

# Tugas

# Kecerdasan Buatan



Heriyanto

1184023

Applied Bachelor of Informatics Engineering

Program Studi D4 Teknik Informatika

Applied Bachelor Program of Informatics Engineering

*Politeknik Pos Indonesia*

Bandung 2021

## **Abstract**

Buku Pedoman ini dibuat dengan tujuan memberikan acuan, bagi mahasiswa Tingkat Akhir dan dosen Pembimbing. Pada intinya buku ini menjelaskan secara lengkap tentang Standar pengerjaan Intership dan Tugas Akhir di Program Studi D4 Teknik Informatika, dan juga mengatur mekanisme, teknik penulisan, serta penilaiannya. Dengan demikian diharapkan semua pihak yang terlibat dalam aktivitas Bimbingan Mahasiswa Tingkat Akhir berjalan lancar dan sesuai dengan standar.

# Contents

<b>1</b>	<b>Mengenal Kecerdasan Buatan dan Scikit-Learn</b>	<b>1</b>
1.1	Teori . . . . .	1
1.1.1	Definisi dan Sejarah Kecerdasan Buatan . . . . .	1
1.1.1.1	Definisi Kecerdasan Buatan . . . . .	1
1.1.1.2	Definisi Kecerdasan Buatan Menurut Para Ahli . . . . .	1
1.1.2	Sejarah Kecerdasan Buatan . . . . .	2
1.1.3	Supervised Learning dan Unsupervised Learning . . . . .	2
1.1.3.1	Supervised Learning . . . . .	2
1.1.3.2	Unsupervised Learning . . . . .	3
1.1.4	Data Set . . . . .	3
1.2	Instalasi . . . . .	3
1.2.1	Instalasi Scikit-Learn pada Anaconda . . . . .	3
1.2.2	Loading an example dataset . . . . .	5
1.2.3	Learning and Predicting . . . . .	8
1.2.4	Conventions . . . . .	10
1.2.4.1	Type Casting . . . . .	10
1.2.4.2	Refitting and updating parameters . . . . .	14
1.2.4.3	Multiclass vs. multilabel fitting . . . . .	15
1.3	Penanganan Error . . . . .	19
1.3.1	Error 1 . . . . .	19
<b>2</b>	<b>Membangun Model Prediksi</b>	<b>21</b>
2.1	Teori . . . . .	21
2.1.1	Binary Classification . . . . .	21
2.1.2	Supervised Learning, Unsupervised Learning, dan Clustering . . . . .	22
2.1.3	Evaluasi dan Akurasi . . . . .	23
2.1.4	Confusion Matrix . . . . .	24
2.1.5	K-Fold Cross Validation . . . . .	26

2.1.6	Decision Tree . . . . .	27
2.1.7	Entropi dan Information Gain . . . . .	27
2.2	Scikit Learn . . . . .	32
2.2.1	Nomor 1 . . . . .	32
2.2.2	Nomor 2 . . . . .	33
2.2.3	Nomor 3 . . . . .	33
2.2.4	Nomor 4 . . . . .	34
2.2.5	Nomor 5 . . . . .	35
2.2.6	Nomor 6 . . . . .	35
2.2.7	Nomor 7 . . . . .	36
2.2.8	Nomor 8 . . . . .	37
2.2.9	Nomor 9 . . . . .	37
2.2.10	Nomor 10 . . . . .	38
2.2.11	Nomor 11 . . . . .	39
2.2.12	Nomor 12 . . . . .	39

<b>Bibliography</b>	<b>41</b>
---------------------	-----------

# List of Figures

1.1	Anaconda Prompt . . . . .	4
1.2	Anaconda Prompt . . . . .	4
1.3	Loading an example dataset . . . . .	5
1.4	Hasil Dataset . . . . .	6
1.5	Digits Data . . . . .	6
1.6	Hasil Digits Data . . . . .	7
1.7	Digits Target . . . . .	7
1.8	Hasil Digits Target . . . . .	7
1.9	Hasil Digits Target . . . . .	8
1.10	Hasil Digits Image . . . . .	8
1.11	Learning and Predicting . . . . .	9
1.12	Hasil Learning and Predicting . . . . .	10
1.13	Hasil Learning and Predicting . . . . .	10
1.14	Hasil X Numpy . . . . .	11
1.15	Hasil X new Numpy . . . . .	12
1.16	Iris Predict . . . . .	12
1.17	Hasil Iris Predict . . . . .	13
1.18	Refitting and updating parameters . . . . .	14
1.19	Hasil Refitting and updating parameters . . . . .	15
1.20	Multiclass Predict . . . . .	16
1.21	Hasil Multiclass Predict . . . . .	16
1.22	Multiclass Predict 2 . . . . .	17
1.23	Hasil Multiclass Predict 2 . . . . .	18
1.24	MultiLabel Fitting . . . . .	18
1.25	Hasil MultiLabel Fitting . . . . .	19
1.26	Hasil MultiLabel Fitting . . . . .	19
1.27	Hasil MultiLabel Fitting . . . . .	20
1.28	Learning and Predicting . . . . .	20

2.1	Contoh Spam Filtering, Garis Miring Biru adalah class boundary . .	21
2.2	Supervised Learning: Dataset Iris . . . . .	22
2.3	Unsupervised Learning . . . . .	23
2.4	Model Evaluasi . . . . .	23
2.5	Akurasi . . . . .	24
2.6	Confusion Matrix . . . . .	25
2.7	5-Fold Cross Validation . . . . .	26
2.8	5-Fold Cross Validation . . . . .	27
2.9	Entropy . . . . .	28
2.10	Entropy Atribut Tunggal . . . . .	28
2.11	Entropy Dua Atribut . . . . .	29
2.12	Langkah 1 . . . . .	29
2.13	Langkah 2 . . . . .	30
2.14	Langkah 3 . . . . .	31
2.15	Langkah 4a . . . . .	32
2.16	Langkah 4b . . . . .	32
2.17	Nomor 1 . . . . .	33
2.18	Nomor 2 . . . . .	33
2.19	Nomor 3 . . . . .	34
2.20	Nomor 4 . . . . .	35
2.21	Nomor 5 . . . . .	35
2.22	Nomor 6 . . . . .	36
2.23	Nomor 6 . . . . .	36
2.24	Nomor 7 . . . . .	36
2.25	Nomor 7 . . . . .	37
2.26	Nomor 8 . . . . .	37
2.27	Nomor 9 . . . . .	38
2.28	Nomor 10 . . . . .	38
2.29	Nomor 11 . . . . .	39
2.30	Nomor 12 . . . . .	40

# Chapter 1

## Mengenal Kecerdasan Buatan dan Scikit-Learn

### 1.1 Teori

#### 1.1.1 Definisi dan Sejarah Kecerdasan Buatan

##### 1.1.1.1 Definisi Kecerdasan Buatan

Kecerdasan buatan adalah sistem kecerdasan yang ditanamkan atau ditambahkan ke dalam teknologi oleh manusia, yang kemudian akan dikembangkan dalam lingkungan ilmiah atau dalam pembentukan entitas ilmiah yang ada. Kecerdasan di sini menekankan pada kemampuan memperoleh pengetahuan baru, yang dapat langsung diterapkan. Meskipun AI memiliki konotasi ilmiah, AI juga merupakan cabang dari ilmu komputer, pembelajaran, perilaku (behaviour), dan adaptasi pada mesin. [15].

##### 1.1.1.2 Definisi Kecerdasan Buatan Menurut Para Ahli

1. Scahlkoff (1990)

Kecerdasan Buatan merupakan bidang studi yang berusaha menerangkan dan meniru perilaku kecerdasan dalam bentuk proses komputasi.[13]

2. Luger dan Stubblefield (1993)

Kecerdasan Buatan merupakan cabang ilmu komputer yang berhubungan dengan otomatisasi perilaku yang cerdas.[13]

3. Rich dan Knight (1991)

Kecerdasan Buatan merupakan studi tentang cara membuat komputer melakukan sesuatu yang sampai saat ini, orang dapat melakukannya lebih baik.[13]

#### 4. Haag dan Keen (1996)

Kecerdasan Buatan adalah bidang studi yang berhubungan dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia dalam sebuah sistem teknologi sehingga sistem tersebut dapat memfasilitasi proses pengambilan keputusan yang biasanya dilakukan oleh manusia.[13]

### 1.1.2 Sejarah Kecerdasan Buatan

Istilah kecerdasan buatan pertama kali diusulkan pada tahun 1956. Hingga saat ini, dengan peningkatan daya komputasi dan kapasitas penyimpanan, penggunaan kecerdasan buatan menjadi semakin populer. Fase penelitian awal proyek AI berlangsung sekitar tahun 1950-an, bertujuan untuk mengeksplorasi tema pemecahan masalah dan metode simbolik.

Pada 1960-an, Departemen Pertahanan AS juga sangat ingin mengembangkan dan melatih komputer dengan kemampuan dasar penalaran manusia. Sekitar tahun 1970-an, proyek DARPA (Defense Advanced Research Projects Agency) berhasil menyelesaikan studi kasus pemetaan jalan. Di awal abad ke-21, tepatnya tahun 2003, DARPA juga berhasil melahirkan asisten pribadi yang cerdas.

Setelah itu, teknologi kecerdasan buatan terus berkembang, dan hingga saat ini memasuki prosedur yang lebih detail dengan menerapkan algoritma deep learning. Di sana, kecerdasan buatan yang dikembangkan dapat melakukan tugas dan memberikan solusi yang lebih kompleks dengan kondisi yang lebih beragam.[8]

### 1.1.3 Supervised Learning dan Unsupervised Learning

#### 1.1.3.1 Supervised Learning

Supervised learning (pembelajaran terarah) adalah metode pembelajaran mesin di mana pengguna mengharapkan hasil, dan informasinya diketahui atau dimiliki oleh sistem. Artinya metode pembelajaran bekerja dengan menggunakan kembali masukan atau data pengguna dan hasil keluaran yang sebelumnya diselesaikan oleh sistem. Supervised Learning itu sendiri dapat dibagi lagi menjadi regresi dan klasifikasi.[14]

Regresi adalah teknik analisis yang digunakan untuk mengidentifikasi hubungan antara dua variabel atau lebih. Tujuan dari regresi adalah untuk menemukan fungsi yang memodelkan data dengan meminimalkan kesalahan atau perbedaan antara nilai prediksi dan nilai sebenarnya.

Klasifikasi adalah teknik yang digunakan untuk mengklasifikasikan beberapa item yang tidak berlabel ke dalam satu set kelas diskrit. Klasifikasi mencoba mempelajari



hubungan antara sekumpulan variabel fitur dan variabel target. Dalam klasifikasi, variabel targetnya adalah tipe kategori.

### **1.1.3.2 Unsupervised Learning**

Unsupervised Learning adalah metode lain dalam materi pembelajaran mesin, di mana tidak ada yang bisa mengetahui hasil yang diharapkan. Tujuan utama dari metode pembelajaran ini adalah agar para penggunanya dapat mengelompokkan objek-objek yang dianggap serupa pada suatu ruang atau area tertentu.[7]

### **1.1.4 Data Set**

Kumpulan data adalah objek yang mirip dengan kamus, yang menyimpan semua data dan beberapa metadata tentang data.[1]

Saat membuat model Machine Learning, data harus dibagi menjadi satu Training Set dan satu Testing Set. Machine Learning harus diberi tahu "Data Set" mana yang akan dicapai, dan "Data Set" yang dapat digunakan untuk mencapai tujuan. "Data Set" untuk dicapai ini adalah Testing Set, dan "Data Set" untuk mencapai ini disebut Training Set.[6]

Training Set adalah bagian dari Data Set yang dilatih untuk membuat prediksi atau menjalankan fungsi algoritma ML. Dengan memberikan petunjuk melalui algoritma sehingga mesin yang dilatih dapat menemukan korelasinya sendiri atau mempelajari pola dari data yang diberikan. Testing Set adalah bagian dari data set yang diuji untuk melihat akurasi, atau dengan kata lain, untuk melihat performanya.[12]

## **1.2 Instalasi**

### **1.2.1 Instalasi Scikit-Learn pada Anaconda**

1. Pastikan Anaconda telah terpasang pada perangkat komputer atau laptop yang anda gunakan.
2. Jika Anaconda belum terpasang pada perangkat anda silahkan kunjungi link berikut untuk tutorial memasang anaconda: <https://docs.anaconda.com/anaconda/install/>.
3. Setelah anaconda terpasang buka command shell anaconda, dengan mengetik di start menu "Anaconda Prompt (ananconda 3)".

