```

Pada contoh program di atas, terdapat list makanan yang berisi 4 elemen, dengan menggunakan pengulangan For, “x” yang akan mengambil elemen dalam list tersebut akan dicetak dan terus berulang hingga setiap elemen telah digunakan.

```
for x in "martabak":
    print(x)
```

Pengulangan dengan For juga dapat digunakan pada sebuah string, dimana setiap huruf dalam string akan menjadi elemen yang akan dicetak pada setiap pengulangan seperti contoh di atas.

Pengulangan For biasa menggunakan fungsi range(), yang akan memberikan angka yang berurutan dengan rentang yang diinginkan. Seperti contoh program di bawah, dengan fungsi range(5), program akan berulang memberikan output nilai dari 0 hingga 5. Hal ini dikarenakan fungsi range() yang dimulai dari 0 hingga 4, karena fungnsi range() yang bersifat ekslusif.

```
for i in range(5):
    print(i)
```

Pengulangan For juga bisa dituliskan didalam pengulangan lainnya atau bisa disebut *nested loop*. Struktur program ini akan bekerja dengan struktur dua dimensi seperti tabel atau matriks.

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
    for y in fruits:
        print(x + " " + y)
```

2.6 Functions

Fungsi merupakan salah satu konsep penting dalam pemrograman. Dengan menggunakan fungsi dapat membuat blok kode yang dapat dipanggil dan digunakan berulang kali. Hal ini akan membantu dalam penulisan program yang lebih terstruktur dan mudah dikelola. Pada dasarnya fungsi berisi sekumpulan syntax yang akan menjalankan tugasnya ketika dipanggil. Fungsi dalam Python dapat menerima input atau yang biasa disebut argumen atau tidak menerima input sama sekali dan akan memberikan sebuah output.

2.6.1 Membuat dan Memanggil Fungsi

Penulisan sebuah fungsi dalam bahasa Python diawali dengan kata kunci “def”, kemudian diikuti dengan nama fungsi dan tanda kurung dan titik dua diakhir. Untuk memanggil sebuah fungsi, hanya perlu menuliskan nama fungsi dengan tanda kurung pada program.

```
def greetings():
    print("Hello, Good morning")

greetings()
```

2.6.2 Fungsi dengan Argumen

Sebuah fungsi juga dapat diberikan input supaya fungsi tersebut dapat bekerja dengan data yang berbeda setiap digunakan. Dalam membuat fungsi dengan argumen, fungsi dibuat dengan menambahkan argumen pada nama fungsi di dalam tanda kurung. Argumen yang dituliskan pada nama fungsi nantinya akan dipanggil pada blok kode di

dalam fungsi. Ketika memanggil sebuah fungsi, input akan dituliskan di dalam tanda kurung dari nama fungsi.

```
def greetings(name):
    print(f"Hello, Good morning {name}")

greetings("John")
```

2.6.3 Fungsi dengan Banyak Argumen

Sebuah fungsi dapat menerima lebih dari satu argument atau input. Penulisan fungsi dengan banyak argument hanya perlu menambahkan argument pada tanda kurung yang dipisahkan dengan tanda koma.

```
def myName(fname, lname):
    print(f"My name is {fname} {lname}")

myName("John", "Adam")
myName(lname="Adam", fname="John")
```

Ketika memanggil sebuah fungsi dengan banyak argumen, penulisan argument hanya perlu ditulis secara berurutan. Penulisan argumen juga bisa secara tidak berurutan dengan cara menuliskan nama argumen atau variabel kemudian menuliskan nilai yang ingin diberikan seperti program diatas pada baris terakhir.

2.6.4 Fungsi dengan Nilai Kembali

Sebuah fungsi dapat memberikan output dengan menggunakan perintah “return”. Dengan menggunakan “return”, sebuah fungsi dapat mengembalikan sebuah nilai tanpa perlu mencetaknya.

```
def multiplication(a, b):
    x = a * b
    return x

result = multiplication(4, 3)
print(f"Result of multiplyng 4 and 3 is {result}")
```

BAB 3
LIBRARY FOR PYTHON

3.1 Pengantar Library

Dalam pembelajaran *machine learning*, pengolahan data merupakan aspek yang sangat penting. Kualitas data yang baik akan memudahkan model menghasilkan prediksi yang akurat. Beberapa library Python populer yang sering digunakan dalam pengolahan data antara lain NumPy dan Pandas untuk manipulasi dan analisis data, serta Matplotlib dan Seaborn untuk visualisasi data. Penguasaan library-library ini penting untuk mempermudah proses eksplorasi, pembersihan, dan penyajian data, yang merupakan bagian integral dari *machine learning*.

3.2 Numpy

NumPy (Numerical Python) adalah library Python yang sangat powerful untuk membuat dan mengolah array multi-dimensional (dikenal juga sebagai matriks atau tensor). NumPy dirancang untuk mempercepat proses komputasi numerik di Python, baik untuk operasi sederhana maupun kompleks seperti aljabar linear, simulasi acak, transformasi Fourier, dan banyak lagi. Banyak library Python populer, seperti Pandas dan SciPy, dibangun di atas NumPy.

Instalasi NumPy

Untuk menginstal NumPy, jalankan perintah berikut di terminal:

```
pip install numpy
```

Setelah berhasil diinstal, kita bisa mengimpornya dengan cara berikut:

```
import numpy as np
```

Membuat Array

Untuk membuat array menggunakan NumPy, kita bisa menggunakan kode berikut:

```
array_1 = np.array([2, 3, 6, 5])
print(array_1)
```

Output:

```
array([2, 3, 6, 5])
```

Slicing Array

Untuk mengambil satu elemen dari array, kita bisa menggunakan indeks:

```
array_1 = np.array([2, 3, 6, 5])
print(array_1[2])
```

Output:

```
6
```

Perlu diingat bahwa indeks dari array Numpy dimulai dari nol.

Kita juga bisa mengambil beberapa elemen sekaligus dengan slicing:

```
array_1 = np.array([2, 3, 6, 5])
print(array_1[0:2])
```

Output:

```
array([2, 3, 6])
```

Bentuk dan Reshaping Array

Untuk mengetahui bentuk (shape) dari array atau matriks pada numpy:

```
array_1 = np.array([2, 3, 6, 5], [3, 1, 2, 4])
print(array_1.shape)
```

Output:

```
(2, 4)
```

Kode di atas akan menghasilkan bentuk matriksnya yaitu (2, 4) atau matriks 2x4.

Untuk melakukan reshaping atau mengubah bentuk matriks, dapat menggunakan fungsi reshape:

```
array_1 = np.array([2, 3, 6, 5])
reshaped_array = array_1.reshape(2,2) # mengubah matriks jadi
2x2
print(reshaped_array)
```

Output:

```
array([[2, 3],
       [6, 5]])
```

Kode di atas akan mengubah bentuk matriks yang awalnya 4x1 menjadi 2x2.

Menggabungkan dan Memisahkan Array

Untuk menggabungkan dua array yaitu dengan menggunakan concatenate sebagai berikut.

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.concatenate((arr1, arr2))
print(arr)
```

Output:

```
array([1, 2, 3, 4, 5, 6])
```

Untuk memisahkan array menjadi beberapa bagian menggunakan array_split:

```
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 4) # 4 adalah jumlah split
print(newarr)
```

Output:

```
[array([1, 2]), array([3, 4]), array([5]), array([6])]
```

Operasi Perhitungan

Penjumlahan, pengurangan, pembagian, dan perkalian menggunakan numpy:

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print(a + b)
print(a - b)
print(a * b)
print(a / b)
```

Melakukan operasi dot:

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
result = np.dot(A, B)
print(result)
```

Menggunakan fungsi agregat seperti sum, mean, min, dan max.

```
A = np.array([[1, 2], [3, 4]])
print(np.sum(A))  # Jumlah semua elemen: 10
print(np.mean(A))  # Rata-rata: 2.5
print(np.min(A))  # Nilai minimum: 1
print(np.max(A))  # Nilai maksimum: 4
```

3.3 Pandas

Pandas adalah sebuah library di Python yang sangat berguna untuk mengolah dan menganalisis data. Library ini dibuat khusus untuk bekerja dengan data yang disimpan dalam bentuk tabel (data frame), mirip dengan tabel di Excel atau database. Pandas sangat membantu dalam berbagai tugas terkait data, seperti *data wrangling*, *exploratory data analysis*, menganalisis pola, dan bahkan digunakan dalam *machine learning*. Untuk memulai menggunakan pandas, pertama kita perlu menginstall librarynya terlebih dahulu.

```
pip install pandas
```

Setelah itu kita bisa import pandas pada code editor kita.

```
import pandas as pd
```

Membuat Data Frame

Untuk membuat data frame atau tabel pada pandas kita bisa menggunakan pd.DataFrame(). Untuk membuat data frame dari list:

```
data = [1, 2, 3, 4, 5]
df = pd.DataFrame(data)
print(df)
```

Output:

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

Contoh lain dengan disertai nama kolom:

```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data, columns=['Name', 'Age'])
print df
```

Output:

```
    Name  Age
0   Alex  10
1    Bob  12
2  Clarke 13
```

Membuat data frame dari dictionary dan mengganti indeks:

```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve',
'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data,
index=['rank1','rank2','rank3','rank4'])
print df
```

Output:

```
       Age   Name
rank1  28    Tom
rank2  34    Jack
rank3  29    Steve
rank4  42    Ricky
```

Membuat data frame dari file csv, xlsx, dan json:

```
df_csv = pd.read_csv("dataset.csv")
df_excel = pd.read_excel("dataset.xlsx")
```

```
df_json = pd.read_json("dataset.json")
```

Membersihkan Data

Untuk menghitung dan menghapus nilai null atau nilai kosong pada data frame:

```
print(df.isnull().sum())
df = df.dropna()
```

Untuk menghitung dan menghapus nilai duplikat pada data frame:

```
df.duplicated().sum()
df = df.drop_duplicates()
```

Eksplorasi Data

Eksplorasi data adalah tahap penting dalam proses analisis data yang bertujuan untuk memahami struktur, pola, dan karakteristik dari dataset yang ada sebelum melakukan analisis lebih lanjut atau pengembangan model.

Untuk menampilkan informasi dari masing-masing kolom:

```
print(df.info())
```

Contoh Output:

```
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   data    5 non-null      int64
```

Untuk menampilkan informasi statistik dari masing-masing kolom:

```
print(df.describe())
```

Contoh Output:

```
          data
count    5.00000
mean     3.20000
std      1.30384
min      2.00000
25%      2.00000
50%      3.00000
75%      4.00000
max      5.00000
```

Menghitung nilai korelasi antar variabel:

```
correlation_matrix = df.corr()
print(correlation_matrix)
```

Mengelompokkan data dalam DataFrame Pandas berdasarkan kolom kategorikal dan menghitung rata-rata dari kolom numerik untuk setiap kelompok:

```
grouped_data =
df.groupby('categorical_column').agg({'numerical_column': 'mea
n'})
print(grouped_data)
```

Manipulasi Data

Menambahkan kolom baru:

misalnya ingin menambahkan kolom baru dengan nama 'square' dengan nilainya adalah pangkat dua dari nilai yang ada di kolom 'value'

