



MODUL PRAKTIKUM MACHINE LEARNING

Penyusun : Izmy Alwiah Musdar

PRODI INFORMATIKA
STMIK KHARISMA MAKASSAR

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT yang telah memberikan nikmat kesehatan dan kesempatan, sehingga penyusunan modul praktikum Machine Learning dapat terlaksana.

Dalam era yang semakin berkembang ini, teknologi menjadi salah satu hal yang sangat penting untuk dikembangkan. Salah satu teknologi yang sedang berkembang dengan pesat adalah Machine Learning. Dalam rangka mempersiapkan mahasiswa untuk menghadapi tantangan masa depan yang semakin kompleks, Prodi Informatika STMIK KHARISMA Makassar menyediakan mata kuliah pilihan praktikum Machine Learning yang diharapkan dapat membantu mahasiswa memahami dan menguasai konsep dasar dan aplikasi Machine Learning.

Modul praktikum Machine Learning ini dirancang untuk memberikan pemahaman yang komprehensif kepada mahasiswa tentang konsep dasar Machine Learning dan aplikasinya dalam berbagai bidang. Modul ini terdiri dari 9 materi, yaitu Pengantar Python, Dasar Machine Learning, Quiz, Regresi Linier, Decision Tree Classifier, Naive Bayes Classifier, Jaringan Syaraf Tiruan, Support Vector Machine, dan Clustering.

Materi-materi tersebut akan membantu mahasiswa untuk memahami konsep dasar Machine Learning dan penerapannya dalam berbagai situasi. Dalam modul praktikum ini, mahasiswa akan dilatih untuk menggunakan berbagai algoritma Machine Learning, mengimplementasikan model Machine Learning pada dataset, serta menganalisis dan mengevaluasi hasil dari model yang telah dibangun.

Dengan adanya modul praktikum ini, diharapkan dapat membantu mahasiswa Prodi Informatika STMIK KHARISMA Makassar untuk memahami konsep dan aplikasi Machine Learning dengan baik. Penyusunan modul ini masih banyak kekurangan. Oleh karena itu, kami mohon kritik dan saran dari para pembaca sehingga modul ini selanjutnya dapat tersusun dengan lebih baik.

Makassar, Oktober 2022

Penyusun

DAFTAR ISI

KATA PENGANTAR	1
DAFTAR ISI	2
MODUL I Pengantar Python 1	4
MODUL II Pengantar Python 2	16
MODUL III Dasar Machine Learning	30
MODUL IV Quiz	38
MODUL V Regresi Linier	41
MODUL VI Decision Tree Classifier	47
MODUL VII Naïve Bayes Classifier	53
MODUL VIII Jaringan Syaraf Tiruan 1	58
MODUL IX Jaringan Syaraf Tiruan 2	65
MODUL X Support Vector Machine	72
MODUL XI Clustering	77
DAFTAR PUSTAKA	78

MODUL I

PENGANTAR PYTHON

A. Pokok Bahasan

1. Instalasi Anaconda Navigator
2. Pengenalan Jupyter Notebook
3. Pengenalan Bahasa Pemrograman Python

B. Tujuan Pembelajaran

1. Mahasiswa mampu memasang IDE Anaconda Navigator
2. Mahasiswa mampu menjalankan Jupyter Notebook
3. Mahasiswa memperoleh gambaran tentang Bahasa Pemrograman Python
4. Mahasiswa mengetahui konsep-konsep penting pemrograman Python sehingga diperoleh pengetahuan dasar tentang membuat kode dengan Python.

C. Dasar Teori

I. Identifier

Terdapat beberapa entitas dalam Python yaitu kelas, fungsi, variabel. Entitas tersebut perlu diberi nama pada saat didefinisikan dalam program. Berikut tata cara penamaan entitas dalam Python :

- Nama bisa kombinasi dari huruf besar dan huruf kecil (a-z atau A-Z).
- Dapat menggunakan angka (0-9) atau underscore (_).
- Tidak bisa dimulai dengan angka. Contoh : 1Variabel (Tidak Valid), Variabel1 (Valid).
- Keyword dalam Python tidak bisa dijadikan nama variabel.
- Selain underscore(_), karakter khusus lain (!, @, dsb) tidak bisa digunakan dalam penamaan.

II. Kode Blok (Indentasi dan Suites)

Tata cara penulisan blok kode pada Python penting untuk diketahui. Terdapat dua konsep penulisan kode blok dalam Python, yaitu Indentasi dan Suites. Indentasi digunakan sebagai penanda blok kode. Setiap baris yang berada pada blok kode yang sama harus memiliki indentasi yang sama. Tidak sama dengan bahasa pemrograman lain dimana indentasi hanya untuk keperluan kerapian penulisan program, pada Python indentasi dibutuhkan untuk menunjukkan kepemilikan .

Suites adalah sekumpulan statement tunggal yang membentuk sebuah blok program. Sebuah header line diikuti dengan sebuah blok kode membentuk statement yang kompleks seperti if, while, def, dan class. Header line terdiri dari keyword diikuti dengan titik dua (:) dan selanjutnya beberapa statemen membentuk sebuah suites.

Contoh penggunaan indentasi dan suite yang benar

```
: x=1
  if (x==1):
    print ("x bernilai 1")
  else :
    print ("x bernilai selain 1")
x bernilai 1
```

III. Tipe Objek Dasar

Semua data pada Python direpresentasikan dengan objek atau relasi antar objek. Setiap objek memiliki identitas, tipe dan nilai. Berikut ini objek dasar yang ada di Python.

Tipe Objek	Contoh Data
None	None
Boolean	True, False
Integer	-1, 0, 1
Long	1L, 98999L
Float	3,3333
Complex	2i+5j
String	"Saya", 'Dia'
Tuple	kosong = () (1, True, 'F')
List	kosong = [] [1, True, 'F']
Set	kosong = set() set(1, True, 'F')
Dictionary	kosong = {} {'1':'A', '2':'AA', True=1, False=0}
File	f = open('filename', 'rb')

IV. Operator Dasar

Operator ini digunakan dalam operasi matematika yang melibatkan angka seperti penjumlahan, pengurangan, perkalian, pembagian, dsb. Berikut ini daftar operator aritmatika :

Operator	Deskripsinya	Contoh
+	Penjumlahan	x + y
-	Pengurangan	x - y
*	Perkalian	x * y
/	Pembagian	x / y
%	Modulus	x % y

** Eksponen	Eksponensiasi	x ** 2
//	Pembagian dengan pembulatan	9 // 2 = 4

V. Operator perbandingan atau relasi

Operator ini digunakan untuk membandingkan antar nilai. Hasil operasi dengan operator ini adalah True dan False. Berikut ini operator perbandingan atau relasi di Python :

Operator	Keterangan	Contoh
==	Bernilai True apabila operand yang dibandingkan bernilai sama	Jika x = 2, y = 2 maka (x==y) mengembalikan nilai True
!=	Bernilai True apabila operand yang dibandingkan bernilai tidak sama	Jika x = 3, y = 2 maka (x!=y) mengembalikan nilai True
>	Bernilai True apabila nilai operand sebelah kiri lebih besar dari nilai operand kanan.	Jika x = 3, y = 2 maka (x>y) mengembalikan nilai True
<	Bernilai True apabila nilai operand sebelah kiri lebih kecil dari nilai operand kanan.	Jika x = 3, y = 2 maka (x<y) mengembalikan nilai False
>=	Bernilai True apabila nilai operand sebelah kiri lebih besar atau sama dengan nilai operand kanan.	Jika x = 2, y = 2 maka (x>=y) mengembalikan nilai True
<=	Bernilai True apabila nilai operand sebelah kiri lebih kecil atau sama dengan nilai operand kanan.	Jika x = 3, y = 2 maka (y<=x) mengembalikan nilai True

VI. Operator Penugasan

Operator penugasan digunakan untuk memberikan nilai ke dalam variabel. Contoh untuk memberikan nilai 2 untuk variabel x, maka dituliskan dengan x = 2. Operator yang digunakan adalah “=”. Terdapat variasi untuk menuliskan penugasan, contohnya untuk menuliskan x = x + 2 maka bisa dituliskan dengan x += 2. Format pengusutan seperti ini juga berlaku untuk operator aritmatika lainnya seperti x *= 2, dst.

VII. Operator Logika

Operator logika terdiri dari AND, OR, dan NOT. Operator-operator ini digunakan untuk mengecek dua buah variable pada kondisi tertentu. Hasilnya bernilai TRUE atau FALSE.

Operator	Keterangan	Contoh
AND	Jika kedua operand TRUE maka kondisi bernilai TRUE	Jika var1=True dan var2=False, maka (var1 AND var2) = False
OR	Jika salah satu operand TRUE maka kondisi bernilai TRUE	Jika var1=True dan var2=False, maka (var1 OR var2) = True
NOT	Bernilai kebalikan dari operand	Jika var1=True, maka (NOT var1) = False

VIII. Operator Keanggotaan

Operator keanggotaan digunakan untuk mengetahui apakah sebuah nilai terdapat pada kumpulan nilai dalam string, list, tuple, set atau dictionary. Operator keanggotaan yang digunakan pada Python adalah **"in" dan "not in"**. Khusus untuk dictionary, operator keanggotaan hanya bisa digunakan untuk kunci, tidak bisa digunakan untuk nilainya.

IX. Struktur Kontrol

Struktur kontrol adalah suatu bagian kode yang bisa mengatur arah jalannya program berdasarkan kondisi tertentu. Terdapat dua jenis struktur kontrol dalam Python, yaitu seleksi dan perulangan.

Seleksi

Seleksi memungkinkan programmer untuk menjalankan aksi berdasarkan apabila kondisi tertentu terpenuhi. Terdapat dua cara untuk menyatakan seleksi, yaitu if dan if..else. Berikut contoh penggunaan seleksi dalam Python :

```
angka = 5
if (angka % 2 == 0) :
    print (angka, "adalah bilangan genap")
else :
    print (angka, "adalah bilangan ganjil")
```

Pada contoh di atas hanya terdapat dua kondisi. Apabila terdapat lebih dari dua kondisi, maka digunakan if bersarang. Contohnya sebagai berikut :

```
nilai = 75

if nilai >=80 :
    print ("A")
elif nilai >=70 :
    print ("B")
elif nilai >=60 :
    print ("C")
elif nilai >=50 :
    nilai ("D")
else :
    nilai ("E")
```

Perulangan

Perulangan merupakan pernyataan yang dapat digunakan untuk menjalankan baris program secara berulang sampai kondisi tertentu terpenuhi. Python menyatakan perulangan dengan dua cara yaitu **for** dan **while**. Berikut contoh penggunaan for dan while :

Contoh perulangan dengan For

```
huruf = ['A', 'B', 'C', 'D', 'E']

print("Mencetak semua huruf dengan for")

for h in huruf :
    print (h)
```

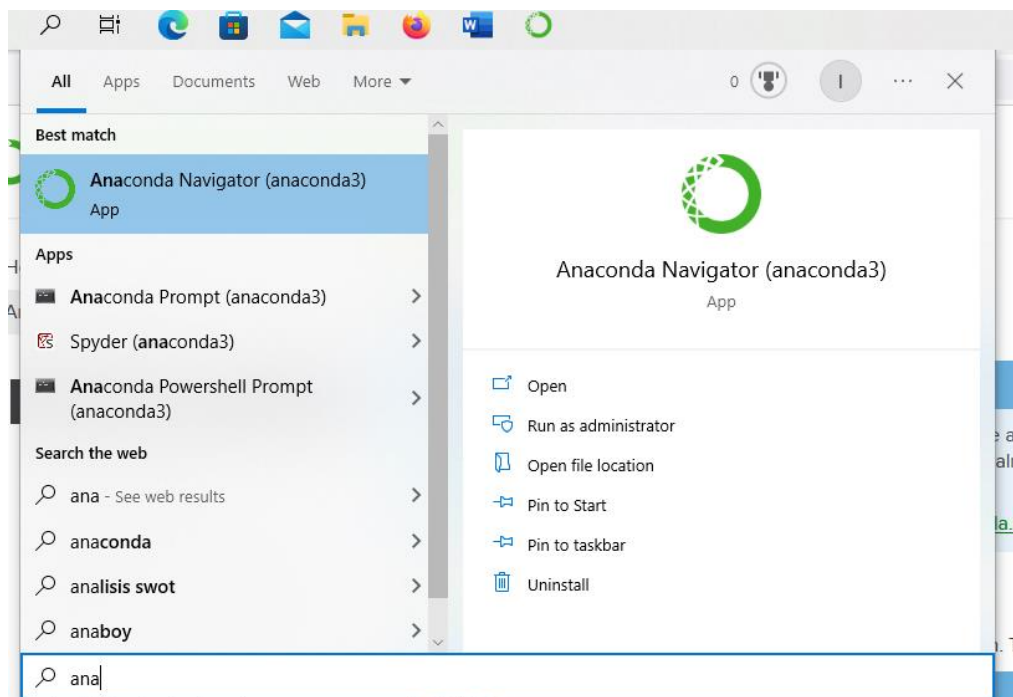
Contoh perulangan dengan while

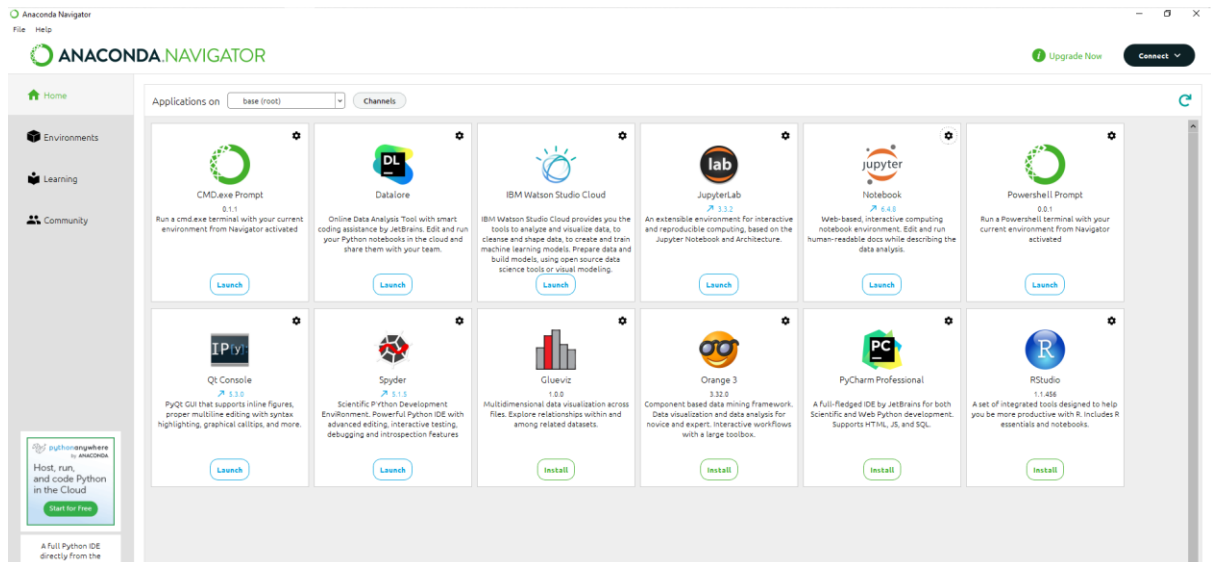
```
while (i < len(nilai1)) :  
    jumlah += nilai1[i]  
    i+=1
```

D. Tahapan Praktikum

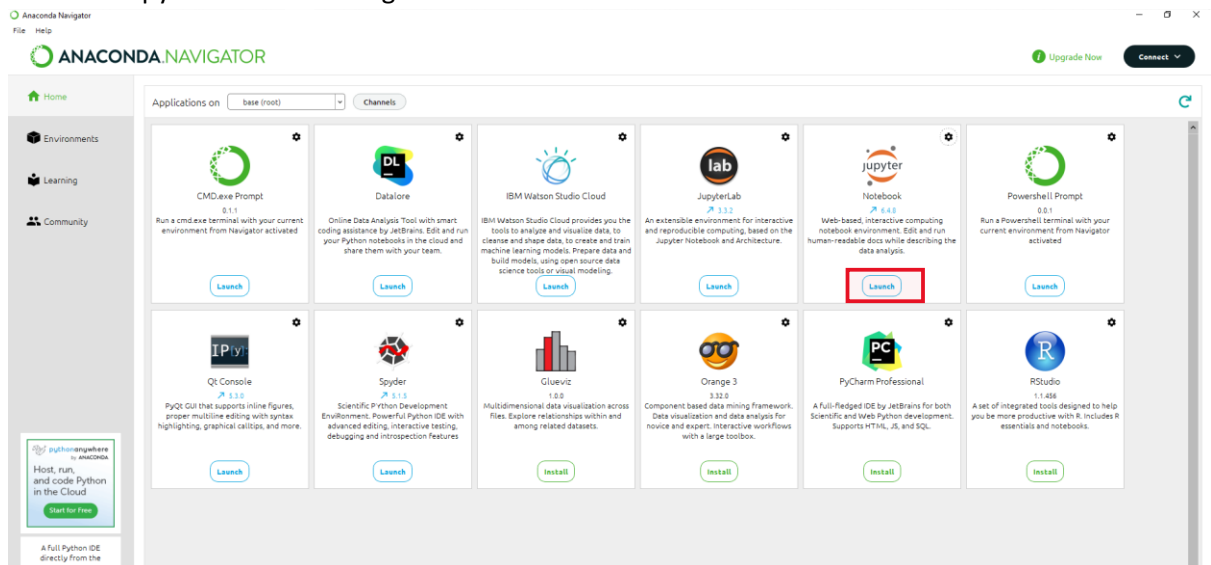
I. Instalasi IDE : Anaconda Navigator

1. Unduh installer dan baca tahapan memasang anaconda di <https://docs.anaconda.com/anaconda/install/windows/>
2. Jalankan Anaconda Navigator