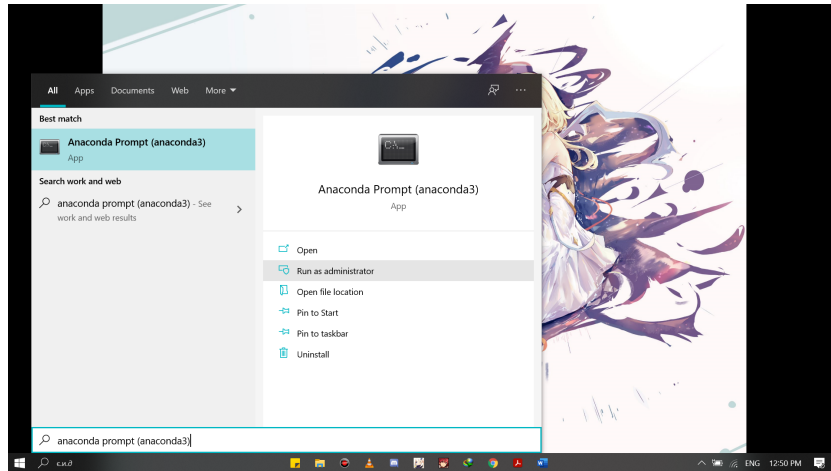


Figure 1.1: Anaconda Prompt

4. Kemudian ketik perintah berikut untuk memasang scikit-learn library pada anaconda: `conda install -c conda-forge scikit-learn`, dan tunggu sampai prosesnya selesai.

```
Administrator: Anaconda Prompt (anaconda3)

(base) C:\WINDOWS\system32>conda install -c conda-forge scikit-learn
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: D:\Aplikasi\anaconda3

added / updated specs:
- scikit-learn

The following packages will be downloaded:

package | build | size | channel
-----|-----|-----|-----
conda-4.9.2 | py38haa244fe_0 | 3.1 MB | conda-forge
python_abi-3.8 | 1_cp38 | 4 KB | conda-forge
Total: 3.1 MB

The following NEW packages will be INSTALLED:

python_abi conda-forge/win-64::python_abi-3.8-1_cp38

The following packages will be SUPERSEDED by a higher-priority channel:

conda pkgs/main::conda-4.9.2-py38haa95532_0 --> conda-forge::conda-4.9.2-py38haa244fe_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
python_abi-3.8 | 4 KB | ##### | 100%
conda-4.9.2 | 3.1 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\WINDOWS\system32>
```

Figure 1.2: Anaconda Prompt

### 1.2.2 Loading an example dataset

Scikit-learn hadir dengan beberapa dataset standar, misalnya dataset iris dan digit untuk klasifikasi dan dataset diabetes untuk regresi. Berikut ini, dimulai dengan interpreter Python menggunakan tool Spyder kemudian memuat set data iris dan digit.[1]

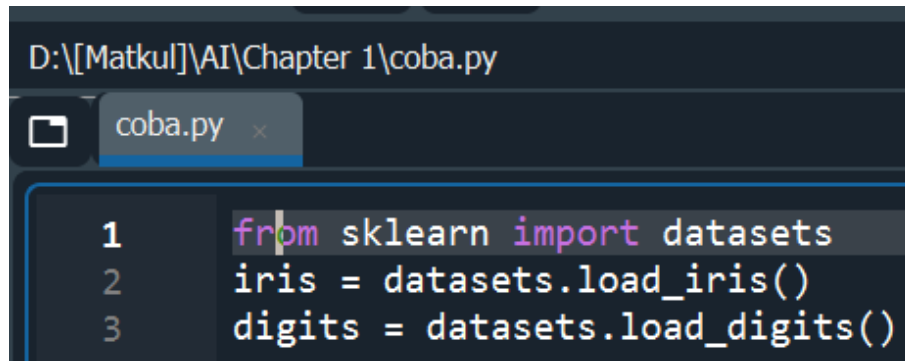
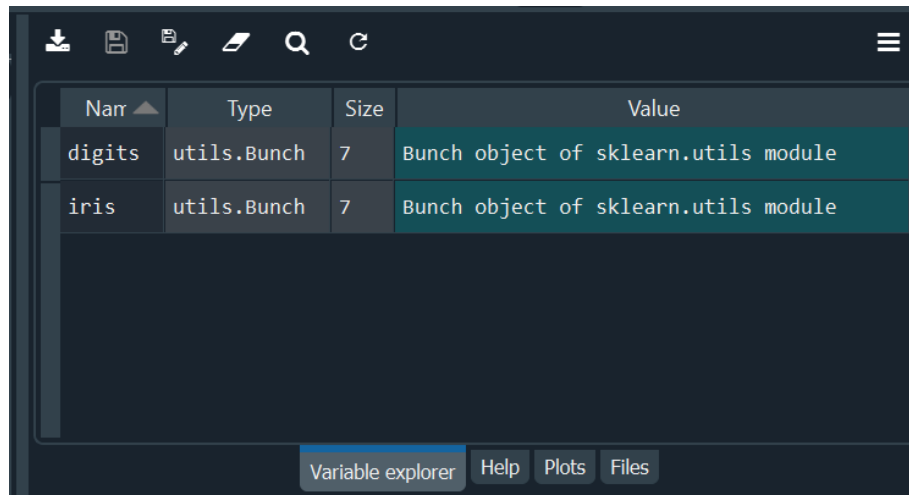
A screenshot of a code editor window. The title bar shows the file path 'D:\[Matkul]\AI\Chapter 1\coba.py'. Below the title bar, there is a tab labeled 'coba.py'. The code editor area contains three lines of Python code: line 1: 'from sklearn import datasets', line 2: 'iris = datasets.load\_iris()', and line 3: 'digits = datasets.load\_digits()'. The code is syntax-highlighted with colors: 'from' is purple, 'sklearn' is blue, 'import' is pink, 'datasets' is blue, 'iris' is blue, 'load\_iris()' is blue, 'digits' is blue, and 'load\_digits()' is blue. Line numbers 1, 2, and 3 are visible on the left side of the code editor.

Figure 1.3: Loading an example dataset

Keterangan

1. Baris pertama: meng-import module datasets dari library sklearn.
2. Baris kedua: iris adalah sebuah variabel dengan nilai yang memuat dataset iris dataset iris.
3. Baris ketiga: digits adalah sebuah variabel dengan nilai yang memuat dataset digits.

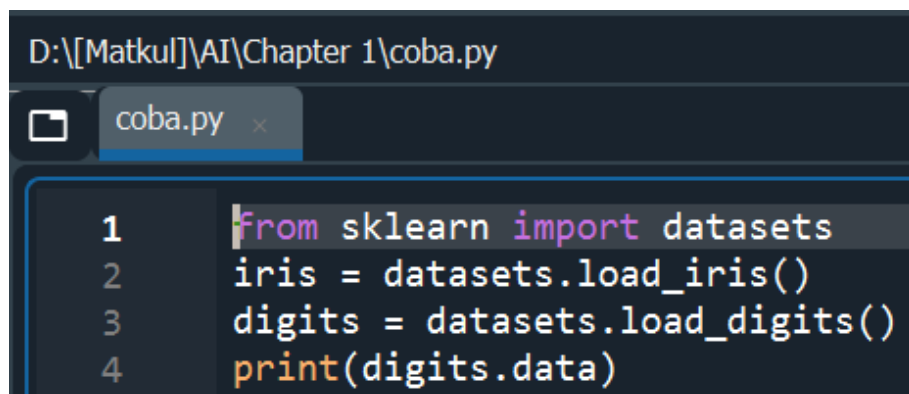
Jika kode tersebut dijalankan akan menghasilkan dataset iris dan digits yang telah dimuat didalam module datasets. Berikut hasil kode yang berjalan tanpa error. Dapat dilihat pada kolom value bahwa dataset-nya telah dimuat.



Name	Type	Size	Value
digits	utils.Bunch	7	Bunch object of sklearn.utils module
iris	utils.Bunch	7	Bunch object of sklearn.utils module

Figure 1.4: Hasil Dataset

Misalnya, untuk kumpulan data digit, Anda dapat menggunakan `digits.data` untuk mengakses fungsi yang dapat digunakan untuk mengklasifikasikan sampel digit:



```
D:\[Matkul]\AI\Chapter 1\coba.py
coba.py x
1 from sklearn import datasets
2 iris = datasets.load_iris()
3 digits = datasets.load_digits()
4 print(digits.data)
```

Figure 1.5: Digits Data

Disini kita menambahkan satu baris, yaitu `print(digits.data)` yang akan menampilkan sampel data digit. Berikut Hasilnya:

```
Console 1/A x
In [3]: runfile('D://[Matkul]/AI/Chapter 1/coba.py',
wdir='D://[Matkul]/AI/Chapter 1')
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]

In [4]: |
```

Figure 1.6: Hasil Digits Data

Selain `digits.data`, `digits.target` memberikan informasi dasar dari kumpulan data digit, yaitu nomor yang sesuai dengan setiap gambar digit yang ingin kita pelajari, berikut kodenya:

```
D:\[Matkul]\AI\Chapter 1\coba.py
coba.py x
1 from sklearn import datasets
2 iris = datasets.load_iris()
3 digits = datasets.load_digits()
4 print(digits.target)
```

Figure 1.7: Digits Target

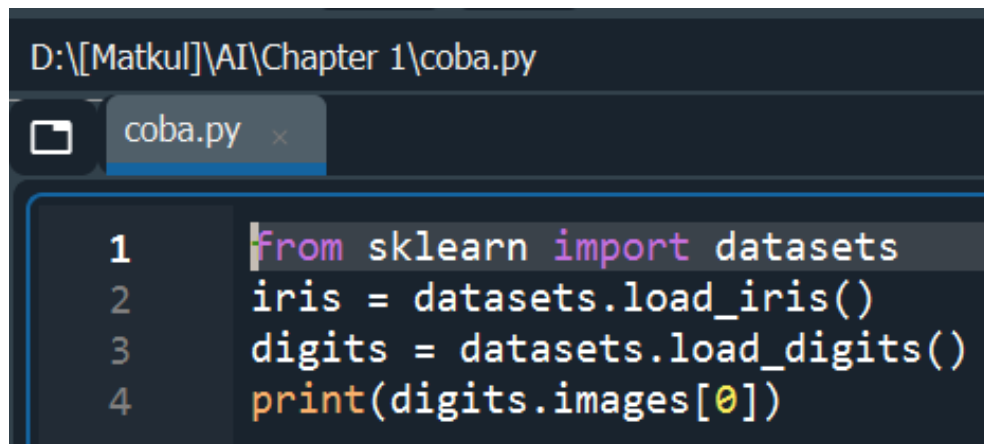
Seperti yang dijelaskan diatas sebelumnya `digits.target` akan memberikan informasi nomor setiap gambar digit yang akan dipelajari. Berikut hasilnya:

```
Console 1/A x
In [9]: runfile('D://[Matkul]/AI/Chapter 1/coba.py',
wdir='D://[Matkul]/AI/Chapter 1')
[0 1 2 ... 8 9 8]

In [10]: |
```

Figure 1.8: Hasil Digits Target

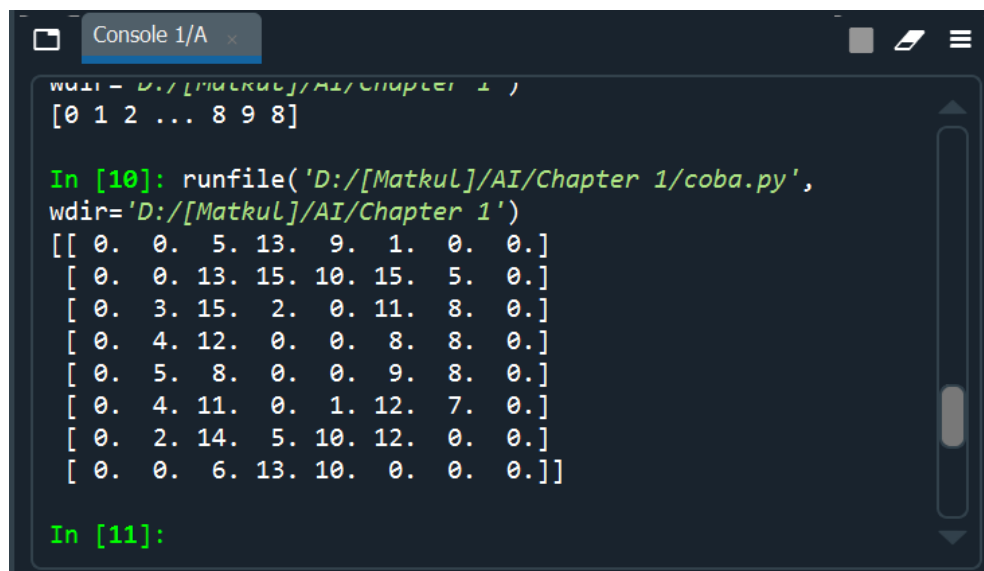
Meskipun data asli mungkin memiliki bentuk yang berbeda, datanya selalu berupa 2D (n-samples, n-features). Untuk digit, setiap sampel asli adalah gambar bentuk (8, 8), yang dapat diakses dengan cara berikut:



```
D:\[Matkul]\AI\Chapter 1\coba.py
coba.py x
1 from sklearn import datasets
2 iris = datasets.load_iris()
3 digits = datasets.load_digits()
4 print(digits.images[0])
```

Figure 1.9: Hasil Digits Target

`digits.images[0]` akan menampilkan digit gambar index ke nol, yang akan menampilkan angka-angka dalam bentuk array.



```
Console 1/A x
wdir = D:\[Matkul]\AI\Chapter 1
[0 1 2 ... 8 9 8]

In [10]: runfile('D:\[Matkul]\AI\Chapter 1\coba.py',
wdir='D:\[Matkul]\AI\Chapter 1')
[[ 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.]]

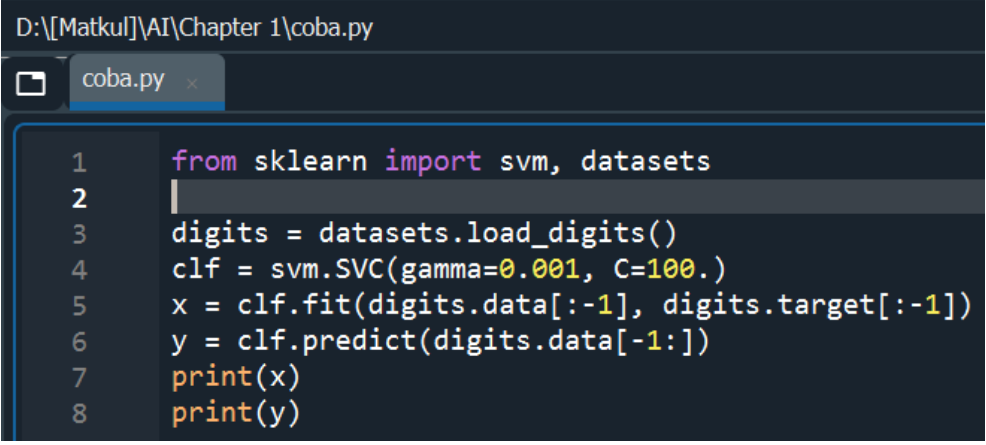
In [11]:
```

Figure 1.10: Hasil Digits Image

### 1.2.3 Learning and Predicting

Dalam scikit-learn, estimator yang digunakan untuk klasifikasi adalah objek Python yang mengimplementasikan metode `fit` (X, y) dan `predict` (T). Contoh prediktor

adalah kelas `sklearn.svm.SVC`, yang mengimplementasikan support vector classification. Estimasi konstruktor diambil sebagai fungsi dari model parameter.



