



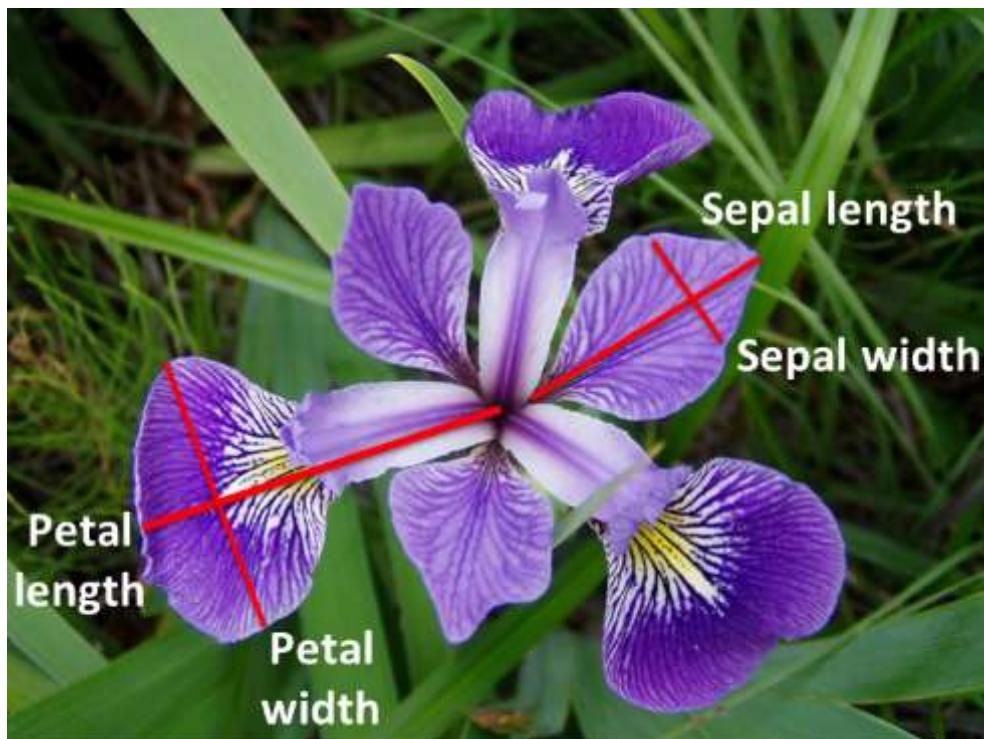
Hands-On

Hands-On ini digunakan pada kegiatan Microcredential Associate Data Scientist 2021

Tugas Mandiri Pertemuan 11

Pertemuan 11 (sebelas) pada Microcredential Associate Data Scientist 2021 menyampaikan materi mengenai Membangun Model 2 (Regresi Non Linier, Support Vector Machine, dll). silakan Anda kerjakan Latihan 1 s/d 20. Output yang anda lihat merupakan panduan yang dapat Anda ikuti dalam penulisan code :)

About Iris dataset



The iris dataset contains the following data (**Before Cleansing**)

- 50 samples of 3 different species of iris (150 samples total)
- Measurements: sepal length, sepal width, petal length, petal width
- The format for the data: (sepal length, sepal width, petal length, petal width)

The variables are:

- sepal_length: Sepal length, in centimeters, used as input.
- sepal_width: Sepal width, in centimeters, used as input.
- petal_length: Petal length, in centimeters, used as input.
- petal_width: Petal width, in centimeters, used as input.
- class: Iris Setosa, Versicolor, or Virginica, used as the target.

Contents

Data Preprocessing

- Include Libraries
- Import DataSet
- Handle Missing Value (*sudah dilakukan pada pert 8*)

Data Visualization

- Scatterplot
- Pairplot
- Barplot
- Violin
- Areaplot
- Correlation

Feature Engineering

Machine learning Model (Regresi Non Linier, Support Vector Machine, dll)

- Logistic Regression
- Naive Bayes
- KNN
- Support Vector Machine

1. Data Preprocessing

Latihan (1)

Melakukan import library yang dibutuhkan

```
In [120]: # import Library pandas
import pandas as pd
# Import Library numpy
import numpy as np

# Import Library matplotlib dan seaborn untuk visualisasi
import matplotlib.pyplot as plt
%matplotlib inline
from scipy import ndimage

# Import Module LinearRegression digunakan untuk memanggil algoritma Linear Regression
from sklearn.linear_model import LinearRegression

# import Module train_test_split digunakan untuk membagi data kita menjadi training dan testing
from sklearn.model_selection import train_test_split

# import modul mean_absolute_error dari Library sklearn
from sklearn.metrics import mean_absolute_error

# import math agar program dapat menggunakan semua fungsi yang ada pada modul math
import math
import seaborn as sns

from sklearn import preprocessing

# me-non aktifkan peringatan pada python
import warnings
warnings.filterwarnings("ignore")
from sklearn import datasets
```

Load Dataset

```
In [121]: #Panggil file (load file bernama Iris_AfterClean.csv.csv) dan simpan dalam dataframes
dataset =pd.read_csv('Iris_AfterClean.csv')
```

Latihan (2)

Review Dataset

```
In [122]: # tampilkan 5 baris awal dataset dengan function head()
dataset.head()
```

Out[122]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	4.6	3.1	1.5	0.2	Iris-setosa
1	5.0	3.6	1.4	0.2	Iris-setosa
2	5.4	3.9	1.7	0.4	Iris-setosa
3	4.9	3.1	1.5	0.1	Iris-setosa
4	5.4	3.7	1.5	0.2	Iris-setosa

```
In [123]: # tampilkan unique value dari species
dataset.Species.unique()
```

Out[123]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)

dari output diatas, dataset ini memiliki tiga varietas tanaman Iris.

```
In [124]: # melihat statistik data untuk data numeric dan non numeric
dataset.describe(include="all")
```