```
df['square'] = df['value']**2
```

Mengurutkan data berdasarkan kolom tertentu :

```
df_sorted = df.sort_values(by='column_name', ascending=True)
```

Filter data:

misalnya ingin mengambil nilai yang lebih dari 100 pada kolom 'value':

```
filtered_data = df[df['value'] >100]
```

Menggabungkan data frame:

```
new_df = pd.concat([df, df_new], axis=0)
```

3.4 Matplotlib

Matplotlib adalah library yang digunakan untuk memvisualisasikan data. Library ini menyediakan berbagai fungsi dan alat untuk membuat berbagai jenis grafik dan plot, seperti grafik garis, grafik batang, histogram, diagram sebar, dan banyak lagi.

Instalasi Matplotlib

Untuk menginstall matplotlib, jalankan perintah ini pada terminal:

```
pip install matplotlib
```

Setelah berhasil diinstall, kita bisa mengimportnya di *code editor*:

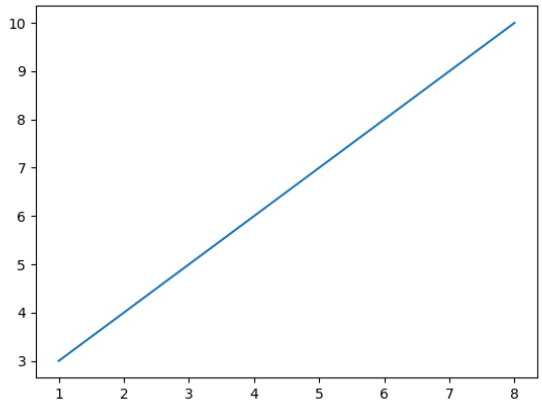
```
import matplotlib.pyplot as plt
```

Membuat Grafik

Untuk membuat *line chart*:

```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show() # untuk menampilkan chart
```

Output:

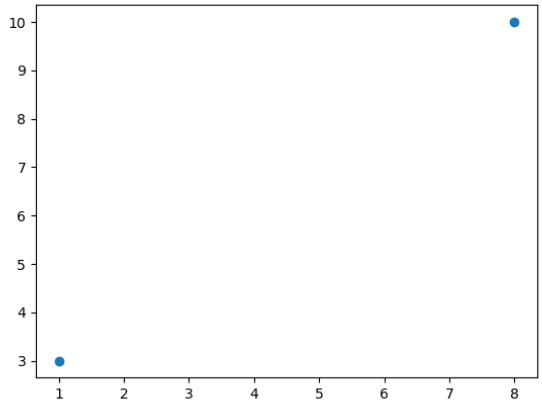


Gambar 1 Membuat Garis

Untuk membuat diagram titik dari data:

```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints, 'o')
plt.show()
```

Output:



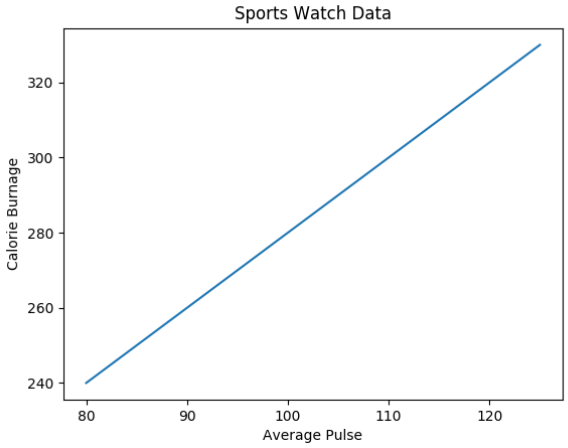
Gambar 2 Membuat Titik Pada Grafik

Untuk menambahkan judul, label x, dan label y pada grafik:

```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y =
np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)
```

```
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```

Output:

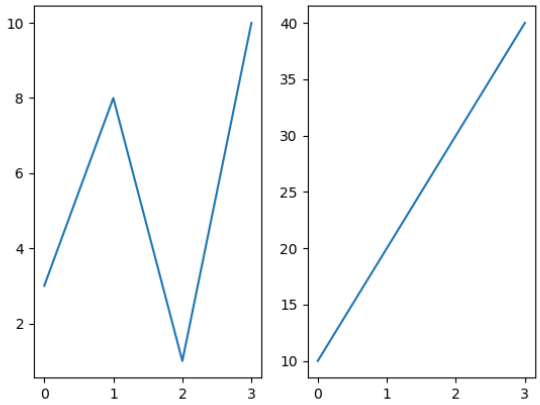


Gambar 3 Menambah Judul dan Label

Untuk membuat dan menampilkan beberapa grafik menggunakan subplot():

```
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1) # the figure has 1 row, 2 columns, and
this plot is the first plot.
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2) # the figure has 1 row, 2 columns, and
this plot is the second plot.
plt.plot(x,y)

plt.show()
```

Output:

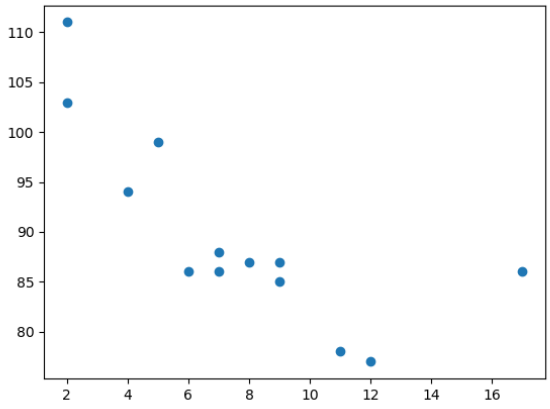


Gambar 4 Menambah Subplot

Membuat scatter plot:

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```

Output:

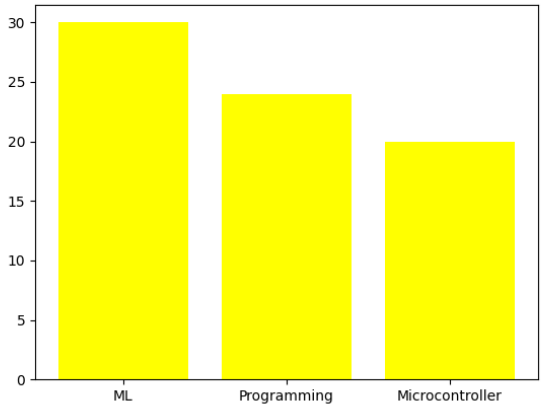


Gambar 5 Scatter Plot

Membuat bar chart:

```
x = np.array(["ML", "Programming", "Microcontroller"])
y = np.array([30, 24, 20])

plt.bar(x, y, color = "yellow")
plt.show()
```

Output:

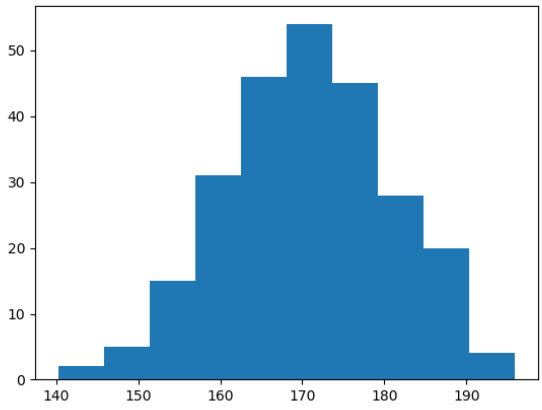


Gambar 6 Membuat Diagram Batang

Membuat histogram:

```
x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```

Output:

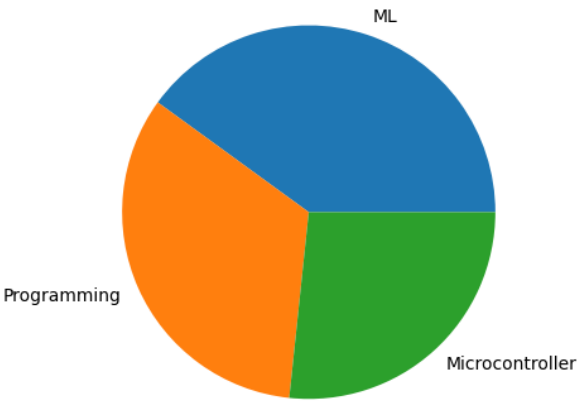


Gambar 7 Membuat Histogram

Membuat pie chart:

```
y = np.array([30,24,20])
mylabels = ["ML", "Programming", "Microcontroller"]