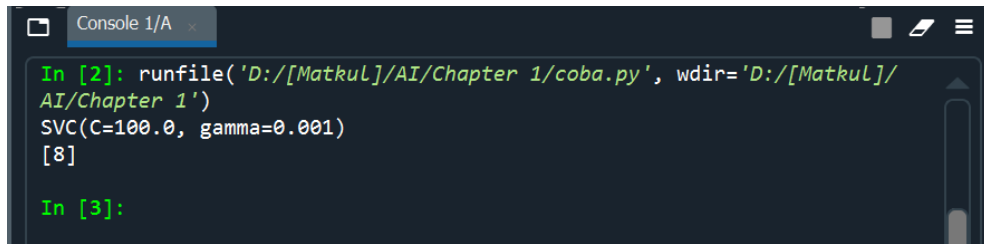
```
D:\[Matkul]\AI\Chapter 1\coba.py
coba.py x
1  from sklearn import svm, datasets
2
3  digits = datasets.load_digits()
4  clf = svm.SVC(gamma=0.001, C=100.)
5  x = clf.fit(digits.data[:-1], digits.target[:-1])
6  y = clf.predict(digits.data[-1:])
7  print(x)
8  print(y)
```

Figure 1.11: Learning and Predicting

#### Keterangan

1. Baris pertama mengambil module `svm` dan `datasets` dari library `sklearn`.
2. Baris ketiga adalah variable dengan nama `digits` yang memiliki nilai untuk memuat dataset digit.
3. Baris keempat variable `clf` yang memiliki value `gamma=0.001` dan `Classification = 100`.
4. Baris kelima variable `x` yang berisi `clf` sebagai classifier kemudian `fit` sebagai method yang mengambil data yang pas dari `digits.data` dan `digits.target` dengan nilai training set `[:-1]`.
5. Baris keenam variable `y` yang berisi `clf` sebagai classifier kemudian method `predict` akan memprediksi `digits.data` dengan nilai training set `[-1:]`.
6. Baris ketujuh dan kedelapan akan menampilkan hasil dari method `fit` dan `predict`.

Hasil dari method `fit` dan `predict`

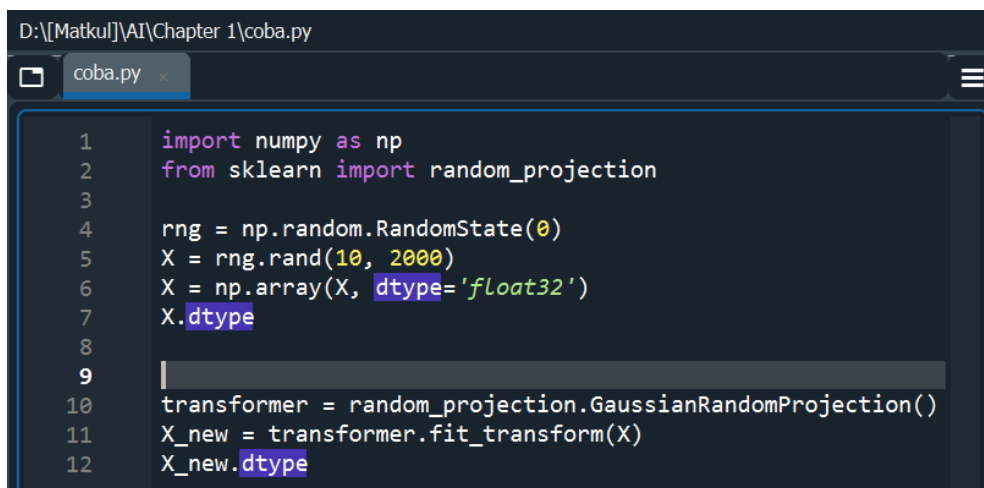


```
Console 1/A
In [2]: runfile('D:/[Matkul]/AI/Chapter 1/coba.py', wdir='D:/[Matkul]/
AI/Chapter 1')
SVC(C=100.0, gamma=0.001)
[8]
In [3]:
```

Figure 1.12: Hasil Learning and Predicting

## 1.2.4 Conventions

### 1.2.4.1 Type Casting



```
D:\[Matkul]\AI\Chapter 1\coba.py
coba.py
1 import numpy as np
2 from sklearn import random_projection
3
4 rng = np.random.RandomState(0)
5 X = rng.rand(10, 2000)
6 X = np.array(X, dtype='float32')
7 X.dtype
8
9
10 transformer = random_projection.GaussianRandomProjection()
11 X_new = transformer.fit_transform(X)
12 X_new.dtype
```

Figure 1.13: Hasil Learning and Predicting

#### Keterangan

1. Baris pertama memanggil library numpy dan dibuat dengan alias np.
2. Baris kedua memanggil module random projection dari library sklearn.
3. Baris keempat adalah variable rng yang mendefinisikan numpy as np, kemudian fungsi random dan attr RandomState.
4. Baris kelima adalah variable X yang mendefinisikan variable rng, kemudian method random yang akan menentukan nilai random dari 10 - 2000.
5. Baris keenam adalah variable X yang akan menampung nilai random sebelumnya kedalam array dengan tipe data float32.



6. Baris ketujuh, akan mengubah tipe data nilai random float32 ke float64.
7. Baris kesepuluh adalah sebuah variable transformer yang mendefinisikan class random projection dan memanggil fungsi GaussianRandomProjection.
8. Baris kesebelas adalah variable X new yang akan melakukan perhitungan pada variable X.
9. Baris kedua belas, merubah tipe data menjadi float64.

Hasil dari method X dan X new.



X - NumPy object array

	0	1	2	3	4	5
0	0.548814	0.715189	0.602763	0.544883	0.423655	0.645
1	0.811518	0.476084	0.523156	0.250521	0.605043	0.302
2	0.292642	0.566518	0.137414	0.349712	0.0532164	0.379
3	0.374794	0.428686	0.683057	0.600948	0.070483	0.157
4	0.45776	0.376918	0.702335	0.207324	0.0742795	0.366
5	0.748268	0.180203	0.389023	0.0376002	0.0117877	0.996
6	0.56469	0.839746	0.376884	0.499676	0.0813024	0.324
7	0.320595	0.249867	0.0310728	0.853923	0.0742608	0.136

Format Resize ☐ Background color Save and Close Close

Figure 1.14: Hasil X Numpy

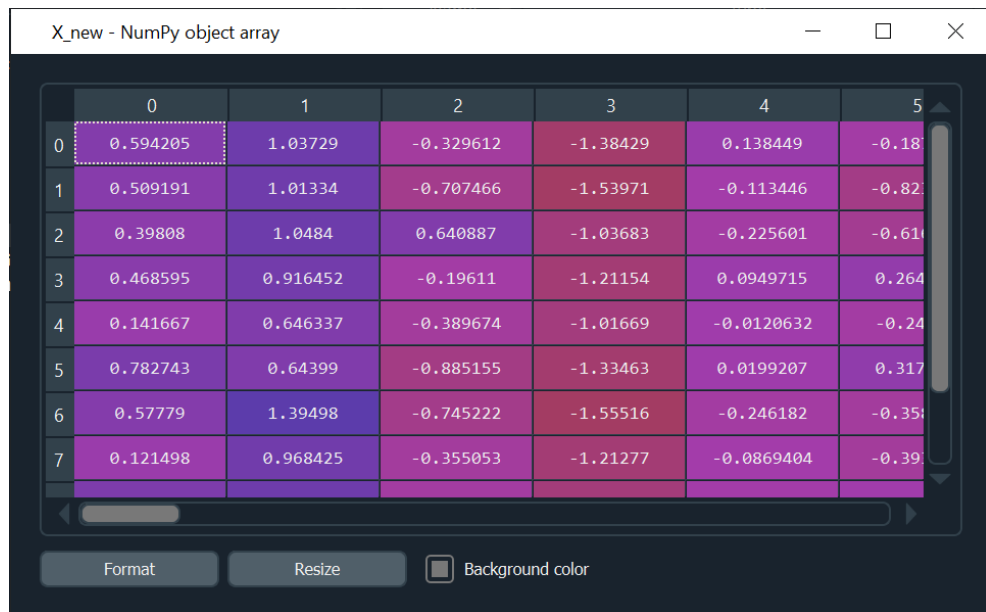


Figure 1.15: Hasil X new Numpy

Di sini, `predict ()` pertama kali mengembalikan array integer, karena `iris.target` (array integer) digunakan dengan tepat. `Prediction ()` kedua mengembalikan larik string karena `iris.target names` cocok untuk penginstalan.

```

D:\[Matkul]\AI\Chapter 1\coba.py
coba.py x
1  from sklearn import datasets
2  from sklearn.svm import SVC
3
4  iris = datasets.load_iris()
5  clf = SVC()
6  X = clf.fit(iris.data, iris.target)
7  print(X)
8
9  Y = list(clf.predict(iris.data[:3]))
10 print(Y)
11
12 Z = clf.fit(iris.data, iris.target_names[iris.target])
13 print(Z)
14
15 M = list(clf.predict(iris.data[:3]))
16 print(M)

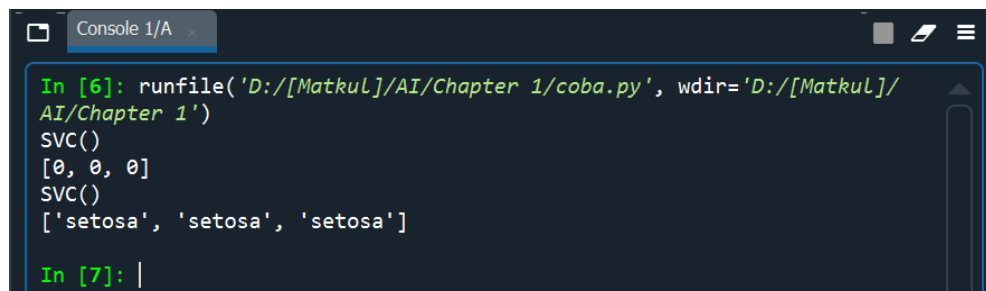
```

Figure 1.16: Iris Predict

Keterangan

1. Baris pertama, memanggil module datasets dari library sklearn.
2. Baris kedua, memanggil module SVC dari library sklearn.svm.
3. Baris keempat, variable iris yang memuat datasets iris.
4. Baris kelima, variable clf yang memanggil method SVC.
5. Baris keenam, variable X yang memanggil classifier kemudian method fit yang memanggil data pas dari iris.data dan target.
6. Baris ketujuh, menampilkan hasil variable X.
7. Baris kesembilan, variable Y yang akan mengembalikan iris predict dalam bentuk array.
8. Baris kesepuluh, menampilkan hasil variable Y.
9. Baris keduabelas, memanggil classifier kemudian method fit yang memanggil data pas dari iris.names, iris.data, dan target.
10. Baris ketigabelas, menampilkan hasil variable Z.
11. Baris kelimabelas, variable M yang akan mengembalikan iris predict kedalam bentuk string.
12. keenambelas, menampilkan hasil variable M.

Hasil dari iris predict.