Out[124]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	140.000000	140.000000	140.000000	140.000000	140
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-virginica
freq	NaN	NaN	NaN	NaN	50
mean	5.902857	3.028571	3.910714	1.262857	NaN
std	0.819365	0.398791	1.720369	0.746825	NaN
min	4.300000	2.200000	1.000000	0.100000	NaN
25%	5.200000	2.800000	1.675000	0.400000	NaN
50%	5.850000	3.000000	4.500000	1.400000	NaN
75%	6.425000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.000000	6.900000	2.500000	NaN

```
In [125]: # Melihat Informasi Lebih detail mengenai struktur DataFrame dapat dilihat menggunakan dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140 entries, 0 to 139
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   SepalLengthCm  140 non-null   float64
 1   SepalWidthCm   140 non-null   float64
 2   PetalLengthCm  140 non-null   float64
 3   PetalWidthCm   140 non-null   float64
 4   Species        140 non-null   object  
dtypes: float64(4), object(1)
memory usage: 5.6+ KB
```

Seperti yang kita lihat di atas distribusi titik data di setiap kelas adalah sama sehingga Iris adalah dataset seimbang

Latihan (3)

Checking if there are any missing values



```
In [126]: # cek jumlah nilai yang hilang / missing values dari setiap kolom dengan function isna()
dataset = dataset.copy(deep=True)
dataset.isna().mean()
```

```
Out[126]: SepalLengthCm    0.0
SepalWidthCm     0.0
PetalLengthCm   0.0
PetalWidthCm    0.0
Species         0.0
dtype: float64
```

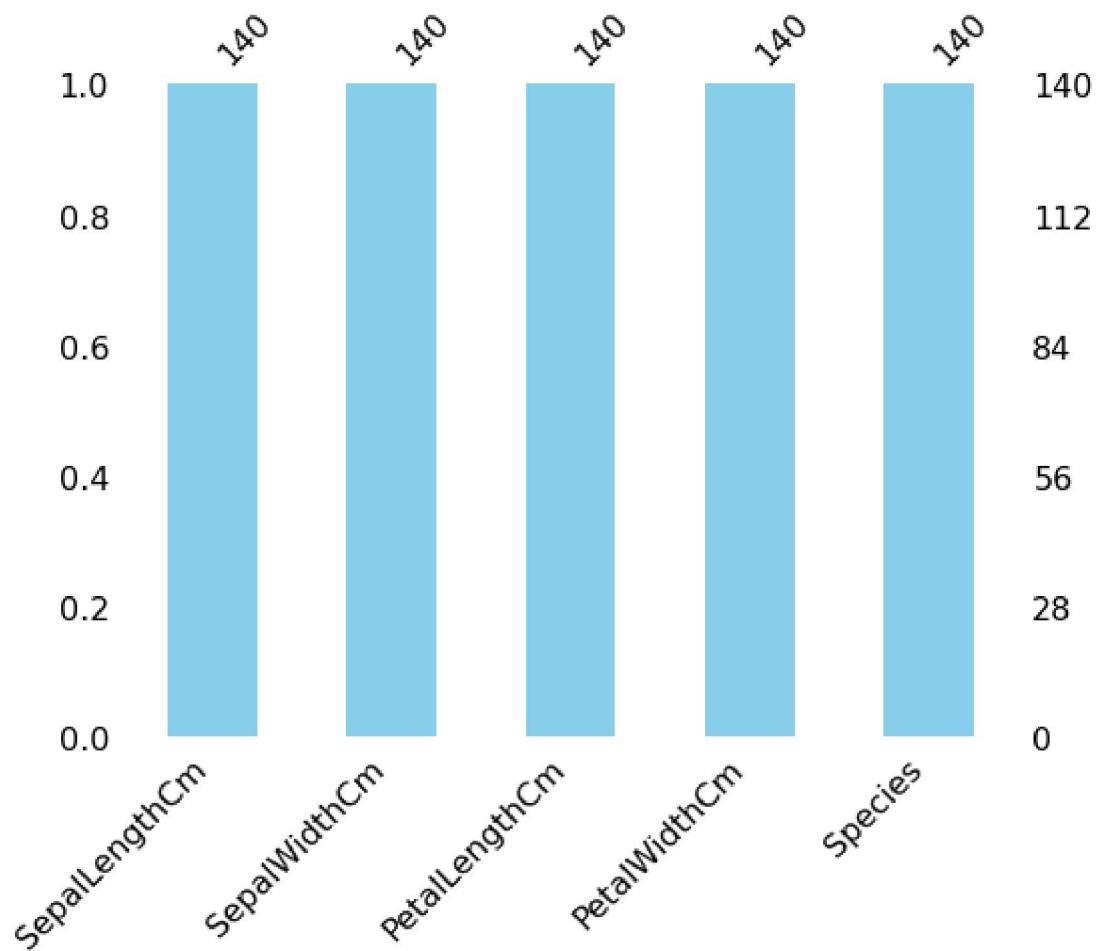
```
In [127]: conda install -c conda-forge/label/gcc7 missingno
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.
```

Note: you may need to restart the kernel to use updated packages.

```
In [128]: # cek missing values dengan visualisasi menggunakan library: Missingno adalah pustaka untuk mengetahui jumlah missing value pada dataset
# jenis: barchart
import missingno as msno
msno.bar(dataset,figsize=(8,6),color='skyblue')
plt.show()
```



Dataset IrisAfterclean.csv ini adalah dataset yang telah melewati proses cleansing pada pertemuan 8 kemarin sehingga dataset ini sudah bersih