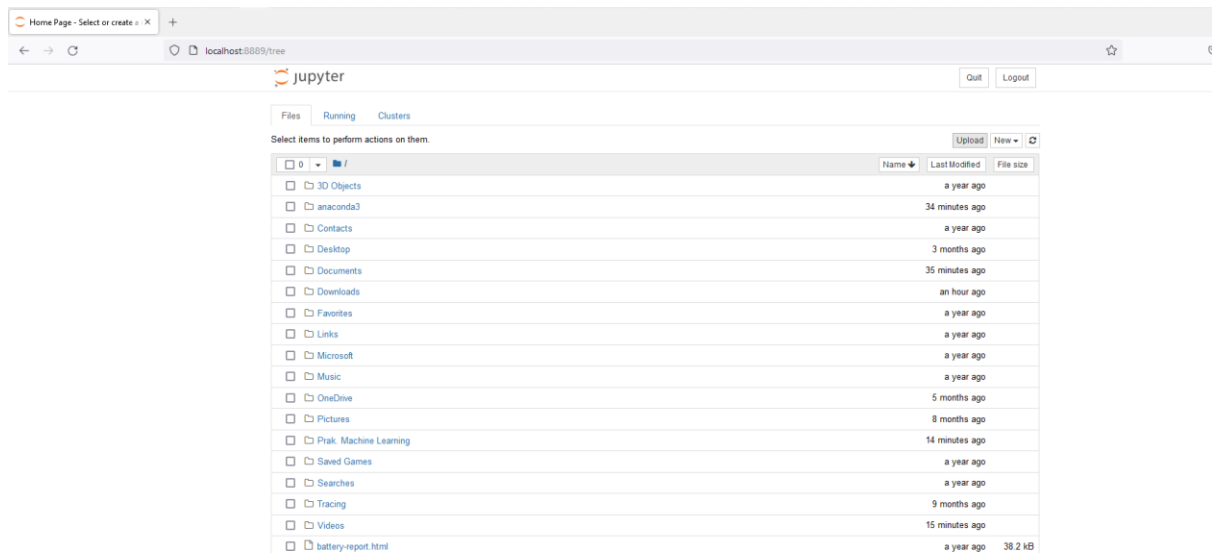


3. Jalankan Jupyter Notebook dengan memilih tombol Launch

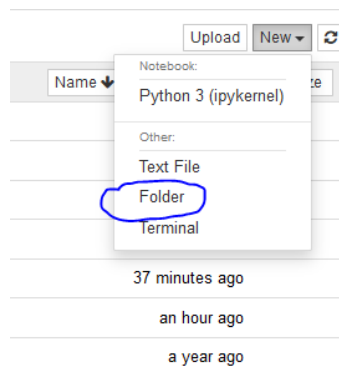


II. Membuat Proyek dengan Jupyter Notebook

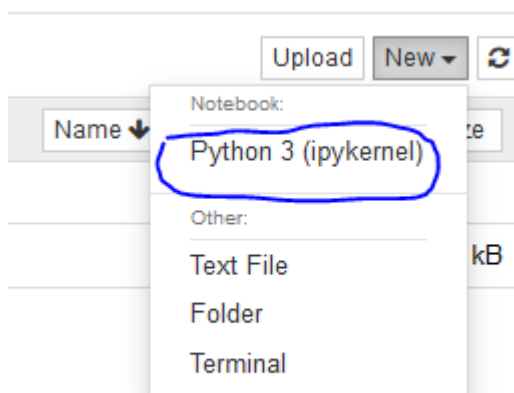
1. Tampilan awal Jupyter Notebook

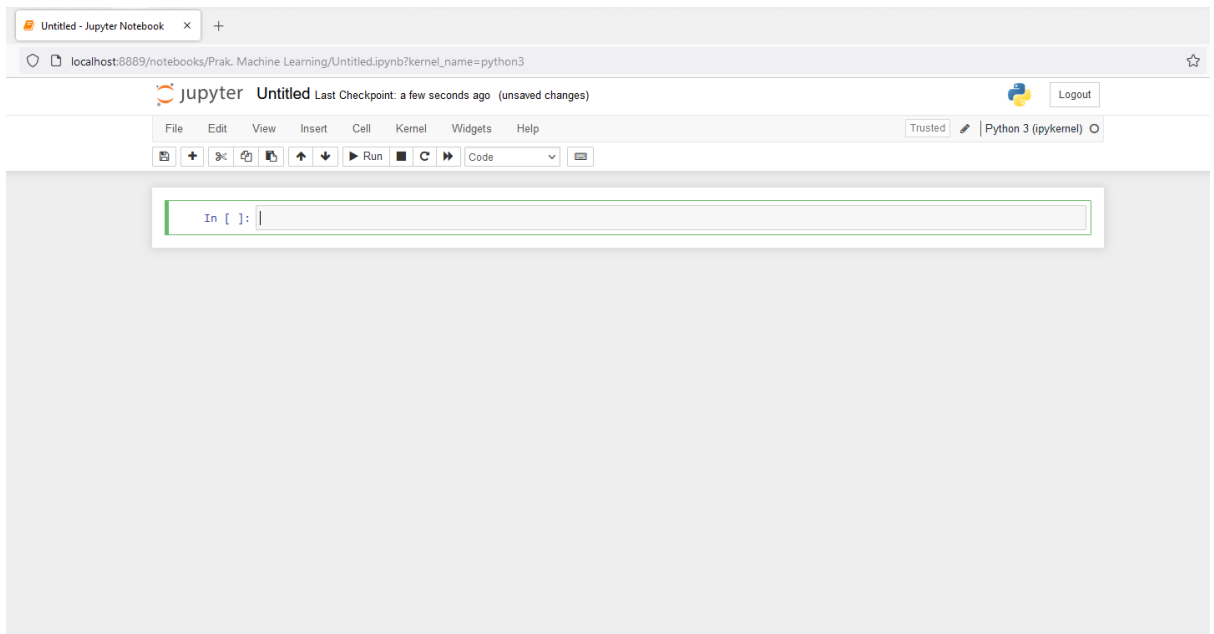


2. Buat Folder untuk menampung proyek anda dengan pilih New -> Folder

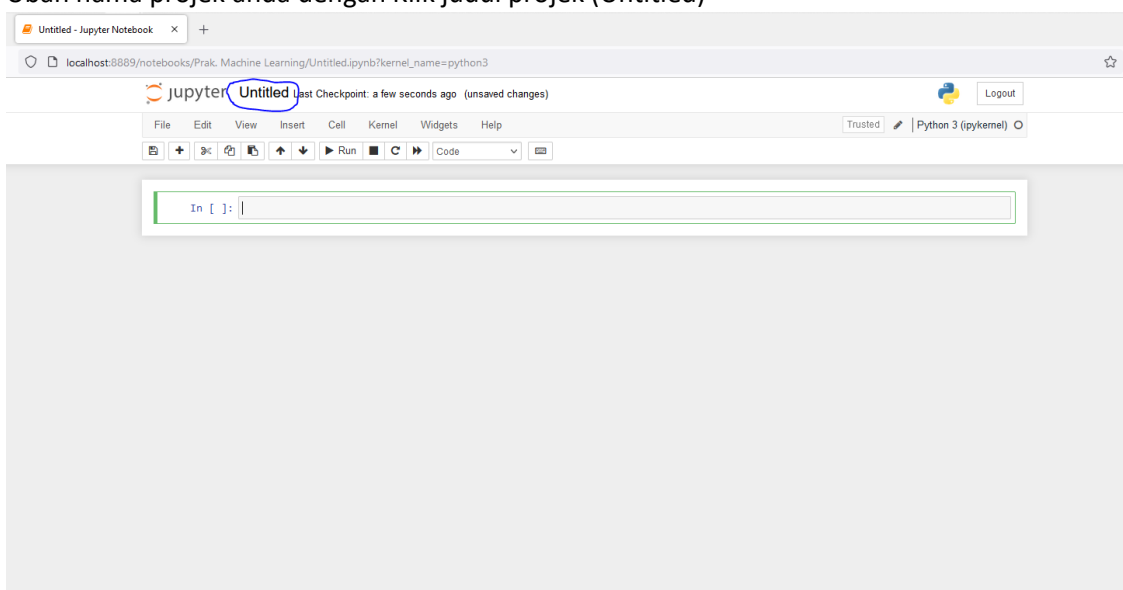


3. Pilih folder telah dibuat. Kemudian buat sebuah proyek dengan pilih New -> Python 3

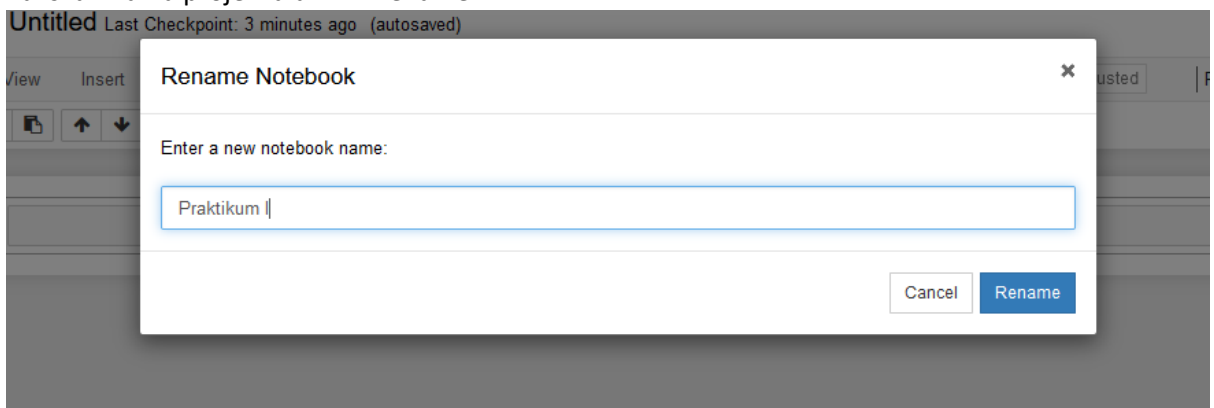




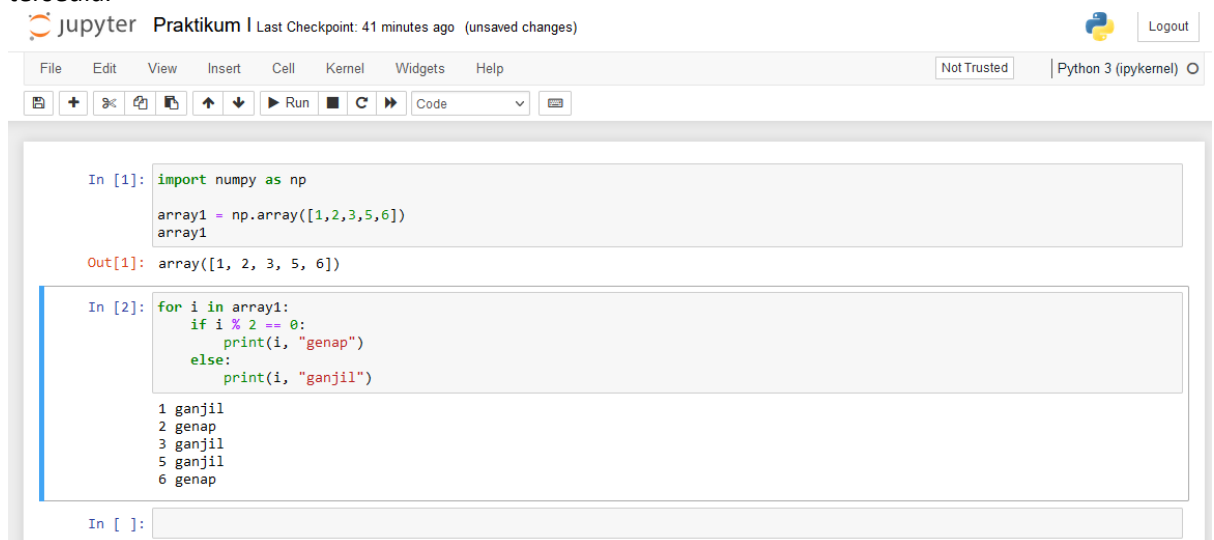
4. Ubah nama proyek anda dengan Klik judul proyek (Untitled)



5. Tuliskan nama proyek lalu Pilih Rename



6. Jupyter Notebook siap digunakan. Silahkan mengetikkan kode program anda pada cell yang tersedia.



The screenshot shows a Jupyter Notebook interface with the title 'jupyter Praktikum I'. The top bar includes a menu (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a 'Not Trusted' warning, and 'Python 3 (ipykernel)'. The first code cell contains:

```
In [1]: import numpy as np
        array1 = np.array([1,2,3,5,6])
        array1
```

The output for the first cell is:

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

The second code cell contains:

```
In [2]: for i in array1:
        if i % 2 == 0:
            print(i, "genap")
        else:
            print(i, "ganjil")
```

The output for the second cell is:

```
1 ganjil
2 genap
3 ganjil
5 ganjil
6 genap
```

III. Pengenalan Python

1. Buat sebuah proyek/notebook di jupyter notebook.
2. Buat program pertama untuk mencetak Halo, Python

```
print("Halo, Python")
```

Hasilnya:

```
Halo, Python
```

3. Contoh indentation dan suites yang benar

```
# Contoh indentation yang benar
print("Contoh Indentasi dan Suites")
x=1
if(x==1):
    print("x bernilai 1")
else:
    print("x bernilai selain 1")
```

Hasilnya :

```
Contoh Indentasi dan Suites
x bernilai 1
```

4. Contoh indentation dan suites yang salah

```
# Contoh indentation yang salah
print("Contoh Indentasi dan Suites")
x=1
if(x==1):
    print("x bernilai 1")
else:
    print("x bernilai selain 1")
```

Hasilnya :

```
Input In [3]
  print("x bernilai 1")
  ^
IndentationError: expected an indented block
```

5. Penggunaan Objek Data

```
#boolean
boolean = True

#float
Float = 3.14

#complex
kompleks = 2+8j

#String
string = 'Tipe String'
string_lagi = "Ini juga String"

print (type(boolean), boolean)
print (type(Float), Float)
print (type(kompleks), kompleks)
print (type(string), string)
print (type(string_lagi), string_lagi)
```

Hasilnya :

```
<class 'bool'> True
<class 'float'> 3.14
<class 'complex'> (2+8j)
<class 'str'> Tipe String
<class 'str'> Ini juga String
```

6. Komentar

```
#Komentar satu baris

"""Komentar baris 1
Komentar baris 2"""

print("Komentar")
```

Hasilnya :

```
Komentar
```

7. Operator Dasar

```

print ("Operator Aritmatika")
x = 10
y = 5

#penjumlahan
print ("Penjumlahan x + y = ", x + y)

#pengurangan
print ("Penjumlahan x - y = ", x - y)

#eksponen
print ("Penjumlahan x ** y = ", x**y)

x = 10
y = 5

print ("\nOperator Relasi")
# operasi sama dengan
print ("x == y", x == y)

#operasi tidak sama dengan
print ("x != y", x != y)

#operasi kurang dari
print ("x < y", x < y)

#operasi lebih besar atau sama dengan
print ("x >= y", x >= y)

print ("\nOperator Logika")
nilai = 70

print ("70 >=80 and 70 <=100 adalah", (nilai >= 80) and (nilai <= 100))
print ("70 >=80 or 70 <=100 adalah", (nilai >= 80) or (nilai <= 100))
print ("not 70>=80 adalah", not (nilai >= 80))

print ("\nOperator Keanggotaan")
kota = "Makassar"
print ("Huruf j ada di Makassar ? ',' in kota)
print ("Huruf j tidak ada di Makassar ? ',' not in kota)

```

Hasilnya :

```

Operator Aritmatika
Penjumlahan x + y = 15
Penjumlahan x - y = 5
Penjumlahan x ** y = 100000

Operator Relasi
x == y False
x != y True
x < y False
x >= y True

```

```
Operator Logika
70 >=80 and 70 <=100 adalah False
70 >=80 or 70 <=100 adalah True
not 70>=80 adalah True

Operator Keanggotaan
Huruf j ada di Makassar ? False
Huruf j tidak ada di Makassar ? True
```

8. Seleksi Dua Kondisi

```
# Seleksi dua kondisi
angka = 5
if (angka % 2 == 0) :
    print (angka, "adalah bilangan genap")
else :
    print (angka, "adalah bilangan ganjil")
```

Hasilnya :

```
5 adalah bilangan ganjil
```

9. Seleksi Bersarang

```
# Seleksi bersarang
nilai = 75

if nilai >=80 :
    print ("A")
elif nilai >=70 :
    print ("B")
elif nilai >=60 :
    print ("C")
elif nilai >=50 :
    nilai ("D")
else :
    nilai ("E")
```

Hasilnya :

```
B
```

10. Perulangan

```
huruf = ['A', 'B', 'C', 'D', 'E']

#Perulangan dengan for
print("Mencetak semua huruf dengan for")
for h in huruf :
    print (h)

#Perulangan dengan while
```

```
print("Menjumlahkan seluruh nilai dalam nilai1 menggunakan while")
nilai1 = [1,3,4,8,9]
i = 0
jumlah = 0
while (i < len(nilai1)):
    jumlah += nilai1[i]
    i+=1
print("Jumlah nilai pada nilai1 adalah ", jumlah)
```

Hasilnya :

Mencetak semua huruf dengan for

A
B
C
D
E

Menjumlahkan seluruh nilai dalam nilai1 menggunakan while

Jumlah nilai pada nilai1 adalah 25

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

MODUL II

PENGANTAR PYTHON 2

A. Pokok Bahasan

1. Perbedaan Penggunaan List, Tuples, Set dan Dictionary
2. Fungsi
3. Modul
4. Input/Output File

B. Tujuan Pembelajaran

1. Mahasiswa mampu membedakan list, tuples, set, dictionary.
2. Mahasiswa mampu membuat list, tuples, set, dan dictionary.
3. Mahasiswa mampu membuat fungsi sederhana.

4. Mahasiswa mampu memanggil modul.
5. Mahasiswa mampu menggunakan fungsi input dan output file.

C. Dasar Teori

I. Struktur Data

Data dalam python direpresentasikan dengan objek. Setiap objek memiliki identitas, tipe, dan nilai. Struktur data dalam python juga termasuk kedalam objek. Beberapa Tipe objek dasar pada Python yang juga merupakan struktur data adalah Tuple, List, Set, dan Dictionary. Keempat jenis struktur data ini memungkinkan objek untuk menampung lebih dari satu nilai dan juga menampung beberapa nilai dengan tipe data yang berbeda.

Penggunaan List, Tuple, Set, dan Dictionary

- **List** digunakan untuk menampung data yang memiliki urutan, nilainya bisa diubah, dan dapat menampung nilai duplikat.
- **Tuple** digunakan untuk menampung data yang memiliki urutan, nilainya tidak bisa diubah, dan dapat menampung nilai duplikat.
- **Set** digunakan untuk menampung data yang tidak memiliki data duplikat, nilainya tidak bisa diubah, dan tidak memiliki urutan tertentu.
- **Dictionary** digunakan untuk menampung data yang memiliki key. Key ini yang digunakan untuk mengakses data. Data dalam dictionary dapat diubah, memiliki urutan tertentu, dan tidak menampung key duplikat.

II. List

List merupakan tipe data yang dapat menampung lebih dari satu nilai. Nilai dalam list tidak harus memiliki tipe yang sama. List didefinisikan dengan menuliskan beberapa nilai dalam kurung siku “[]” dan setiap nilai dipisahkan dengan koma. Berikut beberapa operasi yang dapat digunakan pada List :

Operasi	Ekspresi	Contoh
Membuat list	[item1, item2,..]	warna=["merah", "kuning", "hijau"]
Mengakses item dalam list. Misalnya mengakses item pertama.	list[indeks]	warna[0] warna[0:2]
Mengubah nilai item	list[indeks] = nilai_baru	warna[2] = "biru" warna[1:2] = ["hijau", "biru"]
Panjang list	len(list)	len(warna)
Menambahkan objek ke dalam list. Misalkan menambahkan nilai 9 pada list angka	list.append(obj)	angka.append(9)
Menambahkan objek ke indeks tertentu	list.insert(obj)	warna.insert(3,"cokelat")
Menghapus item	list.remove(obj) list.pop(indeks) del list[indeks]	warna.remove("cokelat") warna.pop(3) warna.pop()

	del list list.clear()	del warna[1]
Iterasi untuk mencetak semua isi list	for x in list : print x	for x in warna : print x for i in range(len(warna)): print(warna[i]) [print(x) for x in warna]
Membuat list berdasarkan kondisi	list = [kondisi]	warna_baru=[x for x in warna if x!= "kuning"]
Mengurutkan item	list.sort()	warna.sort() warna.sort(reverse=True)
Menyalin list	new_list = list.copy() new_list = list(obj_list)	warna_baru=warna.copy() warna_baru=list(warna)
Menggabungkan dua list	Gunakan operasi (+) list.extend(obj_list)	list3 = list1 + list2 list1.extend(list2)
Menghitung jumlah item tertentu dalam list. Contoh menghitung jumlah nilai 3 dalam list angka. Hasilnya adalah 1 karena hanya ada satu nilai 3.	list.count(obj)	angka.count(3)

III. Tuple

Tuple merupakan tipe data yang mirip dengan list namun nilainya tidak bisa diubah. Selain itu, untuk mendefinisikan sebuah tupel dibutuhkan tanda kurung "(")" yang diisi dengan beberapa nilai yang dipisahkan dengan koma. Beberapa operasi yang digunakan pada tuple sebagai berikut :

Operasi	Ekspresi	Contoh
Membuat Tuple	(item1, item2, ...)	tuple1 = (1,3,'a','c') tuple1=() tuple1=('a')
Mengakses item pada tuple	tuple[indeks] tuple[dari_indeks:sampai_indeks]	tuple1[1], tuple1[-1] tuple1[1:3], tuple1[:3]
Menghapus tuple	del nama_tuple	del tuple1
Menghitung Panjang tuple	len(tuple)	len(tuple1)
Iterasi untuk mencetak isi tuple	for x in tuple : print (x)	for x in tuple1 : print (x)
Menggabungkan dua tuple	Gunakan operasi (+)	tuple3 = tuple1 + tuple2
Mengambil nilai maksimum dan minimum dari tuple	min(tuple) max(tuple)	min(tuple1) max(tuple1)
Konversi list ke tuple	tuple(list)	tuple([1,2,3,4])
Konversi tuple ke list	list=list(tuple)	list1=list(tuple1)

Unpack Tuple	(var1, var2) = tuple	gender = ("Perempuan","Laki-laki") (g1, g2) = gender
Mengakses indeks elemen tuple	tuple.index(elemen)	tuple1.index(3)

IV. Sets

Sets adalah implementasi dari himpunan pada matematika. Sehingga operasi matematika untuk himpunan seperti union, intersection, dll dapat dilakukan pada sets. Terdapat tiga karakteristik dari sets, yaitu :

- Item-item dalam set tidak memiliki urutan tertentu.
- Tidak ada item dengan nilai yang sama, sehingga tiap item bernilai unik.
- Item dalam sets dapat diubah.