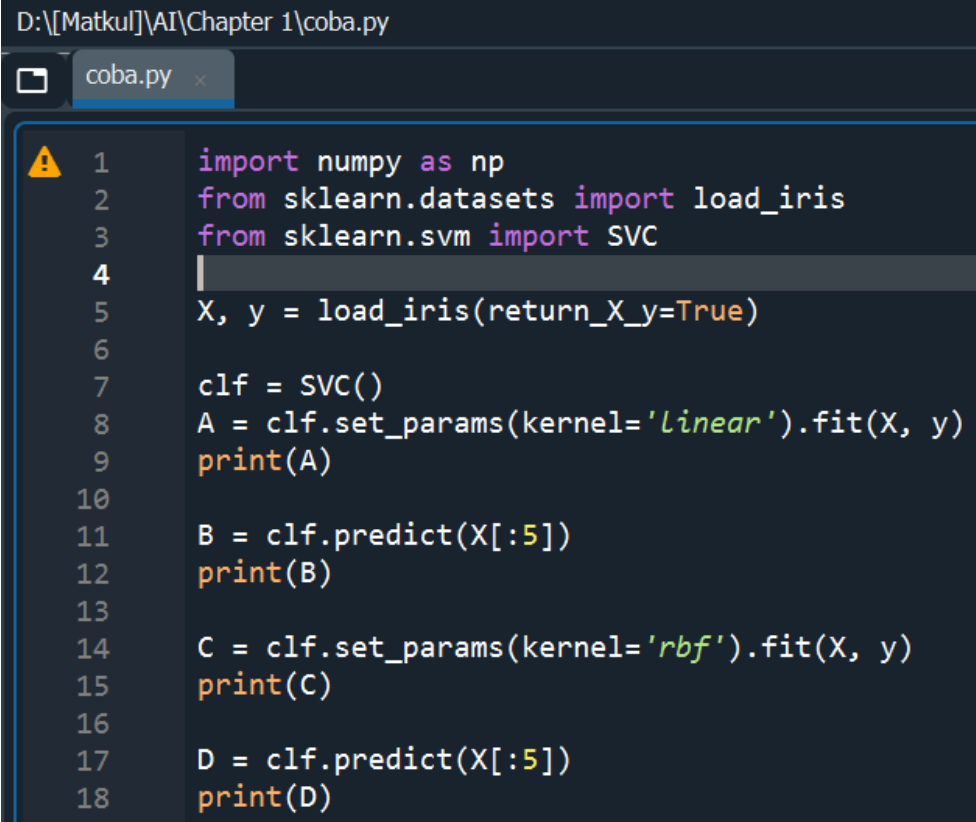
```
Console 1/A  
In [6]: runfile('D:/[Matkul]/AI/Chapter 1/coba.py', wdir='D:/[Matkul]/  
AI/Chapter 1')  
SVC()  
[0, 0, 0]  
SVC()  
['setosa', 'setosa', 'setosa']  
In [7]: |
```

Figure 1.17: Hasil Iris Predict

Di sini, `predict ()` pertama kali mengembalikan array integer, karena `iris.target` (array integer) digunakan dengan tepat. `Prediction ()` kedua mengembalikan string karena `iris.targetnames` cocok untuk penginstalan.

#### 1.2.4.2 Refitting and updating parameters

Hyperparameter dari estimator dapat diperbarui setelah dibuat dengan metode `set_params()`. Memanggil `fit()` beberapa kali akan menimpa `fit()` yang dipelajari sebelumnya:



```
D:\[Matkul]\AI\Chapter 1\coba.py
coba.py x
1 import numpy as np
2 from sklearn.datasets import load_iris
3 from sklearn.svm import SVC
4
5 X, y = load_iris(return_X_y=True)
6
7 clf = SVC()
8 A = clf.set_params(kernel='linear').fit(X, y)
9 print(A)
10
11 B = clf.predict(X[:5])
12 print(B)
13
14 C = clf.set_params(kernel='rbf').fit(X, y)
15 print(C)
16
17 D = clf.predict(X[:5])
18 print(D)
```

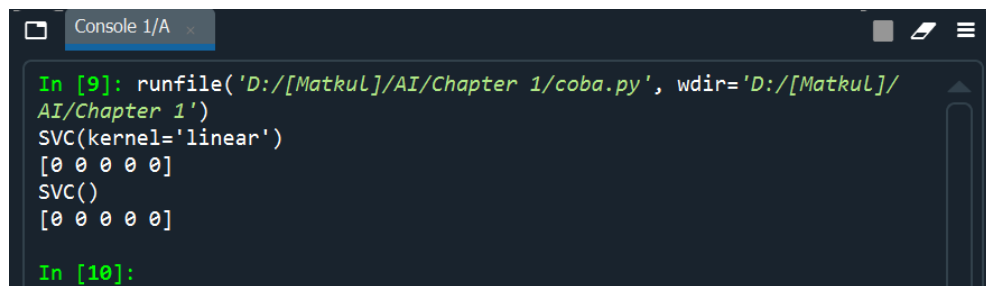
Figure 1.18: Refitting and updating parameters

Keterangan:

1. Baris pertama, memanggil library numpy dengan alias np.
2. Baris kedua, memanggil module load iris dari library sklearn.datasets.
3. Baris ketiga, memanggil module SVC dari library sklearn.svm.
4. Baris kelima, mengisi variable X dan y dengan datasets iris.
5. Baris ketujuh, mendefinisikan clf sebagai fungsi SVC.
6. Baris kedelapan, mengubah rbf menjadi linear melalui `SVC.setparams` kedalam variable A.

7. Baris kesembilan, menampilkan hasil variable A.
8. Baris kesebelas, clf dengan method predict akan menampilkan data X dalam array sebanyak 5 data kedalam variable B.
9. Baris kedua belas, menampilkan hasil variable B.
10. Baris keempat belas, mengubah linear ke rbf melalui SVC.setparams kedalam variable C.
11. Baris kelima belas, menampilkan hasil variable C.
12. Baris ketujuh belas, clf dengan method predict akan menampilkan data X dalam array sebanyak 5 data kedalam variable D.
13. Baris kedelapan belas, menampilkan hasil variable D.

Hasil dari Refitting and updating parameters.



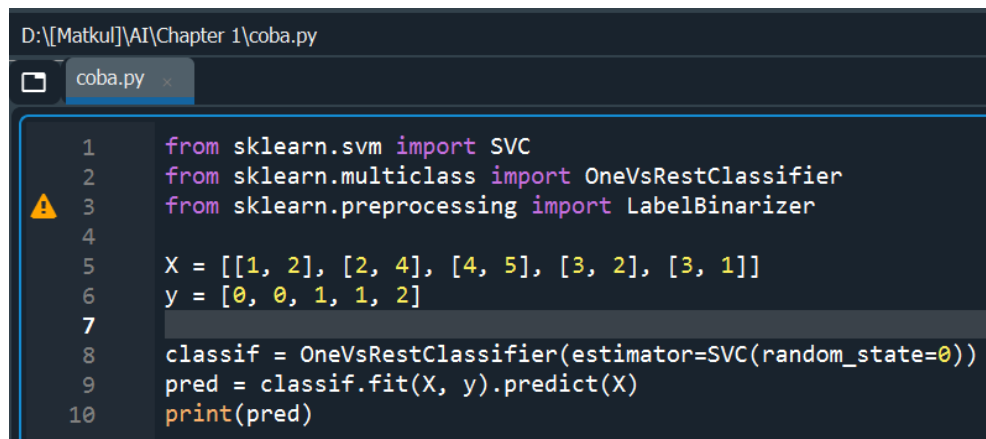
```
Console 1/A
In [9]: runfile('D:/[Matkul]/AI/Chapter 1/coba.py', wdir='D:/[Matkul]/
AI/Chapter 1')
SVC(kernel='linear')
[0 0 0 0 0]
SVC()
[0 0 0 0 0]
In [10]:
```

Figure 1.19: Hasil Refitting and updating parameters

Di sini, kernel default rbf pertama-tama diubah menjadi linier oleh SVC.setparams () setelah membuat estimator, dan kemudian diubah kembali ke rbf untuk mereparasi estimator dan membuat prediksi kedua.

#### 1.2.4.3 Multiclass vs. multilabel fitting

Saat menggunakan multiclass classifiers, tugas learning and prediction yang dilakukan bergantung pada format data target yang sesuai, yang bergantung pada:



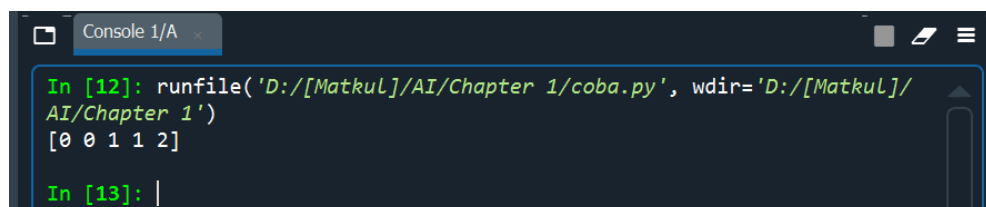
```
D:\[Matkul]\AI\Chapter 1\coba.py
coba.py x
1  from sklearn.svm import SVC
2  from sklearn.multiclass import OneVsRestClassifier
3  from sklearn.preprocessing import LabelBinarizer
4
5  X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
6  y = [0, 0, 1, 1, 2]
7
8  classif = OneVsRestClassifier(estimator=SVC(random_state=0))
9  pred = classif.fit(X, y).predict(X)
10 print(pred)
```

Figure 1.20: Multiclass Predict

Keterangan:

1. Baris pertama, memanggil module SVC dari library sklearn.svm.
2. Baris kedua, memanggil module OneVsRestClassifier dari library sklearn.multiclass.
3. Baris ketiga, memanggil module LabelBinarizer dari library sklearn.preprocessing.
4. Baris kelima dan keenam, variable x dan y yang berisi data array.
5. Baris kedelapan, menggunakan method OneVsRestClassifier dengan fungsi SVC sebagai estimator untuk menghasilkan data random yang didefinisikan kedalam classif.
6. Baris kesembilan, memberikan hasil multiclass prediksi yang sesuai kedalam variable pred.
7. Baris kesepuluh, menampilkan hasil variable pred.

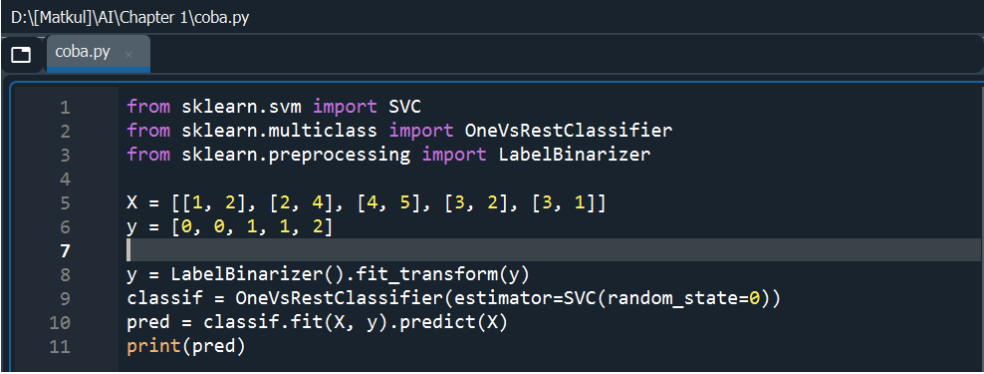
Hasil dari Multiclass Predict.



```
Console 1/A x
In [12]: runfile('D:/[Matkul]/AI/Chapter 1/coba.py', wdir='D:/[Matkul]/
AI/Chapter 1')
[0 0 1 1 2]
In [13]: |
```

Figure 1.21: Hasil Multiclass Predict

Dalam kasus di atas, classifier cocok untuk pred label multiclass, sehingga metode `predict ()` menyediakan prediksi multiclass yang sesuai. Dimungkinkan juga untuk menyesuaikan dengan array 2d indikator label biner:



```
D:\[Matkul]\AI\Chapter 1\coba.py
coba.py
1  from sklearn.svm import SVC
2  from sklearn.multiclass import OneVsRestClassifier
3  from sklearn.preprocessing import LabelBinarizer
4
5  X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
6  y = [0, 0, 1, 1, 2]
7
8  y = LabelBinarizer().fit_transform(y)
9  classif = OneVsRestClassifier(estimator=SVC(random_state=0))
10 pred = classif.fit(X, y).predict(X)
11 print(pred)
```

Figure 1.22: Multiclass Predict 2

Keterangan:

1. Baris pertama, memanggil module SVC dari library sklearn.svm.
2. Baris kedua, memanggil module OneVsRestClassifier dari library sklearn.multiclass.
3. Baris ketiga, memanggil module LabelBinarizer dari library sklearn.preprocessing.
4. Baris kelima dan keenam, variable x dan y yang berisi data array.
5. Baris kedelapan, classifier `fit()` merepresentasi variable y dengan LabelBinarizer.
6. Baris kesembilan, menggunakan method OneVsRestClassifier dengan fungsi SVC sebagai estimator untuk menghasilkan data random yang didefinisikan kedalam classif.
7. Baris kesepuluh, memberikan hasil multiclass prediksi yang sesuai kedalam variable `pred`.
8. Baris kesebelas, menampilkan hasil variable `pred`.