2. Data Visualization

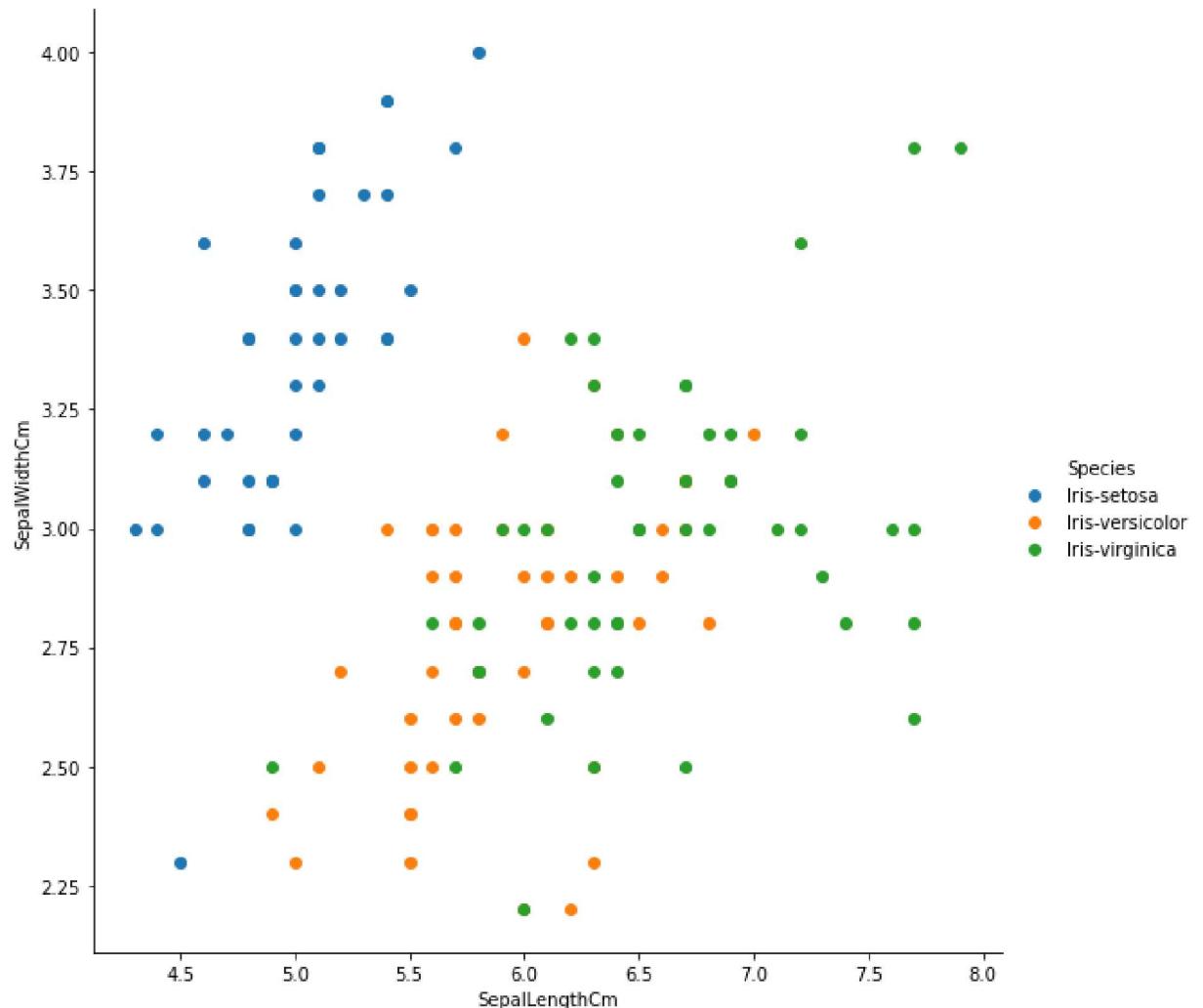
2.1 Scatter Plot

Scatter plot adalah visualisasi data dua dimensi yang menggunakan titik untuk mewakili nilai yang diperoleh untuk dua variabel berbeda, satu diplot sepanjang sumbu x dan yang lainnya diplot sepanjang sumbu y. Kita dapat memplot scatter plot di antara dua fitur.

Latihan (4)

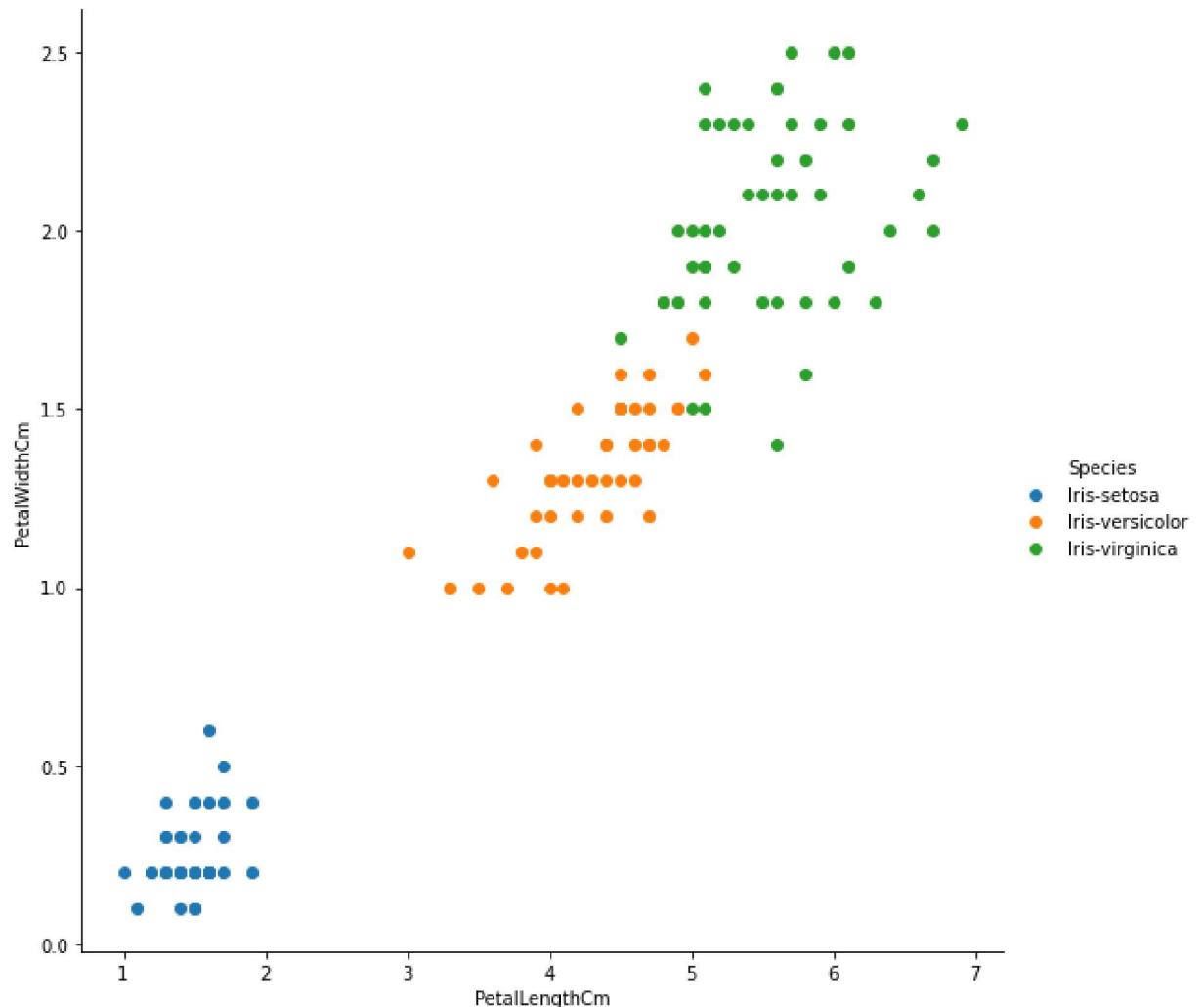
buat visualisasi scatter plot 'Sepal Length' dan 'Sepal Width'

```
In [129]: import matplotlib.pyplot as plt
%matplotlib inline
sns.FacetGrid(dataset,hue='Species',size=8)\n.map(plt.scatter,'SepalLengthCm','SepalWidthCm')\n.add_legend()\npass
```