plt.pie(y, labels = mylabels)
plt.show()
```

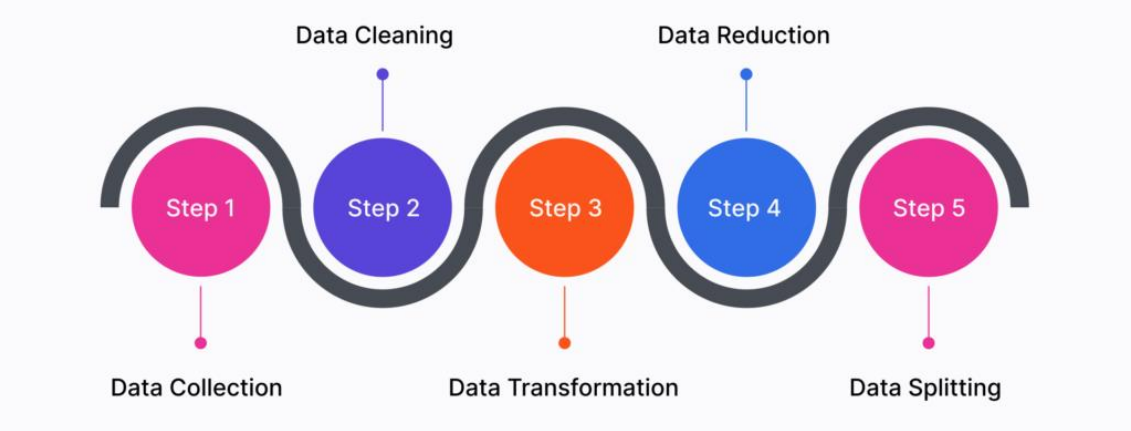
Output:



Gambar 8 Membuat Pie Chart

BAB 4
DATA PREPARATION

Data preparation merupakan proses penting dalam *machine learning* yang akan sangat mempengaruhi performa dan akurasi dari model yang dibuat. Proses ini bertujuan untuk mengolah dan membersihkan data mentah sebelum digunakan menjadi data dengan format yang sesuai untuk algoritma *machine learning* yang digunakan. proses ini termasuk mengumpulkan data, membersihkan data, transformasi data, reduksi data, dan pembagian data.



Gambar 9 Alur Dalam Tahap Data Preparation

Dalam proses pembuatan *machine learning*, tentu memerlukan data yang cukup untuk mendapatkan model *machine learning* dengan hasil yang baik. Data-data ini bisa didapatkan dari berbagai sumber seperti database internal, survei, atau sumber eksternal seperti API dan dataset publik. Terdapat beberapa sumber data yang bersifat public yang dapat digunakan untuk pengembangan *machine learning* seperti Kaggle, UCI *Machine learning* Repository, Goggle Dataset Search, atau Satu Data Indonesia. Data yang didapatkan bisa dalam format yang bermacam-macam seperti csv, excel, json, dan lain-lain. Untuk bisa mengakses data dengan format yang bermacam-macam, bisa menggunakan salah satu pustaka Python yaitu Pandas yang telah dibahas di bab sebelumnya. Data-data ini tentu tidak langsung siap untuk digunakan membuat model *machine learning*, maka dari itu diperlukan tahap pembersihan data.

Proses pembersihan data merupakan proses untuk menangani permasalahan yang ada pada data yang telah dikumpulkan. Seringkali dari data yang didapatkan terdapat

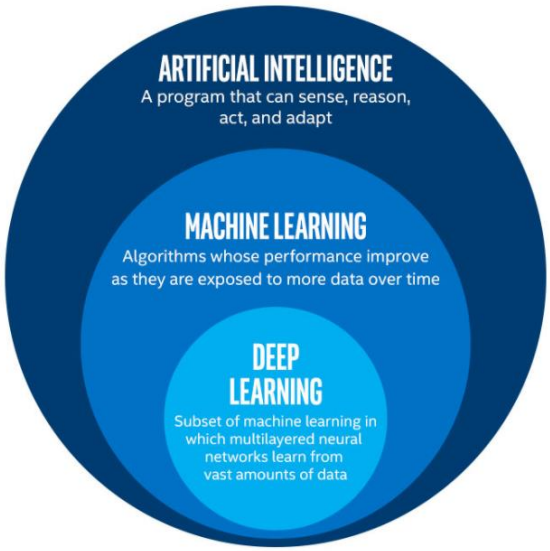
beberapa permasalahan seperti *missing value, outliers*, atau tipe data yang tidak sesuai. Permasalahan ini perlu ditangani sebelum data yang dimiliki digunakan untuk pembuatan model *machine learning.* Metode dalam menangani permsalahan pada data bergantung pada tipe data yang dimiliki dan mengikuti jenis data seperti yang dibutuhkan untuk model *machine learning*. Sebagai contoh ketika menangani *missing value*, permasalahan ini biasa ditemukan dari data numerik yang seringkali tidak memiliki nilai atau bernilai nol. Untuk menangani kasus ini diperlukan pemahaman terhadap data dan tujuan dari model *machine learning* untuk menentukan metode penyelesainnya, seperti menghapus seluruh baris dimana terdapat *missing value*, atau mengisi *missing value* dengan nilai yang didapatkan dari data seperti rata-rata atau median dari kolom data.

Tahap selanjutnya adalah proses transformasi data, dimana data yang telah dibersihkan akan dirubah menjadi format yang dapat dipahami dengan mudah oleh algoritma *machine learning*. Tahap ini mencakup proses mengubah data numerik yang memiliki skala berbeda menggunakan metode normalisasi atau penskalan min-maks. Mengubah data kategori seperti tipe barang atau nama daerah menjadi format numerik juga akan membantu dalam proses algoritma *machine learning.* Proses reduksi data juga dapat membantu algoritma *machine learning* dalam menentukan pola dengan lebih mudah. Reduksi data merupakan proses penyederhanaan data tanpa kehilangan esensinya. Dalam proses ini akan ditentukan fitur yang akan dipertahankan dari kumpulan data yang akan digunakan dalam algoritma *machine learning.*

Tahap terakhir dalam data preparation adalah pembagian data menjadi beberapa subset untuk *training*, *validation,* dan *testing.* Dengan membagi data menjadi beberapa subset, dapat memastikan model *machine learning* yang telah dibuat dapat diuji dengan data baru untuk melihat performa dari model yang dibuat. Tahap ini penting dalam pembuatan model *machine learning* untuk menghindari *overfitting* atau model yang memberikan performa yang baik pada data *training*, namun ketika model digunakan untuk data lain memberikan performa yang kurang baik. Pembagian data biasa dilakukan dengan rasio 70:30 atau 80:20 untuk *training* dan *test*, tergantung jumlah data yang dimiliki.

BAB 5
MACHINE LEARNING

5.1 PENGANTAR MACHINE LEARNING



Gambar 10 Domain Pengetahuan dalam Artificial Intelligence

Machine learning (pembelajaran mesin) adalah cabang dari kecerdasan buatan (AI) yang berfokus pada pengembangan algoritma dan model yang memungkinkan komputer untuk belajar dari data dan membuat prediksi atau keputusan tanpa diprogram secara eksplisit. Pada intinya, *machine learning* menggunakan data untuk menemukan pola, mengenali tren, dan memperbaiki kinerja model seiring berjalannya waktu.

Terminologi dasar dalam *machine learning*:

1. Data: Data adalah fondasi dari *machine learning*. Ini bisa berupa angka, teks, gambar, video, atau jenis informasi lainnya yang digunakan untuk melatih model.
2. Model: Model dalam *machine learning* adalah representasi matematika atau algoritma yang mencoba memetakan input ke output. Model ini dibangun dengan menggunakan data pelatihan dan dapat digunakan untuk membuat prediksi pada data baru.

3. Fitur (Features): Fitur adalah karakteristik atau atribut dari data yang digunakan sebagai input untuk model. Misalnya, dalam analisis rumah, fitur bisa berupa ukuran rumah, lokasi, dan jumlah kamar. Disimbolkan dengan huruf X.
4. Target: adalah output dari model itu sendiri biasanya disimbolkan dengan huruf Y.
5. Label: Dalam pembelajaran terawasi (supervised learning), label adalah output atau jawaban yang benar dari data yang dilabeli. Misalnya, dalam klasifikasi email sebagai "spam" atau "bukan spam", label adalah kategori "spam" atau "non-spam".
6. Training dan Testing:
   - Training: Proses melatih model menggunakan data yang sudah diberi label. Model akan belajar menemukan pola dari data ini.
   - Testing: Proses menguji performa model pada data baru yang belum pernah dilihat untuk mengevaluasi akurasi prediksi model.

Jenis-jenis *machine learning*:

1. Supervised Learning (Pembelajaran Terawasi): Pada jenis ini, model dilatih dengan menggunakan data yang sudah diberi label. Contohnya, memprediksi harga rumah berdasarkan fitur seperti ukuran, lokasi, dan tahun bangunan. Model diajarkan untuk memetakan input ke output berdasarkan contoh-contoh yang sudah ada. Contoh algoritma: KNN, Decision Trees, Support Vector Machines (SVM).
2. Unsupervised Learning (Pembelajaran Tak Terawasi): Pada jenis ini, model mencoba menemukan pola atau struktur dalam data yang tidak memiliki label. Ini digunakan untuk mengelompokkan data berdasarkan kemiripan, atau untuk menemukan asosiasi antara variabel. Contoh algoritma: K-Means Clustering, Principal Component Analysis (PCA), Hierarchical Clustering.
3. Reinforcement Learning (Pembelajaran Penguatan): Dalam pembelajaran ini, agen (agent) belajar untuk membuat keputusan dengan berinteraksi dengan

lingkungannya dan menerima umpan balik dalam bentuk reward atau penalti. Tujuannya adalah memaksimalkan reward kumulatif seiring waktu.

- Contoh aplikasi: Pengendalian robot, game, dan pengendalian sistem otonom.
- Contoh algoritma: Q-Learning, Proximal Policy Optimization (PPO).

Langkah-langkah dalam *machine learning*:

1. Mengumpulkan Data: Data dikumpulkan dari berbagai sumber, seperti sensor, database, atau aplikasi.
2. Preprocessing Data: Data biasanya perlu dibersihkan, dinormalisasi, atau diubah agar sesuai dengan format yang diperlukan oleh algoritma.
3. Membangun Model: Setelah data siap, algoritma *machine learning* dipilih untuk membangun model berdasarkan data pelatihan.
4. Training: Model dilatih menggunakan data pelatihan untuk menemukan pola atau hubungan.
5. Evaluasi: Model diuji menggunakan data uji (testing data) untuk melihat seberapa baik prediksi yang dihasilkannya.
6. Deployment: Model yang dilatih digunakan dalam aplikasi dunia nyata untuk membuat prediksi atau keputusan.

Contoh aplikasi *machine learning*:

- Pengenalan gambar: Mendeteksi objek dalam foto atau video.
- Pengenalan suara: Menentukan kata atau suara yang diucapkan oleh pengguna.
- Deteksi penipuan: Menganalisis pola transaksi untuk mendeteksi aktivitas yang mencurigakan.
- Sistem rekomendasi: Seperti yang digunakan oleh Netflix atau Amazon untuk merekomendasikan film atau produk berdasarkan preferensi pengguna.

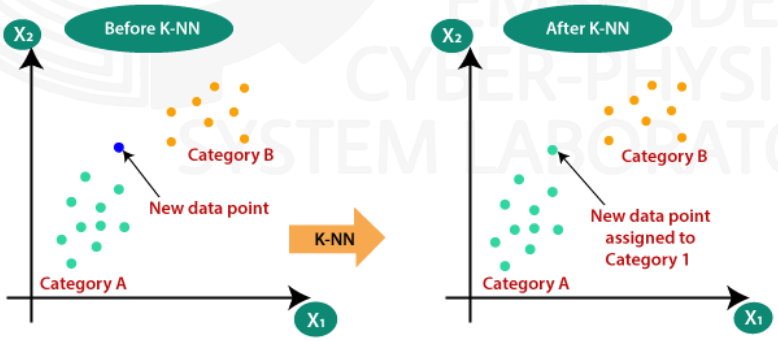
Dengan kemajuan dalam komputasi dan ketersediaan data yang semakin besar, *machine learning* telah menjadi salah satu pilar utama di berbagai bidang teknologi modern.

5.2 ALGORITMA SUPERVISED LEARNING

Supervised Learning menggunakan pendekatan pembelajaran yang mendikte. Mesin akan diberikan data yang sudah diberikan label. Misal sebuah mesin ingin mengklasifikasikan antara anjing dan kucing. Maka kita sebagai manusia harus memberikan mesin sejumlah data yang sudah di berikan label anjing dan kucing pada setiap gambar.

5.2.1 K-Nearest Neighbor (KNN)

KNN adalah salah satu algoritma yang paling populer digunakan dalam mesin learning karena bisa digunakan untuk klasifikasi dan regresi. Algoritma ini mengidentifikasi data baru yang ingin diprediksi dengan melihat jarak terdekat pada data yang lama(neighbor). Misal ada data yang targetnya adalah mengklasifikasi laki-laki dan perempuan. Algoritma KNN akan mengklasifikasikan data tersebut dengan cara melihat data terdekatnya. Jika data terdekat adalah laki-laki maka data prediksi akan menghasilkan laki-laki dan berlaku sebaliknya.



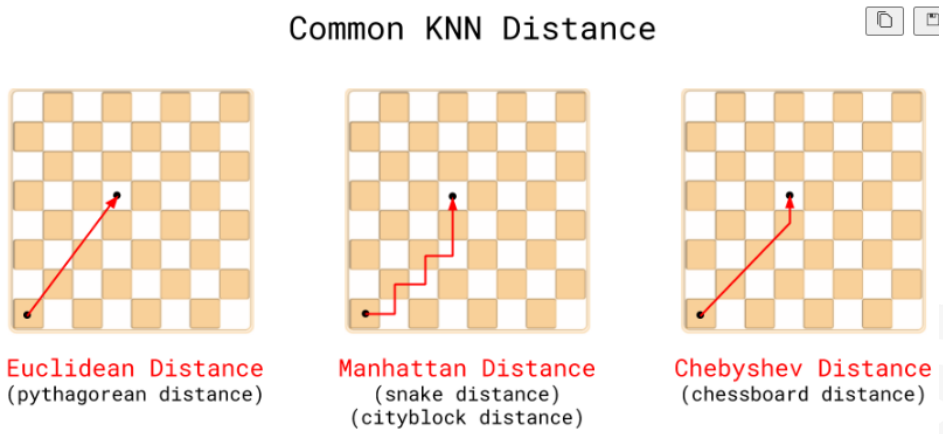
Gambar 11 Ilustrasi KNN

Lebih mudah jika anda melihat gambar di atas. Jika diperhatikan, data baru yang berwarna biru dikategorikan sebagai kategori A setelah di algoritma KNN diterapkan. Hal ini karena data kategori A lebih dekat dengan data baru. Maka dari itu metode perhitungan jarak menjadi parameter penting dalam KNN. Selain itu kita juga dapat menambah jumlah neighbor(tetangga) yang menjadi parameter klasifikasi. Tetangga

yang dimaksud adalah data lama yang menjadi acuan data baru. Selain dua parameter tersebut, kita dapat membobotkan KNN secara uniform atau jarak. Hal ini akan lebih detail pada penjelasan berikutnya.

Untuk lebih jelasnya mari kita melakukan demo menggunakan jupyter notebook lokal atau colab. Pertama import terlebih dahulu paket yang perlu, yaitu LuWiji untuk KNN. Dengan kode berikut.