Hasil dari Multiclass Predict 2

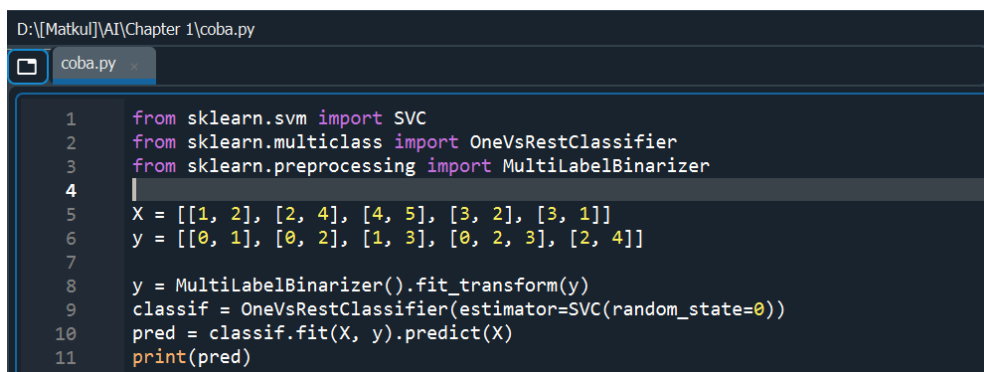


```
Console 1/A  
  
In [16]: runfile('D:/[Matkul]/AI/Chapter 1/coba.py', wdir='D:/[Matkul]/  
AI/Chapter 1')  
[[1 0 0]  
 [1 0 0]  
 [0 1 0]  
 [0 0 0]  
 [0 0 0]]  
  
In [17]:
```

Figure 1.23: Hasil Multiclass Predict 2

Di sini, LabelBinarizer digunakan untuk menyesuaikan pengklasifikasi dengan representasi label biner pred dari y. Dalam kasus ini, predict () mengembalikan pred dalam bentuk array yang mewakili prediksi yang sesuai.

MultiLabel Fitting, dalam kasus ini, beberapa label ditetapkan ke classifier yang sesuai untuk setiap sampel. MultiLabelBinarizer digunakan untuk membuat binarisasi pred multilabel agar sesuai. Akibatnya, predict () mengembalikan larik pred yang berisi beberapa label yang diprediksi untuk setiap instance.



```
D:\[Matkul]\AI\Chapter 1\coba.py  
coba.py  
1 from sklearn.svm import SVC  
2 from sklearn.multiclass import OneVsRestClassifier  
3 from sklearn.preprocessing import MultiLabelBinarizer  
4  
5 X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]  
6 y = [[0, 1], [0, 2], [1, 3], [0, 2, 3], [2, 4]]  
7  
8 y = MultiLabelBinarizer().fit_transform(y)  
9 classif = OneVsRestClassifier(estimator=SVC(random_state=0))  
10 pred = classif.fit(X, y).predict(X)  
11 print(pred)
```

Figure 1.24: MultiLabel Fitting

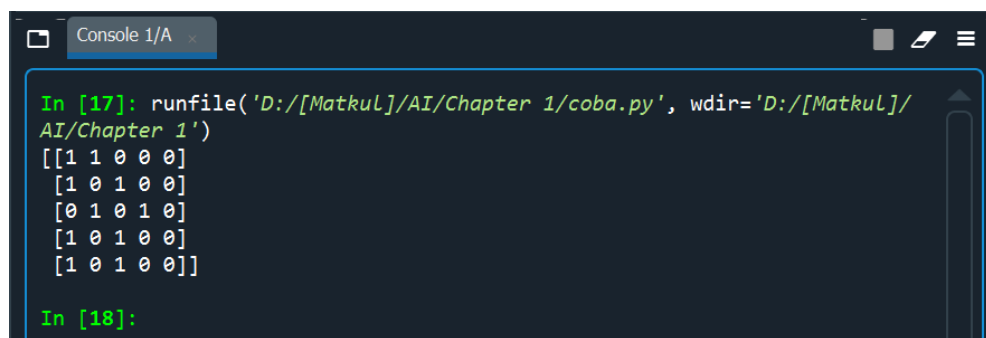
Keterangan:

1. Baris pertama, memanggil module SVC dari library sklearn.svm.
2. Baris kedua, memanggil module OneVsRestClassifier dari library sklearn.multiclass.
3. Baris ketiga, memanggil module LabelBinarizer dari library sklearn.preprocessing.
4. Baris kelima dan keenam, variable x dan y yang berisi data array.
5. Baris kedelapan, classifier fit() merepresentasi variable y dengan MultiLabelBinarizer.



6. Baris kesembilan, menggunakan method OneVsRestClassifier dengan fungsi SVC sebagai estimator untuk menghasilkan data random yang didefinisikan kedalam classif.
7. Baris kesepuluh, memberikan hasil multiclass prediksi yang sesuai kedalam variable pred.
8. Baris kesebelas, menampilkan hasil variable pred.

Hasil dari Multilabel Fitting



```

In [17]: runfile('D:/[Matkul]/AI/Chapter 1/coba.py', wdir='D:/[Matkul]/
AI/Chapter 1')
[[1 1 0 0 0]
 [1 0 1 0 0]
 [0 1 0 1 0]
 [1 0 1 0 0]
 [1 0 1 0 0]]

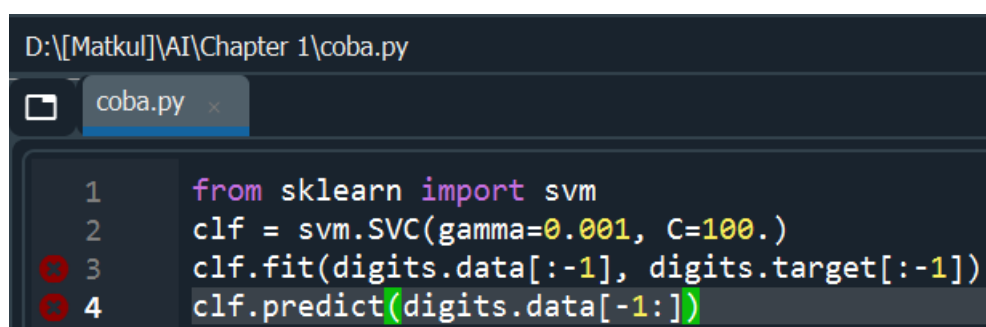
In [18]:

```

Figure 1.25: Hasil MultiLabel Fitting

## 1.3 Penanganan Error

### 1.3.1 Error 1



```

D:\[Matkul]\AI\Chapter 1\coba.py
coba.py
1  from sklearn import svm
2  clf = svm.SVC(gamma=0.001, C=100.)
3  clf.fit(digits.data[:-1], digits.target[:-1])
4  clf.predict(digits.data[-1:])

```

Figure 1.26: Hasil MultiLabel Fitting

Pada baris tiga dan empat terdapat error, yaitu sebagai berikut:

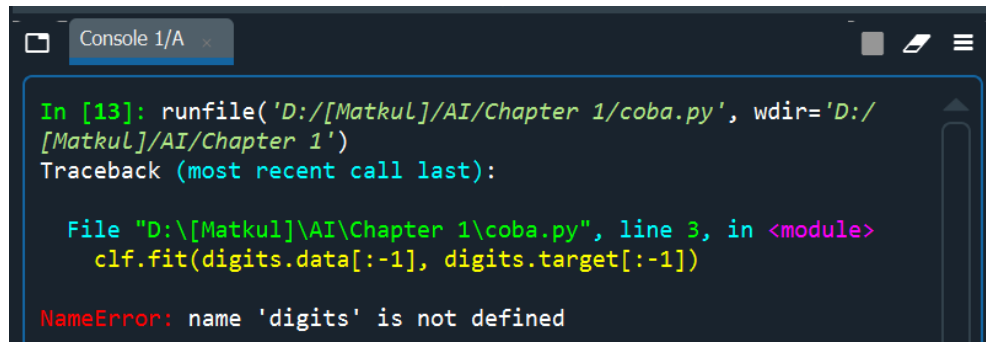
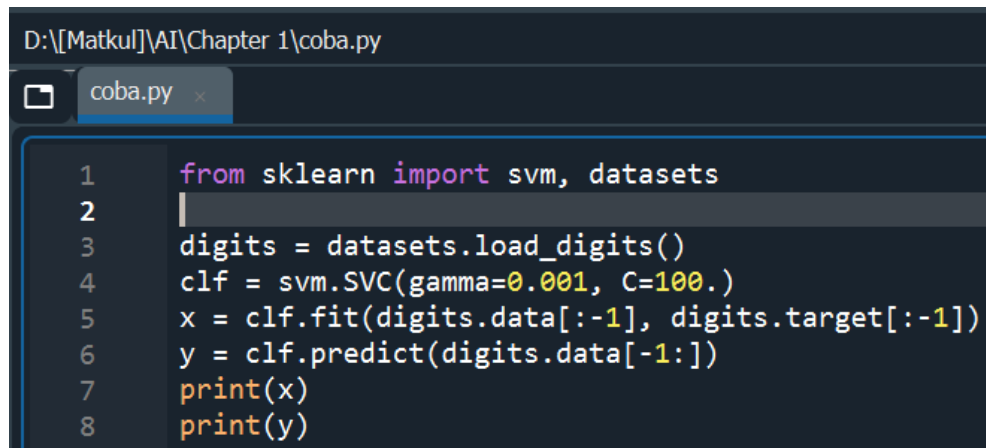
A screenshot of a Jupyter console window titled 'Console 1/A'. It shows the execution of a Python script. The first line is 'In [13]: runfile('D:/[Matkul]/AI/Chapter 1/coba.py', wdir='D:/[Matkul]/AI/Chapter 1')'. Below this, it says 'Traceback (most recent call last):'. Then, it shows the file path and line number: 'File "D:\[Matkul]\AI\Chapter 1\coba.py", line 3, in <module>'. The next line is 'clf.fit(digits.data[:-1], digits.target[:-1])'. Finally, it shows the error: 'NameError: name 'digits' is not defined'.

Figure 1.27: Hasil MultiLabel Fitting

Nama errornya adalah karena "digits" tidak terdefiniskan. Untuk penanganannya kita harus mendefinisikan digits sebagai berikut:

A screenshot of a Python script editor showing a file named 'coba.py'. The script contains the following code:

```
1 from sklearn import svm, datasets
2
3 digits = datasets.load_digits()
4 clf = svm.SVC(gamma=0.001, C=100.)
5 x = clf.fit(digits.data[:-1], digits.target[:-1])
6 y = clf.predict(digits.data[-1:])
7 print(x)
8 print(y)
```

Figure 1.28: Learning and Predicting

Dapat dilihat perbedaannya pada baris pertama dan ketiga, pada baris pertama, memanggil modul datasets yang sebelumnya tidak ada dan kemudian pada baris ketiga, digits didefinisikan.

## Chapter 2

# Membangun Model Prediksi

### 2.1 Teori

#### 2.1.1 Binary Classification

Setiap data dalam klasifikasi biner memiliki atribut kelas yang berisi dua nilai. Nilai suatu kelas dapat dinyatakan sebagai positif atau negatif. 0 atau 1; benar atau salah; dll. Contoh klasifikasi biner: pemfilteran spam (model klasifikasi yang mendeteksi apakah email diklasifikasikan sebagai spam atau non-spam).[5]

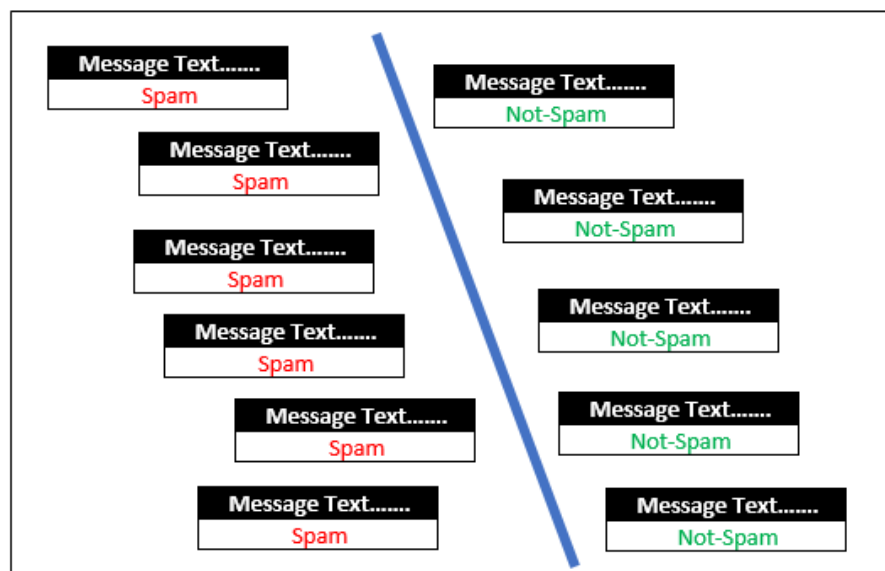


Figure 2.1: Contoh Spam Filtering, Garis Miring Biru adalah class boundary

### 2.1.2 Supervised Learning, Unsupervised Learning, dan Clustering

Supervised Learning biasanya digunakan untuk menyelesaikan masalah klasifikasi dan regresi. Algoritma sangat bergantung pada penerapan input dan output pada kumpulan data tertentu, jadi kami (pengguna / ilmuwan data) memainkan peran utama dalam memverifikasi input dan output ini. Perhatikan gambar berikut.