Berikut ini adalah operasi untuk tipe data sets yang dapat dilakukan di Python :

Operasi	Ekspresi	Contoh
Membuat sebuah set	set{item1, item2, ...} set{} #setkosong	kota=set(['Makassar', 'Jakarta','Surabaya'])
Menambahkan item/elemen pada set	add()	kota.add('Malang')
Menghapus semua item pada set	clear()	kota.clear()
Menyalin isi set ke set lainnya	copy()	kota1 = kota.copy()
Operasi	Ekspresi	Contoh
Mengapus satu item/element	discard() : apabila elemen yang dihapus tidak terdapat pada set maka tidak terjadi apapun remove() : apabila elemen yang dihapus tidak terdapat pada set maka memunculkan pesan error	kota.discard('Malang') kota.remove('Malang')
Mengembalikan nilai yang berbeda antara 2 dua set.	difference()	A = {1,2,3,4,5} B = {4,5,6,7,8} A.difference(B) output : {1,2,3}
Mengapus item set yang ada di set lainnya	difference_update()	A.difference_update(B) output : set([1,2,3])

Mengembalikan nilai yang beririsan dari dua set.	intersection()	A.intersection(B) output : {4,5}
Mengubah item dalam set yang terlibat dalam operasi intersection sesuai hasil intersection	intersection_update()	A.intersection_update(B) print (A) output: set([4,5])
Mengembalikan nilai True apabila tidak ada item yang beririsan	isdisjoint()	A.isdisjoint(B) output : FALSE
Mengembalikan nilai True apabila suatu set merupakan anggota dari set lainnya	issubset()	A.issubset(B) output : FALSE
Mengembalikan nilai True apabila suatu set beranggotakan set lainnya.	issuperset()	A.issuperset(B) output: FALSE
Menampilkan nilai yang tidak beririsan antar set	symmetric_difference()	A.symmetric_difference(B) output: {1, 2, 3, 6, 7, 8}
Mengubah item dalam set sesuai dengan hasil operasi symmetric difference	symmetric_difference_update()	A.symmetric_difference_update(B) print (A) output: set([1, 2, 3, 6, 7, 8])
Operasi	Ekspresi	Contoh
Melakukan operasi union dan hasil operasi menjadi set yang baru	union()	A.union(B) print (A) output : set([1,2,3,4,5])
Mengubah item dalam set sesuai hasil operasi union	update()	A.update(B) print (A) output : set([1,2,3,4,5,6,7,8])
Mengambil jumlah item dalam set	len()	len(A)
Mengambil item terbesar dalam set	max()	max(A)

Mengambil item terkecil dalam set	min()	min(A)
Mengurutkan nilai dalam set tapi tidak mengubah urutan nilai yang tersimpan dalam set	sorted()	sorted(A) output : [4,5,6,7,8]
Menjumlahkan semua item dalam set	sum()	sum(A)

V. Dictionary

Dictionary merupakan tipe data pada Python dimana setiap elemennya memiliki nilai dan key. Elemen pada dictionary dituliskan dalam kurung kurawal. Nilai dan key dipisahkan dengan titik dua (:), dan setiap elemen dipisahkan dengan koma (,). Key harus unik dalam sebuah dictionary dan tidak dapat diubah. Nilai dalam dictionary bisa berupa data duplikat. Berikut oprasi yang bisa dilakukan dengan tipe data dictionary :

Operasi	Ekspresi	Contoh
Membuat sebuah dictionary	dict = {'key1': 'nilai1', 'key2': 'nilai2', ...}	siswa={'Nama': 'Janu', 'Usia': 8, 'Kelas': 'Tiga'}
Mengakses elemen dalam dictionary	dict['key'] get('key')	siswa['Nama'] siswa.get('nama')
Menghapus dictionary	del dict['key']; dict.clear(); del dict;	del siswa['Nama'] siswa.clear() del siswa;
Update isi dictionary	dict['key']=nilai_baru	siswa['Usia']:9
Jumlah elemen dalam dictionary	len(dict)	len(siswa) output : 3
Operasi	Ekspresi	Contoh
Mengambil semua key dalam dictionary	dict.keys()	print("Key Dictionary Siswa : ", siswa.keys())
Mengambil semua elemen dalam dictionary	dict.items()	print("Elemen Siswa : ", siswa.items())
Mencetak nilai dari dictionary	dict.values()	print("Siswa :", siswa.values()) output : Siswa : ['Tiga', 8, 'Janu']
Menggandakan dictionary	dict.copy()	siswa4=siswa.copy()
Membuat sebuah dictionary bersarang	dict = { "siswa1": { 'key1': 'nilai1', 'key2': 'nilai2' }, "siswa2": { 'key1': 'nilai1', 'key2': 'nilai2' }, }	murid = { "siswa1": { 'Nama': 'Janu', 'Usia': 8 }, "siswa2": { 'Nama': 'Sam', 'Usia': 9 }, }

	...	
	}	
Membuat dictionary dari daftar key	<code>dict.fromkeys()</code>	<pre>seq = ('Nama', 'Usia', 'Kelas') siswa = siswa.fromkeys(seq) print("Siswa :", str(siswa))</pre> <p>output :</p> <p>Siswa : {'Nama':None, 'Usia':None, 'Kelas':None}</p>
Mengembalikan nilai default untuk key yang tidak terdapat dalam dictionary	<code>dict.get(key, default=None)</code>	<pre>siswa1={'Nama':'Janu','Usia':8} print("Kelas : ", siswa1.get('Kelas', 'Satu'))</pre> <p>output :</p> <p>Kelas : Satu</p>
Mengecek keberadaan key pada sebuah dictionary. Nilai true akan dikembalikan jika key ditemukan dalam dictionary	<code>dict.has_key(key)</code>	<code>siswa.has_key('Nama')</code>
Menambah dictionary ke dictionary lain dengan key yang berbeda	<code>dict.update(dict2)</code>	<pre>siswa1={'Nama':'Janu','Usia':8} siswa2={'Kelas':'Satu'} siswa1.update(siswa2) print(siswa1)</pre> <p>output:</p> <p>Value :</p> <p>{'Nama':'Janu','Usia':8,'Kelas':'Satu'}</p>

VI. Fungsi

Fungsi merupakan sebuah blok berisi beberapa statement yang saling terkait yang disusun untuk menjalankan aksi tertentu. Fungsi merupakan konsep yang memungkinkan modularitas dan penggunaan kembali kode program. Berikut aturan dalam pendefinisian fungsi di Python :

- Pendefinisian **header** fungsi dilakukan dengan menuliskan keyword **def** diikuti dengan nama fungsi serta tanda kurung dan dituliskan di awal blok fungsi. Diakhir header dituliskan titik dua (:).
- Fungsi dapat memiliki argumen atau parameter yang ditempatkan di dalam tanda kurung.
- Statement-statement dalam fungsi dituliskan setelah header fungsi dan diindentisasi.
- Fungsi dapat mengembalikan nilai. Ketika fungsi tidak mengembalikan nilai maka fungsi menjadi subprosedur.

Sintaks fungsi :

```
def function_name():
```

```
1st block line
2nd block line
...
```

Sintak fungsi dengan argumen :

```
def function_name(parameters):
    1st block line
    2nd block line
    ...
    return[expression]
```

VII. Modul

Modul terdiri dari sejumlah fungsi atau kelas yang independen tapi saling terkait. Pada python terdapat modul bawaan ada juga yang harus diimport ketika ingin digunakan. Modul disebut juga dengan Library. Berikut cara untuk mengimport modul di Python.

```
#Impor semua fungsi dalam modul
import nama_modul
from nama_modul import*

#impor fungsi tertentu dalam modul
from nama_modul import nama_fungsi
```

VIII. Input/Output File

Python memiliki fungsi untuk membaca dan menulis ke file. Proses membaca dan menulis diawali dengan proses membuka file. Setelah operasi membaca atau menulis file perlu ditutup kembali. Berikut ini operasi-operasi untuk operasi file.

Aksi	Ekspresi	Contoh
Membuka file	obj = open(filename, access_mode, buffer)	f = open('data.txt','w')
Membaca file	Fileobject.read(value)	f = open('data.txt','w') f.readlines()
Menutup file	Fileobject.close()	f.close()
Menulis ke file	Fileobject.write(string str)	data=['Sinta\n', 'Jaja\n', 'Rina\n'] f = open('data.txt','w') f.writelines(data) f.close

Pada saat membuka sebuah file maka perlu didefinisikan jenis pembacaan file. Penentuan jenis pembacaan file didefinisikan melalui `access_mode`. Akses mode default dari pembacaan file adalah *mode* baca (r). Berikut pilihan jenis mode lainnya :

Jenis pembacaan	Keterangan
R	File hanya bisa dibaca
Rb	File dibaca dalam format binary
r+	File bisa dibaca dan ditulis
rb+	File dibaca dan ditulis dalam format binary
W	File hanya bisa ditulis
Wb	File ditulis dalam format binary
w+	File dibaca dan ditulis, apabila sudah terdapat file yang sama maka file ditimpa
wb+	File dibaca dan ditulis dalam format binary, apabila sudah terdapat file yang sama maka file ditimpa
A	Membuka file yang memungkinkan isi file ditambahkan, membuat file jika belum ada
Ab	Membuka file yang memungkinkan isi file ditambahkan, membuat file jika belum ada
a+	Membuka file yang memungkinkan isi file ditambahkan dan dibaca, membuat file jika belum ada
ab+	Membuka file yang memungkinkan isi file ditambahkan dan dibaca dalam format binary, membuat file jika belum ada

D. Tahapan Praktikum

I. Struktur Data

1. Membuat list dan melakukan operasi pada list

```
# Membuat list
list1 = ['Januari', 'Februari', 20, 21, 2020, 2021]

# Mengakses nilai dalam list
print ("List1[0] : ", list1[0])
print ("List1[2:5] :", list1[2:5])

# Menambahkan objek dalam list
list1.append(2022)
list1.insert(2, "Maret")
print("Isi list1 setelah penambahan :", list1)

# Menghapus objek dalam list
del list1[7]
list1.remove('Maret')
print("Isi list1 setelah penghapusan : ", list1)
```



```
#Mengurutkan list
angka=[1,4,3,6,5,11]
print ("Pengurutan ascending : ", angka)
angka.sort(reverse=True)
print ("Pengurutan descending : ", angka)

#Mengambil nilai terbesar
print("Nilai terbesar dalam list angka : ", max(angka))
```

Hasilnya:

```
List1[0] : Januari
List1[2:5] : [20, 21, 2020]
Isi list1 setelah penambahan : ['Januari', 'Februari', 'Maret', 20, 21,
2020, 2021, 2022]
Isi list1 setelah penghapusan : ['Januari', 'Februari', 20, 21, 2020,
2021]
Pengurutan ascending : [1, 4, 3, 6, 5, 11]
Pengurutan descending : [11, 6, 5, 4, 3, 1]
Nilai terbesar dalam list angka : 11
```

Lengkapi kode berikut :

```
# Membuat list buah isi dengan objek apel, pisang, nanas, semangka,
kelapa
buah = ---

# Tambahkan objek jeruk ke dalam list buah
buah.---(---)

# Cetak list buah dari data ke-1 sampai ke-3
print ("Data ke-1 s/d ke-3 :", ---[---])
```

2. Membuat tuple dan melakukan operasi pada tuple

```
# Membuat Tuple
tuple1 = ()
print("Isi tuple kosong : ", tuple1)

tuple1 = (1, )
print("Isi tuple dengan 1 item : ", tuple1)

tuple1 = (1,4,'a', 'c', 'd')
print("Isi tuple lebih dari 1 item : ", tuple1)

# Mengakses item dalam tuple
print("Item ke-2 : ", tuple1[1])
print("Dua Item awal : ", tuple1[0:2])
print("Isi tuple1 dari item ke-3 : ", tuple1[2:])

# Jumlah item dalam tuple
print("Jumlah item dalam tuple : ", len(tuple1))
```

```
#Mencetak isi tuple dengan iterasi
print("Isi tuple : ")
for x in tuple1 : print (x)

#Mengecek keberadaan item dalam tuple
print("Apakah ada item a dalam tuple1?", 'a' in tuple1)
```

Hasilnya :

```
Isi tuple kosong : ()
Isi tuple dengan 1 item : (1,)
Isi tuple lebih dari 1 item : (1, 4, 'a', 'c', 'd')
Item ke-2 : 4
Dua Item awal : (1, 4)
Isi tuple1 dari item ke-3 : ('a', 'c', 'd')
Jumlah item dalam tuple : 5
Isi tuple :
1
4
a
c
d
Apakah ada item a dalam tuple1? True
```

Lengkapi kode berikut :

```
# Membuat tuple score dengan objek bad, fair, good, excellent
score = ---
print("Isi tuple score : ", ---)

# Jumlah item dalam tuple
print("Jumlah item dalam tuple score : ", ---(---))

# Mengakses item dalam tuple
print("Score ke-1 : ", ---)
```

3. Membuat set dan melakukan operasi pada set

```
#inisialisasi set
kota = {"Makassar", "Jakarta", "Surabaya"}
print("Isi set : ", kota)

#menambahkan elemen
kota.add("Malang")
print("Isi set setelah penambahan 1 elemen : ", kota)

#menambahkan beberapa elemen
kota.update(["Manado", "Jambi"])
print("Isi set setelah operasi update : ", kota)
```

```

#Menghapus sebuah elemen
kota.remove('Malang')
print("Isi set setelah penghapusan 1 elemen : ", kota)

#Menghapus sebuah elemen, walaupun 'Malang' telah dihapus,
#tidak ada pesan error yang ditampilkan
kota.discard('Malang')

#operasi pop menghapus sebuah item secara random dari set
print ("Elemen ", kota.pop(), "dihapus dari set kota")
print(kota)

#menampilkan jumlah elemen dalam set
print("Jumlah elemen dalam set : ", len(kota))

```

Hasilnya :

```

Isi set : {'Surabaya', 'Makassar', 'Jakarta'}
Isi set setelah penambahan 1 elemen : {'Surabaya', 'Makassar',
'Jakarta', 'Malang'}
Isi set setelah operasi update : {'Surabaya', 'Malang', 'Jakarta',
'Manado', 'Jambi', 'Makassar'}
Isi set setelah penghapusan 1 elemen : {'Surabaya', 'Jakarta',
'Manado', 'Jambi', 'Makassar'}
Elemen Surabaya dihapus dari set kota
{'Jakarta', 'Manado', 'Jambi', 'Makassar'}
Jumlah elemen dalam set : 4

```

Lengkapi kode berikut :

```

# Membuat set buah isi dengan objek apel, pisang, nanas, semangka,
kelapa
buah = ---
print("Isi set buah : ", ---)

# Tambahkan objek jeruk dan nanas
buah.---([---])
print("Isi set buah : ", ---)

#Menghapus objek semangka
---.---(---)
print("Isi set buah setelah penghapusan 1 elemen : ", ---)

```

4. Membuat dictionary dan melakukan operasi pada dictionary

```

#Membuat dictionary
siswa = {'Nama': 'Janu', 'Usia':8, 'Kelas':'Tiga'}
print("Isi dictionary: ", siswa)

#Mengakses elemen dalam dictionary
print("Nilai dari key Nama : ", siswa['Nama'])

```

```

#Copy isi dictionary
siswa1 = siswa.copy()
print("Dictionary siswa1 : ",siswa1)
# Menghapus elemen dictionary
siswa1.clear()
# Menghapus kamus siswa1
print("Dictionary siswa1 : ",siswa1)
del siswa1

#mengubah nilai dictionary
siswa['Usia'] = 9
print("Isi dictionary: ", siswa)

#Menampilkan nilai dari key tertentu
print("Usia Janu : ", siswa.get('Usia'))

#Menambah dictionary ke dictionary lainnya
siswa2 = {'Nama':'Sam', 'Usia':7}
kelasSiswa = {'Kelas' : 'Dua'}
siswa2.update(kelasSiswa)
print("Isi dictionary siswa2 : ", siswa2)

```

Hasilnya :

```

Isi dictionary: {'Nama': 'Janu', 'Usia': 8, 'Kelas': 'Tiga'}
Nilai dari key Nama : Janu
Dictionary siswa1 : {'Nama': 'Janu', 'Usia': 8, 'Kelas': 'Tiga'}
Dictionary siswa1 : {}
Isi dictionary: {'Nama': 'Janu', 'Usia': 9, 'Kelas': 'Tiga'}
Usia Janu : 9
Isi dictionary siswa2 : {'Nama': 'Sam', 'Usia': 7, 'Kelas': 'Dua'}

```

Lengkapi kode berikut :

```

#Membuat dictionary produk dengan nm_barang : Air Mineral, merek :
Aqua, harga : 3000
produk = {'nm_barang' : '---', '---':'Aqua' , 'harga':---}
print("Isi dictionary: ", ---)

#menampilkan nilai dari key nm_barang
print("Nama Barang : ", ---)

#menampilkan nilai dari key merek
print("Merek : ", ---.get())

#mengubah nilai harga menjadi 3500
produk[---]=---
print("Harga : ", ---)

```

II. Fungsi

```

#fungsi
def jumlah_dua_angka(x,y=0):
    return x + y

#subprosedur
def jumlah_banyak_angka(angka):
    jml=0
    for x in angka:
        jml = jml + x
    print(jml)

#memanggil fungsi
print("Hasil fungsi :")
print (jumlah_dua_angka(5))
print (jumlah_dua_angka(5,2))

#memanggil subprosedur
print("Hasil subprosedur :")
jumlah_banyak_angka(list([1,2,3,4,5]))
jumlah_banyak_angka(list([4,3,5,6]))

```

Hasilnya :

```

Hasil fungsi :
5
7
Hasil subprosedur :
15
18

```

III. Modul

```

#Impor semua fungsi dalam modul
import pandas as pd
import numpy

#Impor fungsi tertentu dalam modul
from matplotlib.pyplot import bar

# Menggunakan fungsi membuat DataFrame
df = pd.DataFrame(numpy.random.randint(0,100,size=(5, 2)), columns=list('AB'))
print(df)

# Menggunakan fungsi membuat bar chart

```

```
gender = ["Perempuan", "Laki-laki"]
jumlah = [200, 250]
bar(gender, jumlah)
```

IV. Input/Output File

```
#Membuat file dengan nama data. \n adalah karakter untuk baris baru
data = ['Sinta\n', 'Jaja\n', 'Rina\n']
f = open('data.txt', 'w')
f.writelines(data)

#Membaca file
f = open('data.txt')
print (f.readlines())
f.close()
```

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

MODUL III

DASAR MACHINE LEARNING

A. Pokok Bahasan

1. Paket Machine Learning Python
2. Paket Analisis Data : Numpy, Pandas
3. Feature Engineering

B. Tujuan Pembelajaran

1. Mahasiswa mengenal paket analisis data di Python
2. Mahasiswa mampu menggunakan Paket NumPy dan Pandas untuk analisis data

3. Mahasiswa mampu melakukan Feature Engineering dengan menggunakan paket Machine Learning Python.

C. Dasar Teori

I. Paket Machine Learning

Terdapat banyak library di python terkait dengan machine learning. Secara garis besar paket terbagi atas dua yaitu paket analisis data dan paket inti machine learning. Paket analisis data terdiri dari kumpulan paket dengan fungsi matematis dan sains yang digunakan untuk preprocessing data dan tranformasi data. Paket inti machine learning adalah kumpulan paket yang menyediakan fungsi algoritma machine learning yang dapat digunakan untuk mengekstrak pola dari data. Terdapat empat jenis paket yang umum digunakan untuk analisis data. Paket ini adalah NumPy, SciPy, Pandas, Matplotlib, dan Seaborn.

II. NumPy

NumPy adalah library inti untuk komputasi ilmiah di Python. Library ini menyediakan objek array multidimensional dan perkakas untuk pengelolaan array. Array di NumPy merupakan kumpulan nilai dengan tipe data yang sama. Inisialisasi array dapat dilakukan dengan mendefinisikan list bersarang. Pengaksesan elemen array pada NumPy menggunakan kurung siku. Berikut beberapa fungsi pada library NumPy :

Operasi	Ekspresi	Contoh
Membuat array	<code>numpy.array([list_data])</code> <code>numpy.array([list_data])</code>	<code>arr = numpy.array([1,2,3])</code> <code>arr2d = numpy.array([1,2,3],[4,5,6])</code>
Mengakses elemen array	<code>array[indeks]</code>	<code>arr[0]</code> <code>arr[0,1]</code>
<i>Slicing Array</i>	<code>array[indeksAwal:indeksAkhir]</code> <code>array[indeksAwal:indeksAkhir:step]</code> <code>array2D[indeksAwal:indeksAkhir, indeksAwal:indeksAkhir]</code>	<code>arr[1:5]</code> <code>arr[2:]</code> <code>arr[:3]</code> <code>arr[1:5:2]</code> <code>arr[0:2, 1:4]</code>
Menyalin array	<code>array.copy()</code>	<code>arr1 = arr.copy()</code>
Operasi	Ekspresi	Contoh
Mengakses ukuran array	<code>array.shape</code>	<code>arr.shape</code>
Mengubah ukuran array	<code>array.reshape(jml_baris, jml_kolom)</code>	<code>arr.reshape(5, 3)</code>
<i>Join Array</i>	<code>numpy.concatenate((arr1,arr2))</code>	<code>arr= numpy.concatenate((array1,array2))</code>
<i>Splitting Array</i>	<code>numpy.array_split(array, jml_split)</code>	<code>arr_split = numpy.array_split(arr,3)</code>
Mencari elemen di Array	<code>numpy.where(array=elemen)</code>	<code>x = numpy.where(arr=3)</code>
Mengurutkan array	<code>numpy.sort(array)</code>	<code>numpy.sort(arr)</code>
<i>Filter Array</i>	<code>array[filter]</code>	<code>arr1=arr[arr<8]</code> <code>filter = arr%2==0</code> <code>arr1=arr[filter]</code>

Daftar fungsi NumPy yang lebih lengkap dapat dilihat pada dokumentasi di <https://numpy.org/doc/stable/reference/>

III. Pandas

Pandas merupakan library pada Python untuk menangani data relasional. Terdapat dua struktur data pada Pandas yaitu Series dan DataFrame. Series merupakan tipe data untuk objek satu dimensi, serupa dengan sebuah kolom dalam spreadsheet atau tabel SQL. Data dalam series memiliki indeks yang dimulai dari 0 sampai N. Sedangkan DataFrame merupakan tipe data untuk objek dua dimensi, serupa dengan sebuah tabel dalam SQL atau spreadsheet. Beberapa fungsi yang disediakan oleh library Pandas :

Operasi	Ekspresi	Contoh
Membaca data csv	<code>pandas.read_csv(filepath)</code>	<code>pandas.read_csv('data.csv')</code>
Membuat Series	<code>pandas.Series(data,index)</code>	<code>Pandas.Series(['Makassar','Gowa', 'Maros'], index=[1,2,3])</code>
Membuat objek dua dimensi. Objek ini serupa dengan tabel atau spreadsheet	<code>pandas.DataFrame(data=None, index=None, columns=None)</code>	<code>d = {'col1': [1, 2], 'col2': [3, 4]}</code> <code>df = pandas.DataFrame(data=d)</code> <code>pandas.DataFrame(numpy.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]), columns=['a', 'b', 'c'])</code>
Statistik dasar	<code>DataFrame.describe()</code>	<code>df.describe()</code>
Menampilkan data 5 atau n data teratas	<code>DataFrame.head()</code>	<code>df.head()</code> <code>df.head(n=3)</code>
Menampilkan judul kolom	<code>DataFrame.columns</code>	
Menampilkan nilai dataframe	<code>DataFrame.values</code>	
Merge/Join DataFrame	<code>pandas.concat([df1,df2])</code> <code>df1.append(df2)</code>	Join 2 dataframe pada kolom <code>Pandas.concat([df1,df2], axis=1)</code>
Grouping	<code>DataFrame.groupby().fungsiagregat()</code>	<code>df.groupby(['kota']).max()</code>
Operasi	Ekspresi	Contoh
Mengakses dataframe berdasarkan indeks	<code>DataFrame.loc[i:j]</code>	Menampilkan data indeks 0 sampai 2 : <code>DataFrame.loc[0:2]</code>
Mengakses dataframe berdasarkan baris dan kolom tertentu	<code>DataFrame.iloc[baris,kolom]</code>	Mengambil data 2 baris pertama dan 2 kolom pertama : <code>df.iloc[0:2,0:2]</code>
Filter berdasarkan nama kolom	<code>DataFrame[Filter]</code>	<code>df[df['x1'] > 5]</code>
Mean	<code>DataFrame.mean()</code>	
Max	<code>DataFrame.max()</code>	
Min	<code>DataFrame.min()</code>	
Sum	<code>DataFrame.sum()</code>	

count	DataFrame.count()	
-------	-------------------	--

Daftar fungsi Pandas yang lebih lengkap dapat dilihat pada dokumentasi di <https://pandas.pydata.org/docs/reference/index.html>

IV. Feature Engineering

Kualitas output algoritma machine learning sangat bergantung pada kualitas data input. Proses untuk memperoleh data dengan fitur yang layak disebut dengan *feature engineering*. *Feature engineering* merupakan aspek yang paling penting dalam membangun sistem *machine learning* yang efisien. Beberapa penerapan feature engineering antara lain :

- Penanganan data hilang
Pengangan data hilang dapat dilakukan dengan empat acara yaitu menghapus data, mengisi dengan nilai pusat data, mengganti dengan nilai random, dan menggunakan model prediktif.
- Penanganan data kategorikal
Penanganan data kategorikan adalah mengubah data kategorikal menjadi data numerik sehingga dapat digunakan untuk membuat model. Terdapat dua Teknik untuk penanganan data kategorikal, yaitu :
 - Membuat dummy variable
 - Mengubah dalam bentuk angka
- Normalisasi Data
Normalisasi data adalah proses mengubah skala nilai dari variabel sehingga setiap variabel memiliki jangkauan nilai yang sama. Normalisasi data dapat dilakukan dengan menggunakan Min-Max Scaling dan Z-scores.
- Diskritisasi
Diskritisasi adalah proses mengubah data numerik menjadi data kategorikal. Hal ini biasanya dilakukan apabila pembuatan model perlu menggunakan data kategorikal sedangkan terdapat variable data dengan tipe numerical. Salah Teknik diskritisasi adalah binning.