```
import Luwiji
illustration.knn_distance
```



Gambar 12 Ilustrasi Jenis Jarak KNN

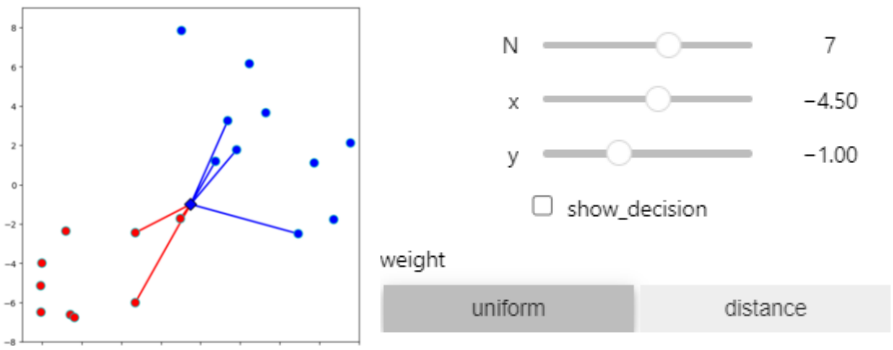
Seperti yang disebutkan sebelumnya. Jarak tetangga menjadi parameter penting untuk KNN. Dalam mencari jarak terdapat beberapa metode seperti gambar di atas. Metode ini pun memiliki kelemahan dan keuntungannya masing-masing. Tapi tidak akan dijelaskan secara mendetail dalam modul ini.

Kemudian untuk melakukan demo dan ilustrasi KNN digunakan agar kita bisa lebih memahami bagaimana cara kerja KNN. Silahkan gunakan baris kode berikut.

```
demo.knn()
```

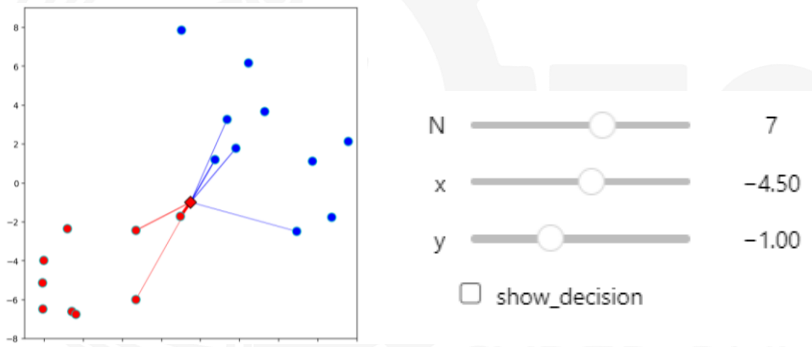
dari demo ini kita dapat merubah N tetangga. Kita juga dapat mengubah posisi data baru yang berbentuk persegi (x,y) dan merubah pembobotan. Seperti yang disinggung sebelumnya pembobotan akan menentukan bagaimana data baru diklasifikasikan. Pembobotan uniform akan menyebabkan data baru diklasifikasikan berdasarkan jumlah tetangga terdekat yang paling banyak. Pada gambar dibawah

tetangga terbanyak adalah biru, maka data baru diklasifikasikan sebagai biru. Namun jika kita ubah



Gambar 13 Pembobotan Uniform

pembobotan menjadi distance. Maka klasifikasi data baru akan menjadi merah seperti gambar di bawah. Hal ini karena tetangga yang lebih dekat akan mendapatkan bobot yang lebih tinggi.



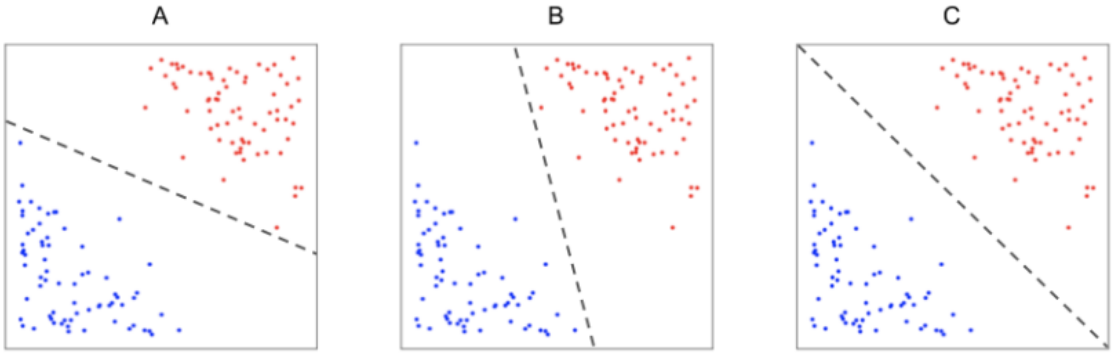
Gambar 14 Pembobotan Distance

5.2.2 Suport Vector Machine (SVM)

Mari perhatikan ilustrasi A, B, dan C. Ilustrasi tersebut adalah contoh klasifikasi gedung A dan gedung B. Titik-titik merah adalah gedung A dan yang biru adalah gedung B. Jika kita disuruh untuk memilih mana garis putus-putus yang paling tepat untuk memisahkan data tersebut. Maka secara spontan kita akan menjawab gambar C. Kenapa?

Akurasi gambar A, B, dan C sebenarnya semuanya 100%. Tapi apa yang membedakan? Kalau kita lihat Gambar A, perhatikan titik merah yang berada dekat sekali dengan garis putus-putus yang kita buat sebagai pembatas klasifikasi. Jika kita punya data baru dengan warna yang sama bergeser sedikit saja lebih ke bawah. Maka data baru

kita akan diklasifikasikan sebagai warna biru. Artinya model A terlalu sensitif. Mari kita lihat gambar C, garis pemisah terletak lebih jauh dari titik biru maupun merah terdekatnya. Dalam bahasa matematika, pemisah pada gambar C marginnya lebih lebar dibandingkan gambar A dan B.



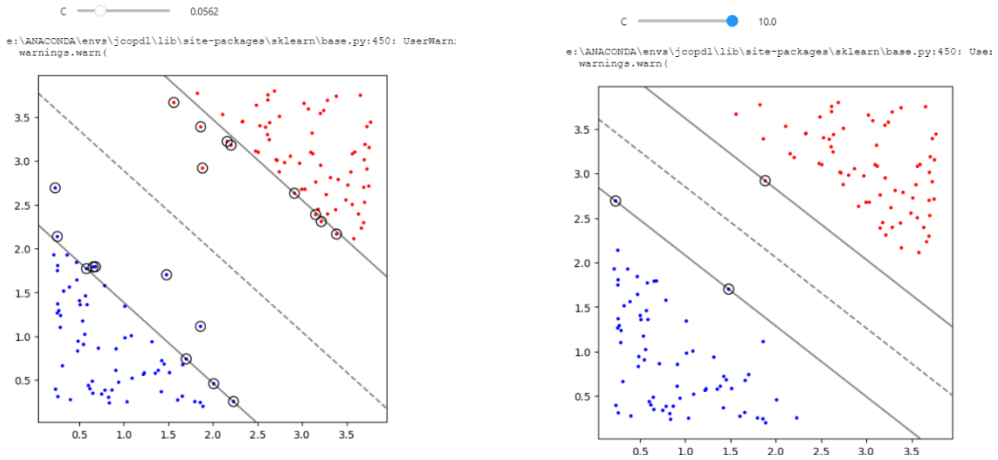
Gambar 15 Ilustrasi SVM

Inilah SVM, bagaimana cara membuat pemisahan dengan margin terbesar. Ini bisa dianalogikan seperti membuat jalan raya terbesar, gedung paling tepi jalan disebut sebagai suport vector karena digunakan untuk menemukan margin terbesar (jalan raya terbesar). Maka SVM bisa disederhanakan seperti mesin pencari gedung-gedung tersebut.

C-Penalty Parameter

Perhatikan gambar di bawah ini, ada nilai x yang sangat dekat dengan kumpulan data bulat, kita bisa bilang outlayer. Jika kita memisahkan dengan hanya melihat nilai x tersebut maka model kita akan terlalu sensitif. Sehingga perlu adanya toleransi terhadap data-data yang nilainya agak melenceng dari data-data yang lain.

Kita menginginkan model kita seperti gambar di bawah ini agar model tidak terlalu sensitif. Untuk itu dibutuhkan parameter penalty yang di notasikan sebagai c. Ini akan membuat model lebih toleransi terhadap data. Liat ilustrasi berikut. Semakin besar C => semakin besar penalty terhadap kesalahan => lebih sensitive. Semakin kecil C => semakin toleran terhadap kesalahan

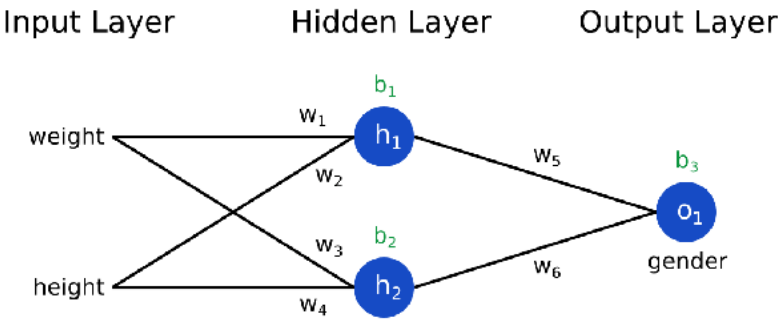


Gambar 16 C-Penalty Parameter

5.2.3 Neural Networks (NN)

Pada bagian ini Neural Networks hanya diperkenalkan singkat. NN biasanya dikenal masuk ke ranah Deep Learning, subset dari *Machine learning* yang berfokus pada pembelajaran berbasis Neural Networks. Secara umum Neural Networks biasanya perlu label sehingga kami masukan ke supervised learning.

Neural networks (jaringan saraf tiruan) adalah model komputasi yang terinspirasi oleh cara kerja otak manusia dalam memproses informasi. Mereka terdiri dari lapisan-lapisan neuron buatan yang saling terhubung, yang dapat "belajar" dari data untuk mengenali pola atau membuat prediksi. Neural networks banyak digunakan dalam berbagai tugas pembelajaran mesin (*machine learning*) seperti pengenalan gambar, pemrosesan bahasa alami, dan prediksi deret waktu.



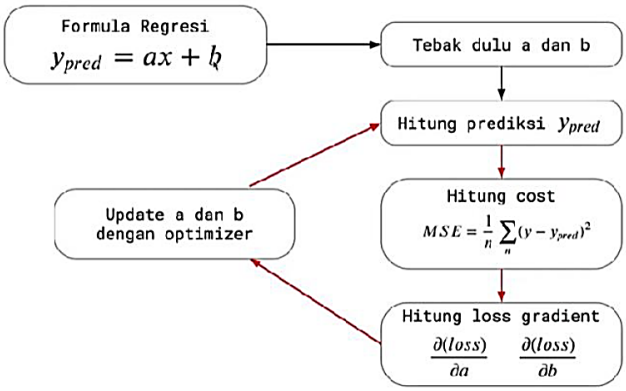
Gambar 17 Contoh Neural Network Sederhana

Bagian-bagian utama neural networks:

1. Neuron (Nodes): Setiap neuron menerima input, mengolahnya menggunakan fungsi aktivasi, lalu menghasilkan output yang diteruskan ke neuron berikutnya.
2. Lapisan (Layers):
   - Input layer: Lapisan pertama yang menerima data input.
   - Hidden layer: Lapisan-lapisan di antara input dan output yang memproses data secara bertahap.
   - Output layer: Lapisan terakhir yang menghasilkan prediksi atau hasil dari neural network.
3. Bobot dan Bias (Weights and Biases): Setiap sambungan antar neuron memiliki bobot yang menunjukkan seberapa kuat hubungan antar neuron. Bias menambahkan fleksibilitas pada neuron untuk menyesuaikan output.
4. Fungsi Aktivasi: Fungsi matematis yang menentukan apakah neuron akan "menyala" (menghasilkan output). Contoh fungsi aktivasi adalah ReLU, sigmoid, dan tanh.
5. Backpropagation: Algoritma yang digunakan untuk menyesuaikan bobot dan bias berdasarkan kesalahan pada output jaringan, memungkinkan jaringan belajar dari data.

Cara kerja neural networks:

1. Feedforward: Data mengalir dari lapisan input ke lapisan output melalui hidden layers.
2. Training: Neural network dilatih menggunakan data contoh (training data). Jaringan belajar dengan memperbaiki bobotnya melalui proses backpropagation, untuk meminimalkan kesalahan antara prediksi dan hasil sebenarnya.
3. Testing: Setelah dilatih, neural network digunakan untuk membuat prediksi pada data baru.



Gambar 18 Alur Kerja Neural Networks

5.3 ALGORITMA UNSUPERVISED LEARNING

Algoritma Unsupervised Learning tidak menjadi fokus modul ini. Kasus-kasus tertentu memang memerlukan pembelajaran berbasis unsupervised. Sehingga pengenalan dua algoritma unsupervised learning dimaksudkan untuk mengenalkan konsep dasarnya pada anda. Dua hal umum yang biasanya dilakukan dengan unsupervised learning adalah Clustering dan Dimensionality Reduction. Penjelasan lebih lanjut anda bisa simak penjelasan berikut.

5.3.1 K-Means

K-Means, sebuah algoritma unsupervised learning, digunakan untuk mengelompokkan data ke dalam beberapa cluster K berdasarkan kesamaan karakteristik. Pada awalnya, langkah pertama adalah memilih K centroid (titik pusat cluster) secara acak. Selanjutnya, setiap titik data akan diberikan ke dalam kluster yang memiliki pusat terdekat, dihitung menggunakan metode jarak Euclidean. Setelah semua data point diberi tugas, centroid dihitung kembali sebagai hasil penjumlahan dari semua data point dalam cluster tersebut dibagi dengan jumlah data point yang ada. Proses ini diulang berulang kali sampai tercapai konvergensi, yang mana data point tidak lagi berpindah cluster atau perubahan centroid menjadi sangat kecil. Pada akhirnya, terbentuklah K cluster di mana data point dalam satu cluster memiliki tingkat kemiripan yang tinggi, sementara adanya perbedaan signifikan antar cluster.

5.3.2 PCA

Principal Component Analysis (PCA) adalah cara untuk menyederhanakan data yang rumit. Bayangkan kamu punya data dengan banyak kolom atau fitur, dan setiap kolom mewakili informasi tertentu. PCA membantu dengan menemukan cara untuk menggabungkan beberapa kolom ini menjadi kolom baru yang lebih sedikit, namun tetap menyimpan sebagian besar informasi penting. Jadi, alih-alih bekerja dengan data yang sangat banyak dan membingungkan, kamu bisa fokus hanya pada beberapa kolom yang paling penting. Ini sangat berguna untuk membuat analisis lebih mudah dan cepat, sambil tetap mempertahankan gambaran besar dari data yang ada.

BAB 6
PROYEK

Proyek ini akan menggunakan dataset titanic, dimana fitur-fitur atau kolom-kolom pada data tersebut akan dimodelkan untuk memprediksi selamat/tidak selamatnya seseorang dalam kecelakaan kapal.

Nomenklatur dataset:

PassengerId	ID penumpang (integer)
Survived	Survived or Not
Pclass	Class of Travel (1st class paling mahal)
Name	Name of Passenger
Sex	Gender
Age	
SibSp	Number of Sibling/Spouse aboard
Parch	Number of Parent/Child aboard
Ticket	
Fare	
Cabin	
Embarked	The port in which a passenger has embarked. C = Cherbourg S = Southampton Q = Queenstown

Gambar 19 Nomenklatur Dataset Titanic

6.1 Pengenalan Dan Persiapan Data

Sebelum memulai proyek anda, siapkan data titanic.csv. Buat notebook baru dengan jupyter notebook, dan jangan lupa environment sudah terinstall dengan benar. Sesudah membuat notebook baru,

Langkah ke -1 import package berikut.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from luwiji.knn import illustration, demo
```

