

Modul 1 - Introduction to Machine Learning with Python

Kemampuan Akhir yang Direncanakan

- Mahasiswa mampu menguasai konsep dasar pembelajaran mesin dan merancang solusi permasalahan untuk studi kasus tertentu.

Pengenalan Python

- Python menjadi bahasa pemrograman yang luas digunakan, khususnya aplikasi *data science* dan *machine learning*.
- Python menggabungkan kelebihan bahasa pemrograman yang *general-purposed* dengan kemudahan penggunaan *library* domain pengolahan data seperti MATLAB atau R
- *Library* yang dimiliki Python telah mencakup pengolahan data, visualisasi, statistik, pemrosesan teks, pemrosesan gambar, dan lain sebagainya.
- Kita dapat berinteraksi langsung dengan code baik menggunakan terminal atau IDE seperti Jupyter Notebook, Google Colab, dan VS Code dengan ekstensi Jupyter Notebook.

Jupyter Notebook dan Anaconda

- Dalam mata kuliah ini kita perlu menggunakan Anaconda sebagai *environment* untuk pengembangan model *machine learning*. Unduh dengan tautan: <https://www.anaconda.com/download>
- Isikan email untuk mendapatkan link unduh Anaconda, silakan melakukan instalasi mengikuti instruksi yang berjalan. Pastikan untuk mencentang opsi menambahkan Anaconda pada PATH di OS dan opsi agar kernel Anaconda dapat diakses aplikasi lain.
- Kemudian untuk membuka Jupyter Notebook, buka Anaconda Prompt dan ketik command `jupyter notebook`
- Tunggu hingga proses loading selesai dan Jupyter Notebook akan tampil pada *browser default* yang terinstal pada perangkat.

VS Code dan Ekstensi Jupyter Notebook

- Sebagai alternatif kita dapat menggunakan Visual Studio Code sebagai IDE dengan menambahkan ekstensi Jupyter Notebook.

- Instalasi VS Code kemudian pilih menu View -> Extensions, cari extension Jupyter dan lakukan instalasi.
- Untuk membuat file baru, pilih menu File -> New File -> Jupyter Notebook.

Sintaks-sintaks Dasar Python

Print dan Comment

- Print digunakan untuk menampilkan pesan, maka pada notebook yang sudah dibuat sebelumnya, ketikkan sesuai yang tertampil pada contoh.
- Comment pada Python menggunakan tanda pagar (#), baris yang diberikan comment tidak akan tereksekusi.
- Kemudian tekan tombol run cell di sebelah kiri cell atau ctrl+enter untuk menjalankan sintaks tersebut sehingga hasilnya muncul seperti berikut.

```
In [ ]: #ini adalah comment, tidak akan tereksekusi
print("Pembelajaran Mesin dan Pembelajaran Mendalam adalah Machine Learning and Deep Learning")
```

Pembelajaran Mesin dan Pembelajaran Mendalam adalah Machine Learning and Deep Learning

- Salah satu bentuk format print pada Python dapat menggunakan format seperti berikut:

```
In [ ]: name="Nama Anda"
age=28
print('My name is: {one}, and my age is: {two}'.format(one=name,two=age))
print('My name is: {}, and my age is: {}'.format(age,name))
```

My name is: Nama Anda, and my age is: 28
My name is: 28, and my age is: Nama Anda

Tipe-tipe Data Dalam Python

1. Bilangan

- Tipe data Numbers pada Python dapat kita assign ke dalam variabel seperti contoh berikut:

```
In [ ]: varA = 13
varB = 21.5
varC = varA*varB-varA
varD = varC/(varB+varA)
print(varA*varB)
print(varC)
print(varD)
print(varA*varA)
```

```
279.5
266.5
7.72463768115942
169
```

2. String

- Tipe data String pada Python dapat kita assign dengan tanda petik tunggal ('') atau ganda ("") seperti contoh berikut:

```
In [ ]: varStr1='Contoh string petik tunggal'
varStr2="Contoh string petik ganda"
varStr3="Contoh penggunaan keduanya: i'm possible"
print(varStr1,varStr2,varStr3)
```

```
Contoh string petik tunggal Contoh string petik ganda Contoh penggunaan keduanya: i'm possible
```