D. Tahapan Praktikum

I. Numpy

1. Array 1 dimensi

```
import numpy as np

# Membuat array 1 dimensi
a = np.array([0, 1, 2, 3])

# Slicing
print("Elemen indeks ke-1 :", a[0])
print("Elemen ke-1 s/d ke-3 :", a[0:3])

# Mengubah elemen array
a[0] = 5
```

```
print("Isi array a setelah diubah : ", a)
```

2. Array 2 dimensi

```
import numpy as np

# Membuat array 2 dimensi
b = np.array([[0, 1, 2, 3],[4,5,6,7]])
print("Isi array 2 Dimensi : \n", b)

# Slicing
print("Elemen baris ke-1 :", b[0,])
print("Elemen baris ke-1 kolom ke-2 s/d ke-3 :", b[0,1:3])

# Mengubah elemen array
b[0,0] = 10
print("Isi array b setelah diubah : \n", b)

# Ukuran Array
print("Ukuran Array b : ", b.shape)

# Filter
arr_filter = b[b%2==0]
print("Isi array arr_filter")
print(arr_filter)
```

II. Pandas

1. Membuat DataFrame dan Series

```
import pandas as pd
import numpy as np

#Membuat sebuah objek series
s = pd.Series(np.array([1,2,3,np.nan,5,6]), index=['A','B','C','D','E','F'])
print("Series")
print(s)

#Membuat sebuah objek DataFrame
data = {'Gender' : ['F', 'M', 'M'], 'Emp_ID':['E01','E02','E03'],
        'Age':[25,27,25]}
df=pd.DataFrame(data, columns=['Emp_ID','Gender','Age'])
print("\n\nDataFrame")
print(df)
```

2. Membaca data dari sebuah file csv

```
import pandas as pd

#Membaca data iris
df_iris = pd.read_csv('iris.csv')

print("Fungsi head()")
```

```
print(df_iris.head())
print("\nFungsi display()")
display(df_iris.head(n=10))
print("\nFungsi describe()")
print(df_iris.describe())
```

3. Membaca data dari *link*

```
import pandas as pd

# Membaca data golf
df_golf = pd.read_csv('https://raw.githubusercontent.com/iizmyy/prak-machine-
learning/main/dataset/golf-dataset.csv')
print(df_golf)

# Ukuran dataframe
print("\nUkuran df_golf : ",df_golf.shape)

# Atribut dataframe
print("\nAtribut : ", df_golf.columns.values)
```

4. Slicing dan Filtering Data

```
#Slicing menggunakan fungsi loc
print("\nSlicing dengan fungsi loc")
golf_slice = df_golf.loc[0:3,['Outlook','Temp', 'Play Golf']]
print(golf_slice)

#Slicing menggunakan fungsi iloc
print("\nSlicing dengan fungsi iloc")
golf_slice = df_golf.iloc[0:5,2:5]
print(golf_slice)

# Filtering
print("\nFiltering Humidity : Normal")
filter_humidity_normal = df_golf['Humidity'] == 'Normal'
print(df_golf[filter_humidity_normal])

print("\nFiltering Outlook : Rainy")
print(df_golf[df_golf['Outlook']=='Rainy'])

print("\nFiltering Outlook : Rainy dan Humidity : Normal")
df_golf_filter = df_golf[(df_golf['Outlook']=='Rainy') & (filter_humidity_normal)]
print(df_golf_filter)
```

III. Feature Engineering

1. Penanganan Data Hilang

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

```

# Membuat sebuah dictionary
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# Membuat dataframe dari dictionary
df = pd.DataFrame(dict)
print("Data Awal")
print(df)

# Mengisi data hilang dengan nilai rata-rata
for i in df.columns:
    df[i].fillna(df[i].mean(), inplace=True)

print("\nData Setelah penanganan data hilang")
print(df)

```

2. Penanganan Data Kategorikal

```

import pandas as pd

df = pd.DataFrame({
    'A' : ['high', 'medium', 'low', 'medium'],
    'B' : [10,20,30,40]}, index=[0,1,2,3])

print(df)

# 1. Membuat dummy variabel
# Menggunakan fungsi get_dummies di Pandas
df_with_dummies = pd.get_dummies(df, prefix='A', columns=['A'])
print("\nMembuat Dummy Variabel : \n", df_with_dummies)

# Mengubah kategorikal ke angka
# Menggunakan fungsi factorize dari Pandas
df['A_pd_factorized'] = pd.factorize(df['A'])[0]
print("\nMengubah Kategorika ke numerik : \n", df)

```

3. Normalisasi Data

```

import pandas as pd
from sklearn import preprocessing

#Membaca data iris
df_iris = pd.read_csv('iris.csv')

#Slicing
iris_data = df_iris.iloc[0:10,:4]
iris_target = df_iris.iloc[0:10,4]
data = pd.concat([iris_data, iris_target], axis=1)
print("\nData sebelum normalisasi")
display(data)

```

```

# Normalisasi
# 1. Z-Score
zscore_scale = preprocessing.StandardScaler().fit(iris_data)
data_zscore = zscore_scale.transform(iris_data)

# 2. Min-Max Scaling
minmax_scale = preprocessing.MinMaxScaler().fit(iris_data)
data_minmax = minmax_scale.transform(iris_data)

print("\nHasil normalisasi Z-Score")
df_data_zscore = pd.DataFrame(data_zscore, columns=iris_data.columns)
data = pd.concat([df_data_zscore, iris_target], axis=1)
display(data)

print("\nHasil normalisasi Min-Max Scaling")
df_data_minmax = pd.DataFrame(data_minmax, columns=iris_data.columns)
data = pd.concat([df_data_minmax, iris_target], axis=1)
display(data)

```

4. Binning

```

import pandas as pd

# Membuat data dari list
data_penduduk = [['Alex', 10], ['Bob', 16], ['Dian', 26], ['James', 24], ['Sam', 69]]
df = pd.DataFrame(data=data_penduduk, columns=['Nama', 'Usia'])
print(df)

# Binning dengan fungsi cut di Pandas
df['Binned'] = pd.cut(df['Usia'], bins=[0,10,19,60,100])

category = ['Anak', 'Remaja', 'Dewasa', 'Lanjut Usia']
df['Kategori'] = pd.cut(df['Usia'], bins=[0,10,19,60,100], labels=category)

print("\nData Setelah Diskritisasi : \n",df)

```

Latihan Praktikum

ID_Karyawan	Gender	Usia	Jabatan
E01	P	20	Staf
E02	L	19	HRD
E03	L	40	Manajer
E04	L	31	Manajer
E05	P	27	Staf

```

import pandas as pd

# Buat dataframe sesuai dengan table di atas
data_kary = {---}

df_karyawan=pd.---(---)

```

```

display(---)

# Filtering karyawan dengan gender perempuan
print("\nFiltering Gender : Perempuan")
filter_gender = df_karyawan[---] == ---
print(df_karyawan[---])

# Filtering karyawan dengan usia kurang dari 25 tahun
print("\nFiltering gender : Perempuan dan Usia kurang dari 25 tahun")
print(df_karyawan[(---) & (---)])

# Membuat dummy variabel gender
df_kary_dummies = pd.---(---, prefix='---', columns=---)
print("\nMembuat Dummy Variabel : \n", ---)

```

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

MODUL IV QUIZ

Quiz ini menggunakan data harga rumah di Jakarta Selatan. Data memiliki 6 atribut , yaitu

LT : Luas Tanah (m²)

LB : Luas Bangunan (m²)

JKT : Jumlah Kamar Tidur

JKM : Jumlah Kamar Mandi

GARASI : Ketersediaan garasi

HARGA : Harga rumah (Milyar)

1. Baca data dari file Harga Rumah Jaksel.csv dan tampilkan deskripsi data.

```
import --- as ---

# Baca Data
df_rumah = ---.---(---)

# Tampilkan 10 data teratas
display(---.head(---))

# Tampilkan deskripsi dataframe df_rumah
---.describe()
```

2. Tampilkan data rumah yang harganya kurang dari 1 M dan memiliki garasi

```
df_rumah_1 = df_rumah[(---) & (---)]
display(df_rumah_1)

# Cetak jumlah rumah dengan harga kurang dari 1 M yang memiliki garasi
print('Jumlah rumah : ', df_rumah_1.count()[0])
```

3. Tampilkan data rumah dengan 3 kamar tidur , 3 kamar mandi dan harga kurang dari 1.5 M

```
df_rumah_2 = df_rumah[---]
display(df_rumah_2)

# Cetak jumlah rumah
print('Jumlah rumah : ', df_rumah_2.--- [0])
```

4. Tampilkan data rumah dengan minimal luas bangunan 100-150 dan memiliki garasi

```
filter_lb = (---) & (---)
df_rumah_3 = df_rumah[filter_lb & (---)]
display(---)

# Tampilkan harga rumah termurah berdasarkan data df_rumah_3
print("Harga termurah : ", ---.min(), "M")

# Tampilkan harga rumah termahal berdasarkan data df_rumah_3
print("Harga termahal : ", ---.max(), "M")
```

5. Tampilkan data rumah dengan harga termahal dan termurah berdasarkan filter data nomor 4.

```
print("Data rumah termurah :")
display(df_rumah_3[df_rumah_3[---] == ---.min()])

print("Data rumah termahal :")
display(df_rumah_3[df_rumah_3[---] == ---.max()])
```

6. Tampilkan luas tanah, luas bangunan, jumlah kamar tidur, dan harga dari rumah yang harganya kurang dari 600 juta

```
# df_rumah_4 = df_rumah[---]
display(df_rumah_4[[---,---,---,---]])
```

7. Konversi nilai dari variabel garasi menjadi numerik

```
df_rumah_dummies = pd.---(---)
display(---)
```

8. Normalisasi atribut luas tanah, luas bangunan, jumlah kamar tidur, dan jumlah kamar mandi

```
from --- import ---

# Slicing atribut yang akan dinormalisasi
data_normal = df_rumah_dummies.iloc[---]

# Normalisasi dengan minmax scaling
minmax_scale = ---(---)
data_normal = ---(---)

print("\nHasil normalisasi Min-Max Scaling")
df_normal = pd.DataFrame(data_normal, columns=['LT','LB','JKT', 'JKM'])

# Menggabungkan data normalisasi dengan data atribut lainnya
df_data_normal = pd.concat([---, df_rumah_dummies.iloc[---]], axis=1)
display(df_data_normal)
```

9. Binning harga rumah pada data rumah yang telah dinormalisasi pada nomor 8 dengan kategori :
Murah : 0-600 juta, Cukup Mahal: 600 juta -1.5 M , Mahal : 1.5- 5M , Sangat Mahal 5-250M

```
kategori = [---]
df_data_normal['KATEGORI'] = ---(---)
display(df_data_normal)
```

10. Berdasarkan data rumah nomor 9 tentukan jumlah data rumah untuk setiap kategori.

```
print("Jumlah data rumah berdasarkan kategori harga")
print(---[---].value_counts())
```


MODUL V

REGRESI LINIER

A. Pokok Bahasan

1. Regresi linier sederhana
2. Regresi linier berganda

B. Tujuan Pembelajaran

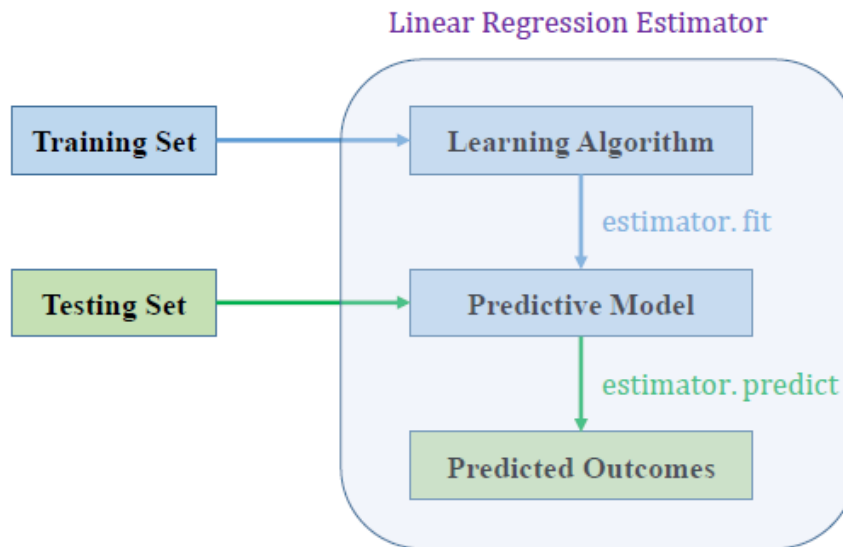
1. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model dengan regresi linier sederhana.
2. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model dengan regresi linier berganda.

C. Dasar Teori

Regresi adalah salah satu teknik yang dapat digunakan memperkirakan nilai dari suatu variable dengan mempelajari relasi variable tersebut dengan variable lain yang relevan. Salah satu jenis regresi adalah regresi linier. Regresi linier menggunakan relasi antara titik-titik data untuk menentukan

sebuah garis lurus yang melewati titik data tersebut. Garis yang dihasilkan data digunakan untuk memprediksi nilai. Regresi linier terdiri dari dua jenis yaitu regresi linier sederhana dan regresi linier berganda.

Tahapan memprediksi nilai menggunakan regresi linier ditunjukkan dengan Gambar 1.



Gambar 1. Proses Regresi Linier
[Sumber : Tayo, 2018 url : pub.towardsai.net/]

I. Regresi Linier Sederhana

Regresi linier sederhana merupakan regresi dengan satu variable bebas, yaitu prediksi sebuah nilai berdasarkan satu variable bebas ini. Persamaan regresi linear sederhana sebagai berikut :

$$y = w_0 + w_1x$$

II. Regresi Linier berganda

Regresi linier berganda merupakan regresi yang memiliki lebih dari satu variable bebas, yaitu prediksi suatu nilai berdasarkan dua atau lebih variabel. Persamaan regresi linier berganda sebagai berikut :

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n$$

dimana :

w_0 : intercept

$w (w_1, w_2, \dots, w_n)$: vector bobot (koefisien)

x : variable independent

Python menyediakan library scikit learn yang dapat digunakan untuk membuat model regresi. Import library untuk membangun model regresi sebagai berikut :

```
from sklearn import linear_model
```

Berikut beberapa fungsi dalam kelas linear_model :

Fungsi	Deskripsi
fit(X, y)	Training model linear menggunakan data latih dengan atribut bebas X dan target y.
predict(X)	Melakukan prediksi untuk data X menggunakan model linear.

D. Tahapan Praktikum

I. Regresi Linear Sederhana

1. Menyiapkan data *training*

```
import pandas as pd
import matplotlib.pyplot as plt

nilai = {
    "jam_belajar" : [2,2,2.5,4,5,5,7,8,8,9],
    "nilai_ujian" : [57,60,66,73,76,79,81,90,95,100]
}

# Data Training
df_nilai = pd.DataFrame(nilai)
display(df_nilai)

# Scatter Plot
df_nilai.plot(kind='scatter', x='jam_belajar', y='nilai_ujian', title='Nilai Vs Lama Belajar')
```

2. Training Data dan Prediksi Nilai

```
import numpy as np
from sklearn import linear_model

# Variabel bebas dan variabel target
X = df_nilai[['jam_belajar']]
y = df_nilai['nilai_ujian']

# Membuat objek regresi linear
regr_s = linear_model.LinearRegression()

# Training model menggunakan data latih
regr_s.fit(X.values, y)

# Nilai koefisien dan nilai intercept
print("Koefisien : ", regr_s.coef_)
```

```

print("Intercept : ", regr_s.intercept_)
print("Model regresi : y= {} + {}x".format(regr_s.intercept_,regr_s.coef_[0]))

# Memprediksi dengan satu data uji
jam_belajar_uji = 3
nilai_ujian_pred = regr_s.predict([[jam_belajar_uji]])

print("\nPrediksi nilai ujian dengan jam belajar {} jam :
{}".format(jam_belajar_uji,nilai_ujian_pred[0]))

# Memprediksi lebih dari satu data uji
jam_belajar_uji = pd.DataFrame(np.array([[3],[3.5],[6]]), columns=['jam_belajar'])
nilai_ujian_pred = regr_s.predict(jam_belajar_uji.values)

# Hasil prediksi
print("\nHasil Prediksi 3 data uji")
jam_belajar_uji['prediksi_nilai_ujian'] = nilai_ujian_pred
print(jam_belajar_uji)

```

3. Membuat Regression Plot

```

import seaborn as sns

#Regression Plot
sns.lmplot(x="jam_belajar", y="nilai_ujian", data=df_nilai, line_kws={"color": "C1"}, ci=None)

```

II. Regresi Linier Berganda

Regresi linier berganda menggunakan data rumah di Jakarta Selatan. File data yang digunakan adalah HARGA RUMAH JAKSEL.csv dengan atribut :

LT : Luas Tanah (m2)
 LB : Luas Bangunan (m2)
 JKT : Jumlah Kamar Tidur
 JKM : Jumlah Kamar Mandi
 GARASI : Ketersediaan garasi
 HARGA : Harga rumah (Milyar)

```

import numpy as np
import pandas as pd
from sklearn import linear_model

# Load dataset rumah jaksel
df_rumah = pd.read_csv('HARGA RUMAH JAKSEL.csv')

# Training Data
rumah_X_train = df_rumah.loc[:200, ('LT','LB','GARASI')]
rumah_X_train = pd.get_dummies(rumah_X_train, prefix='GRS',columns=['GARASI'])

```

```

rumah_y_train = df_rumah.loc[:200,'HARGA']

# Membuat objek regresi linier
regr_b = linear_model.LinearRegression()

# Training model dengan data latih
regr_b.fit(rumah_X_train.values, rumah_y_train)

# Nilai koefisien dan nilai intercept
print("Koefisien : ", regr_b.coef_)
print("Intercept : ", regr_b.intercept_)

# Prediksi harga sebuah rumah dengan LT:550, LB:700, Garasi : ADA
rumah_x_test1 = np.array([[550,700,1,0]])
harga_rumah_pred = regr_b.predict(rumah_x_test1)
print("\nPrediksi rumah LT=550, LB=700, Garasi=ADA :", harga_rumah_pred, "M")

# Prediksi harga beberapa rumah
rumah_X_test2 = df_rumah.loc[201:205, ('LT','LB','GARASI')]
rumah_X_test2 = pd.get_dummies(rumah_X_test2, prefix='GRS',columns=['GARASI'])
harga_rumah_pred = regr_b.predict(rumah_X_test2.values)

# Hasil Prediksi dalam tabel
print("\nHasil Prediksi data test 2")
rumah_Y_test2 = df_rumah.loc[201:205, 'HARGA']
rumah_test2 = pd.concat([rumah_X_test2, rumah_Y_test2], axis=1)
rumah_test2['HARGA_PRED'] = harga_rumah_pred
print(rumah_test2)

```

Latihan Praktikum

1. Amati kode program pada percobaan praktikum linear regresi berganda. Berdasarkan hasil pengamatan anda, berikan penjelasan untuk potongan kode berikut :

Kode Python	Deskripsi
df_rumah = pd.read_csv('HARGA RUMAH JAKSEL.csv')	
rumah_X_train = df_rumah.loc[:200, ('LT','LB','GARASI')]	
rumah_X_train = pd.get_dummies(rumah_X_train, prefix='GRS',columns=['GARASI'])	
rumah_y_train = df_rumah.loc[:200,'HARGA']	
regr_b.fit(rumah_X_train.values, rumah_y_train)	

harga_rumah_pred = regr_b.predict(rumah_x_test1)	
rumah_x_test1 = np.array([[550,700,1,0]])	
rumah_X_test2 = df_rumah.loc[201:205, ('LT','LB','GARASI')]	
rumah_Y_test2 = df_rumah.loc[201:205, 'HARGA'] rumah_test2 = pd.concat([rumah_X_test2, rumah_Y_test2], axis=1) rumah_test2['HARGA_PRED'] = harga_rumah_pred	

2. Berdasarkan nilai koefisien dan intercept, tuliskan persamaan regresi yang dihasilkan pada percobaan praktikum regresi linear berganda.
3. Buat kode program untuk memprediksi harga rumah dengan kriteria berikut :

LB	LT	GARASI
120	80	ADA
100	120	TIDAK
100	120	ADA

Berapa prediksi harga untuk masing-masing rumah ?