Mari memahami review apa saja package yang kita import dan apa kegunaannya.

1. Numpy menyediakan fungsi yang siap pakai untuk memudahkan kita melakukan perhitungan saintifik seperti matriks, aljabar, statistik, dan sebagainya. Numpy ini digunakan karena lebih lengkap dan mudah dibandingkan modul math yang ada di python. Selain itu numpy juga lebih cepat karena menggunakan bahasa C dan sebagian python.
2. Pandas adalah library open source pada Python yang sering digunakan untuk memproses data yang meliputi pembersihan data, manipulasi data, hingga

melakukan analisis data. Ketika melakukan suatu analisis, kita tidak bisa menggunakan data mentah.

3. Matplotlib adalah library python paling populer untuk melakukan visualisasi data yang lebih menarik dan mudah dipahami sehingga matplotlib akan terasa lebih alami untuk dipelajari. Matplotlib disusun oleh John Hunter di tahun 2002, dan di desain agar dapat digunakan selayaknya menggunakan MATLAB. Matplotlib dapat digunakan untuk memvisualisasikan data secara 2D maupun 3D dan menghasilkan gambar berkualitas yang bahkan dapat anda simpan dalam berbagai format gambar, seperti JPEG dan PNG. Jika, anda menggunakan python script maka anda perlu menginstall matplotlib terlebih dahulu. Tapi, jika kebetulan anda menggunakan python melalui anaconda meliputi spyder dan jupyter notebook maka, anda tidak perlu menginstalnya lagi karena sudah menjadi built-in library.
4. LuWiji adalah paket yang penuh dengan ilustrasi dan widget pembelajaran mesin. Ini dirancang untuk membantu instruktur dalam mengajarkan pembelajaran mesin

Langkah ke-2 setelah import package adalah, import dataset. Dataset disini berupa titanic.csv.

```
df=pd.read_csv("titanic.csv", index_col="PassengerId") #membaca data
df.head() #memeperlihatkan data bagian atasnya saja
```

Untuk membaca data, digunakan pandas yang disingkat pd. Kemudain pilih datasetnya dan pilih indeks “PassengerId” (ingat index tidak boleh masuk dalam pemodelan). Jika benar anda akan mendapatkan hasil sebagai berikut.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Gambar 20 Dataset Titanic dalam Tabular

Langkah ke -3 Hapus data yang tidak berpengaruh terhadap model. Kalau kita perhatikan kolom nama dan nomor tiket sebenarnya tidak berpengaruh terhadap keselamatan seseorang, maka dapat kita buang dengan cara.

```
df.drop(columns=["Name", "Ticket"], inplace=True)
```

Untuk membuang data ini sebenarnya tergantung dari bagaimana anda atau orang yang ahli menginterpretasikannya.

Langkah ke -4 Lihat data yang kosong. Melihat data yang kosong atau bolong adalah hal yang penting karena berpengaruh pada hasil dari model. Terlalu banyak data

kosong menyebabkan prediksi menjadi salah, maka dari itu perlu penanganan untuk data yang bolong. Salah satu cara penanganan untuk data bolong adalah dengan cara menghapus keseluruhan baris data atau diganti dengan nilai rata-rata data. Tapi jika data yang bolong terlalu banyak, maka lebih baik baris data dihapus semuanya karena rata-ratanya tidak bisa mewakili semua data yang bolong.