buat visualisasi scatter plot 'Petal Length' dan 'Petal Width'

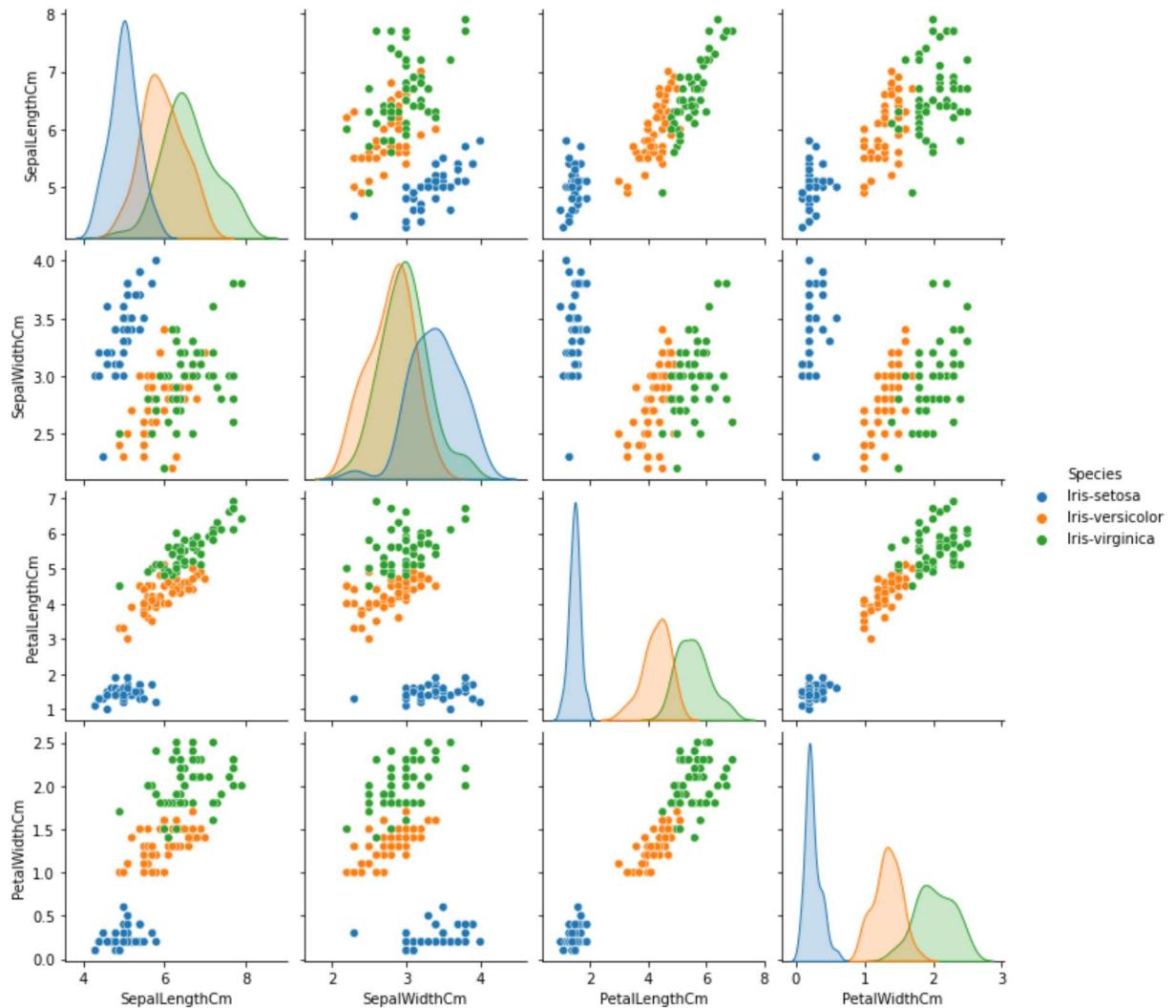
```
In [130]: import matplotlib.pyplot as plt
%matplotlib inline
sns.FacetGrid(dataset,hue='Species',size=8)\n.map(plt.scatter,'PetalLengthCm','PetalWidthCm')\n.add_legend()\npass
```