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

MODUL VI

DECISION TREE CLASSIFIER

A. Pokok Bahasan

1. Decision Tree Data Kategorial
2. Decision Tree Data Numerik

B. Tujuan Pembelajaran

1. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model decision tree untuk data kategorial.
2. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model decision tree untuk data numerik.

C. Dasar Teori

Algoritma decision tree merupakan salah satu algoritma machine learning yang cukup populer. Algoritma ini termasuk kedalam algoritma supervised learning yang dapat digunakan untuk memecahkan masalah klasifikasi. Algoritma ini menghasilkan model yang berbentuk struktur pohon.

Struktur pohon keputusan terdiri dari simpul root, cabang, dan simpul daun. Setiap simpul melambangkan sebuah atribut data, setiap cabang melambangkan output dari kondisi pengujian, dan setiap simpul daun menyatakan label kelas.

Output dari model decision tree mudah untuk diinterpretasikan dan menyediakan aturan yang mengarah ke keputusan. Data training digunakan untuk membangun sebuah model decision tree, dimana akan ditentukan variabel pembagi dan nilai pembagi. Terdapat beberapa hal yang perlu diperhatikan dalam implementasi algoritma decision tree, yaitu :

1. Pertama, semua data training dianggap sebagai root.
2. Nilai dari setiap atribut harus bertipe kategorial. Apabila terdapat atribut numerik maka perlu dilakukan diskritisasi sebelum model dibuat.
3. Data training didistribusi secara rekursif berdasarkan nilai atributnya.
4. Penentuan root atau atribut pemilah didasarkan pada ukuran impurity setiap atribut.

Algoritma Decision Tree

Proses pembuatan decision tree sebagai berikut :

- Pohon dibuat secara top-down.
- Dimulai dari sebuah pohon kosong dan mula-mula semua data training dianggap sebagai root.
- Data input dibagi secara rekursif berdasarkan atribut yang terpilih.
- Pemilihan atribut pemilah didasarkan pada ukuran impurity antara lain gini indeks atau information gain (entropy).
- Proses pembentukan tree berhenti ketika memenuhi kondisi berikut :
 - Semua data sampel pada semua simpul telah memiliki kelas yang sama.
 - Tidak ada atribut tersisa untuk proses partisi – label pada simpul daun ditentukan dari label data terbanyak pada simpul tersebut.
 - Tidak ada sampel data yang tersisa untuk dipartisi.

Pemilihan Atribut Pemilah

Tantangan utama pada implementasi decision tree adalah pada pemilihan atribut pemilah terbaik. Pemilihan atribut ini yang menentukan atribut yang menjadi root ataupun simpul-simpul pada setiap subtree. Pemilihan atribut pemilah menggunakan nilai impurity dari setiap atribut. Terdapat 2 ukuran impurity yang banyak digunakan, yaitu : information gain dan Gini Index.

Information Gain

Information gain merupakan ukuran impurity yang digunakan untuk mengestimasi kandungan informasi pada sebuah atribut. Nilai information gain dapat diperoleh melalui nilai entropy. Entropy digunakan untuk mengukur impurity pada sebuah dataset. Information gain diperoleh dari menghitung selisih antara entropy data sebelum dibagi dan rata-rata entropy dataset setelah dibagi berdasarkan nilai atribut.

Rumus untuk entropy sebagai berikut :

$$Entropy = \sum_{i=1}^C -p_i * \log_2(p_i)$$

Dimana, c adalah jumlah kelas dan p_i adalah probabilitas data kelas i terhadap semua objek data dalam dataset.

Rumus information gain sebagai berikut :

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Dimana, $|S_v|$ adalah jumlah objek untuk nilai v pada atribut A dan $|S|$ adalah jumlah objek data pada sampel data.

ID3 (Iterative Dichotomiser) merupakan algoritma Decision Tree yang menggunakan entropy untuk menghitung information gain. Atribut dengan nilai information gain tertinggi dipilih sebagai atribut pemilah terbaik.

Gini Index

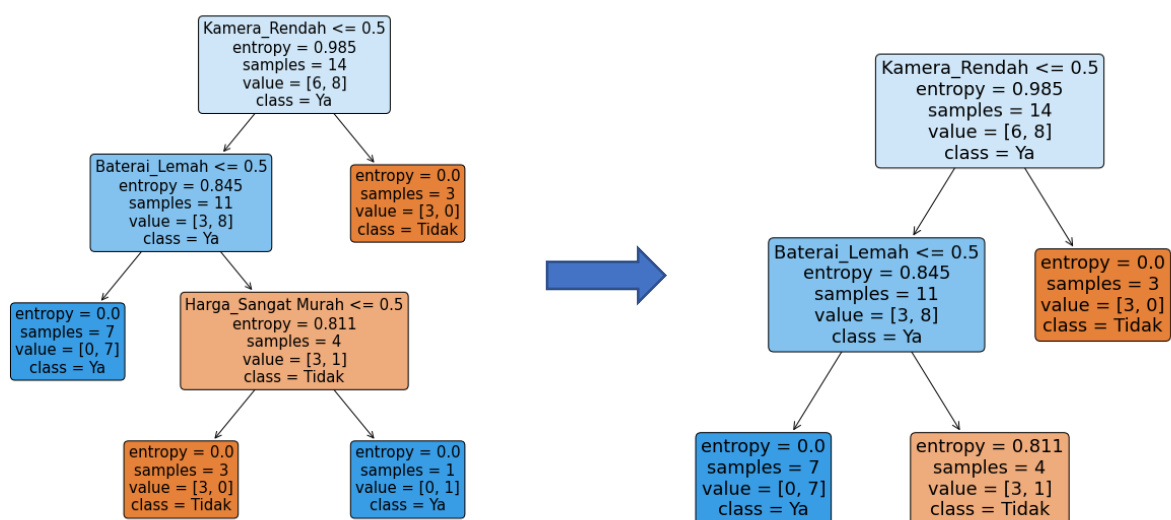
Gini index merupakan ukuran pemilah atribut terbaik yang digunakan pada algoritma CART (Classification and Teggession Tree). Rumus Gini Index sebagai berikut :

$$Gini = 1 - \sum_{i=1}^c (p_i)^2$$

dimana p_i adalah probabilitas untuk setiap label kelas i .

Pruning

Pruning adalah proses menghentikan proses pembuatan subtree di level tertentu apabila ukuran kebaikan simpul lebih kecil dari nilai threshold yang telah ditetapkan. Contoh decision tree sebelum dan setelah pruning ditunjukkan pada Gambar 1.



Gambar 1. Decision Tree Sebelum dan Setelah Pruning

Library Algoritma Decision Tree

Python menyediakan library scikit learn yang dapat digunakan untuk membuat model decision tree. Import library untuk membangun model regresi sebagai berikut :

```
from sklearn.tree import DecisionTreeClassifier
```

Berikut beberapa fungsi dalam kelas DecisionTreeClassifier :

Fungsi	Deskripsi
fit(X, y)	Training model decision tree menggunakan data latih dengan atribut bebas X dan target y.
predict(X)	Melakukan prediksi untuk data X menggunakan model decision tree.

D. Tahapan Praktikum

I. Decision Tree Variabel Kategorial

Decision Tree untuk data dengan variabel kategorial menggunakan dataset terkait dengan apakah suatu handphone direkomendasikan untuk dibeli oleh *vlogger* pemula. Rekomendasi handphone berdasarkan atribut resolusi kamera, kekuatan baterai, dan harga handphone.

1. Import Library

```
import pandas as pd
import matplotlib.pyplot as plt
```

2. Import Dataset

```
df_hp = pd.read_csv('handphone.csv')
display(df_hp.head())
df_hp.shape
```

3. Menentukan atribut bebas dan atribut target atau kelas

```
fitur = ['Baterai', 'Kamera', 'Harga']

X = pd.get_dummies(df_hp[fitur], prefix=['Baterai', 'Kamera', 'Harga'])
y = df_hp['Layak_Direkomendasikan']
display(X)
```

4. Decision Tree Classifier menggunakan Information Gain

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

# Membuat objek decision tree
clf_tree = DecisionTreeClassifier(criterion='entropy', max_depth=2, random_state=0)
```

```
# Train Model
clf_tree = clf_tree.fit(X.values, y)

# Decision Tree
plt.figure(figsize=(10,10))
tree.plot_tree(clf_tree, feature_names=X.columns, class_names=y.sort_values().unique(),
               filled=True, rounded=True)
plt.show()

# Ekstrak aturan dari decision tree
aturan = tree.export_text(clf_tree, feature_names=list(X.columns))
print(aturan)
```

5. Prediksi

```
# HP15 : Baterai : Cukup, Kamera : Rendah, Harga : Sangat Mahal
pred_rekomendasi = clf_tree.predict([[1,0,0,1,0,0,0,1,0]])
print('Prediksi Rekomendasi HP15 : ', pred_rekomendasi)

# HP16 : Baterai : Cukup, Kamera : Tinggi, Harga : Mahal
pred_rekomendasi = clf_tree.predict([[1,0,0,0,0,1,1,0,0,0]])
print('Prediksi Rekomendasi HP16 : ', pred_rekomendasi)
```

II. Decision Tree Variabel Numerik

Decision Tree untuk data dengan variabel numerik menggunakan dataset terkait dengan apakah direkomendasikan untuk bermain sepak bola berdasarkan variabel temperature (C), humidity(%), weather, dan wind.

1. Import library dan membaca data

```
import pandas as pd
import matplotlib.pyplot as plt

df_football = pd.read_csv('football.csv')
display(df_football.head())
df_football.shape
```

2. Penentuan atribut bebas dan target (kelas)

```
fitur=['weather', 'temperature', 'humidity', 'wind']
X = pd.get_dummies(df_football[fitur], prefix=['weather','wind'], columns=['weather','wind'])
y = df_football.play
```

3. Decision Tree Classifier dengan Gini Index

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
```

```
# Membuat objek decision tree
clf_tree = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=0)

# Train Model
clf_tree = clf_tree.fit(X.values, y)

# Decision Tree
plt.figure(figsize=(15,20))
tree.plot_tree(clf_tree, feature_names=X.columns, class_names=y.sort_values().unique(),
               filled=True, rounded=True)
plt.show()

# Ekstrak aturan dari decision tree
aturan = tree.export_text(clf_tree, feature_names=list(X.columns))
print(aturan)
```

4. Prediksi

```
# Play15 : weather : Sunny, temp : 37 , humidity : 90 , wind : False
pred_rekomendasi = clf_tree.predict([[37,90,0,0,1,1,0]])
print('Prediksi Play15 : ', pred_rekomendasi)

# Play16 : weather : Cloudy, temp : 27 , humidity : 70 , wind : True
pred_rekomendasi = clf_tree.predict([[28,70,1,0,0,0,1]])
print('Prediksi Play16 : ', pred_rekomendasi)
```

Latihan Praktikum

1. Buat kode program untuk membuat model decision tree dari data berikut. Ukuran impurity dapat menggunakan entropy(information gain) atau gini index. (Sumber Data : Suyanto, 2018)

Handphone	Baterai (Jam)	Kamera (MP)	Harga (Juta)	Layak_Direkomendasikan
H1	26	8	1.2	Ya
H2	27	13	15	Ya
H3	28	5	6	Ya
H4	25	2	5	Tidak
H5	23	10	1	Ya
H6	20	7	3.5	Ya
H7	22	7	10	Ya
H8	24	8	2	Ya
H9	21	3	4	Tidak
H10	16	13	0.8	Ya
H11	12	10	12	Tidak
H12	14	5	5	Tidak
H13	18	5	3	Tidak
H14	15	3	14	Tidak

2. Screenshot decision tree yang dihasilkan dari soal nomor 1 dan jelaskan beberapa aturan klasifikasi yang anda dapat amati dari model decision tree tersebut.
3. Prediksi rekomendasi HP dengan kriteria berikut :

- a. Baterai : 22, Kamera : 5 , Harga : 3
- b. Baterai : 20 , Kamera : 8 , Harga : 3.5

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

MODUL VII

NAÏVE BAYES CLASSIFIER

A. Pokok Bahasan

1. Gaussian Naïve Bayes
2. Confusion Matrix

B. Tujuan Pembelajaran

1. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model Gaussian Naïve Bayes.
2. Mahasiswa mampu menggunakan membuat confusion matrix dan menghitung akurasi dari model yang dibuat.

C. Dasar Teori

Algoritma naïve bayes adalah algoritma klasifikasi berbasis teorema bayes. Naïve bayes dalam melakukan klasifikasi data mengasumsikan bahwa setiap fitur data tidak saling terikat. Sebagai contoh untuk menentukan seseorang layak diberikan kredit pada suatu bank maka dilihat dari pendapatan, riwayat pinjaman sebelumnya, transaksi perbankannya, dan usia. Walaupun fitur-fitur ini sebenarnya memiliki keterkaitan, namun fitur ini dianggap tidak saling terikat. Asumsi ini yang membuat naïve bayes sederhana dalam komputasi, dan oleh karena itu disebut naïve.

Naïve bayes melakukan klasifikasi dengan menghitung probabilitas posterior, yaitu nilai probabilitas dari setiap label kelas terhadap suatu objek dengan beberapa fitur. Formula untuk menghitung probabilitas posterior sebagai berikut:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

Dimana :

$P(C_i|X)$ adalah probabilitas posterior kelas terhadap suatu objek data

$P(X|C_i)$ adalah probabilitas fitur objek X terhadap kelas

$P(C_i)$ adalah probabilitas prior dari kelas

$P(X)$ probabilitas prior setiap fitur objek yang diprediksi

Namun, karena diketahui bahwa setiap fitur objek prediksi adalah sama untuk setiap kelas maka nilai $P(X)$ tidak perlu disertakan dalam penghitungan probabilitas posterior. Sehingga, persamaan perhitungan probabilitas posterior cukup menggunakan formula berikut :

$$P(C_i|X) = P(X|C_i)P(C_i)$$

Tahapan Algoritma Naïve Bayes

Proses pembuatan klasifikasi dengan naïve bayes, umumnya memiliki tahapan sebagai berikut :

- Tentukan objek yang akan diprediksi;
- Hitung probabilitas prior untuk setiap label kelas;
- Hitung probabilitas bersyarat untuk setiap atribut terhadap setiap label kelas;
- Hitung probabilitas setiap kelas berdasarkan objek yang diprediksi;
- Bandingkan nilai probabilitas setiap label kelas, label kelas dengan probabilitas tertinggi yang menjadi label kelas dari objek yang diprediksi .

Jenis Algoritma Naïve Bayes

- Gaussian Naïve Bayes.

Algoritma ini digunakan ketika objek yang diprediksi memiliki fitur yang bernilai kontinu atau numerik. Probabilitas fitur yang bernilai kontinu terhadap kelas dihitung dengan persamaan berikut :

$$P(x_k|C_i) = \frac{1}{\sigma_{ik} \sqrt{2\pi}} e^{-\frac{(x_k - \mu_{ik})^2}{2\sigma_{ik}^2}}$$

Dimana μ_{ik} dan σ_{ik} adalah mean dan standar deviasi dari nilai-nilai pada atribut A_k untuk kelas C_i

- Multinomial Naïve Bayes

Multinomial Naïve bayes adalah penggunaan naïve bayes untuk distribusi data multinomial. Jenis naïve bayes ini yang digunakan pada klasifikasi text.

- **Bernoulli Naïve Bayes**
Bernoulli Naïve Bayes digunakan pada data dengan banyak fitur dan fiturnya bernilai biner. Pada implementasi di klasifikasi teks maka fiturnya adalah muncul tidaknya kata dalam teks bukan jumlah kemunculan kata dalam teks.

Aplikasi Naïve Bayes

Naïve bayes adalah salah satu algoritma yang mudah dan cepat dalam proses klasifikasi. Algoritma ini cocok diterapkan pada data dengan volume yang besar. Algoritma ini telah berhasil diterapkan pada beberapa aplikasi berikut :

- Spam Filtering
- Text Classification
- Sentiment Analysis
- Recommender System

Library Algoritma Naïve Bayes

Python menyediakan library scikit learn yang dapat digunakan untuk membuat model naïve bayes. Pada praktikum ini jenis naïve bayes yang digunakan adalah Gaussian Naïve Bayes. Cara import library untuk membangun model dengan Gaussian Naïve Bayes sebagai berikut :

```
from sklearn.naive_bayes import GaussianNB
```

Berikut beberapa fungsi dalam kelas GaussianNB :

Fungsi	Deskripsi
fit(X, y)	Training model naïve bayes menggunakan data latih dengan atribut bebas X dan target y.
predict(X)	Melakukan prediksi untuk data X menggunakan model naïve bayes.

Confusion Matrix

Confusion Matrix adalah tabel yang dapat digunakan untuk mengetahui performansi dari sebuah model klasifikasi yang dibuat. Format dari tabel confusion matrix sebagai berikut :

		Prediksi Kelas	
		FALSE	TRUE
Kelas Sebenarnya	FALSE	Jumlah TN	Jumlah FP
	TRUE	Jumlah FN	Jumlah TP

Gambar 1. Confusion Matrix

- **True Negatives (TN)** : Jumlah data dengan kelas sebenarnya FALSE dan diprediksi sebagai FALSE.
- **False Positif (FP)** : Jumlah data dengan kelas sebenarnya FALSE dan diprediksi sebagai TRUE.
- **False Negatives (FN)** : Jumlah data dengan kelas sebenarnya TRUE dan diprediksi sebagai FALSE.
- **True Positives (TP)** : Jumlah data dengan kelas sebenarnya TRUE dan diprediksi sebagai TRUE.

Berdasarkan nilai pada confusion matrix, kita bisa mengukur performansi suatu model dengan beberapa ukuran sebagai berikut :

Ukuran	Deskripsi	Formula
Akurasi	Berapa % prediksi yang benar ?	$(TP+TN)/(TP+TN+FP+FN)$
Rasio kesalahan klasifikasi	Berapa % prediksi yang salah ?	$(FP+FN)/(TP+TN+FP+FN)$
Presisi	Berapa % prediksi positif yang benar dari seluruh data yang diprediksi positif?	$TP/(TP+FP)$
Recall	Berapa % prediksi positif yang benar dari seluruh data positif?	$TP/(TP+FN)$

Dataset

Pada praktikum ini dataset yang digunakan adalah dataset pembelian produk berdasarkan iklan di media sosial. Dataset ini dapat digunakan untuk memprediksi apakah seseorang akan membeli produk dari iklan di media sosial. Jumlah data dalam dataset sebanyak 400 data, dan jumlah atribut adalah 5 dimana 3 atribut dapat dijadikan fitur/prediktor dan 1 atribut sebagai target/kelas. Nama-nama atribut dataset ini sebagai berikut :

- **User ID** : ID pengguna, atribut ini tidak perlu digunakan dalam pembuatan model machine learning.
- **Gender** : Nominal dengan nilai Male dan Female.
- **Age** : Numerik berupa usia pelanggan.
- **EstimatedSalary** : Numerik berupa estimasi gaji tahunan pengguna dalam dollar.
- **Purchased** : Nilai dari Purchased adalah 1 dan 0, dimana 1 berarti membeli dan 0 tidak membeli.