- Selain itu, variabel string dapat diperlakukan sebagai array of character seperti contoh berikut:

```
In [ ]: varStr="I love Machine Learning"
print(varStr)
print(varStr[7:])
print(varStr+" and Data Science")
print(varStr*2)
```

```
I love Machine Learning
Machine Learning
I love Machine Learning and Data Science
I love Machine LearningI love Machine Learning
```

3. Sequence

a. List

- List pada Python dapat dibuat menggunakan simbol [] seperti array di bahasa sebelah, seperti contoh berikut:

```
In [ ]: list=['I','love',"machine","learning",2023]
print(list)

['I', 'love', 'machine', 'learning', 2023]
```

- List pada Python dapat dimanipulasi dan diakses elemennya seperti contoh berikut:

```
In [ ]: list=['I','love',"machine","learning",2023]
list.append("until the end")
print(list)
print(list[1])
print(list[2:4])
print(list[:4])
list[1]="learn"
print(list)
len(list)

['I', 'love', 'machine', 'learning', 2023, 'until the end']
love
['machine', 'learning']
['I', 'love', 'machine', 'learning']
['I', 'learn', 'machine', 'learning', 2023, 'until the end']

Out[ ]: 6
```

b. Dictionary

- Dictionary pada Python adalah tipe data yang menyerupai tabel dan terdiri atas key dan value.
- Dictionary dibuat dengan simbol kurung kurawal { } dan ditutup dengan kurung kurawal lagi }. Contohnya sebagai berikut:

```
In [ ]: myDict={}
myDict['one']=1
myDict[2]="My age is 27 years old"
myDict["3"]=30

print(myDict['one'])
print(myDict[2])
print(myDict.keys())
print(myDict.values())
```

```
1
My age is 27 years old
dict_keys(['one', 2, '3'])
dict_values([1, 'My age is 27 years old', 30])
```

c. Tuple

- Tuple adalah baris data yang dapat menyimpan berbagai tipe variabel.
- Perbedaan Tuple dengan list adalah ukuran dan isi tuple tidak bisa berubah.
- Tuple dibuat dengan simbol kurung () seperti contoh berikut:

```
In [ ]: tp =('time','to','repair','your gadget',1,2,3)

print(tp[0])

tp[2]="wash"
```

```
time
```

```
-----
```

```
Cell In[10], line 5
  1 tp =('time','to','repair','your gadget',1,2,3)
  2
  3 print(tp[0])
----> 4 tp[2]="wash"
```

```
Traceback (most recent call last)
```

```
TypeError: 'tuple' object does not support item assignment
```

Pemilihan/Selection

- Pemilihan atau selection pada Python kurang lebih sama dengan bahasa pemrograman yang lain, hanya saja yang membedakan adalah tidak ada penggunaan kurung kurawal.
- Statemen yang berjalan pada pemilihan dibedakan dengan identasi 1 tab.
- Untuk sintaks yang digunakan adalah if, else, dan elif (else if) yang diikuti dengan titik dua untuk membuka sintaks. Contohnya sebagai berikut:

```
In [ ]: varName="AGP"

if len(varName)<10:
    print("You're lucky")
elif len(varName)<20 and len(varName)>=10:
    print("You're very lucky")
else :
    print("You're super lucky")
```

You're lucky

Perulangan/Loop

- Perulangan pada Python menggunakan for dan while.
- Untuk loop for menggunakan sintaks: `for var in statement:`

```
In [ ]: list=["one", "two", "three"]
parameter=[0,1,2,3,4,5]

temp=0
for number in parameter:
    print(number)
    temp=temp+number

print("Nilai temp:",temp)

for isi in list:
    print(isi)

for item in list:
    print(item+item)
```

```
0  
1  
2  
3  
4  
5  
Nilai temp: 15  
one  
two  
three  
oneone  
twotwo  
threethree
```

- Sedangkan while menggunakan sintaks: `while(statement)`

```
In [ ]: i = 1  
        while i <=10:  
            print("i is: {}".format(i))  
            i=i+1
```

```
i is: 1  
i is: 2  
i is: 3  
i is: 4  
i is: 5  
i is: 6  
i is: 7  
i is: 8  
i is: 9  
i is: 10
```