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

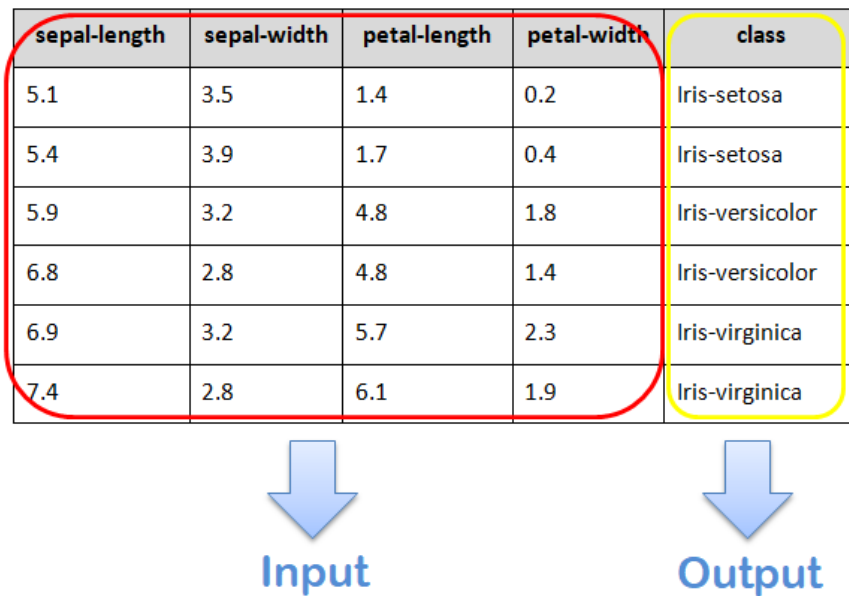


Figure 2.2: Supervised Learning: Dataset Iris

Pada gambar di atas, input merupakan variabel yang akan diamati, lalu output adalah hasilnya. Dataset iris telah memiliki kategori yang menjadi class, jika kita memasukkan input baru akan keluar hasil sesuai dengan kategori, pada contoh di atas keluar kategori iris-setosa.

Unsupervised Learning berarti tidak mengawasi dan memandu model machine learning, tetapi membiarkan model tersebut belajar dengan sendirinya untuk menemukan informasi yang mungkin tidak terlihat oleh mata manusia. Perhatikan gambar berikut.

ID	Age	Education	Experience
1	30	2	6
2	28	1	3
3	44	3	22
4	37	1	8
5	41	2	19
6	26	2	5

Figure 2.3: Unsupervised Learning

Seperti yang dapat dilihat dari contoh data di atas, data yang diberikan hanya berupa variabel masukan, tanpa label (Output). Dalam hal ini, model berlatih pada kumpulan data, kemudian mencari informasi dan menarik kesimpulannya sendiri dari data tersebut.

Clustering atau pengelompokan merupakan salah satu masalah dalam penggunaan teknik unsupervised learning. Contoh cluster termasuk segmentasi perbankan pelanggan atau segmentasi berita online.

Pada dasarnya, jika kumpulan data tidak memiliki label class, teknik unsupervised learning akan digunakan. Dengan kata lain, algoritma dapat secara otomatis membagi data menjadi beberapa cluster berdasarkan kriteria tertentu (seperti kesamaan).[10]

### 2.1.3 Evaluasi dan Akurasi

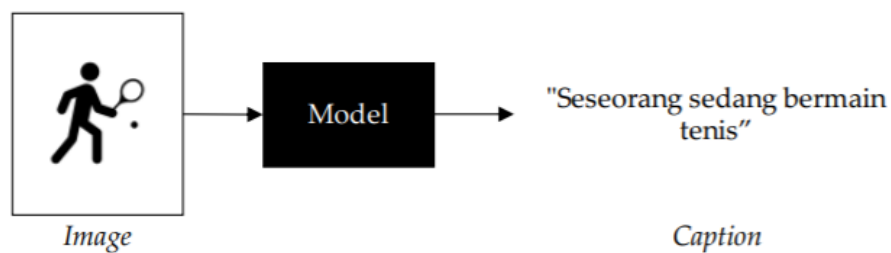


Figure 2.4: Model Evaluasi

Pada gambar di atas, bagaimana Anda mengevaluasi apakah model dapat memberikan penjelasan yang mencerminkan situasi gambar? Misalnya, jika gambar diberikan "Seseorang sedang bermain tenis" dan model memprediksi "Seseorang bermain bola basket", model memberikan interpretasi yang salah. Namun, jika Anda melihat proporsi dari jumlah kata yang sama, Anda berharap ada banyak kata di output yang cocok dengan prediksi, yaitu "seseorang sedang bermain".[11]

Akurasi adalah indikator kinerja paling sederhana dan sering digunakan saat mengevaluasi kinerja model. Akurasi didefinisikan sebagai proporsi prediksi yang benar dibagi dengan jumlah sampel. Mengingat output yang diperlukan (juga dikenal sebagai "gold standard")  $y$  dan hasil prediksi  $\hat{y}$ , dengan menghitung rasio jumlah elemen yang sama antara  $y$  dan  $\hat{y}$ . Secara matematis akurasi dirumuskan dengan persamaan, dimana  $N$  merepresentasikan jumlah sampel.[11]

$$\text{Akurasi} = \frac{1}{N} \sum_{i=1}^N \text{verdict}_i$$

$$\text{verdict}_i = \begin{cases} 1, & y_i = \hat{y}_i \\ 0, & y_i \neq \hat{y}_i \end{cases}$$

Figure 2.5: Akurasi

#### 2.1.4 Confusion Matrix

Confusion Matrix juga biasa disebut sebagai Error Matrix. Pada dasarnya, confusion matrix memberikan informasi tentang perbandingan hasil klasifikasi yang dilakukan oleh sistem (model) dengan hasil klasifikasi yang sebenarnya. Matriks konfusi berbentuk tabel matriks yang mendeskripsikan performa model klasifikasi pada rangkaian data uji dengan nilai sebenarnya yang diketahui. Gambar di bawah ini adalah matriks konfusi, yang berisi 4 kombinasi berbeda dari nilai prediksi dan nilai aktual. Lihatlah gambar di bawah ini:[2]

		<b>Actual Values</b>	
		<b>1 (Positive)</b>	<b>0 (Negative)</b>
<b>Predicted Values</b>	<b>1 (Positive)</b>	<b>TP</b> (True Positive)	<b>FP</b> (False Positive) <i>Type I Error</i>
	<b>0 (Negative)</b>	<b>FN</b> (False Negative) <i>Type II Error</i>	<b>TN</b> (True Negative)

Figure 2.6: Confusion Matrix

- True Positive (TP)

Ini adalah data positif yang prediksinya benar. Misalnya, pasien menderita kanker (tingkat 1), dan model memprediksi bahwa pasien menderita kanker (tingkat 1).

- True Negative (TN)

Menunjukkan data negatif yang diperkirakan benar. Misalnya, pasien tidak mengidap kanker (level 2), dan model memprediksi bahwa pasien tidak mengidap kanker (level 2).

- False Postive (FP) — Type I Error

Ini adalah angka negatif, tetapi ramalannya adalah angka positif. Misalnya pasien tidak menderita kanker (kategori 2), tetapi model yang memprediksi bahwa pasien tersebut menderita kanker (kategori 1).

- False Negative (FN) — Type II Error

Ini adalah angka positif, tetapi ramalannya adalah angka negatif. Misalnya, seorang pasien mengidap kanker (grade 1), tetapi model memprediksi bahwa pasien tersebut tidak mengidap kanker (grade 2).

### 2.1.5 K-Fold Cross Validation

K-fold cross validation merupakan teknik verifikasi dalam pengembangan model verifikasi terpisah, dimana verifikasi mengukur training error dengan menggunakan data uji atau test data untuk pengujian. Pengembangan cross validation sendiri dikarenakan terdapat kelemahan pada model sebelumnya yaitu sampel dipilih secara random, sehingga test error sampling tidak dapat menetapkan kelas secara terstruktur. Walaupun hasil yang didapat dapat dimaksimalkan, tes yang lebih efektif tidak dapat dicapai. Kemudian muncul cross validation, yang dapat bekerja dengan cepat dengan pengambilan sampel yang lebih terstruktur. Oleh karena itu, meskipun dalam jumlah pengujian, data yang berbeda dari eksperimen atau literasi sebelumnya akan digunakan untuk mendapatkan beberapa training dataset dan testing dataset.[9]

Percobaan 1	Test	Train	Train	Train	Train
Percobaan 2	Train	Test	Train	Train	Train
Percobaan 3	Train	Train	Test	Train	Train
Percobaan 4	Train	Train	Train	Test	Train
Percobaan 5	Train	Train	Train	Train	Test

Figure 2.7: 5-Fold Cross Validation

Percobaan di atas merupakan contoh validasi silang 5 kali lipat, yang artinya percobaan tersebut perlu dilakukan 5 tahap. Percobaan 1, untuk mengubah bagian pertama dari partisi menjadi data test, dan partisi lainnya menjadi data training, dan seterusnya.

Dari 5 hasil percobaan ini, kita akan menggunakan confusion matrix untuk mencatat nilai evaluasi kinerja model, kemudian menentukan nilai rata-rata setiap percobaan. Kemudian, akan ditemukan eksperimen mana yang dapat digunakan sebagai referensi untuk menggunakan model algoritma yang dipilih.



### 2.1.6 Decision Tree

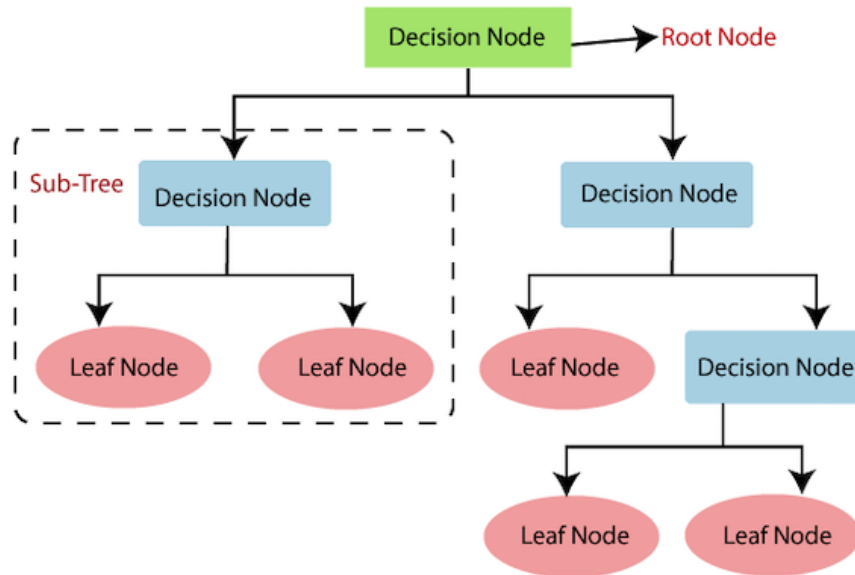


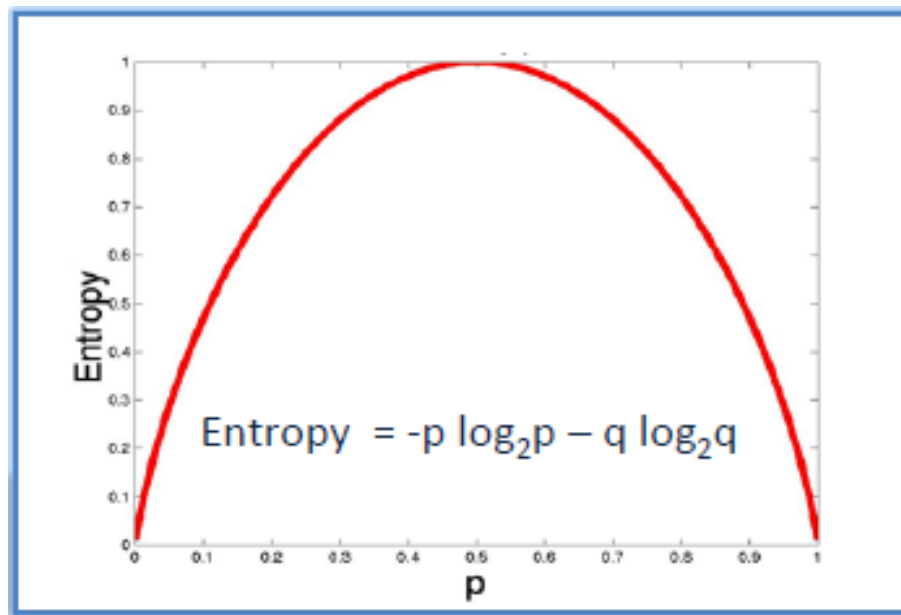
Figure 2.8: 5-Fold Cross Validation

Decision Tree adalah diagram yang dapat membantu Anda memilih salah satu dari beberapa opsi pengoperasian. Biasanya, dimulai dengan satu atau lebih node. Kemudian, cabang node menunjukkan opsi yang tersedia. Selain itu, setiap cabang tersebut akan memiliki cabang baru. Oleh karena itu, cara ini dinamakan "Tree" karena mirip dengan pohon yang banyak cabangnya. Decision tree dapat membuat berbagai pilihan dan menyelidiki kemungkinan hasil dari pilihan tersebut. Selain itu, Anda juga dapat melihat potensi risiko dan keuntungan dari setiap opsi. Mengutip dari Venngage, decision tree mengandung tiga unsur, yaitu:[4]

- Root Node (Akar): tujuan akhir atau keputusan besar yang harus dibuat.
- Branches (Ranting): berbagai opsi tindakan.
- Leaf Node (Daun): kemungkinan hasil atas setiap opsi tindakan.