Sumber dataset : <https://www.kaggle.com/datasets/rakeshrau/social-network-ads>

D. Tahapan Praktikum

1. Import Library dan dataset

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Import dataset
df_ads = pd.read_csv('Social_Network_Ads.csv')
print(df_ads.head())
print("Ukuran data : ", df_ads.shape)
```

2. Menampilkan ringkasan data

```
df_ads.info()
```

3. Menampilkan statistik data


```
df_ads.describe()
```

4. Menentukan fitur(X) dan kelas(y). Fitur yang digunakan adalah atribut Age dan EstimatedSalary

```
X = df_ads.iloc[:, [2,3]]
y = df_ads.iloc[:, 4]

print(X)
print(y)
```

5. Membagi data set menjadi 75% (300 data) sebagai data latih dan 25%(100 data) akan digunakan sebagai data uji. Untuk membagi dataset menjadi data latih dan data uji, kita dapat menggunakan **train_test_split** dari library sklearn.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size = 0.25,
                                                    random_state=0)
# Mencetak jumlah data latih dan data uji
print("Jumlah data latih : {} data".format(X_train.shape[0]))
print("Jumlah data uji : {} data".format(X_test.shape[0]))
```

6. Normalisasi data dengan Z score agar distribusi data menjadi distribusi normal (Gaussian Distribution)

```
from sklearn.preprocessing import StandardScaler

sc= StandardScaler()
X_train_norm = sc.fit_transform(X_train)
X_test_norm = sc.transform(X_test)

# mencetak data yang telah dinormalisasi
print(X_train_norm[0:5])
print(X_test_norm[0:5])
```

7. Membuat model dengan Gaussian Naïve Bayes

```
from sklearn.naive_bayes import GaussianNB

clf = GaussianNB()
clf.fit(X_train_norm, y_train)
```

8. Memprediksi kelas dari data uji

```
y_pred = clf.predict(X_test_norm)
print(y_pred)
```

9. Menampilkan hasil prediksi data uji beserta atribut data lainnya dalam bentuk tabel.

```
data_hasil_prediksi = pd.DataFrame(X_test, columns=['Age', 'EstimatedSalary'])
data_hasil_prediksi = pd.concat([data_hasil_prediksi, pd.Series(y_test, name='Purchased')],
axis=1)

data_hasil_prediksi['Purchased_Pred'] = y_pred
data_hasil_prediksi.head(20)
```

10. Membuat Confusion Matrix untuk menghitung akurasi

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
print("Jumlah data dengan label 0 dan hasil prediksi 0 (TN) :", cm[0,0])
print("Jumlah data dengan label 0 dan hasil prediksi 1 (FP):", cm[0,1])
print("Jumlah data dengan label 1 dan hasil prediksi 1 (TP):", cm[1,1])
print("Jumlah data dengan label 1 dan hasil prediksi 0 (FN):", cm[1,0])

akurasi = ((cm[0,0]+cm[1,1])/(cm[0,0]+cm[1,1]+cm[0,1]+cm[1,0]))*100
print("Akurasi : {} %".format(akurasi))
```

11. Membuat Diagram Confusion Matrix

```
from sklearn import metrics

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm)
cm_display.plot()
plt.show()
```

Latihan Praktikum

1. Buat kode program untuk membuat model naïve bayes untuk memprediksi potensi seseorang membeli produk dari iklan di media sosial berdasarkan fitur **Gender, Age, dan EstimatedSalary**. Gunakan dataset `Social_Network_ads.csv` dengan pembagian **80% data latih dan 20% data uji**.
2. Berapa akurasi dari model yang ada buat?
3. Buat diagram confusion matrix dari hasil pengujian model naïve bayes anda.

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

MODUL VIII

JARINGAN SYARAF TIRUAN 1

A. Pokok Bahasan

1. Perceptron
2. Akurasi model dengan confusion matrix

B. Tujuan Pembelajaran

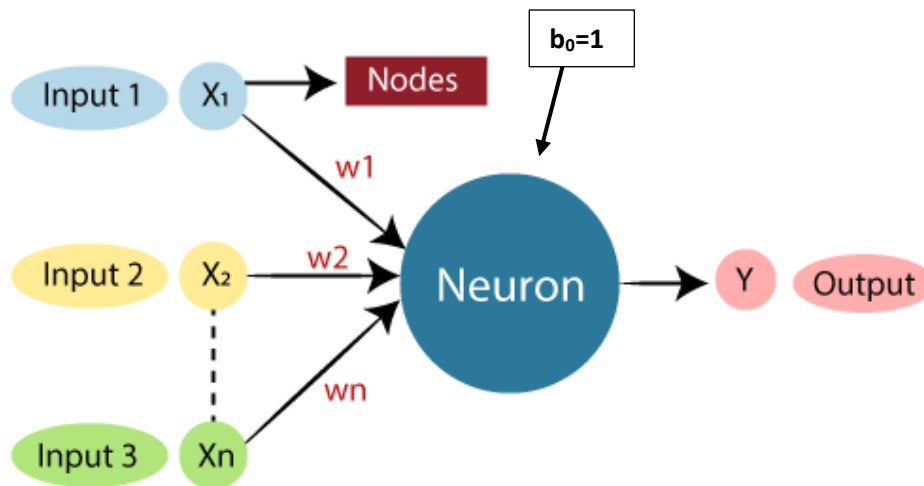
1. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model perceptron
2. Mahasiswa mampu mengimplementasikan perceptron untuk klasifikasi data menggunakan Bahasa Pemrograman Python.
3. Mahasiswa mampu menguji akurasi prediksi pada model perceptron.

C. Dasar Teori

Jaringan syaraf tiruan (JST) adalah salah satu temuan yang paling penting di bidang Machine Learning. JST merupakan metode yang mencoba meniru cara kerja sistem syaraf manusia dimana JST memiliki memiliki neuron yang saling terkoneksi satu dengan yang lainnya. Neuron yang saling terkoneksi adalah neuron-neuron yang berada pada layer yang berbeda. Secara umum JST memiliki 3 layer, yaitu input, hidden layer, dan output layer.

Perceptron

Perceptron pertama kali diperkenalkan pada tahun 1943 oleh McCulloch dan Pitts. Perceptron merupakan salah satu jenis arsitektur JST yang hanya memiliki dua layer yaitu input layer dan output layer. Arsitektur JST ini hanya dapat menghasilkan model linear, dimana untuk permasalahan klasifikasi hanya mampu mengklasifikasikan data ke dalam dua kelas. Gambar 1 menunjukkan arsitektur perceptron.



Gambar 1. Perceptron
[Sumber : javapoint.com]

Layer input pada perceptron terdiri dari node input yang digunakan untuk menerima masukan data training. Setiap node input akan terkoneksi dengan node pada layer output. Koneksi antar node direpresentasikan dengan bobot dan bias. Sedangkan, node pada layer output menghasilkan nilai output.

Sebuah neuron dideskripsikan secara matematis sebagai berikut :

$$u_k = \sum_{j=1}^m w_{kj} x_j$$

dan

$$y_k = \varphi(u_k + b_k)$$

Dimana x_1, x_2, \dots, x_n adalah input, $w_{k1}, w_{k2}, \dots, w_{kn}$ adalah bobot dari setiap koneksi k , u_k adalah kombinasi linier dari output yang dihasilkan node, b_k adalah bias, φ adalah fungsi aktivasi, dan y_k adalah output dari sebuah neuron.

Fungsi aktivasi yang dapat digunakan di perceptron adalah

- **Fungsi Threshold**
- **Fungsi Linear Piecewise**
- **Fungsi Sigmoid**

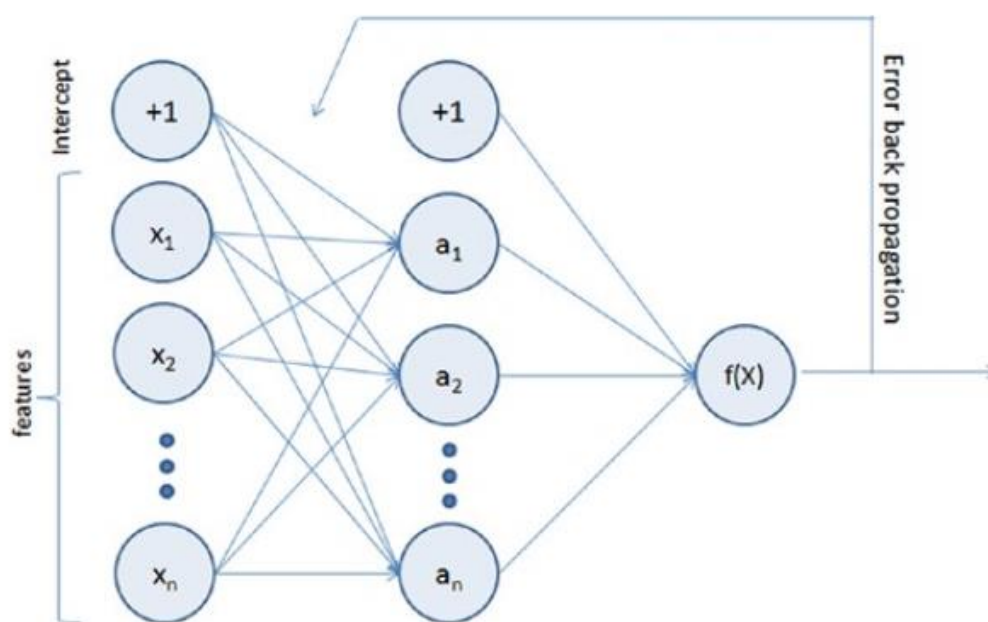
Algoritma Perceptron

1. Inisialisasi $w = 0$

2. Lakukan komputasi berikut untuk semua i dari $i=1, \dots, m$. Mulai proses dengan memasukkan data x dan target d .
3. Hitung $g(x_i) = \text{sgn}(wx')$ dimana $\text{sgn}(\cdot)$ adalah fungsi signum. Semua nilai di atas 0 diberi nilai +1 sebaliknya -1.
4. Jika $d_i \neq g(x_i)$, update w , menggunakan $w = w + \eta[d_i - g(x_i)] x_i$
5. Sampai semua $1 \leq i \leq m$ diperoleh $g(x_i) = d_i$

Multilayer Perceptron

Multilayer perceptron merupakan arsitektur jaringan syaraf tiruan yang terdiri dari layer input, hidden layer, dan output layer. Tidak seperti perceptron, arsitektur ini mampu mengklasifikasi data ke lebih dari dua kelas. Cara kerja arsitektur ini dimulai dengan node pada layer input akan menerima data input kemudian diproses di node-node pada hidden layer. Node output menghasilkan nilai output yang akan dihitung errornya yaitu selisih nilai sebenarnya dengan nilai yang dihasilkan node output. Nilai error diminimalisasi menggunakan metode gradient descent yang menghitung loss function untuk menemukan nilai bobot dan bias yang optimal. Gambar 2 menunjukkan gambar arsitektur Multilayer Perceptron dengan satu hidden layer.



Gambar 2. Multilayer Perceptron
[Sumber : Swamynathan M., 2017]

Algoritma Back Propagation

Algoritma back propagation merupakan salah satu algoritma untuk proses training data pada multilayer perceptron. Hasil dari proses training adalah nilai w dan b yang paling optimal. Berikut langkah training data dengan algoritma Back Propagation :

1. Inisialisasi setiap nilai w dan b dengan nilai random.
2. Hitung nilai output setiap data latih dengan nilai w dan b yang ditentukan sebelumnya. Pada langkah ini diperoleh output prediksi setiap data.
3. Hitung nilai error output, $\text{Error} = \text{Output sebenarnya} - \text{output prediksi}$.

4. Dari output layer, lakukan back propagation ke hidden layer untuk menyesuaikan setiap bobot dan bias dengan menggunakan gradient descent.
5. Ulangi langkah 2 sampai kondisi berhenti terpenuhi. Kondisi berhenti dapat menggunakan maksimum iterasi atau perubahan w dan b sangat kecil atau mendekati 0.

Library Algoritma Perceptron

Python menyediakan library scikit learn yang dapat digunakan untuk membuat model perceptron. Membuat model dengan perceptron dapat dilakukan dengan import kelas sklearn.linear_model.Perceptron. Perintah untuk mengimport kelas sebagai berikut :

```
from sklearn.linear_model import Perceptron
```

Pada kelas perceptron terdapat parameter yang dapat ditentukan pada saat membuat objek perceptron, beberapa parameter yang digunakan pada praktikum ini adalah :

Parameter	Deskripsi
n_iter	Jumlah iterasi maksimum proses training data. Nilai default = 1000
eta0	Learning rate. Nilai default = 1
tol	Kriteria berhenti. Iterasi akan berhenti ketika perubahan bobot < tol. Nilai default = 1e-3

Beberapa atribut dari kelas perceptron antara lain :

Atribut	Deskripsi
Classes_	Label kelas dari data
Coef_	Nilai bobot model perceptron
Intercept_	Nilai bias
n_iter_	Jumlah iterasi yang terjadi pada saat proses training sampai kriteria berhenti terpenuhi.

Berikut beberapa fungsi dalam kelas perceptron:

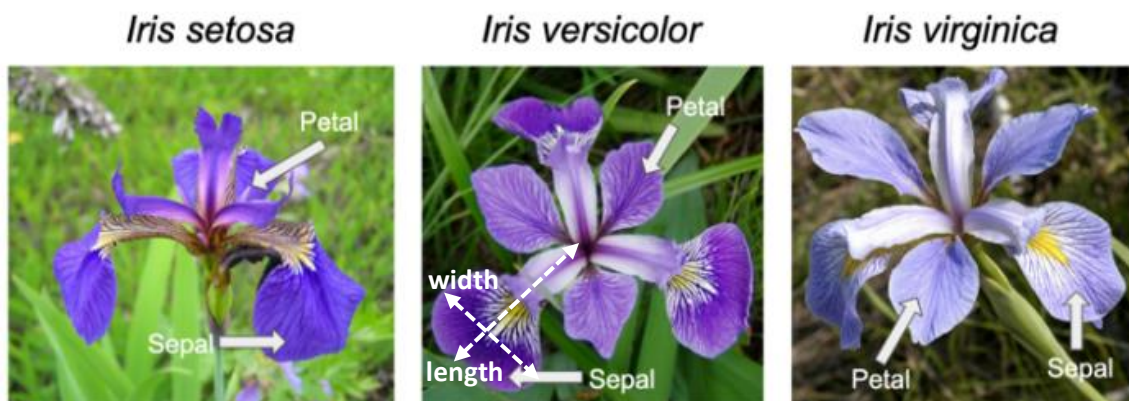
Fungsi	Deskripsi
fit(X, y, coef_init=None, intercept_init=None, sample_weight=None)	Training model perceptron menggunakan data latih dengan atribut bebas atau prediktor X dan target y. Parameter coef_init dapat digunakan untuk menentukan bobot awal sedangkan intercept_init digunakan untuk menentukan nilai bias awal. Apabila kedua nilai ini tidak ditentukan maka nilai bobot dan bias awal ditentukan secara random.
predict(X)	Melakukan prediksi kelas untuk data X menggunakan model perceptron.

Dataset

Pada praktikum ini dataset yang digunakan adalah dataset bunga Iris. Dataset ini digunakan untuk melakukan klasifikasi jenis bunga Iris. Jumlah data dalam dataset sebanyak 150 data, dan jumlah atribut adalah 5 dimana 4 atribut dapat dijadikan fitur/prediktor dan 1 atribut sebagai target/kelas. Atribut dataset ini sebagai berikut :

- **sepal length (cm)** : Panjang daun bunga.
- **sepal width (cm)** : Lebar daun bunga.
- **petal length (cm)** : Panjang kelopak bunga.
- **petal width (cm)** : Lebar kelopak bunga.
- **target** : Kelas data yang berisi 3 jenis bunga yaitu setosa, versicolor, dan virginica dengan label 0 untuk jenis setosa, 1 untuk jenis versicolor, dan 2 untuk virginica.

Dataset ini tersedia di library scikit learn sehingga untuk dapat diimport dengan menuliskan kode `from sklearn.datasets import load_iris` di Python. Gambar 3 menampilkan contoh bunga iris beserta keterangan atribut dan jenis bunganya.



Gambar 3 Bunga Iris

D. Tahapan Praktikum

1. Import Library dan dataset

```
import numpy as np
from sklearn.datasets import load_iris

data_iris = load_iris(as_frame=True)

display(data_iris.data)
print(data_iris.target)
print("Nilai Label : ", data_iris.target_names)
```

2. Mengubah label kelas untuk Iris virginica menjadi 1 dan bukan jenis iris virginica menjadi 0. Hal ini dilakukan karena perceptron hanya mampu melakukan klasifikasi data ke dalam dua kelas.

Sehingga pada praktikum ini dataset yang memiliki 3 label kelas diubah menjadi dua kelas yaitu jenis iris virginica dan bukan jenis virginica (versicolor dan setosa)

```
targets = (data_iris.target==2).astype(np.int8)
print(np.array(targets))
```

3. Menampilkan data menggunakan scatter plot. Memvisualisasikan data dapat membantu untuk mengetahui karakteristik data yang akan dimodelkan.

```
import matplotlib.pyplot as plt

fig, ax1 = plt.subplots(2, 1, figsize=(9,12))
colors = {0:'red', 1:'blue'}
ax1[0].scatter(data_iris.data.iloc[:,0],data_iris.data.iloc[:,1], c=targets.map(colors))
ax1[1].scatter(data_iris.data.iloc[:,2],data_iris.data.iloc[:,3], c=targets.map(colors))

ax1[0].set(xlabel="Sepal Length", ylabel='Sepal Width')
ax1[1].set(xlabel="Petal Length", ylabel='Petal Width')
plt.show()
```

4. Membagi data set menjadi 75% (112 data) sebagai data latih dan 25%(38 data) akan digunakan sebagai data uji. Untuk membagi dataset menjadi data latih dan data uji, kita dapat menggunakan **train_test_split** dari library sklearn.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size = 0.25,
                                                    random_state=0)
# Mencetak jumlah data latih dan data uji
print("Jumlah data latih : {} data".format(X_train.shape[0]))
print("Jumlah data uji : {} data".format(X_test.shape[0]))
```

5. Membuat model data dengan jaringan syaraf tiruan jenis Perceptron menggunakan kelas Perceptron pada library scikit learn. Perceptron melakukan training data dengan menggunakan kriteria berhenti berupa maksimum iterasi 100 atau perubahan bobot < 0.001 serta learning rate 0.1.

```
from sklearn.linear_model import Perceptron

clf_p = Perceptron(max_iter=100, tol=0.001, eta0=0.1, random_state=0)
clf_p.fit(x_train, y_train)
```

6. Mencetak beberapa atribut model hasil training yaitu bobot (coef_), bias(intercept_), dan jumlah iterasi yang dilakukan pada proses training.

```
print("Bobot Optimum: ", clf_p.coef_)
print("Bias: ", clf_p.intercept_)
print("Jumlah iterasi : ", clf_p.n_iter_)
```


7. Memprediksi kelas dari data uji dan mencetak hasil prediksi

```
y_prediksi = clf_p.predict(x_test)

print("Target : ", np.array(y_test))
print("Prediksi Target : ", y_prediksi)
```

8. Membuat Confusion Matrix untuk menghitung akurasi

```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn import metrics

cm = confusion_matrix(y_test, y_prediksi)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cm)
cm_display.plot()

akurasi = accuracy_score(y_test, y_prediksi)*100
print("Akurasi : {} %".format(akurasi))
```

9. Memprediksi apakah bunga iris dengan sepal length 6.7 cm, sepal width 3 cm, petal length 5.2 cm dan, petal width 2.3 cm termasuk jenis virginica atau bukan jenis virginica.

```
y_prediksi_x = clf_p.predict([[6.7, 3, 5.2, 2.3]])
print("Prediksi jenis bunga iris : ", y_prediksi_x)
```

Latihan Praktikum

1. Modifikasi kode pada percobaan 4 dengan beberapa kondisi pembagian dataset berikut. Lakukan training dan testing data dan lengkapi nilai akurasi prediksi untuk masing-masing kondisi pada tabel berikut. Tuliskan apa yang anda dapat simpulkan dari mengamati nilai akurasi tersebut.

Pembagian data set	Akurasi Prediksi
Data latih 80%, Data Uji 20 %	
Data latih 70%, Data Uji 30 %	
Data latih 60%, Data Uji 40 %	
Data latih 40%, Data Uji 60 %	

2. Modifikasi kode pada percobaan 5 untuk maksimum iterasi sebesar 1, 5, 10, dan 15 untuk pembagian dataset yaitu data latih 75% dan data uji 25 % . Bandingkan akurasi prediksi yang dihasilkan untuk masing-masing maksimum iterasi. Berikan kesimpulan anda berdasarkan hasil pengamatan akurasi tersebut.

Iterasi Maksimum	Akurasi Prediksi
1	
5	
10	

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

MODUL IX

JARINGAN SYARAF TIRUAN 2

A. Pokok Bahasan

1. Multilayer Perceptron
2. Akurasi model dengan confusion matrix

B. Tujuan Pembelajaran

1. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model Multilayer Perceptron.
2. Mahasiswa mampu mengimplementasikan Multilayer Perceptron untuk klasifikasi dataset multi kelas menggunakan Bahasa Pemrograman Python.
3. Mahasiswa mampu menguji akurasi prediksi pada model multilayer perceptron.