Untuk melihat data bolong gunakan baris kode berikut,

```
df.isnull().sum()
```

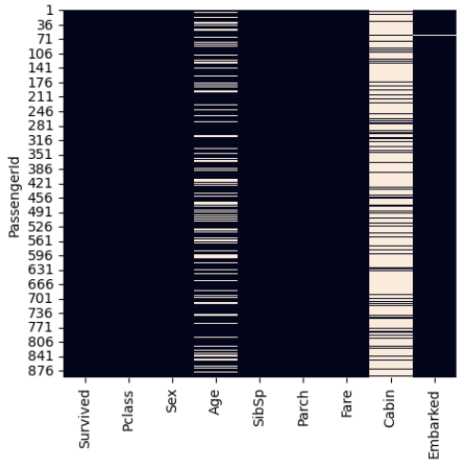
Hasilnya adalah:

```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Cabin       687
Embarked      2
dtype: int64
```

Gambar 21 Cek data Kosong

Selain itu kita juga dapat menggunakan visualisasi dengan menggunakan library seaborn untuk melihat data yang bolong.

```
import seaborn as sns
sns.heatmap(df.isnull())
```



Gambar 22 Cek Data Kosong Dengan Seaborn

Dari data tersebut dapat dilihat bahwa banyak nilai yang kosong pada kolom Age dan Cabin sehingga kolom ini dapat kita hapus dengan cara berikut.

```
df.drop(columns=['Age', 'Cabin'], inplace=True)
```

Setelah menghapus data tersebut, mari kita masukan(impute) data Embarked karena masih ada sedikit data kosong pada Embarked. Sebelumnya kita cari dulu nilai yang paling masuk akal untuk dimasukan dengan cara:

```
df.Embarked.value_counts()
```

Hasilnya adalah:

```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

Gambar 23 Cek Total Data

Kita masukkan “S” sebagai data di Embarked, nilai ini dimasukkan karena yang paling banyak diantara yang lain. Berikut adalah baris kodenya.

```
df.fillna({"Embarked":"S"}, inplace=True)
```

dengan memasukan data tersebut maka lengkaplah sudah data kita. Kita dapat melihat datanya lagi untuk mengecek apakah semua data sudah terisi atau belum dengan menggunakan seaborn seperti sebelumnya.

6.2 Persiapan Fitur dan Target Untuk Model

Perlu dipahami dalam memodelkan sesuatu, terdapat kolom fitur dan kolom target. Contohnya pada gambar di bawah ini.

Feature Variables

Target Variable

sepal-length	sepal-width	petal-length	petal-width	class
5.1	3.5	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
5.9	3.2	4.8	1.8	Iris-versicolor
6.8	2.8	4.8	1.4	Iris-versicolor
6.9	3.2	5.7	2.3	Iris-virginica
7.4	2.8	6.1	1.9	Iris-virginica
6.2	2.8	4.8	1.8	?

Gambar 24 Ilustrasi Fitur dan Target

Kolom fitur adalah kolom yang digunakan untuk memprediksi target. Sedangkan kolom target adalah hal yang ingin kita prediksi. Pada kasus dataset titanic ini, targetnya adalah selamat/tidak selamat "Survived", sedangkan fiturnya adalah kolom selain "Survived". Biasanya fitur dinamai dengan X(kapital) sedangkan target dinamai dengan y(kecil).

```
X= df.drop(columns="Survived") #input feature yang dimasukan
ke model (adalah semua kategori selain survived)
y= df.Survived #target

X=pd.get_dummies(X, columns=["Pclass", "Sex", "Embarked"])
#untuk mengubah data kategorikal menjadi numerikal

X
```

6.3 Pemodelan dengan KNN menggunakan Scikit-Learn

Selanjutnya kita bisa training data titanic yang sudah kita siapkan sebelumnya.

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X,y)
knn.score(X,y)
```

Dari hasil training dengan algoritme KNN, anda akan mendapatkan score sekitar 87% atau lebih, akan berbeda setiap kita melatih model.

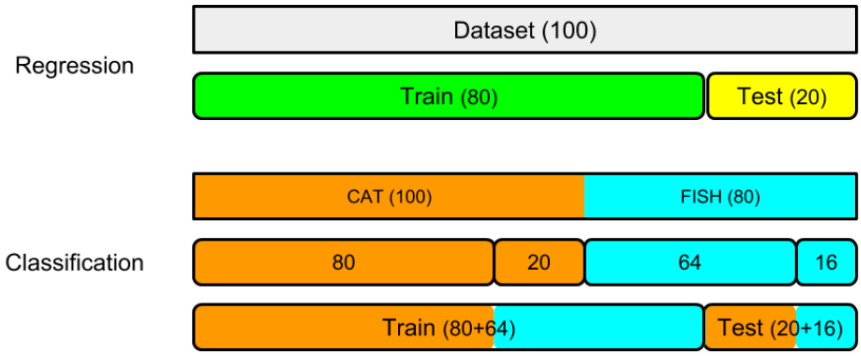
Score ini adalah berupa score training(latihan) model oleh mesin. Kita masih belum bisa memvalidasi apakah mesin sudah bisa memprediksi dengan baik jika kita buat data baru untuk di tes pada model yang sudah kita latih. Maka dari itu kita sebenarnya tidak bisa menggunakan semua data untuk training, tapi perlu adanya data splitting (pembagian data). Pembagian data ini berupa data latihan(training) dan ujian(test). Agar mesin bisa berlatih membuat model, kemudian di uji langsung dengan data tes.

6.4 Datasplitting

Silahkan gunakan baris kode berikut untuk menampilkan ilustrasi dari datasplitting.

```
from sklearn.model_selection import train_test_split
illustration.train_test_split
```

kita akan mendapatkan ilustrasi berikut.



Gambar 25 Datasplitting

Ada perbedaan jika kita ingin membagi data antara regresi dan klasifikasi. Pada regresi data di acak dan kemudian di-*split* langsung, biasanya 80% data training(latihan) dan 20% data ujian. Sedangkan pada klasifikasi, karena datanya berupa kategori maka, tiap kategori akan dibagi masing-masing seperti ilustrasi di atas. Hal ini menghindari tidak seimbangnya data training maupun data tes. Misal sesuai ilustrasi data cat dan fish di atas, jika kita acak dan campur saja cat dan fish, kemudian datanya kita split maka ada kemungkinan data tes atau ujian hanya berupa fish saja atau cat saja. Datasplitting ini di sebut stratify split.

Kita lanjutkan ke kode berikutnya untuk split data.

```
X= df.drop(columns="Survived") #input/feature yang dimasukan
ke model (adalah semua kategori selain survived)
y= df.Survived #target

X=pd.get_dummies(X, columns=["Pclass", "Sex", "Embarked"])

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, stratify=y, random_state=42) #untuk clasifikasi
ini menggunakan statify split

X_train.shape, X_test.shape, y_train.shape, y_test.shape # cek
apakah data test terlalu sedikit
```

Lalu kita train dan uji data yang sudah kita split.

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
knn.score(X_train, y_train), knn.score(X_test, y_test)
Hasil:
```

```
(0.91, 0.72)
```

Maka didapatkan score 91% akurasi untuk training dan 72% akurasi untuk ujian. Dari sini kita dapat mengetahui bahwa hasil ujian masih jauh lebih rendah dibandingkan hasil latihan. Ini bisa kita tingkatkan lagi pada bahasan berikutnya.

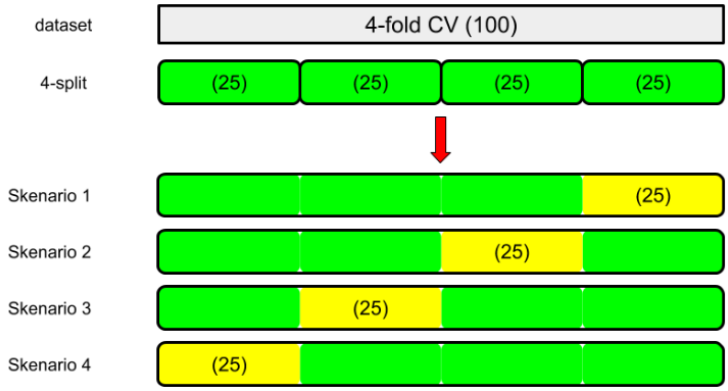
Dari data spliting yang sudah kita lakukan, sebenarnya masih ada kekurangannya. Kekurangannya adalah bisa jadi model saat latihan(training) mendapat data yang mudah, kemudian saat ujian(test) model mendapat data yang sulit sehingga score akhir ujian akan lebih rendah daripada score saat latihan. Ada efek mujur saat kita training maupun test. Hal ini dapat diatasi dengan K-Fold Cross Validation.

6.5 K-Fold Cross Validation

Silahkan copy baris kode berikut,

```
from sklearn.model_selection import cross_val_score
illustration.kfold_cv
```

Anda akan mendapatkan ilustrasi dari K-Fold Cross Validation seperti berikut.



Gambar 26 K-Fold Cross Validation

Misal dataset mempunyai total 100 data. K-Fold Cross Validation membagi data tersebut sesuai kebutuhan kita. Dalam gambar jumlah pembagian yang digunakan adalah 4. Sehingga dataset training 75% dan dataset test 25%. Berbeda dengan splitting biasa, K-Fold Cross Validation akan mentraining datanya berulang kali dengan data yang berbeda-beda. Pada ilustrasi kita bisa lihat, training dan test yang terjadi adalah 4 skenario. Dari semua skenario ini kemudian score hasil training dan score hasil test akan dirata-rata.

```
knn = KNeighborsClassifier(n_neighbors=1)
```

```
cross_val_score(knn, X, y, cv=5) #cv berapa fold (lipatan)
Hasil:
array([0.636, 0.634, 0.833, 0.73033, 0.71348])
```

Hasil dari rata-rata K-Fold Cross Validation adalah seperti berikut.

```
cross_val_score(knn, X, y, cv=5).mean()
Hasil : 0.70377
```

Dari hasil yang sudah didapat, kita masih bisa memperbaikinya dengan melakukan improvement bagian data ataupun improvement bagian model dengan parameter tuning.

6.6 Improvement Data Melalui Feature Scaling

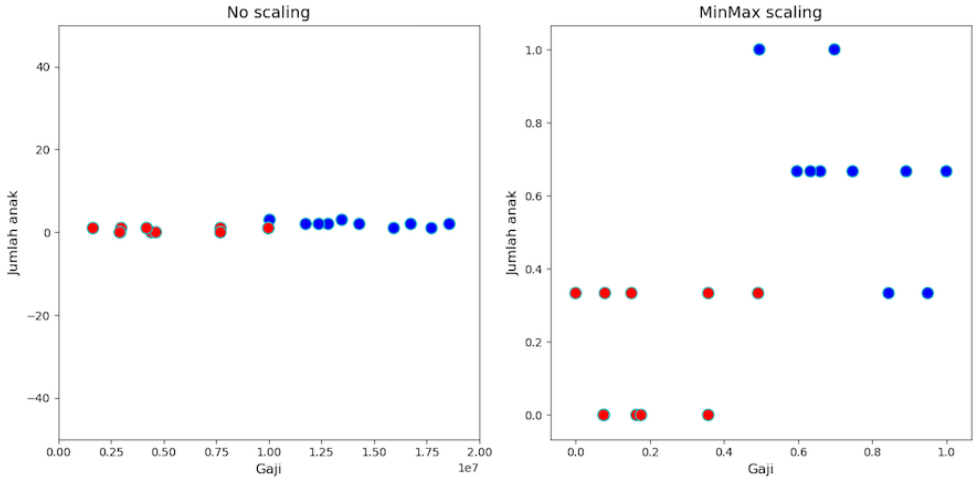
Scaling data adalah hal yang penting dilakukan jika dataset kita memiliki orde yang berbeda jauh. Misal ada data hubungan antara jumlah anak dan gaji. Jika dicari distance menggunakan KNN nya maka ini akan terlalu jauh karena orde gaji yang terlalu tinggi, maka diperlukan feature scaling. Tidak semua algoritma membutuhkan feature scaling. KNN adalah contoh algoritma yang bisa memanfaatkan feature scaling. Langsung saja kita coba, kita akan mencoba dengan fungsi scaler MinMax yang sudah disediakan skit-learn, yang mana scaler ini akan mencari nilai minimum dan maksimum tiap fitur. Kemudian setelah itu di scaling.