- Kita dapat menentukan range menggunakan sintaks range atau membuat numpy array dengan sintaks np.arange seperti berikut:

```
In [ ]: import numpy as np  
  
for i in range(0,21,4):  
    print(i)
```

```
n_test=np.arange(1,11,2)

print(n_test)

for item in n_test:
    print(item)
```

```
0
4
8
12
16
20
[1 3 5 7 9]
1
3
5
7
9
```

Function atau Module

- Untuk pembuatan fungsi dan modul/prosedur, keduanya menggunakan sintaks: `def name_func_or_module(params):`
- Perbedaannya adalah function menggunakan sintaks `return` untuk mengembalikan nilai, sedangkan module/prosedur tidak. Contoh function:

```
In [ ]: def luasSegitiga(alas,tinggi):
    luas=0.5*(alas*tinggi)
    return luas

a=3
t=4

print(luasSegitiga(a,t))
```

6.0

- Contoh module:

```
In [ ]: def PrintNamaXKali(nama,jumlah):
    print(nama*jumlah)

name = "Nama anda"
jumlah=10

PrintNamaXKali(name,jumlah)
```

Nama andaNama andaNama andaNama andaNama andaNama andaNama andaNama andaNama anda

Konsep Dasar Pembelajaran Mesin

Latihan

Deskripsi Dataset

- Kita akan menggunakan dataset bawaan scikit-learn yaitu Iris Dataset.
- Iris Dataset diperkenalkan oleh ahli statistik Inggris Ronald Fisher pada tahun 1936 sebagai contoh dari linear discriminant analysis.
- Dataset ini berisi informasi empat karakteristik bunga Iris, yaitu panjang sepal, lebar sepal, panjang petal, dan lebar petal(dalam satuan centimeter).
- Dataset ini terdiri dari tiga spesies Iris: Iris setosa, Iris virginica, dan Iris versicolor. Masing-masing spesies diwakili dengan 50 sampel.

1. Eksplorasi Singkat Dataset

- Pertama-tama data dimuat ke dalam suatu variabel. Kita memasukan dataset yang ada ke dalam DataFrame dengan cara sebagai berikut:

```
In [ ]: #import library yang dibutuhkan
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris

#memuat dataset dalam dataframe
iris=load_iris()
df_iris=pd.DataFrame(data=iris.data,columns=iris.feature_names,index=None)
```

```
df_iris.head(20)
```

Out[]: **sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)**

0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3.0	1.4	0.1
13	4.3	3.0	1.1	0.1
14	5.8	4.0	1.2	0.2
15	5.7	4.4	1.5	0.4
16	5.4	3.9	1.3	0.4
17	5.1	3.5	1.4	0.3
18	5.7	3.8	1.7	0.3
19	5.1	3.8	1.5	0.3

- Data yang hilang akan membuat algoritma menjadi error, untuk itu kita perlu menanganinya. Kita dapat melakukan pengecekan pada data apabila ada data yang bernilai null, kosong, atau NaN dengan cara seperti berikut:

```
In [ ]: #cek data null, kosong, nan
print("data null \n",df_iris.isnull().sum())
print("data kosong \n",df_iris.empty)
print("data nan \n",df_iris.isna().sum())
```

```
data null
  sepal length (cm)      0
  sepal width (cm)       0
  petal length (cm)      0
  petal width (cm)       0
  dtype: int64
data kosong
  False
data nan
  sepal length (cm)      0
  sepal width (cm)       0
  petal length (cm)      0
  petal width (cm)       0
  dtype: int64
```

- Dataset dapat kita deskripsikan secara statistik menggunakan fungsi describe() dan menghasilkan luaran seperti berikut:

```
In [ ]: df_iris.describe()
```

Out[]:

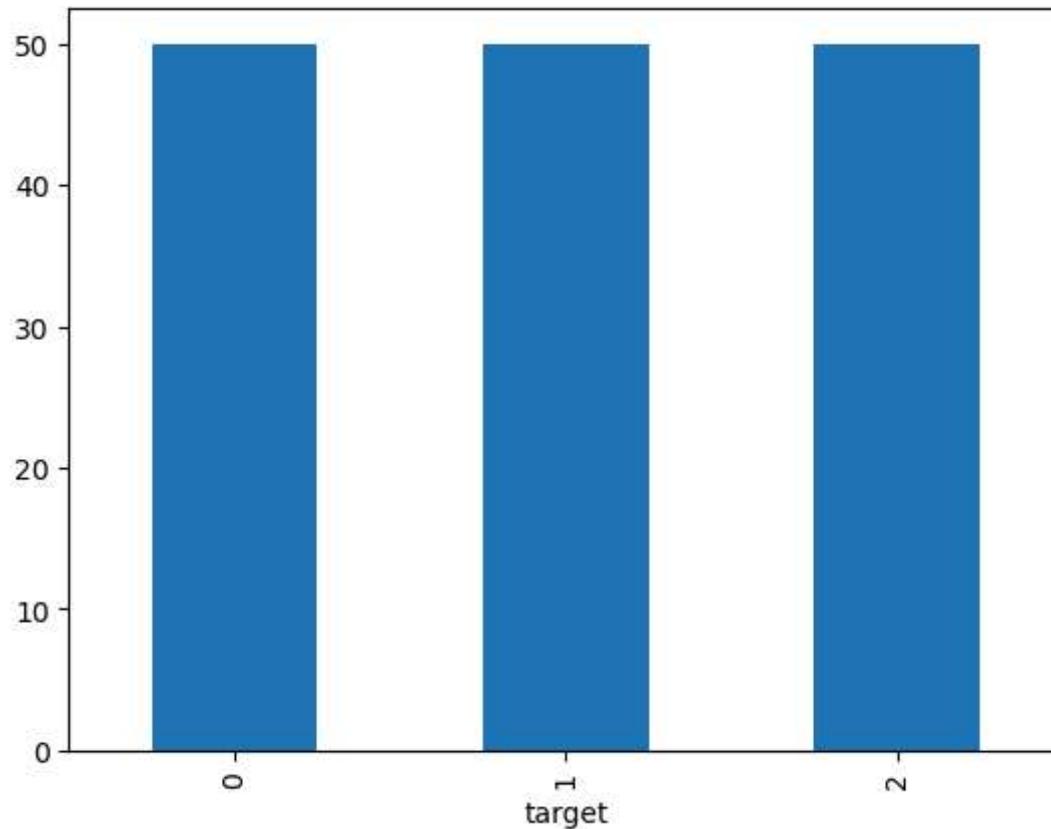
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

- Langkah berikutnya adalah exploratory data analysis atau analisis eksploratif data. Eksplorasi data secara statistik dapat kita lakukan dengan membuat representasi dalam bentuk grafik seperti berikut:

In []:

```
df_iris['target']=iris.target.astype(int)
freq=df_iris.target.value_counts()
freq.plot(kind='bar')
```

Out[]: <Axes: xlabel='target'>



2. Train-Test Split

- Sebelum kita membuat model dari dataset kita perlu membagi dataset menjadi set data training dan data testing.
- Data training merupakan bagian dari dataset yang digunakan untuk melatih algoritme pembelajaran mesin. Algoritme akan mempelajari ciri tiap class data dari masing-masing feature.
- Sedangkan test set adalah bagian dari dataset yang digunakan untuk pengujian. Test set dianggap sebagai set data yang belum pernah dilihat atau dipelajari oleh model machine learning kita.
- Secara umum rasio antara training set dan test set adalah 70%:30% dan 75%:25%.

```
In [ ]: #import library untuk split dataset
from sklearn.model_selection import train_test_split
```

```
x= df_iris.drop('target',axis=1)
y = df_iris.target

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)

print("bentuk X_train",X_train.shape)
print("bentuk X_test",X_test.shape)

print("bentuk y_train",y_train.shape)
print("bentuk y_test",y_test.shape)

print("y_train \n",y_train)
print("y_test \n",y_train)
```

```
bentuk X_train (112, 4)
bentuk X_test (38, 4)
bentuk y_train (112,)
bentuk y_test (38,)
y_train
 61    1
92    1
112   2
2     0
141   2
...
9     0
103   2
67    1
117   2
47    0
Name: target, Length: 112, dtype: int32
y_test
 61    1
92    1
112   2
2     0
141   2
...
9     0
103   2
67    1
117   2
47    0
Name: target, Length: 112, dtype: int32
```