C. Dasar Teori**Library Algoritma Multilayer Perceptron**

Python menyediakan library scikit learn yang dapat digunakan untuk membuat model dengan multilayer perceptron. Cara import library untuk membangun model dengan multilayer perceptron sebagai berikut :

```
from sklearn.neural_network import MLPClassifier
```

Pada kelas MLPClassifier terdapat parameter yang dapat ditentukan pada saat membuat objek MLPClassifier, beberapa parameter yang digunakan pada praktikum ini adalah :

Parameter	Deskripsi
activation	Fungsi aktivasi di hidden layer. Fungsi aktivasi yang bisa digunakan adalah identity , logistic , tanh , dan relu . Defaultnya adalah relu.

alpha	Parameter untuk membatasi ukuran bobot sehingga menghindari overfitting. Nilai default 0.0001.
batch_size	Parameter untuk menentukan jumlah data sampel yang digunakan pada saat feed forward (stochastic optimizer). Nilai default 'auto'.
beta_1	Decay rate untuk estimasi momen pertama, bernilai [0,1), default 0.9. Hanya digunakan ketika menggunakan solver adam.
beta_2	Decay rate untuk estimasi momen kedua, bernilai [0,1), default 0.999. Hanya digunakan ketika menggunakan solver adam
early_stopping	Ketika bernilai True maka secara otomatis menyisihkan 10% training data untuk validasi dan akan menghentikan proses training jika skor validasi tidak mengalami perubahan nilai yang lebih besar dari nilai tol untuk sejumlah n_iter_no_change iterasi. Apabila bernilai False maka iterasi pada saat training data akan berhenti ketika training loss tidak mengalami perubahan yang lebih besar dari nilai tol untuk n_iter_no_change iterasi secara berturut-turut. Efektif digunakan ketika menggunakan solver 'sgd' atau 'adam'
epsilon	Angka yang sangat kecil untuk mencegah pembagian dengan nol dalam implementasi, default=1e-8. Hanya digunakan ketika menggunakan solver adam.
hidden_layer_sizes	Jumlah node untuk setiap hidden layer, default=(100,) yaitu multilayer perceptron memiliki satu hidden layer dengan 100 node. Jika hidden_layer_sizes=(5,4,2) maka terdapat 3 hidden layer dimana jumlah neuron pada hidden layer 1 adalah 5, hidden layer 2 adalah 4, dan hidden layer 3 adalah 2.
learning_rate	Cara memperbarui learning rate, default='constant'. Terdapat 3 pilihan nilai yaitu constant : learning rate nilainya konstan sesuai dengan nilai learning_rate_init, invscaling : learning rate nilainya berangsur-angsur berkurang sesuai dengan inverse scaling exponent dari power_t, dan adaptive : learning rate akan tetap konstan sesuai dengan learning_rate_init selama nilai loss function terus berkurang, jika tidak maka learning rate dibagi dengan 5.
learning_rate_init	Learning rate awal, default=0.001. Hanya digunakan ketika solver menggunakan stochastic gradient distance ('sgd') dan adam.
max_iter	Iterasi maksimum untuk proses training data.
momentum	Momentum untuk update gradient descent, bernilai [0,1), default=0.9. Hanya digunakan ketika solver menggunakan stochastic gradient distance ('sgd').
nesterovs_momentum	Nesterov momentum digunakan ketika momentum > 0 dan solver menggunakan stochastic gradient distance ('sgd'), default = True.
power_t	Digunakan untuk update learning rate yang efektif, digunakan ketika learning_rate diatur sebagai 'invscaling' dan solver menggunakan stochastic gradient distance ('sgd').
random_state	Menentukan nilai random untuk bobot dan bias awal, train-test split ketika early stopping digunakan, dan batch sampling ketika solver 'sgd' dan 'adam', default=None.
shuffle	Mengacak sampel data untuk setiap iterasi, digunakan ketika solver='sgd' atau 'adam', default=True.
solver	Solver untuk optimasi bobot. Terdapat 3 pilihan solver yaitu quasi-Newton ('lbfgs'), stochastic gradient descent ('sgd'), dan adam ('adam'), default='adam'.

tol	Nilai kondisi berhenti. Ketika perubahan nilai loss tidak berubah lebih besar dari nilai tol untuk n_iter_no_change iterasi berturut-turut maka iterasi training data berhenti dan dianggap konvergen.
validation_fraction	Proporsi dataset yang digunakan untuk validasi untuk early stopping, bernilai [0,1) , default=0.1. Hanya digunakan ketika early_stopping bernilai True.
verbose	Apakah akan mencetak pesan kemajuan ke stdout, default=False.
warm_start	Ketika bernilai True, maka menggunakan kembali solusi pada pemanggilan fungsi fit sebelumnya sebagai inisialisasi, jika false maka menghapus solusi sebelumnya.

Beberapa atribut dari kelas MLPClassifier antara lain :

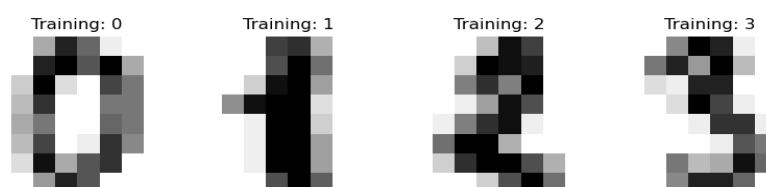
Atribut	Deskripsi
coefs_	Nilai bobot model perceptron
intercepts_	Nilai bias
n_iter_	Jumlah iterasi yang terjadi pada saat proses training sampai kriteria berhenti terpenuhi.
n_layers_	Jumlah layer
n_outputs_	Jumlah node output
out_activation_	Nama fungsi aktivasi output

Berikut beberapa fungsi dalam kelas MLPClassifier :

Fungsi	Deskripsi
fit(X, y)	Training model multilayer perceptron menggunakan data latih dengan atribut bebas atau prediktor X dan target y.
predict(X)	Melakukan prediksi kelas untuk data X menggunakan model multilayer perceptron.

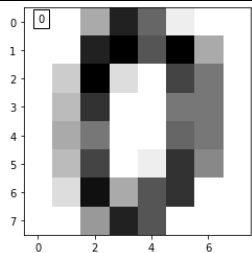
Dataset

Pada praktikum ini dataset yang digunakan adalah dataset **Digits** berisi gambar tulisan tangan untuk angka 0 - 9 . Dataset ini digunakan untuk memprediksi sebuah digit angka yang dituliskan dengan tulisan tangan. Dataset ini terdiri dari 1797 data dimana Setiap data adalah gambar tulisan tangan *grayscale* dengan ukuran 8x8 piksel. Jumlah atribut prediktor sebanyak 64 karena setiap atribut mewakili nilai setiap piksel pada gambar. dan sebuah atribut target. Atribut target berisi angka sesuai dengan yang dituliskan. Gambar 1 menunjukkan contoh sampel dataset



Gambar 1. Dataset Digits
(Sumber : scikit-learn.org)

Tabel 1 menunjukkan gambar tulisan tangan 0 dengan atribut prediktor berupa nilai setiap piksel gambar.

Gambar	Nilai Piksel
	<pre>[[0. 0. 5. 13. 9. 1. 0. 0.] [0. 0. 13. 15. 10. 15. 5. 0.] [0. 3. 15. 2. 0. 11. 8. 0.] [0. 4. 12. 0. 0. 8. 8. 0.] [0. 5. 8. 0. 0. 9. 8. 0.] [0. 4. 11. 0. 1. 12. 7. 0.] [0. 2. 14. 5. 10. 12. 0. 0.] [0. 0. 6. 13. 10. 0. 0. 0.]]</pre>

D. Tahapan Praktikum

1. Import Library.

```
import matplotlib.pyplot as plt

from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.datasets import load_digits
```

2. Load data set dan menampilkan contoh dataset.

```
# load data
digits = load_digits()
print("%d sampel data"%len(digits.target))

# Contoh gambar tulisan tangan
fig = plt.figure(figsize=(4,4))
plt.imshow(digits.images[0], cmap=plt.cm.gray_r)
plt.text(0, 0, str(digits.target[0]), bbox=dict(facecolor='white'))
plt.show()

# Mencetak atribut prediktor sebuah sampel data
print(digits.data[0].reshape(8,8))
```

3. Menampilkan 32 gambar data sampel.

```
fig = plt.figure(figsize=(6,6))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(32):
    ax = fig.add_subplot(8,8,i+1, xticks=[], yticks=[])
    ax.imshow(digits.images[i], cmap=plt.cm.gray_r)
    ax.text(0, 1, str(digits.target[i]), bbox=dict(facecolor='white'))
```

4. Membagi data set menjadi 80% (1437 data) data latih dan 20% (360 data) data uji. Untuk membagi dataset menjadi data latih dan data uji, kita dapat menggunakan **train_test_split** dari library sklearn.

```
X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target,
                                                    test_size=0.2, random_state=0)

print("Jumlah data training : ",len(y_train))
print("Jumlah data uji : ",len(y_test))
```

5. Menampilkan 32 gambar data training.

```
# Plot Data training
fig = plt.figure(figsize=(6,6))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(32):
    ax = fig.add_subplot(8,8,i+1, xticks=[], yticks=[])
    ax.imshow(X_train.reshape(-1,8,8)[i], cmap=plt.cm.gray_r)
    ax.text(0, 1, str(y_train[i]), bbox=dict(facecolor='white'))
```

6. Normalisasi data dengan Z score agar distribusi data menjadi distribusi normal (Gaussian Distribution)

```
scaler = StandardScaler()
scaler.fit(X_train)

X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

7. Membuat model data dengan jaringan syaraf tiruan dengan arsitektur multilayer perceptron menggunakan kelas MLPClassifier dari library scikit learn. MLPClassifier melakukan training data dengan menggunakan 1 hidden layer dengan 5 neuron, fungsi aktivasi relu, iterasi maksimum 100, learning rate=0.1.

```
# Inisialisai ANN Classifier
mlp = MLPClassifier(hidden_layer_sizes=(5), activation = 'relu', max_iter=100,
                    learning_rate='constant', learning_rate_init=0.1)

# Train the classifier with the training data
mlp.fit(X_train_scaled, y_train)

print(mlp)
```

8. Memprediksi kelas dari data uji

```
# Prediksi label data uji
y_prediksi= mlp.predict(X_test_scaled)
```

9. Menampilkan hasil prediksi data uji dalam bentuk gambar. Jika label berwarna hijau maka prediksi benar, jika berwarna merah maka prediksi salah.

```
# Plot Data Uji

fig = plt.figure(figsize=(6,6))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(48):
    ax = fig.add_subplot(8, 8, i+1, xticks=[], yticks=[])
    ax.imshow(X_test.reshape(-1,8,8)[i], cmap=plt.cm.gray_r)

    # Label data uji
    if y_prediksi[i] == y_test[i]:
        ax.text(0, 1, y_prediksi[i], color='green', bbox=dict(facecolor='white'))
    else :
        ax.text(0, 1, y_prediksi[i], color='red', bbox=dict(facecolor='white'))
```

10. Membuat diagram Confusion Matrix dan menghitung akurasi

```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn import metrics

cm = confusion_matrix(y_test, y_prediksi)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cm)
cm_display.plot()

akurasi = accuracy_score(y_test, y_prediksi)*100
print("Akurasi : {} %".format(akurasi))
```

Latihan Praktikum

1. Modifikasi kode pada percobaan 7 dengan beberapa fungsi aktivasi dan learning rate yang berbeda. Lakukan training dan testing data dan lengkapi nilai akurasi prediksi untuk masing-masing fungsi aktivasi dan learning rate. Tuliskan apa yang anda dapat simpulkan dari mengamati nilai akurasi tersebut.

Fungsi Aktivasi	Learning Rate	Akurasi Prediksi
relu	0.6	
	0.1	
logistic	0.6	
	0.1	

tanh	0.6	
	0.1	

- Modifikasi kode pada percobaan 7 dengan tujuh pilihan arsitektur MLP pada tabel berikut. Bandingkan akurasi prediksi yang dihasilkan untuk masing-masing pilihan arsitektur. Berikan kesimpulan anda berdasarkan hasil pengamatan akurasi tersebut.

No	Arsitektur	Akurasi Prediksi
1	1 hidden layer, 1 neuron	
2	1 hidden layer, 5 neuron	
3	1 hidden layer, 10 neuron	
4	1 hidden layer, 100 neuron	
5	Hidden layer 1, 1 neuron Hidden layer 2, 1 neuron	
6	Hidden layer 1, 5 neuron Hidden layer 2, 5 neuron	
7	Hidden layer 1, 5 neuron Hidden layer 2, 4 neuron Hidden layer 3, 3 neuron	

LAPORAN :

- Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
- Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
- Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

MODUL X

SUPPORT VECTOR MACHINE

A. Pokok Bahasan

1. Support Vector Machine
2. Akurasi model dengan confusion matrix

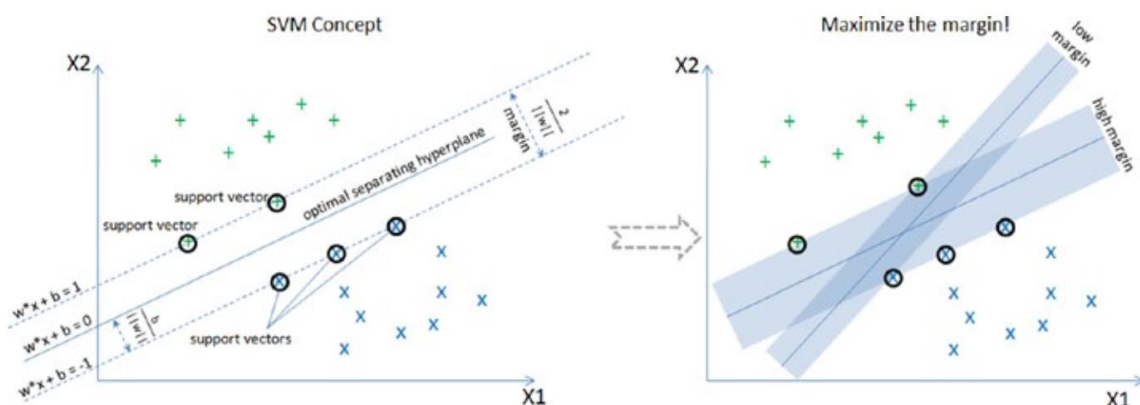
B. Tujuan Pembelajaran

1. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model support vector machine.
2. Mahasiswa mampu mengimplementasikan support vector machine untuk klasifikasi data menggunakan Bahasa Pemrograman Python.
3. Mahasiswa mampu menguji akurasi prediksi pada support vector machine.

C. Dasar Teori

Support Vector Machine (SVM)

Algoritma SVM diusulkan oleh Vladimir N. Vapnik dan Alexey Ya. Chervonenkis pada tahun 1963. Tujuan SVM adalah untuk menentukan hyperplane yang membagi dua kelas secara optimal.



Hyperplane yang optimal ketika margin antara hyperplane dan data observasi berjarak maksimum. Gambar 1 menunjukkan kemungkinan hyperplane pada SVM.

Gambar 1. Ilustrasi Hyperlane SVM

[Sumber : Swamynathan M., 2017]

Python menyediakan library scikit learn yang dapat digunakan untuk membuat model klasifikasi dengan SVM. Cara import library untuk membangun model dengan support vector classifier sebagai berikut :

```
from sklearn.svm import SVC
```

Pada kelas SVC terdapat parameter yang dapat ditentukan pada saat membuat objek SVC, beberapa parameter yang digunakan pada praktikum ini adalah :

Parameter	Deskripsi
c	Parameter regulasi. Nilai default = 1.0.
kernel	Mendefinisikan kernel yang digunakan SVC. Kernel : {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default = 'rbf'.
degree	Derajat polynomial dari fungsi kernel ('poly'). Nilai default=3.
gamma	Nilai {'scale', 'auto'}. Nilai default='scale'.
tol	Toleransi untuk kriteria berhenti.
max_iter	Maksimum iterasi.

Beberapa atribut dari kelas SVC antara lain :

Atribut	Deskripsi
classes_	Label kelas
intercept_	Konstanta atau bias pada fungsi hyperplane
n_iter_	Jumlah iterasi proses training
support_	indeks support vector
support_vectors_	Support vector
n_support_	Jumlah support vector tiap kelas

Berikut beberapa fungsi dalam kelas SVC :

Fungsi	Deskripsi
fit(X, y)	Training model SVC menggunakan data latih dengan atribut bebas atau prediktor X dan target y.
predict(X)	Melakukan prediksi kelas untuk data X menggunakan model SVC.

Untuk informasi yang lebih komprehensif mengenai implementasi SVC dengan scikit learn dapat diakses pada halaman <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>.

Dataset

Pada praktikum ini dataset yang digunakan adalah dataset bunga Iris. Dataset ini digunakan untuk melakukan klasifikasi jenis bunga Iris. Jumlah data dalam dataset sebanyak 150 data, dan jumlah atribut adalah 5 dimana 4 atribut dapat dijadikan fitur/prediktor dan 1 atribut sebagai target/kelas. Atribut dataset ini sebagai berikut :

- **sepal length (cm)** : Panjang daun bunga.
- **sepal width (cm)** : Lebar daun bunga.
- **petal length (cm)** : Panjang kelopak bunga.
- **petal width (cm)** : Lebar kelopak bunga.
- **target** : Kelas data yang berisi 3 jenis bunga yaitu setosa, versicolor, dan virginica dengan label 0 untuk jenis setosa, 1 untuk jenis versicolor, dan 2 untuk virginica.

Dataset ini tersedia di library scikit learn sehingga untuk dapat diimport dengan menuliskan kode `from sklearn.datasets import load_iris` di Python.

D. Tahapan Praktikum

1. Import Library.

```
from sklearn import datasets
import numpy as np
import pandas as pd
```

2. Load data set dan menampilkan label kelas dan atribut dataset. Dataset hanya menggunakan atribut petal length dan petal width sebagai atribut prediktor.

```
iris = datasets.load_iris()

X = iris.data[:, [2,3]]
y = iris.target

print("Label Kelas: ", np.unique(y), iris.target_names)
print("Atribut prediktor : ", iris.feature_names)
```

3. Normalisasi data dengan Z score agar distribusi data menjadi distribusi normal (Gaussian Distribution).

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
sc.fit(X)
X = sc.transform(X)

print(X[0:5,:])
```

4. Membagi data set menjadi 70% data latih dan 30% data uji. Untuk membagi dataset menjadi data latih dan data uji, kita dapat menggunakan **train_test_split** dari library sklearn.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

```
print("X Training:", X_train[0:5,:])
print("y Training:", y_train[0:5])
```

5. Membuat model support vector classifier dengan kernel linear dan nilai C=1. Selanjutnya, mentraining data.

```
from sklearn.svm import SVC

clf_svc = SVC(kernel='linear', C=1, random_state=0)
clf_svc.fit(X_train, y_train)
```

6. Mencetak beberapa atribut model SVC diantaranya bobot, bias, support vector, dan alpha.

```
print("Koefisien model : ", clf_svc.coef_)
print("Bias model : ", clf_svc.intercept_)
print("Indeks Support Vector : ", clf_svc.support_)
print("Support vector : ", clf_svc.support_vectors_)
print("Jumlah support vector: ", clf_svc.n_support_)
print("Nilai alpha : ", np.abs(clf_svc.dual_coef_))
```

7. Prediksi kelas data uji dan mencetak hasil prediksi.

```
y_prediksi = clf_svc.predict(X_test)
print(y_prediksi)
```

8. Membuat confusion matrix dan menghitung akurasi model SVC

```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn import metrics

cm = confusion_matrix(y_test, y_prediksi)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cm)
cm_display.plot()

akurasi = accuracy_score(y_test, y_prediksi)*100
print("Akurasi : {} %".format(akurasi))
```

Latihan Praktikum

1. Modifikasi kode pada percobaan 5 dengan fungsi **kernel linear** dan nilai C yang berbeda sesuai tabel berikut. Lakukan training dan testing data dan lengkapi nilai akurasi prediksi untuk masing-masing kondisi nilai C. Tuliskan apa yang anda dapat simpulkan dari mengamati nilai akurasi tersebut. (Screenshot kode program dan keluaran akurasi prediksi)

Nilai C	Akurasi Prediksi
---------	------------------

0.01	
0.1	
0.5	
1	
5	
10	

2. Modifikasi kode pada percobaan 5 dengan **fungsi kernel** yang berbeda. Lakukan training dan testing data dan lengkapi nilai akurasi prediksi untuk masing-masing fungsi kernel, derajat, dan gamma. Tuliskan apa yang anda dapat simpulkan dari mengamati nilai akurasi tersebut. (Screenshot kode program dan keluaran akurasi prediksi)

Fungsi Kernel	C	Derajat (d)	Akurasi Prediksi
Polynomial	1	1	
	1	2	
	1	5	
	1	10	
	1	15	

Fungsi Kernel	C	Gamma	Akurasi Prediksi
RBF	1	0.1	
	1	1	
	1	10	

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

PRAKTIKUM XI

CLUSTERING

A. Pokok Bahasan

1. K-Means Clustering
2. Metode Elbow
3. Metode Silhoutte
4. Hierarchical Clustering

B. Tujuan Pembelajaran

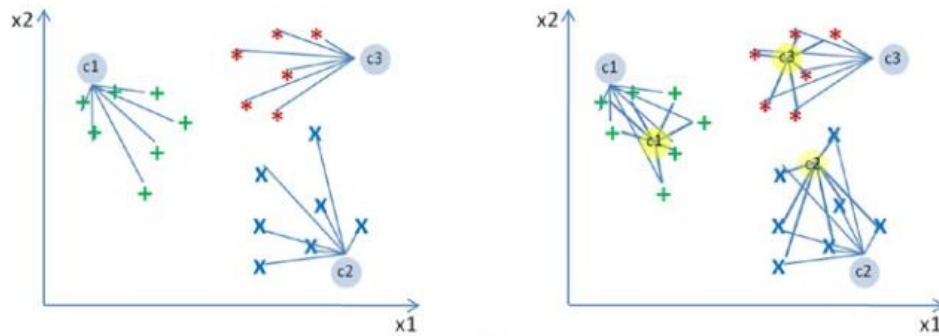
1. Mahasiswa mampu menggunakan Paket Scikit learn untuk membuat model K-means clustering dan Hierarchical clustering.
2. Mahasiswa mampu mengimplementasikan algoritma K-means dan Hierarchical clustering menggunakan bahasa Python untuk mengelompokkan data.
3. Mahasiswa mampu menerapkan metode elbow dan Silhoutte untuk menentukan jumlah cluster optimal.

C. Dasar Teori

Clustering termasuk ke dalam teknik pembelajaran unsupervised. Pembelajaran unsupervised mengidentifikasi kelompok data (cluster) dari data dalam sebuah dataset berdasarkan kemiripan antar data. Algoritma clustering yang populer antara lain adalah K-means dan agglomerative hierarchical clustering.

I. K-means

K-means merupakan algoritma clustering yang cukup sederhana. Data dikelompokkan dalam cluster sedemikian rupa sehingga data dalam cluster yang sama memiliki kemiripan yang tinggi dan tingkat kemiripan antara cluster rendah. Setiap data hanya dapat berada dalam satu cluster. Ilustrasi proses pengelompokkan data ke dalam 3 cluster menggunakan K-means ditunjukkan pada Gambar 1.