### 2.1.7 Entropi dan Information Gain

Pohon keputusan dibangun dari simpul akar dari atas ke bawah dan melibatkan pembagian data menjadi subset yang berisi contoh nilai yang sama (homogen). Algoritma ID3 menggunakan entropi untuk menghitung keseragaman sampel. Jika sampel benar-benar seragam, entropinya nol; jika distribusi sampel sama, entropi adalah 1.[3]



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Figure 2.9: Entropy

1. Entropi yang menggunakan tabel frekuensi atribut tunggal:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5



$$\begin{aligned} \text{Entropy}(\text{PlayGolf}) &= \text{Entropy}(5,9) \\ &= \text{Entropy}(0.36, 0.64) \\ &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94 \end{aligned}$$

Figure 2.10: Entropy Atribut Tunggal

2. Entropi yang menggunakan tabel frekuensi dua atribut:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



$$\begin{aligned}
 E(\text{PlayGolf}, \text{Outlook}) &= P(\text{Sunny}) \cdot E(3,2) + P(\text{Overcast}) \cdot E(4,0) + P(\text{Rainy}) \cdot E(2,3) \\
 &= (5/14) \cdot 0.971 + (4/14) \cdot 0.0 + (5/14) \cdot 0.971 \\
 &= 0.693
 \end{aligned}$$

Figure 2.11: Entrophy Dua Atribut

Information gain didasarkan pada pengurangan entropi setelah kumpulan data dibagi dengan atribut. Keseluruhan proses membangun pohon keputusan adalah menemukan atribut yang mengembalikan perolehan informasi tertinggi (yaitu, cabang yang paling homogen).

1. Langkah Pertama: Hitung Entropi.

$$\begin{aligned}
 \text{Entropy}(\text{PlayGolf}) &= \text{Entropy}(5,9) \\
 &= \text{Entropy}(0.36, 0.64) \\
 &= - (0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\
 &= 0.94
 \end{aligned}$$

Figure 2.12: Langkah 1

2. Langkah 2: Kemudian bagi kumpulan data menjadi atribut yang berbeda. Hitung entropi setiap cabang. Kemudian tambahkan secara proporsional untuk mendapatkan nilai entropi dari pembagian tersebut. Kurangi entropi yang dihasilkan dari entropi sebelum pemisahan. Hasilnya adalah penurunan perolehan informasi atau entropi.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			


		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

$$\begin{aligned} G(\text{PlayGolf}, \text{Outlook}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Outlook}) \\ &= 0.940 - 0.693 = 0.247 \end{aligned}$$

Figure 2.13: Langkah 2

- Langkah 3: Pilih atribut dengan perolehan informasi terbesar sebagai simpul keputusan, bagi kumpulan data menjadi beberapa cabang, dan ulangi proses yang sama pada setiap cabang.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

Outlook	Sunny	Outlook	Temp	Humidity	Windy	Play Golf
		Sunny	Mild	High	FALSE	Yes
		Sunny	Cool	Normal	FALSE	Yes
		Sunny	Cool	Normal	TRUE	No
		Sunny	Mild	Normal	FALSE	Yes
	Sunny	Mild	High	TRUE	No	
	Overcast	Overcast	Hot	High	FALSE	Yes
		Overcast	Cool	Normal	TRUE	Yes
		Overcast	Mild	High	TRUE	Yes
		Overcast	Hot	Normal	FALSE	Yes
	Rainy	Rainy	Hot	High	FALSE	No
		Rainy	Hot	High	TRUE	No
		Rainy	Mild	High	FALSE	No
		Rainy	Cool	Normal	FALSE	Yes
		Rainy	Mild	Normal	TRUE	Yes

Figure 2.14: Langkah 3

4. Langkah 4a: Cabang dengan entropi 0 adalah simpul daun.

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes

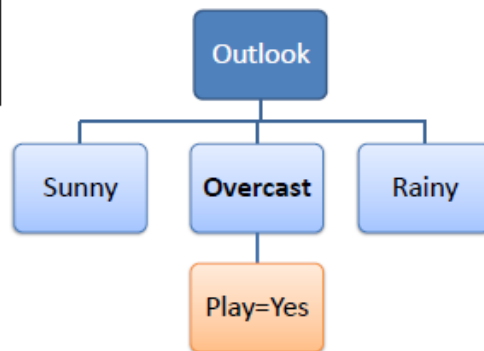


Figure 2.15: Langkah 4a

5. Langkah 4b: Cabang dengan entropi lebih besar dari 0 akan dipecah lagi.

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No

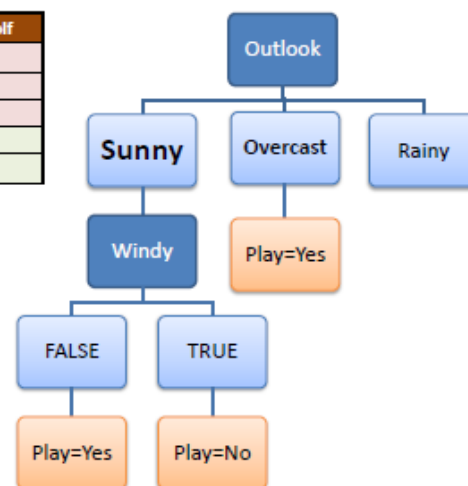


Figure 2.16: Langkah 4b

## 2.2 Scikit Learn

Dataset ambil di <https://github.com/PacktPublishing/Python-Artificial-Intelligence-Projects-for-Beginners> folder Chapter01. Tugas anda adalah, dataset ganti menggunakan student-mat.csv dan mengganti semua nama variabel dari kode di bawah ini dengan nama-nama makanan (NPM mod 3=0), kota (NPM mod 3=1), buah (NPM mod 3=2).

### 2.2.1 Nomor 1

```

1 import pandas as pd #import library pandas dengan alias pd
2 akiba = pd.read_csv('D:/[Matkul]/AI/Chapter 2/Python-Artificial-
    Intelligence-Projects-for-Beginners-master/Chapter01/dataset/student
    -mat.csv', sep=';') #sebuah variable akiba untuk memanggil file csv
3 len(akiba) #untuk menghitung jumlah data di file csv

```

Hasilnya:

```

In [1]: import pandas as pd
...: akiba = pd.read_csv('D:/[Matkul]/AI/Chapter 2/Python-Artificial-
Intelligence-Projects-for-Beginners-master/Chapter01/dataset/student-mat.csv',
sep=';')
...: len(akiba)
Out[1]: 395

```

Figure 2.17: Nomor 1

## 2.2.2 Nomor 2

```

1 akiba['pass'] = akiba.apply(lambda row: 1 if (row['G1']+row['G2']+row['
    G3'])>= 35 else 0, axis=1) #membuat label binary pass atau fail
    berdasarkan G1G2G3 dengan batas data lebih atau sama dengan 35
2 akiba = akiba.drop(['G1', 'G2', 'G3'], axis=1) #menghitung G1,2,3
3 akiba.head() #menampilkan data

```

Hasilnya

```

In [2]: akiba['pass'] = akiba.apply(lambda row: 1 if (row['G1']+row['G2']
+row['G3'])>= 35 else 0, axis=1) #membuat label binary pass atau fail
berdasarkan G1G2G3 dengan batas data lebih atau sama dengan 35
...: akiba = akiba.drop(['G1', 'G2', 'G3'], axis=1) #menghitung G1,2,3
...: akiba.head() #menampilkan data
Out[2]:
  school sex  age address famsize  ... Dalc  Walc  health absences pass
0     GP  F   18      U    GT3  ...   1     1     3         6     0
1     GP  F   17      U    GT3  ...   1     1     3         4     0
2     GP  F   15      U    LE3  ...   2     3     3        10     0
3     GP  F   15      U    GT3  ...   1     1     5         2     1
4     GP  F   16      U    GT3  ...   1     2     5         4     0

[5 rows x 31 columns]

```

Figure 2.18: Nomor 2

## 2.2.3 Nomor 3

```

1 #memanggil kolom pada file csv
2 akiba = pd.get_dummies(akiba, columns=['sex', 'school', 'address', '
    famsize', 'Pstatus', 'Mjob',
3                                     'Fjob', 'reason', 'guardian', '
4                                     schoolsup', 'famsup', 'paid',
                                     'activities', 'nursery', 'higher',
5                                     , 'internet', 'romantic'])
akiba.head() #menampilkan data

```

Hasilnya

```
In [3]:
....: akiba = pd.get_dummies(akiba, columns=['sex', 'school', 'address',
'famsize', 'Pstatus', 'Mjob',
....:                                     'Fjob', 'reason', 'guardian',
'schoolsup', 'famsup', 'paid',
....:                                     'activities', 'nursery',
'higher', 'internet', 'romantic'])
....: akiba.head() #menampilkan data
Out[3]:
   age  Medu  Fedu  ...  internet_yes  romantic_no  romantic_yes
0   18     4     4  ...              0             1              0
1   17     1     1  ...              1             1              0
2   15     1     1  ...              1             1              0
3   15     4     2  ...              1             0              1
4   16     3     3  ...              0             1              0

[5 rows x 57 columns]
```

Figure 2.19: Nomor 3

## 2.2.4 Nomor 4

```
1 akiba = akiba.sample(frac=1) #variable akiba dengan isi data sample
2 akiba_train = akiba[:500] #membagi data untuk training
3 akiba_test = akiba[500:] #membagi data untuk test
4
5 akiba_train_att = akiba_train.drop(['pass'], axis=1) #menghapus data
   yang telah lewat(pass) kemudian diinputkan
6 akiba_train_pass = akiba_train['pass'] #mengambil data pass saja
7
8 akiba_test_att = akiba_test.drop(['pass'], axis=1) #menghapus data yang
   telah lewat(pass) kemudian diinputkan
9 akiba_test_pass = akiba_test['pass'] #mengambil data pass saja
10
11 akiba_att = akiba.drop(['pass'], axis=1) #menghapus data yang telah
   lewat(pass) kemudian diinputkan
12 akiba_pass = akiba['pass'] #mengambil data pass saja
13
14 import numpy as np #import library numpy dengan alias np
15 print ("Passing: %a out of %a (%.2f%%)" % (np.sum(akiba_pass), len(
   akiba_pass), 100*float(np.sum(akiba_pass)) / len(akiba_pass))) #
   menampilkan data
```

Hasilnya



```

In [4]: akiba = akiba.sample(frac=1) #variable akiba dengan isi data sample
...: akiba_train = akiba[:500] #membagi data untuk training
...: akiba_test = akiba[500:] #membagi data untuk test
...:
...: akiba_train_att = akiba_train.drop(['pass'], axis=1) #menghapus data
yang telah lewat(pass) kemudian diinputkan
...: akiba_train_pass = akiba_train['pass'] #mengambil data pass saja
...:
...: akiba_test_att = akiba_test.drop(['pass'], axis=1) #menghapus data yang
telah lewat(pass) kemudian diinputkan
...: akiba_test_pass = akiba_test['pass'] #mengambil data pass saja
...:
...: akiba_att = akiba.drop(['pass'], axis=1) #menghapus data yang telah
lewat(pass) kemudian diinputkan
...: akiba_pass = akiba['pass'] #mengambil data pass saja
...:
...: import numpy as np #import library numpy dengan alias np
...: print ("Passing: %a out of %a (%.2f%%)" % (np.sum(akiba_pass),
len(akiba_pass), 100*float(np.sum(akiba_pass)) / len(akiba_pass))) #menampilkan
data
Passing: 166 out of 395 (42.03%)

```

Figure 2.20: Nomor 4

### 2.2.5 Nomor 5

```

1 from sklearn import tree #import library tree dari sklearn
2 tokyo = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5) #
   membuat decision tree dengan max depth adalah 5
3 tokyo = tokyo.fit(akiba_train_att, akiba_train_pass) #memasukkan data
   untuk decision tree

```

Hasilnya

```

In [5]: from sklearn import tree #import library tree dari sklearn
...: tokyo = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
#membuat decision tree dengan max depth adalah 5
...: tokyo = tokyo.fit(akiba_train_att, akiba_train_pass) #memasukkan data
untuk decision tree

```

Figure 2.21: Nomor 5

### 2.2.6 Nomor 6

```

1 import graphviz #import library graphviz
2 dot_data = tree.export_graphviz(tokyo, out_file=None, label="all",
   impurity=False,
3                                     proportion=True, feature_names=list(
   akiba_train_att),
4                                     class_names=["fail", "pass"], filled=
   True, rounded=True) #mendefinisikan data yang akan divisualisasikan
   di decision tree
5 graph = graphviz.Source(dot_data) #memasukkan data ke variable graph
6 graph #menampilkan gambar