Latihan (5)

buat visualisasi Pair Plots dari data iris

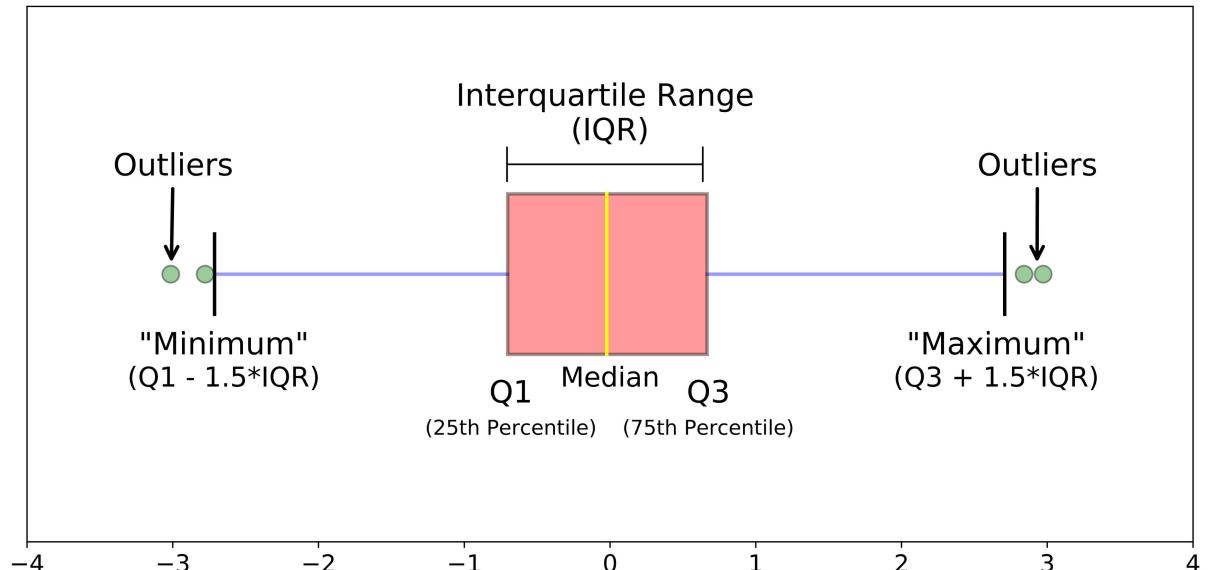
```
In [131]: # buat visualisasi Pair Plots dari data iris dengan parameter hue='species'  
sns.pairplot(dataset,hue='Species')  
pass
```



dari grafik kita dapat melihat scatter plot antara dua fitur dan distribusinya, dari sebaran di atas petal length memisahkan iris setosa dari yang tersisa, dari plot antara petal length dan petal width kita dapat memisahkan bunga

2.3 BoxPlot

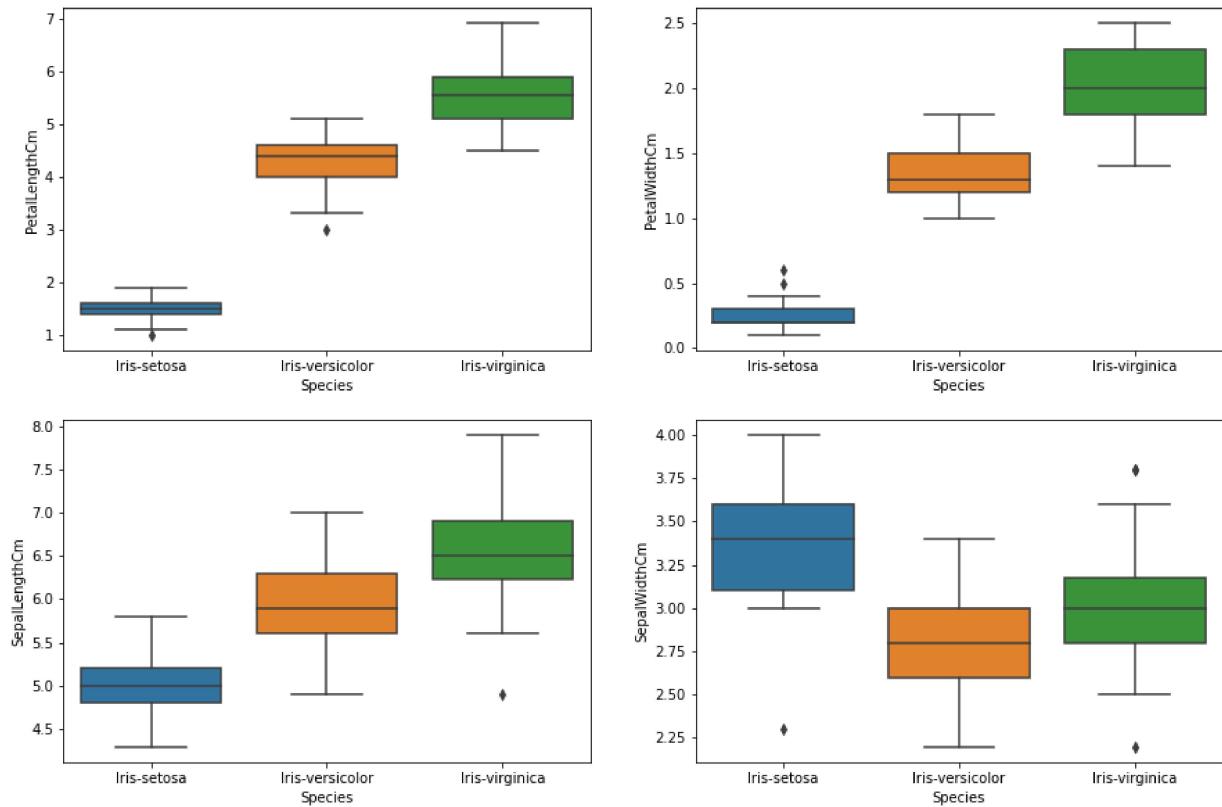
boxplot adalah cara standar untuk menampilkan distribusi data berdasarkan ringkasan lima angka ("minimum", kuartil pertama (Q1), median, kuartil ketiga (Q3), dan "maksimum"). Ini dapat memberi tahu kita tentang outlier dan apa nilainya. Ini juga dapat memberi tahu kita apakah data kita simetris, seberapa ketat data kita dikelompokkan, dan bagaimana jika data kita miring.