Gambar 1. Ilustrasi K-means
[Sumber : Swamynathan M., 2017]

Tahap algoritma K-means adalah

1. Tentukan jumlah cluster (k) dan tentukan pusat cluster (centroid) awal secara random.
2. Hitung jarak setiap data dengan masing-masing pusat cluster.
3. Kelompokkan setiap data ke dalam cluster yang pusatnya memiliki jarak terpendek dengan data tersebut.
4. Hitung pusat cluster baru dengan mencari nilai rata-rata dari data yang menjadi anggota pada cluster tersebut.
5. Ulangi langkah 2 – 4 hingga tidak ada lagi data yang berpindah ke cluster lain, atau perpindahan pusat cluster lebih kecil dari nilai toleransi.

Perhitungan jarak menggunakan persamaan **euclidean distance** berikut :

$$d(a, b) = \sqrt{\sum_{i=1}^N (a_i - b_i)^2}$$

Python menyediakan library scikit learn yang dapat digunakan untuk membuat model clustering K-means. Cara import library untuk membangun model dengan K-means sebagai berikut :

```
from sklearn.cluster import KMeans
```

Pada kelas KMeans terdapat parameter yang dapat ditentukan pada saat membuat objek SVC, beberapa parameter yang digunakan pada praktikum ini adalah :

Parameter	Deskripsi
n_clusters	Mendefinisikan jumlah cluster. Nilai default = 1.0.
init	Mendefinisikan cara memilih centroid awal. Init : {'k-means++', 'random', default = 'k-means++'.
max_iter	Iterasi maksimum. Nilai default=300.
tol	Nilai toleransi perubahan centroid untuk kondisi berhenti. Nilai default='1e-4'.

Beberapa atribut dari kelas KMeans antara lain :

Atribut	Deskripsi
cluster_centers_	Titik centroid cluster
labels_	Cluster tiap data
n_iter_	Jumlah iterasi proses training

Berikut beberapa fungsi dalam kelas KMeans :

Fungsi	Deskripsi
fit(X)	Proses klasterisasi dataset X.
predict(X)	Prediksi cluster terdekat dari data X.

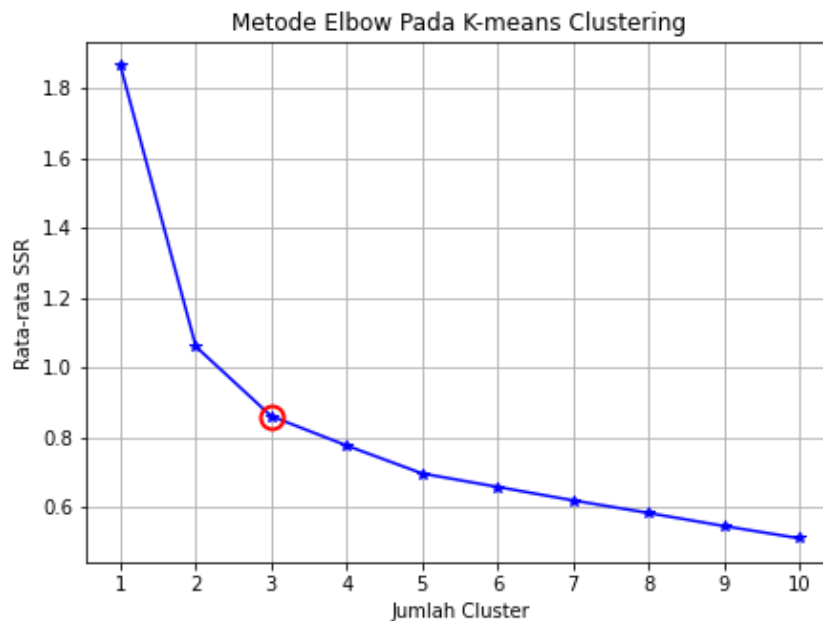
Untuk informasi yang lebih komprehensif mengenai implementasi K-means dengan scikit learn dapat diakses pada halaman <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>.

II. Metode Elbow

Salah satu kekurangan dari algoritma K-means adalah jumlah cluster harus ditentukan di awal. Jumlah cluster yang ditentukan tersebut tidak menjamin adalah jumlah cluster ideal. Terdapat dua cara yang dapat digunakan untuk menemukan jumlah k ideal, yaitu :

- Metode Elbow
- Metode Average Silhouette

Metode elbow melakukan clustering menggunakan k-means untuk cakupan nilai k, misalnya 1 sampai 10 dan kemudian menghitung sum of squared error (SSE) atau persentase variansi untuk setiap k. Selanjutnya, plot nilai cluster terhadap nilai SSE pada diagram garis dan perhatikan bentuk siku pertama dari garis. Ini menunjukkan jumlah cluster ideal. SSE bernilai 0 jika jumlah k sama dengan jumlah titik data karena tidak ada error antara data dengan pusat clusternya. Oleh karena itu, tujuan metode elbow adalah memilih nilai k yang kecil yang memiliki nilai SSE yang rendah. Nilai k tersebut membentuk siku pada diagram garis. Untuk persentase variansi, semakin bertambah k maka nilai persentase variansi semakin meningkat. Gambar 2 menunjukkan diagram garis dengan elbow terbentuk pada k bernilai 3.



Gambar 2. Metode Elbow dengan K bernilai 3.

III. Metode Average Silhouette

Metode average silhouette pertama kali diperkenalkan pada tahun 1986 oleh Peter J. Rousseeuw untuk menggambarkan konsistensi data dalam cluster. Nilai silhouette berada pada rentang -1 sampai 1, dimana semakin besar nilainya maka data-data yang berada di cluster yang sama memiliki kedekatan yang tinggi (jaraknya dekat) dan data yang berada pada cluster yang berbeda memiliki kedekatan yang rendah (jaraknya jauh).

Nilai silhouette untuk setiap data dihitung dengan persamaan berikut :

$$s(i) = \frac{b(i) - a(i)}{\max \{a(i), b(i)\}}$$

Dimana :

$a(i)$: rata-rata jarak antara data i dengan data lainnya pada cluster yang sama

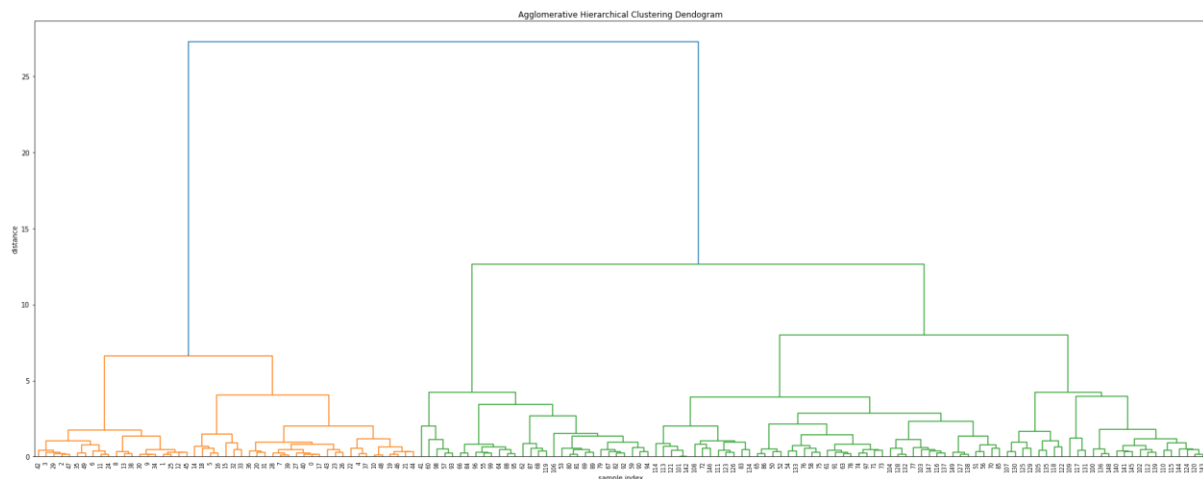
$b(i)$: rata-rata jarak terdekat antara data i dengan cluster lainnya.

Sehingga average silhouette untuk sebuah hasil klusterisasi adalah nilai rata-rata $s(i)$ dari seluruh titik data pada dataset.

IV. Hierarchical Clustering

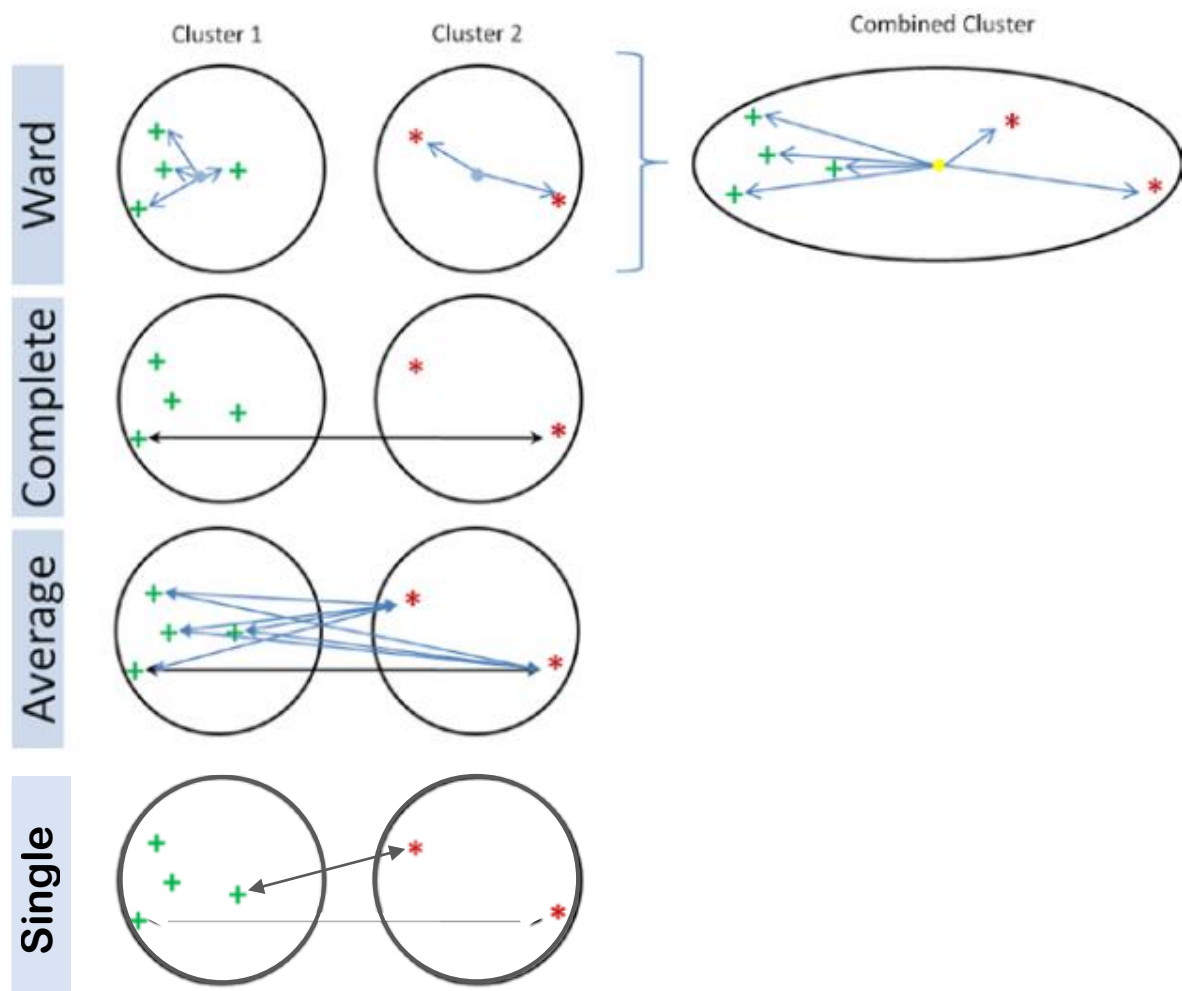
Hierarchical clustering adalah teknik klusterisasi yang tidak memerlukan pendefinisian jumlah kluster di awal karena proses klusterisasi dilakukan dengan membangun hirarki cluster. Terdapat dua strategi dalam hierarchical clustering yaitu pertama, Agglomerative dimana proses cluster dimulai dengan setiap data adalah sebuah cluster kemudian secara iteratif menggabungkan dua buah cluster yang memiliki kesamaan tinggi sampai membentuk sebuah hirarki cluster. Kedua, Divisive adalah kebalikan dari agglomeratif dimana proses cluster dimulai dengan seluruh data pada dataset berada di dalam satu cluster kemudian secara iteratif membagi dua cluster sampai dengan setiap data berada di cluster yang berbeda. Strategi yang umum digunakan dalam hierarchical clustering adalah

agglomerative hierarchical clustering. Hasil klasterisasi dari hierarchical clustering adalah hirarki cluster yang digambarkan dalam bentuk dendrogram seperti ditunjukkan pada Gambar 3.



Gambar 3 Dendrogram Agglomerative Hierarchical Clustering

Penggabungan dua buah cluster didasarkan pada ukuran keterhubungan (linkage). Dua buah cluster dengan ukuran keterhubungan yang lebih kecil akan digabungkan terlebih dahulu. Terdapat beberapa ukuran keterhubungan yang dapat digunakan diantaranya single linkage, complete linkage, average linkage, dan ward. Gambar 4 menampilkan ilustrasi masing-masing jenis linkage.



Gambar 4. Linkage Agglomerative Hierarchical Clustering

[Sumber : Swamynathan M., 2017]

V. Dataset

Pada praktikum ini dataset yang digunakan adalah dataset bunga Iris. Dataset ini digunakan untuk melakukan klasifikasi jenis bunga Iris. Jumlah data dalam dataset sebanyak 150 data, dan jumlah atribut adalah 5 dimana 4 atribut dapat dijadikan fitur/prediktor dan 1 atribut sebagai target/kelas. Atribut dataset ini sebagai berikut :

- **sepal length (cm)** : Panjang daun bunga.
- **sepal width (cm)** : Lebar daun bunga.
- **petal length (cm)** : Panjang kelopak bunga.
- **petal width (cm)** : Lebar kelopak bunga.
- **target** : Kelas data yang berisi 3 jenis bunga yaitu setosa, versicolor, dan virginica dengan label 0 untuk jenis setosa, 1 untuk jenis versicolor, dan 2 untuk virginica.

Dataset ini tersedia di library scikit learn sehingga untuk dapat diimport dengan menuliskan kode `from sklearn.datasets import load_iris` di Python.

D. Tahapan Praktikum

I. Kmeans

1. Import Library.

```
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
import pandas as pd
```

2. Load dataset sebagai data frame dan menghilangkan spasi pada judul kolom.

```
iris = datasets.load_iris()

iris = pd.DataFrame(data=np.c_[iris.data, iris.target], columns=iris.feature_names+['species'])
iris['species'] = iris['species'].astype(int)
display(iris.head())

# menghilangkan spasi pada judul kolom
iris.columns = iris.columns.str.replace(' ', '')
iris.head()
```

3. Normalisasi data dengan Z score agar distribusi data menjadi distribusi normal (Gaussian Distribution).

```
from sklearn.preprocessing import StandardScaler

X = iris.iloc[:, :4]
y = iris.species
```

```

sc = StandardScaler()
sc.fit(X)
X = sc.transform(X)

print(X[0:5,:])

```

4. Membuat model klasterisasi dengan Kmeans dan melakukan proses clustering menggunakan k-means dengan jumlah cluster sebanyak tiga.

```

from sklearn.cluster import KMeans

model_kmeans = KMeans(n_clusters=3, random_state=0)
model_kmeans.fit(X)

print(model_kmeans.labels_)

```

5. Menghitung akurasi hasil klasterisasi dengan membandingkan cluster sebenarnya dan cluster hasil klasterisasi.

```

from sklearn.metrics import accuracy_score, classification_report

iris['pred_species'] = np.choose(model_kmeans.labels_, [1, 0, 2]).astype(np.int64)

akurasi = accuracy_score(iris.species, iris.pred_species)*100
print("Akurasi : {}".format(akurasi))

```

6. Menampilkan perbandingan visualisasi data hasil clustering dan label data sebenarnya.

```

fig, ax1 = plt.subplots(2, 2, figsize=(18,15))
colors = {0:'red', 1:'blue', 2:'green'}
ax1[0][0].scatter(iris['sepalength(cm)'], iris['sepalwidth(cm)'], c=iris['species'].map(colors),
s=80)
ax1[0][1].scatter(iris['sepalength(cm)'], iris['sepalwidth(cm)'],
c=iris['pred_species'].map(colors), s=80)
ax1[1][0].scatter(iris['petallength(cm)'], iris['petalwidth(cm)'], c=iris['species'].map(colors),
s=80)
ax1[1][1].scatter(iris['petallength(cm)'], iris['petalwidth(cm)'],
c=iris['pred_species'].map(colors), s=80)

ax1[0][0].set(xlabel="Sepal Length", ylabel='Sepal Width', title='Sepal (Actual)')
ax1[0][1].set(xlabel="Sepal Length", ylabel='Sepal Width', title='Sepal (Predicted)')
ax1[1][0].set(xlabel="Petal Length", ylabel='Petal Width', title='Petal (Actual)')
ax1[1][1].set(xlabel="Petal Length", ylabel='Petal Width', title='Petal (Predicted)')
plt.show()

```

II. Metode Elbow

Menentukan jumlah cluster ideal menggunakan metode elbow untuk nilai k 1 - 10.

```
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans

K = range(1,11)
kmeans = [KMeans(n_clusters=k).fit(X) for k in K]
centroids = [k.cluster_centers_ for k in kmeans]

D_k = [cdist(X, c, 'euclidean') for c in centroids]
cldx = [np.argmin(D, axis=1) for D in D_k]
dist = [np.min(D, axis=1) for D in D_k]
avgWSS = [sum(d)/X.shape[0] for d in dist]

# membuat diagram garis metode elbow
kldx=2
plt.figure(figsize=(7,5))
plt.plot(K, avgWSS, 'b*-')
plt.plot(K[kldx], avgWSS[kldx], marker='o', markersize=12,
         markedgewidth=2, markedgewidthcolor='r', markerfacecolor='None')
plt.grid(True)
plt.xticks(K)
plt.xlabel('Jumlah Cluster')
plt.ylabel('Rata-rata SSR')
plt.title('Metode Elbow Pada K-means Clustering')
plt.show()
```

III. Metode Average Silhouette

1. Menghitung skor average silhouette untuk hasil klasterisasi dengan nilai k 2 – 9.

```
from sklearn.metrics import silhouette_score
from matplotlib import cm

score = []
for n_cluster in range(2,10):
    model_kmeans = KMeans(n_clusters=n_cluster)
    model_kmeans.fit(X)

    labels = model_kmeans.labels_
    centroids = model_kmeans.cluster_centers_
    score.append(silhouette_score(X, labels, metric='euclidean'))

print(score)
```

2. Visualisasi skor average silhouette menggunakan diagram garis.

```
plt.figure(figsize=(7,5))
n_cluster = range(2,10)
plt.plot(n_cluster, score, 'b*-')
plt.grid(True)
plt.xticks(K)
plt.xlabel('Jumlah Cluster')
```

```
plt.xlabel('siloutte score')
plt.title('Siloutte for K-means')
plt.show()
```

IV. Hierarchical Clustering

1. Membuat model agglomerative clustering dengan jumlah cluster tiga.

```
from sklearn.cluster import AgglomerativeClustering

model_agglo = AgglomerativeClustering(n_clusters=3)
model_agglo.fit(X)

print(model_agglo.labels_)
```

2. Menghitung akurasi hasil klasterisasi dengan agglomerative clustering

```
from sklearn.metrics import accuracy_score

iris['pred_species_agglo'] = np.choose(model_agglo.labels_, [2, 0, 1]).astype(np.int64)

akurasi = accuracy_score(iris.species, iris.pred_species_agglo)*100
print("Akurasi : {}".format(akurasi))
```

3. Menggambar dendrogram hasil klasterisasi.

```
from scipy.cluster.hierarchy import cophenet, dendrogram, linkage
from scipy.spatial.distance import pdist

# membuat matriks linkage
Z = linkage(X, 'ward')
c, coph_dists = cophenet(Z, pdist(X))

# menampilkan dendrogram
plt.figure(figsize=(25,10))
plt.title('Agglomerative Hierarchical Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(
    Z,
    leaf_rotation=90.,
    leaf_font_size=8.,
)
plt.tight_layout()
```

LAPORAN :

1. Tuliskan laporan hasil praktikum yang anda lakukan. Screenshot setiap tahapan praktikum yang anda lakukan (kode dan hasil).
2. Tuliskan apa yang anda dapat simpulkan dari praktikum yang anda lakukan.
3. Laporan dikumpulkan sebelum praktikum selanjutnya. File laporan berformat Pdf.

DAFTAR PUSTAKA

- Admin. (2022). *What do you mean by Neural Network?*
<https://training.javatpoint.com/recurrent-neural-network>
- Developers, S. (n.d.-a). *Recognizing hand-written digits*. https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html
- Developers, S. (n.d.-b). *sklearn.cluster.KMeans*. <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>
- Inc, A. (2022). *Anaconda Navigator* (2.0.3). Anaconda Inc.
<https://docs.anaconda.com/anaconda/install/>
- NumFOCUS, I. (2022). *API reference*. <https://pandas.pydata.org/docs/reference/index.html>
- NumPy Developers. (2022). *NumPy Reference*. <https://numpy.org/doc/stable/reference/>
- Raushan, R. (n.d.). *Social Network Ads*. <https://www.kaggle.com/datasets/rakeshrau/social-network-ads>
- Suyanto. (2018). *Machine Learning : Tingkat Dasar dan Lanjut*. Informatika.
- Swamynathan, M. (2017). *Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python* (1st ed.). Apress.
- Tayo, B. O. (2018). *Machine Learning: Python Linear Regression Estimator Using Gradient Descent*. <https://pub.towardsai.net/machine-learning-python-linear-regression-estimator-using-gradient-descent-b0b2c496e463>