```

Hasilnya



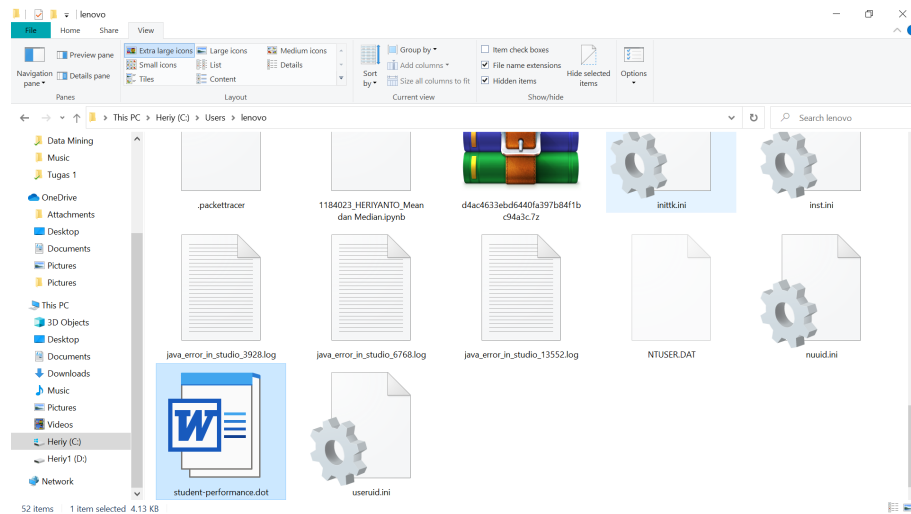


Figure 2.25: Nomor 7

## 2.2.8 Nomor 8

```
1 tokyo.score(akiba_test_att , akiba_test_pass) #menghitung prediksi nilai yang akan datang
```

Hasilnya error, kenapa hasilnya error karena hasil prediksi dari student-mat.csv itu nol(0), jadi tidak menampilkan nilai prediksi. Untuk menampilkan nilai prediksi minimal nilainya harus 1.

```
akan datang

File "D:\Aplikasi\Anaconda3\lib\site-packages\sklearn\base.py", line 499, in
score
    return accuracy_score(y, self.predict(X), sample_weight=sample_weight)

File "D:\Aplikasi\Anaconda3\lib\site-packages\sklearn\tree\_classes.py", line
427, in predict
    X = self._validate_X_predict(X, check_input)

File "D:\Aplikasi\Anaconda3\lib\site-packages\sklearn\tree\_classes.py", line
389, in _validate_X_predict
    X = check_array(X, dtype=DTYPE, accept_sparse="csr")

File "D:\Aplikasi\Anaconda3\lib\site-packages\sklearn\utils\validation.py",
line 77, in inner_f
    return f(**kwargs)

File "D:\Aplikasi\Anaconda3\lib\site-packages\sklearn\utils\validation.py",
line 650, in check_array
    raise ValueError("Found array with %d sample(s) (shape=%s) while a"

ValueError: Found array with 0 sample(s) (shape=(0, 56)) while a minimum of 1 is
required.
```

Figure 2.26: Nomor 8

## 2.2.9 Nomor 9

```

1 from sklearn.model_selection import cross_val_score #import module
  cross_val_score dari sklearn.model_selection
2 okinawa = cross_val_score(tokyo, akiba_att, akiba_pass, cv=5) #pembagian
  data menjadi 5 bagian
3 print ("Accuracy: %0.2f (+/- %0.2f)" % (okinawa.mean(), okinawa.std() *
  2)) #menampilkan hasil dari rata-rata dan standar deviasi dikali 2

```

Hasilnya

```

In [12]: from sklearn.model_selection import cross_val_score #import module
cross_val_score dari sklearn.model_selection
...: okinawa = cross_val_score(tokyo, akiba_att, akiba_pass, cv=5)
#pembagian data menjadi 5 bagian
...: print ("Accuracy: %0.2f (+/- %0.2f)" % (okinawa.mean(), okinawa.std() *
2)) #menampilkan hasil dari rata-rata dan standar deviasi dikali 2
Accuracy: 0.54 (+/- 0.07)

```

Figure 2.27: Nomor 9

## 2.2.10 Nomor 10

```

1 for max_depth in range(1, 20): #pengulangan max depth dalam jangkauan
  1-20
2   tokyo = tree.DecisionTreeClassifier(criterion="entropy", max_depth=
  max_depth) #membuat decision tree
3   okinawa = cross_val_score(tokyo, akiba_att, akiba_pass, cv=5) #
  pembagian data menjadi 5 bagian
4   print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth,
  okinawa.mean(), okinawa.std() * 2)) #menampilkan hasil dari rata-
  rata dan standar deviasi dikali 2

```

Hasilnya

```

...: okinawa = cross_val_score(tokyo, akiba_att, akiba_pass, cv=5)
#pembagian data menjadi 5 bagian
...: print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth,
okinawa.mean(), okinawa.std() * 2)) #menampilkan hasil dari rata-rata dan
standar deviasi dikali 2
Max depth: 1, Accuracy: 0.58 (+/- 0.01)
Max depth: 2, Accuracy: 0.58 (+/- 0.09)
Max depth: 3, Accuracy: 0.57 (+/- 0.08)
Max depth: 4, Accuracy: 0.57 (+/- 0.05)
Max depth: 5, Accuracy: 0.54 (+/- 0.06)
Max depth: 6, Accuracy: 0.58 (+/- 0.16)
Max depth: 7, Accuracy: 0.58 (+/- 0.16)
Max depth: 8, Accuracy: 0.54 (+/- 0.10)
Max depth: 9, Accuracy: 0.54 (+/- 0.11)
Max depth: 10, Accuracy: 0.54 (+/- 0.05)
Max depth: 11, Accuracy: 0.52 (+/- 0.08)
Max depth: 12, Accuracy: 0.54 (+/- 0.07)
Max depth: 13, Accuracy: 0.53 (+/- 0.11)
Max depth: 14, Accuracy: 0.53 (+/- 0.12)
Max depth: 15, Accuracy: 0.55 (+/- 0.11)
Max depth: 16, Accuracy: 0.56 (+/- 0.13)
Max depth: 17, Accuracy: 0.54 (+/- 0.11)
Max depth: 18, Accuracy: 0.55 (+/- 0.13)
Max depth: 19, Accuracy: 0.53 (+/- 0.08)

```

Figure 2.28: Nomor 10

### 2.2.11 Nomor 11

```
1 depth_acc = np.empty((19,3), float) #membuat array baru
2 i = 0 #variable i dengan isi 0
3 for max_depth in range(1, 20): #pengulangan dalam jangkauan 1-20
4     tokyo = tree.DecisionTreeClassifier(criterion="entropy", max_depth=
5     max_depth) #membuat decision tree
6     okinawa = cross_val_score(tokyo, akiba_att, akiba_pass, cv=5) #
7     pembagian data menjadi 5 bagian
8     depth_acc[i,0] = max_depth #meng-inputkan data max depth ke array
9     depth_acc
10     depth_acc[i,1] = okinawa.mean() #meng-inputkan rata-rata ke array
11     depth_acc
12     depth_acc[i,2] = okinawa.std() * 2 #meng-inputkan nilai standar
13     deviasi ke array depth_acc
14     i += 1
15 depth_acc
```

```
...:
...: depth_acc
Out[15]:
array([[1.00000000e+00, 5.79746835e-01, 1.01265823e-02],
       [2.00000000e+00, 5.79746835e-01, 8.97217476e-02],
       [3.00000000e+00, 5.72151899e-01, 8.06955820e-02],
       [4.00000000e+00, 5.69620253e-01, 5.06329114e-02],
       [5.00000000e+00, 5.54430380e-01, 8.38123816e-02],
       [6.00000000e+00, 5.67088608e-01, 1.28491924e-01],
       [7.00000000e+00, 5.84810127e-01, 1.29485690e-01],
       [8.00000000e+00, 5.36708861e-01, 1.11622317e-01],
       [9.00000000e+00, 5.54430380e-01, 1.17004760e-01],
       [1.00000000e+01, 5.26582278e-01, 8.25797794e-02],
       [1.10000000e+01, 5.36708861e-01, 1.19390644e-01],
       [1.20000000e+01, 5.49367089e-01, 1.08122311e-01],
       [1.30000000e+01, 5.21518987e-01, 1.31450683e-01],
       [1.40000000e+01, 5.46835443e-01, 1.05481856e-01],
       [1.50000000e+01, 5.44303797e-01, 1.16565716e-01],
       [1.60000000e+01, 5.51898734e-01, 1.11622317e-01],
       [1.70000000e+01, 5.29113924e-01, 1.47965967e-01],
       [1.80000000e+01, 5.44303797e-01, 1.01265823e-01],
       [1.90000000e+01, 5.56962025e-01, 1.09769536e-01]])
```

Figure 2.29: Nomor 11

### 2.2.12 Nomor 12

```
1 import matplotlib.pyplot as plt #import library matplotlib dengan alias
   plt
2 fig, ax = plt.subplots() #membuat plot baru
```

```

3 ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2]) #
   mengisi data plot
4 plt.show() #menampilkan data plot

```

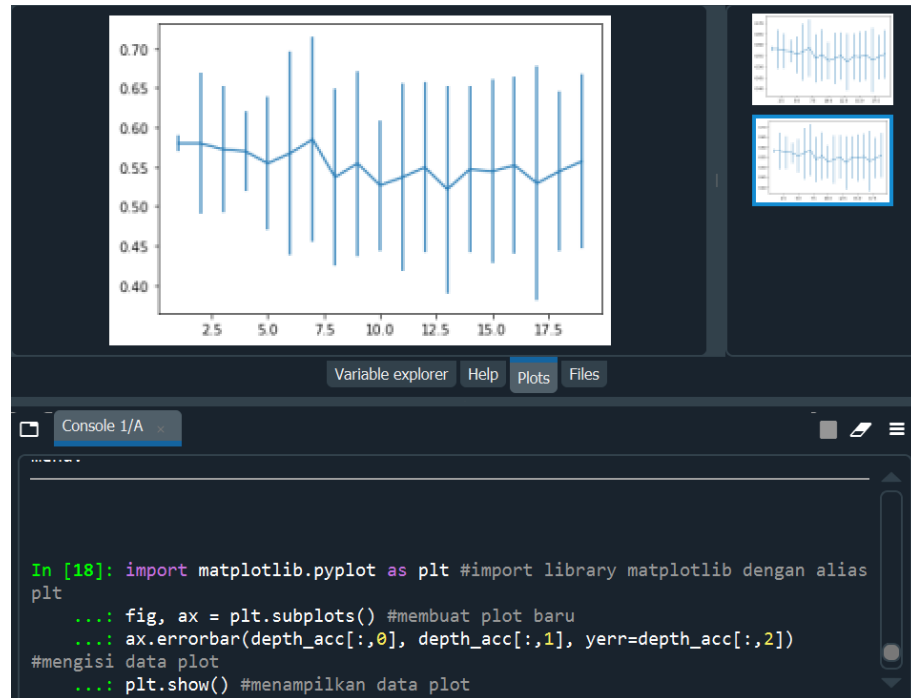


Figure 2.30: Nomor 12

# Bibliography

- [1] An introduction to machine learning with scikit-learn scikit-learn. <https://scikit-learn.org/stable/tutorial/basic/tutorial.html>. Accessed: 2021-03-06.
- [2] Confusion matrix untuk evaluasi model pada supervised learning. <https://medium.com/@ksnugroho/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f>. Accessed: 2021-03-14.
- [3] Data mining – decision tree. <https://danymochtars.wordpress.com/2017/03/07/data-mining-decision-tree/>. Accessed: 2021-03-14.
- [4] Decision tree: Pengertian, plus minus dan cara membuatnya. <https://glints.com/id/lowongan/decision-tree-adalah/.YE8yZJ0zbb0>. Accessed: 2021-03-14.
- [5] Jenis klasifikasi berdasarkan tipe output-nya. <https://zidny.dosen.itelkom-pwt.ac.id/jenis-klasifikasi/>. Accessed: 2021-03-14.
- [6] Memisah dataset menjadi training-set dan test-set. <https://medium.com/machine-learning-id/memisah-dataset-menjadi-training-set-dan-test-set-19b45dd52f6d>: :text=. Accessed: 2021-03-06.
- [7] Mengenal jenis pembelajaran mesin supervised learning dan unsupervised learning. <https://medium.com/kelompok1/mengenal-jenis-pembelajaran-mesin-supervised-learning-dan-unsupervised-learning-c588881e8ef5>. Accessed: 2021-03-06.
- [8] Pemanfaatan Artificial Intelligence di Era Revolusi Industri 4.0 sejarah kecerdasan buatan. <https://www.sekawanmedia.co.id/apa-itu-artificial-intelligence/>. Accessed: 2021-03-06.
- [9] Pengujian data dengan cross validation. <https://www.pengalaman-edukasi.com/2020/04/apa-itu-k-fold-cross-validation.html>. Accessed: 2021-03-14.

- [10] Perbedaan supervised dan unsupervised learning.  
<https://ilmudatapy.com/perbedaan-supervised-dan-unsupervised-learning/>.  
Accessed: 2021-03-14.
- [11] Seleksi fitur dan metode evaluasi. <https://wiragotama.github.io/resources/ebook/parts/JWGI-intro-to-ml-chap9-secured.pdf>. Accessed: 2021-03-14.
- [12] Training dan test set - artificial intelligence and data science.  
<https://www.megabagus.id/training-set-test-set/>. Accessed: 2021-03-06.
- [13] I DASAR ARTIFICIAL INTELLIGENCE AI. Kecerdasan buatan. 2011.
- [14] Sang Made Naufal Caesarya Mahardhika, Moch Arif Bijaksana, and Siti Sa'adah. Analisis monolingual alignment dalam mengukur nilai kesamaan teks semantik pada pasangan ayat al-quran dengan penambahan sistem supervised learning. *eProceedings of Engineering*, 5(1), 2018.
- [15] Yuniar Purwanti, Uman Suherman, and Endang Dimyati. Analisis kebutuhan bahan ajar pada mata kuliah kecerdasan buatan sesuai pembelajaran berbasis proyek. *TEKNOLOGI PEMBELAJARAN*, 2(2), 2017.