Latihan (6)

buat visualisasi box plot dari setiap kolom feature terhadap species

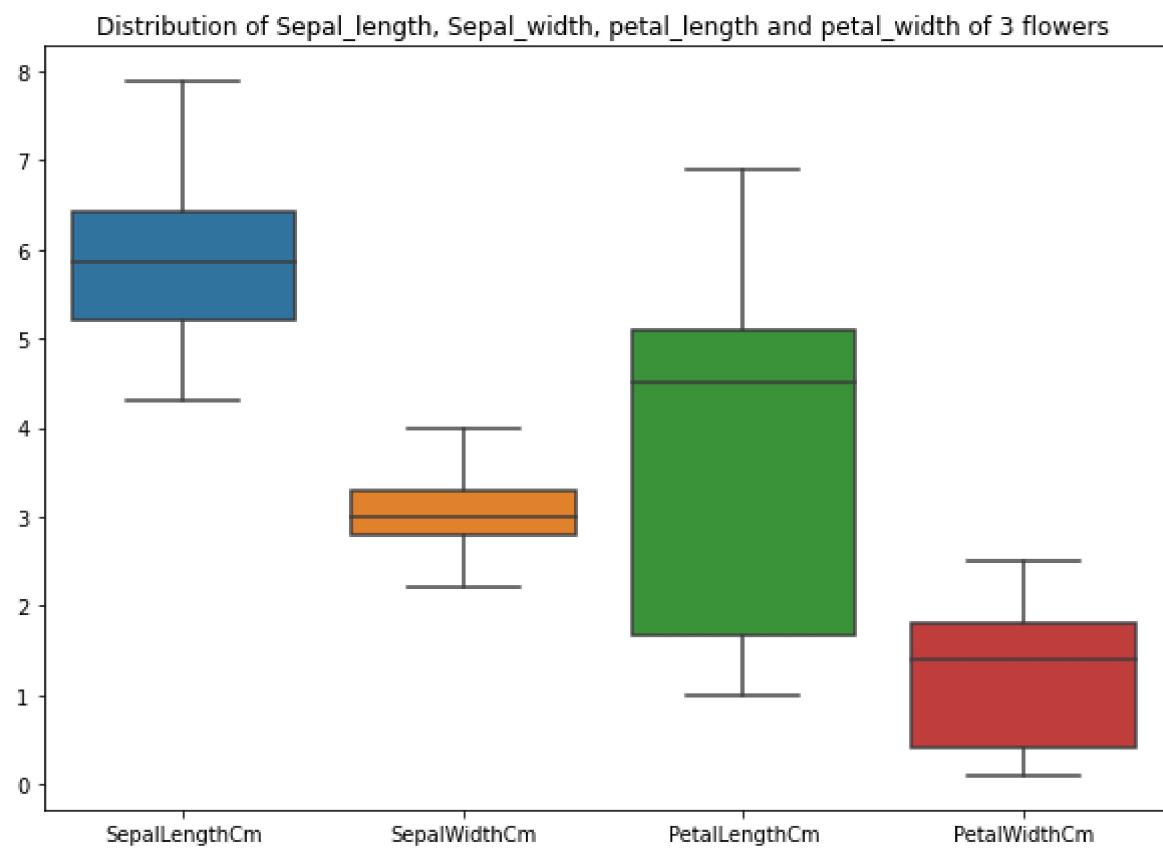
```
In [132]: plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.boxplot(x='Species',y='PetalLengthCm',data=dataset)
plt.subplot(2,2,2)
sns.boxplot(x='Species',y='PetalWidthCm',data=dataset)
plt.subplot(2,2,3)
sns.boxplot(x='Species',y='SepalLengthCm',data=dataset)
plt.subplot(2,2,4)
sns.boxplot(x='Species',y='SepalWidthCm',data=dataset)
plt.show()
```



Latihan (7)