```
from sklearn.preprocessing import MinMaxScaler
#mencari nilai max dan min
scaler = MinMaxScaler()
scaler.fit(X_train)

#membungkus hasil scaler dengan variabel
X_train_scaled=scaler.transform(X_train)
X_test_scaled=scaler.transform(X_test)

# training dan testing menggunakan KNN
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train_scaled, y_train)

#Score KNN
knn.score(X_train_scaled, y_train), knn.score(X_test_scaled,
y_test)

Hasil:
```

```
(0.8890449438202247, 0.6983240223463687)
```

Untuk lebih memahami, mari kita lakukan demo. Demo ini menggunakan data jumlah anak dan gaji. Gaji biasanya menggunakan angka jutaan. Sedangkan jumlah anak tidak sampai puluhan. Maka dari itu diperlukan scaling.

```
demo.knn_scaling()
illustration.knn_scaling
```



Gambar 27 Scaling Data

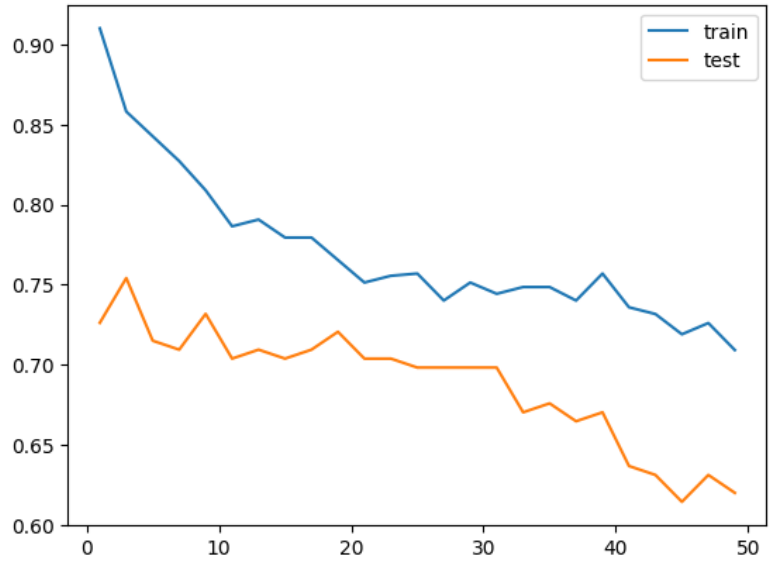
Jika kita tidak melakukan scaling maka jarak antar data terlihat sangat dekat pada gambar di atas. Hal ini akan sangat mempengaruhi bagaimana KNN memodelkan data karena jarak menjadi parameter yang penting.

6.7 Improvement Melalui Parameter Tuning

Untuk mendapatkan score yang lebih baik, kita perlu tuning parameter algoritma kita. Pada KNN parameter yang dapat kita tuning adalah jumlah tetangga, metode perhitungan jarak, dan pembobotan yang berbeda. Tapi bagaimana kita melakukannya, padahal kita belum tahu berapa parameter yang memberikan hasil yang terbaik? Maka kita bisa membuat perulangan untuk mencoba semua kemungkinan. Berikut adalah tuning jumlah tetangga tanpa kita scaling.

```
neighbor =range(1, 51, 2) # menggunakan range 1-51 tetangga
train_score = [] #membuat list score yang disimpan dalam
variabel
test_score = []
```

```
for k in neighbor: #ini digunakan untuk mencoba semua
kemungkinan tetangga
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    train_score.append(knn.score(X_train, y_train)) #memasukan
score ke list
    test_score.append(knn.score(X_test, y_test))

plt.plot(neighbor, train_score, label="train") #plot score dan
jumlah tetangga
plt.plot(neighbor, test_score, label="test")
plt.legend()
```



Gambar 28 Train Test Score

Dari tabel kita dapatkan score maksimal berada pada jumlah tetangga 3. Kita bisa print saja dengan baris kode berikut.

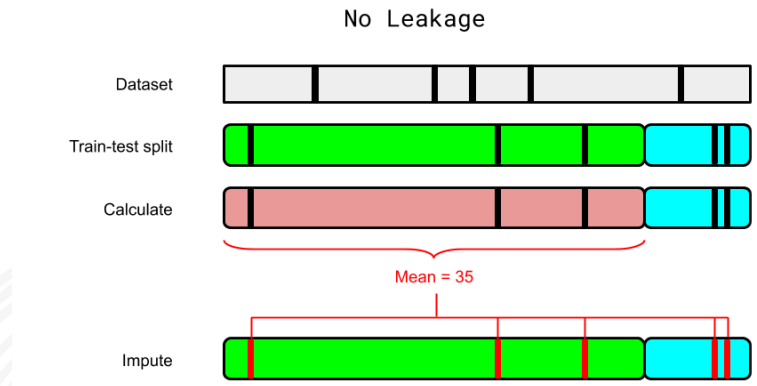
```
print(f"Max test score: {np.max(test_score)}")
print(f"n_neighbor: {neighbor[np.argmax(test_score)]}")
```

Hasilnya, Max test score: 0.754 dengan n_neighbor: 3

6.8 Avoid Data Leakage

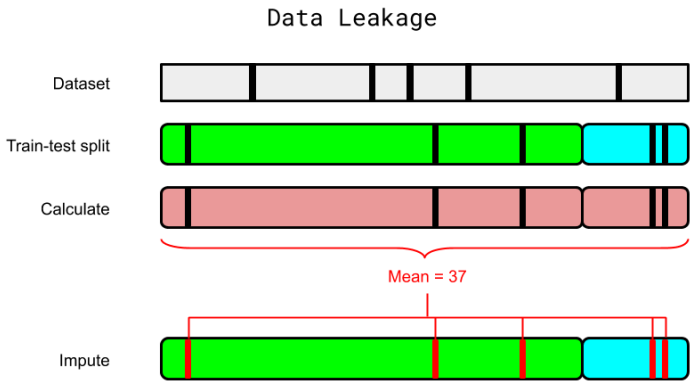
Apa itu data leakage atau kebocoran data. Kebocoran data terjadi ketika data yang seharusnya tersembunyi malah diketahui mesin. Data leakage bisa terjadi karena

beberapa hal yaitu imputation, scaling, atau k-fold. Pada gambar di bawah adalah contoh kebocoran data pada data tes dengan imputation. Seharusnya mesin tidak mengetahui data tes sama sekali, sehingga hasil tes murni karena model itu sendiri. Maka dari itu imputation dilakukan dengan menginput rata-rata dari data training saja, tanpa memasukkan data tes.



-

Gambar 29 No Data Leakage



Gambar 30 Data Leakage

Bisa kita rangkum. Bocornya informasi paling sering terjadi ketika:

1. Imputation
    - Kita malah impute menggunakan informasi dari seluruh data, harusnya hanya train data saja
    - Spit sebelum imputation, kemudian impute dengan rata2 train data saja
2. Scaling

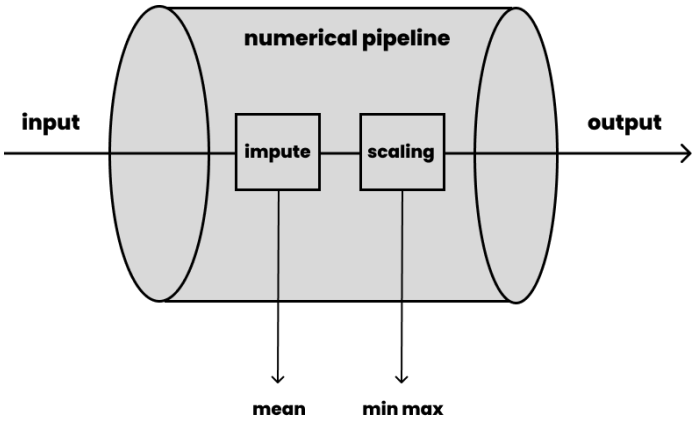
- Kita malah scaling menggunakan informasi dari seluruh data, harusnya hanya train data saja
- Solusi: fit_transform pada train, transform pada test

3. K-fold pada train-test-split
    - K-k-fold menyebabkan data kita menyentuh data test yang seharusnya tersembunyi
    - Ibaratnya seperti kita boleh retake ujian, lama-kelamaan kita akan tahu soal ujiannya
    - Solusi: train-val-test

Maka dari itu kita perlu menggunakan splitting yang lebih baik, coba lihat gambar berikut. Splitting dilakukan diawal sebelum dilakukan k-fold biru(tes) dan hijau(train). Kemudian K-Fold dilakukan hanya pada data yang berwarna hijau. Data ini kemudian dibagi lagi untuk sebagai data tes awal(validation) sehingga data leakage hanya terjadi pada data yang hijau ini. Dari skenario ini, data tes sesungguhnya tidak akan pernah dilihat oleh mesin dan tidak ada kebocoran data.

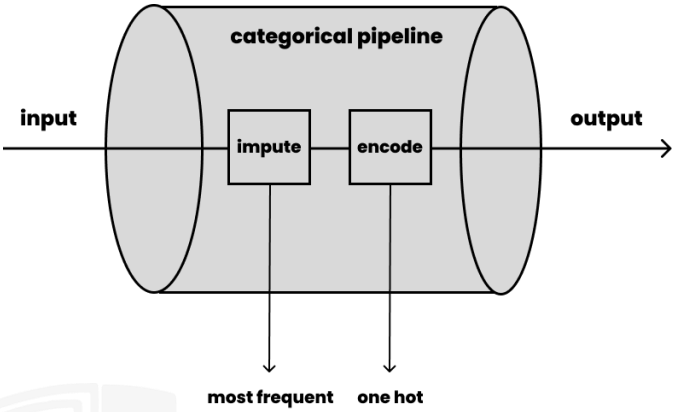
6.9 Menggunakan Pipeline Untuk Memudahkan Pekerjaan

Daripada melakukan koding satu persatu. Akan lebih mudah jika kita menggunakan yang namanya pipeline. Seperti namanya pipa, akan ada input yang masuk ke pipa. Setelah itu di dalam pipa akan ada proses yang akan mengolah dataset dan hasilnya akan disimpan dalam pipeline. Ini akan jauh lebih memudahkan karena kita tidak perlu menyimpan tiap output proses dan dimasukkan ke proses yang lain. Pipeline sudah disediakan oleh sckit-learn. Kita hanya menulis baris kode apasaja yang kita butuhkan untuk prosesnya. Selain itu kita juga perlu membedakan antara numerical dan kategorikal pipeline.



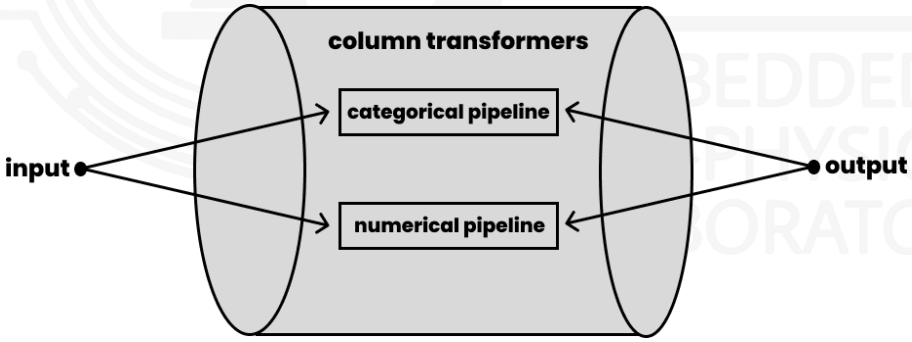
Gambar 31 Numerical Pipeline

Untuk numerical pipeline kita seperti proses sebelumnya bahwa kita perlu meng impute data bolong dan scaling jika diperlukan.



Gambar 32 Categorical Pipeline

Untuk categorical pipeline kita perlu meng impute data bolong dan penting adalah encode data dari categorical menjadi numerical agar bisa dibaca oleh mesin. Encoder yang biasa dipakai adalah onehot encoder, ini juga sudah disediakan oleh scikit-learn. Kemudian kita bisa membungkus numerical pipeline dan categorical pipeline dengan pipeline yang baru atau disebut column transformer.



Gambar 33 Column Transformer

Mari kita tulis kodenya. Pertama kita perlu meng import package yang perlu dari scikit learn.