3. Melatih Model, Melakukan Prediksi, dan Evaluasi Model

- Model dari algoritme dilatih menggunakan training set dan ada pengaturan parameter.
- Parameter yang diatur sesuai dengan algoritme yang digunakan.
- Setelah dilatih, kita melakukan prediksi terhadap test set dan menguji performa dari model yang sudah terbentuk.
- Berikut adalah contoh code dari pemanggilan model dan pelatihannya:

```
In [ ]: #import Library untuk model machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

#buat obyek model machine Learning
KNN = KNeighborsClassifier(n_neighbors=2, p=2)
DT = DecisionTreeClassifier(random_state=0, max_depth=3)

#Latih model machine Learning
KNN.fit(X_train,y_train)
DT.fit(X_train,y_train)
```

```
Out[ ]: ▾ DecisionTreeClassifier ⓘ ?
```

```
DecisionTreeClassifier(max_depth=3, random_state=0)
```

- Untuk melakukan prediksi kita dapat mencoba dengan menginputkan sebuah data baru, belum menggunakan data test set. Karena dataset iris terdiri dari empat feature, kita dapat membuat data baru menggunakan array yang terdiri dari 4 elemen. Misalkan kita menginputkan nilainya dengan ketentuan sebagai berikut:

NPM : AA BB CCDDE (Ex: 20 07 12345)

- Sepal length : A.A * 2
- Sepal width : B.B * 3
- Petal length : C.C
- Petal width : D.D
- Sepal length : $2.0 * 2 = 4.0$
- Sepal width : $0.7 * 3 = 2.1$
- Petal length : $1.2 * 2 = 2.4$

- Petal width : $3.4 / 2 = 1.7$

```
In [ ]: #buat array untuk X baru yang akan diprediksi
X_new=np.array([[4.0,2.1,2.4,1.7]])

print("X_new yang akan diprediksi",X_new.shape)

#prediksi label dari X baru
knn_predict=KNN.predict(X_new)
print("Label prediksi KNN",knn_predict)

dt_predict=DT.predict(X_new)
print("Label prediksi DT",dt_predict)
```

X_new yang akan diprediksi (1, 4)

Label prediksi KNN [1]

Label prediksi DT [2]

```
c:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
c:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

- Hasil prediksi tunggal seperti contoh sebelumnya belum bisa menentukan akurasi atau performa ketepatan model dalam memprediksi label suatu data.
- Evaluasi dapat kita lakukan menggunakan test set yang sudah kita simpan tadi.
- Langkah pengkodeannya adalah sebagai berikut:

```
In [ ]: y_pred_knn=KNN.predict(X_test)

y_pred_dt=DT.predict(X_test)

print("Hasil prediksi KNN pada X_test:",y_pred_knn)
print("Hasil prediksi DT pada X_test:",y_pred_dt)
```

```
Hasil prediksi KNN pada X_test: [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0  
2]  
Hasil prediksi DT pada X_test: [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0  
2]
```

```
In [ ]: print("Akurasi model KNN dengan fungsi score", round(KNN.score(X_test,y_test),3))  
print("Akurasi model DT perbandingan prediksi vs label", round(DT.score(X_test,y_test),3))
```

```
Akurasi model KNN dengan fungsi score 0.974  
Akurasi model DT perbandingan prediksi vs label 0.974
```

- Setelah melakukan pelatihan dan pengujian model kita dapat menyimpan model menggunakan library Pickle.

```
In [ ]: import pickle  
  
with open('knn_dt_iris_model.pkl','wb') as f:  
    pickle.dump((KNN,DT), f)  
  
print("Model KNN dan DT berhasil disimpan")
```

```
Model KNN dan DT berhasil disimpan
```