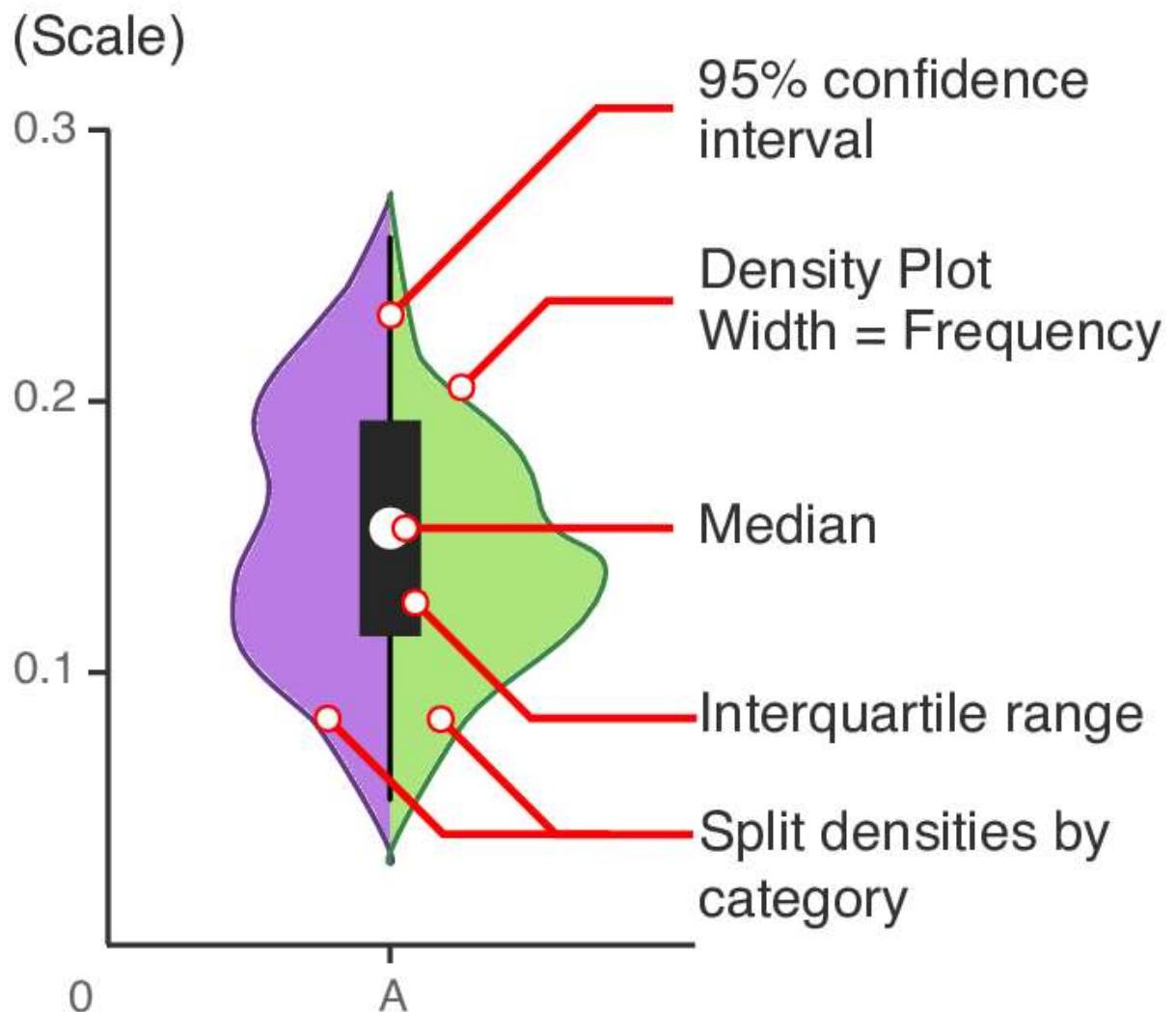
buat visualisasi box plot distribusi setiap kolom feature

```
In [133]: plt.subplots(figsize=(10,7))
sns.boxplot(data=dataset).set_title("Distribution of Sepal_length, Sepal_width, petal_length and petal_width of 3 flowers")
plt.show()
```



2.4 Violin

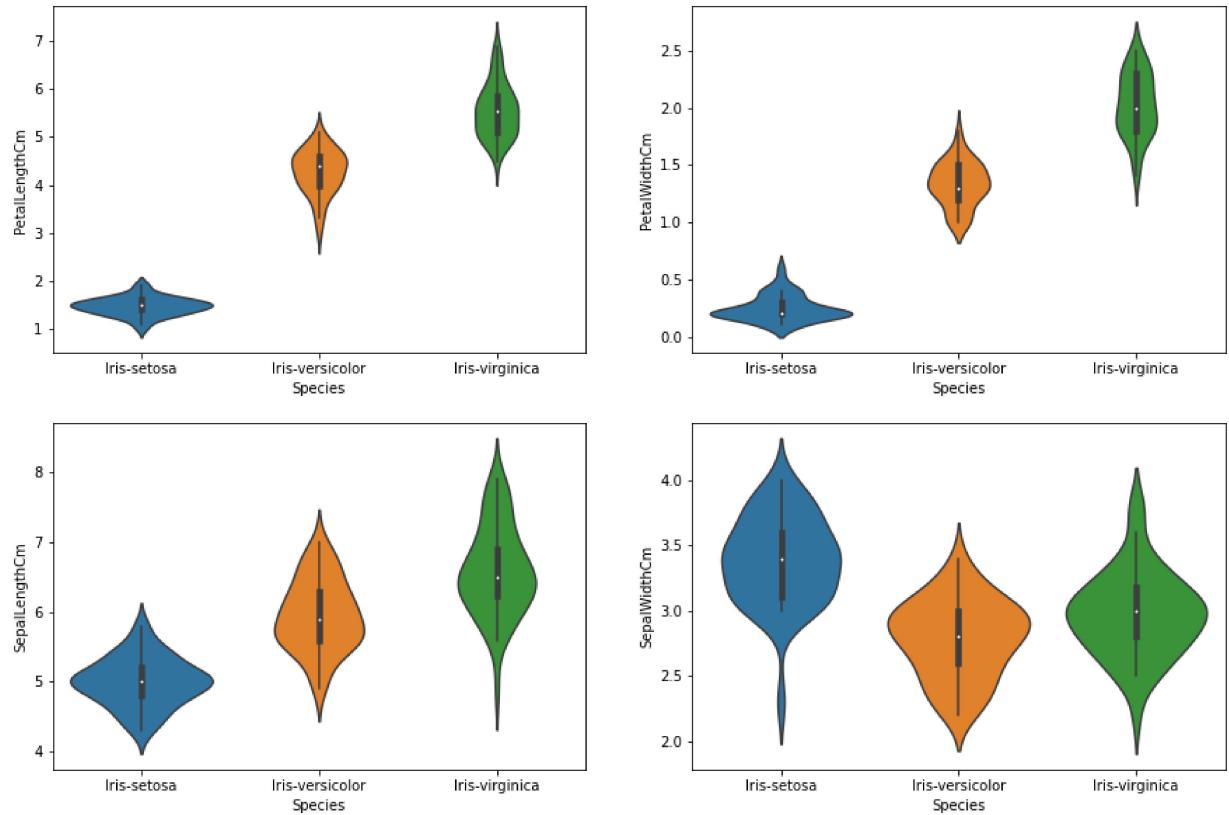
Violin Plot adalah metode untuk memvisualisasikan distribusi data numerik dari variabel yang berbeda. Ini mirip dengan Box Plot tetapi dengan plot yang diputar di setiap sisi, memberikan lebih banyak informasi tentang perkiraan kepadatan pada sumbu y. Kepadatan dicerminkan dan dibalik dan bentuk yang dihasilkan diisi, menciptakan gambar yang menyerupai biola. Kelebihan dari Violin Plot adalah dapat menampilkan nuansa dalam distribusi yang tidak terlihat dalam boxplot. Di sisi lain, boxplot lebih jelas menunjukkan outlier dalam data.



