Statistic for Data Science

Scikit-Learn

In [1]:
```
#Import libraries

import numpy as np
import pandas as pd
```

In [2]:
```
#import dataset iris
from sklearn.datasets import load_iris
```

In [3]:
```
#read data
iris = load_iris()
print(iris.data.shape)
```

```
(150, 4)
```

In [4]:
```
#jumlah baris dan kolumnya: 150, dan 4
```

In [5]:
```
#mengubah data frame agar lebih memudah untuk melakukan memproses data

dataset_iris = load_iris(as_frame =True)
```

In [6]:
```
dataset_iris.data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [7]:
```
#melihat description
dataset_iris.data.describe()
```

Out[7]:

	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

In [8]:
```
#random sampling
dataset_iris.data.sample(n=3, random_state=1)
```

Out[8]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
14	5.8	4.0	1.2	0.2
98	5.1	2.5	3.0	1.1
75	6.6	3.0	4.4	1.4

In [9]:
```
#representative sampling

import random

dataset_iris.data.sample(n=10, random_state = random.randint(0,5))
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
128	6.4	2.8	5.6	2.1
18	5.7	3.8	1.7	0.3
130	7.4	2.8	6.1	1.9
105	7.6	3.0	6.6	2.1
107	7.3	2.9	6.3	1.8
78	6.0	2.9	4.5	1.5
83	6.0	2.7	5.1	1.6
14	5.8	4.0	1.2	0.2
5	5.4	3.9	1.7	0.4
133	6.3	2.8	5.1	1.5

In [10]:
```
#representative sampling

#sample yang setiap barisnya adalah unik, artinya jika ada sample dengan nilai yang sama,
#sample tersebut akan ditimpa dengan ke sample yang lama
#sehingga jumlah sample akan menjadi lebih sedikit

dataset_iris.data.sample(n=10, replace=True, random_state=1)
```

Out[10]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
37	4.9	3.6	1.4	0.1
140	6.7	3.1	5.6	2.4
72	6.3	2.5	4.9	1.5
137	6.4	3.1	5.5	1.8
133	6.3	2.8	5.1	1.5
79	5.7	2.6	3.5	1.0
144	6.7	3.3	5.7	2.5
129	7.2	3.0	5.8	1.6
71	6.1	2.8	4.0	1.3
134	6.1	2.6	5.6	1.4

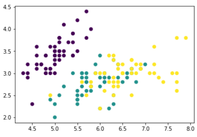
Melihat Penyebaran Data

In [11]:
```
#Import matlotlib

import matplotlib.pyplot as plt
```

In [12]:
```
#visualisasi

plt.scatter(dataset_iris.data['sepal length (cm)'], dataset_iris.data['sepal width (cm)'], c=dataset_iris.target)
```

```
<matplotlib.collections.PathCollection at 0x2381e9e6ac0>
```

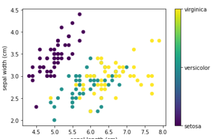
Out[12]:



In [13]:
```
#menambahkan funcformatter

formatter = plt.FuncFormatter(lambda i, *args: iris.target_names[int(i)])

plt.scatter(dataset_iris.data['sepal length (cm)'], dataset_iris.data['sepal width (cm)'], c=dataset_iris.target)
plt.colorbar(ticks=[0, 1, 2], format=formatter)
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
```

```
Text(0, 0.5, 'sepal width (cm)')
```

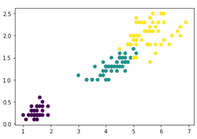
Out[13]:



In [14]:
```
#lakukan visualisasi terhadap Petal !!!

plt.scatter(dataset_iris.data['petal length (cm)'], dataset_iris.data['petal width (cm)'], c=dataset_iris.target)
```

```
<matplotlib.collections.PathCollection at 0x2381ebacc10>
```

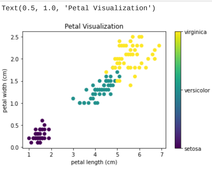
Out[14]:



In [15]:
```
#menambahkan funcformatter

formatter = plt.FuncFormatter(lambda i, *args: iris.target_names[int(i)])

plt.scatter(dataset_iris.data['petal length (cm)'], dataset_iris.data['petal width (cm)'], c=dataset_iris.target)
plt.colorbar(ticks=[0, 1, 2], format=formatter)
plt.xlabel('petal length (cm)')
plt.ylabel('petal width (cm)')
plt.title('Petal Visualization')
```

```
Text(0.5, 1.0, 'Petal Visualization')
```

Out[15]:



In [16]:
```
#testing data with Scikit-Learn

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(dataset_iris.data, dataset_iris.target, test_size=0.33, random_state=42)
```

Data Model metode Logistic Regression dan Evaluasi

In [17]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

In [18]:
```
model=LogisticRegression(solver='liblinear')
model.fit(X_train, y_train)
```

```
LogisticRegression(solver='liblinear')
```

In [19]:
```
model.score(X_train, y_train)
```

```
0.95
```

In [20]:
```
model.score(X_test, y_test)
```

```
1.0
```

Evaluation

In [21]:
```
#Test the model
predictions = model.predict(X_test)
print(predictions)# printing predictions

print()# Printing new Line

#Check precision, recall, f1-score
print( classification_report(y_test, predictions) )

print( accuracy_score(y_test, predictions))
```

```
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 1 1 0 0 1 2 2 1 2]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        15
           2       1.00      1.00      1.00        16

    accuracy                           1.00        50
   macro avg       1.00      1.00      1.00        50
weighted avg       1.00      1.00      1.00        50

1.0
```

In [22]:
```
!jupyter nbconvert --to html "./00000054804_Christopher Darren_Week5_LAB_TAMBAHAN_DIKIT.ipynb" --output-dir="./"
```

```
[NbConvertApp] Converting notebook ./00000054804_Christopher Darren_Week5.ipynb to html
[NbConvertApp] Writing 725478 bytes to 00000054804_Christopher Darren_Week5.html
```