```
#Preprocessor

from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
```

Kemudian kita membuat pipline yang diisi dengan proses-proses yang diperlukan. Pada data numerical kita perlu impute data bolong dan juga scaling data. Jika anda belum paham impute dan scaling bisa dibaca ulang di bagian tersebut. Proses ini diperlukan jika memang ada data yang bolong dan perlu di scale. Jika tidak perlu maka anda bisa menghapusnya. Strategy mean(rata-rata) artinya data bolong akan dimasukkan data yang sudah dirata-rata.

```
numerical_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="mean")),
    ("scaler", MinMaxScaler())
    ])
```

Pada categorical pipeline, proses pertama memasukan data bolong (impute) tapi perbedaan nya dengan strategy most frequent. Artinya data yang dimasukan untuk data bolong adalah data yang paling sering muncul. Kemudian encoder dengan menggunakan onehot. Encoder berarti mengubah data kategori menjadi numerical. Misal pada kasus dataset titanic ini encoder berupa selamat=1 dan tidak selamat=0.

```
categorical_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder())
    ])
```

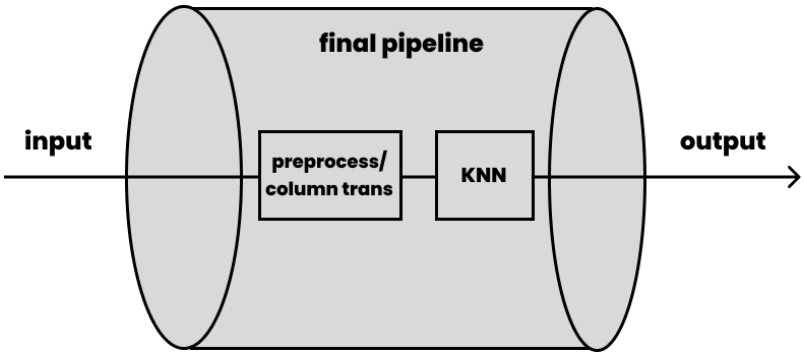
Dari kedua pipeline tersebut, kita masukan lagi ke dalam pipeline columntransformer. Sehingga tiap fitur-fitur yang ada pada dataset numeric maupun kategorik dapat kita pisahkan sesuai dengan pipelinnya.

```
from sklearn.compose import ColumnTransformer
```

Fitur apa saja yang termasuk numerik dan kategorik dapat kita masukan ke pipeline.

```
preprocessor = ColumnTransformer([
    ("numeric", numerical_pipeline,["SibSp", "Parch","Fare"]),
#kita masukan fitur yang numerik yaitu "SibSp", "Parch","Fare"

    ("categoric", categorical_pipeline,["Pclass",
"Sex","Embarked"])
]) # memasukkan fitur kategorik
```

Semua pipeline yang sudah kita buat dapat kita bungkus lagi menggunakan pipeline terakhir yang di dalamnya sudah ada algoritma kita. Algoritma KNN menjadi pipeline terakhir untuk proses ini.



Gambar 34 Final Pipeline

```
from sklearn.neighbors import KNeighborsClassifier

pipeline = Pipeline([
    ("prep", preprocessor), #memasukkan preprocessor yang kita
sudah buat
    ("algo", KNeighborsClassifier()) #Algoritma yang kita
gunakan
])
```

Semua pipeline kita sudah siap. Tapi ada suatu hal yang belum dilakukan. Yaitu bagaimana kita mencoba semua kemungkinan parameter KNN sehingga mendapatkan score terbaik. Hal itu akan dibahas di bagian berikutnya.

6.10 Tuning dan Cross Validation dengan Tools GridSearchCV

Kita akan menggunakan tools GridSearch Cross Validation, semua kemungkinan parameter akan dicoba menggunakan tools ini. Disinilah gunanya pipeline karena kita tidak perlu menyimpan semua hasil dari Cross Validation.

Dengan tools gridsearch ini, kita akan mendapatkan nilai cross validation terbaik. Maka menggunakan gridsearch adalah strategi terbaik untuk mendapatkan score tinggi dengan waktu yang cepat. Misal score validation terbaik ada dengan jumlah tetangga 3 dengan pembobotan distance, maka itulah yang menjadi model yang kita pakai.

Untuk Menggunakannya kita perlu membuat kombinasi parameter apa saja yang dilakukan. Parameter ini dapat dilihat melalui dokumentasi scikit learn. Untuk KNN

sendiri, yang perlu di tuning adalah jumlah tetangga “algo__n_neighbor”, pembobotan, dan metode jarak. Syntax ini sudah di tentukan oleh scikit learn.

```
from sklearn.model_selection import GridSearchCV
parameter = {
    "algo__n_neighbors": range(1,51,2), #banyaknya tetangga
    "algo__weights":["uniform","distance"], #pembobotan
    "algo__p":[1,2] #jenis jarak KNN yaitu 1=manhatan dan
2=euclidian
}

model = GridSearchCV(pipeline, parameter, cv=3, n_jobs=-1,
verbose= 1) #cv artinya berapa fold(pembagian data), n_jobs=-1
artinya berapa core cpu yang digunakan -1 artinya semua core,
kemudian verbose artinya tampilkan report

model.fit(X_train, y_train)
model.score(X_train, y_train), model.score(X_test, y_test)

Hasil:
(0.8174157303370787, 0.7821229050279329)
```

Kita bisa lihat saat kita menggunakan GridSearch, kita hanya perlu memasukkan algoritma kita yang sudah dibungkus pipeline. Kemudian parameter yang sudah kita buat. Dari sini workflow kita untuk memodelkan data sudah selesai. Tinggal save dan load model. Hal yang sama dilakukan untuk algoritma lain, misalnya SVM, Random Forest, Atau XGBoost. Kita hanya perlu mengganti parameter yang perlu, kemudian algoritma yang digunakan. Tentu ada hal yang lain perlu diperhatikan tergantung pada algoritmanya. Selebihnya alur kerjanya sama.

6.11 Prediction, Save and Load Model

Membuat data baru untuk diprediksi harus menggunakan format yang sama seperti modelnya. Karena input saat kita melatih model adalah dataframe, maka kita harus membuatnya persis sama. Demikian pula fiturnya harus sama seperti saat kita memasukkan input ke model yang kita latih. Mari kita contohkan dengan dataset berikut.

```
from luwiji.knn import illustration, demo
illustration.jack_and_rose
```

Name: Rose DeWitt Bukater
With: Mother and Fiance
Class: 1st class
Sex: Female
Boarded: Southampton

source: https://jamescameronstitanic.fandom.com/wiki/Rose_DeWitt_Bukater

Name: Jack Dawson
With: -
Class: 3rd class
Sex: Male
Boarded: Southampton

source: https://jamescameronstitanic.fandom.com/wiki/Jack_Dawson

Gambar 35 Jack and Rose Pediction

Dari data tersebut, kita masukan dulu berdasarkan fitur yang sudah kita modelkan. Kita perlu mengecek fitur apa saja yang perlu kita masukkan ke model dengan melihat data input sebelumnya.

```
X.iloc[0:1]
```

Hasilnya:

PassengerId	Pclass	Sex	SibSp	Parch	Fare	Embarked
1	3	male	1	0	7.25	S

Kemudian kita buat data baru sesuai fitur yang ada dengan kode berikut. Lalu kita prediksi.

```
data =[
    [1, "female", 1, 1, 80, "S"],
    [1, "male", 0, 0, 5, "S"]
]

X_pred = pd.DataFrame(data, index=["Rose", "Jake"],
columns=X.columns)
X_pred
```

Data yang sudah kita buat, dibungkus dengan variabel X_pred dengan index sesuai namanya. Setelah itu data siap untuk di prediksi. Kita dapat lihat prediksinya seperti berikut.

```
X_pred["Survived"] = model.predict(X_pred)
X_pred
```

	Pclass	Sex	SibSp	Parch	Fare	Embarked	Survived
Rose	1	female	1	1	80	S	1
Jake	1	male	0	0	5	S	0

Dari hasil prediksi Jake tidak selamat sedangkan Rose selamat. Dari sini kita sudah memprediksi model dengan data baru yang kita buat sendiri. Namun jika ada dataset yang berupa format csv, maka bisa kita masukan ke variabel X_pred untuk dibaca, kemudian diprediksi dengan cara yang sama.

Hasil model tentunya dapat kita simpan, disini kita menggunakan package jcopml dalam proses save dan load model. Setelah save model, akan ada folder dengan nama "model" yang muncul. Disanalah file model disimpan.

```
from jcopml.utils import save_model
save_model(model, "knn_titanic.pkl")
save_model(model.best_estimator_, "knn_titanic_small.pkl")
```

Untuk load model dan prediksi data kita. Maka kita bisa lakukan berikut.

```
from jcopml.utils import load_model
model_load = load_model("model/knn_titanic_small.pkl")
model_load.predict(X_pred)
```

X_pred adalah variabel yang sudah kita masukan data baru yang ingin kita prediksi tadi. Seperti yang disebutkan sebelumnya X_pred juga bisa kita isi dengan file csv yang sudah jadi, atau bisa juga format file lainnya yang didukung pandas.

DAFTAR PUSTAKA

Geron, Aurelien. 2019. Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow. O'Reilly Media, Inc

Wira, JCOP Untuk Indonesia. https://www.youtube.com/@JCOpUntukIndonesia

The Pecan Team. (2023, November 21). *Data Preparation for Machine Learning: The Ultimate Guide to Doing It Right*. Retrieved from pecan: https://www.pecan.ai/blog/data-preparation-for-machine-learning/