Latihan (8)

buat visualisasi violin plot setiap kolom feature

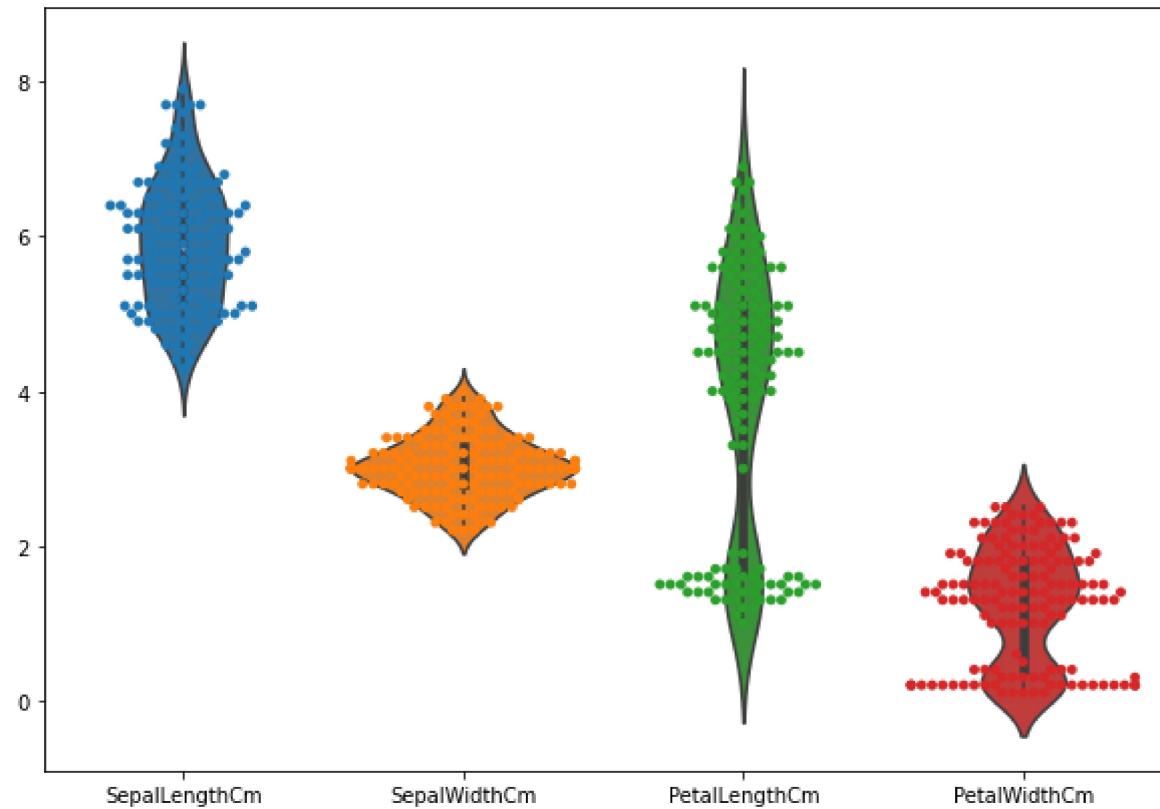
```
In [134]: plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=dataset)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidthCm',data=dataset)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=dataset)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=dataset)
pass
```



Latihan (9)

buat visualisasi violin plot dengan swarm plot

```
In [135]: plt.subplots(figsize=(10,7))
sns.violinplot(data=dataset)
sns.swarmplot(data=dataset)
pass
```



2.5 Area Plot

Area Plot memberi kita representasi visual dari Berbagai dimensi bunga Iris dan jangkauannya dalam dataset.

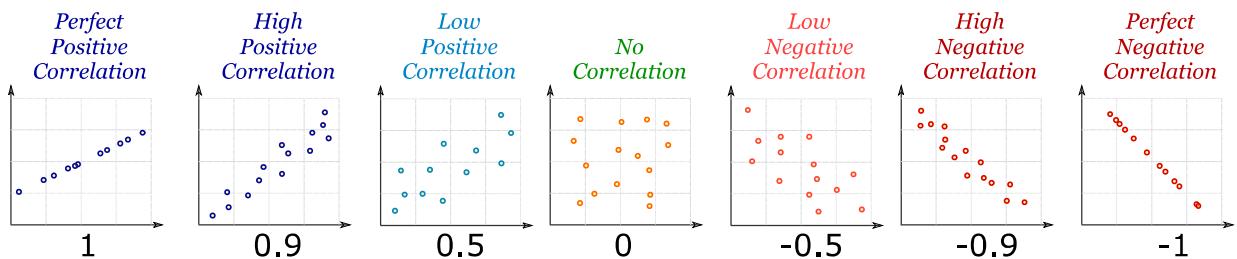
Latihan (10)

buat visualisasi area plot pada setiap feature kolom

```
In [136]: dataset.plot.area(y=[ 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'])
```



2.6 Correlation



Sekarang, ketika kami melatih algoritma apa pun, jumlah fitur dan korelasinya memainkan peran penting. Jika ada fitur dan banyak fitur yang sangat berkorelasi, maka melatih suatu algoritma dengan semua fitur akan mengurangi akurasi. Dengan demikian pemilihan fitur harus dilakukan dengan hati-hati. Dataset ini memiliki fitur yang lebih sedikit tetapi kita masih akan melihat korelasinya.

Latihan (11)

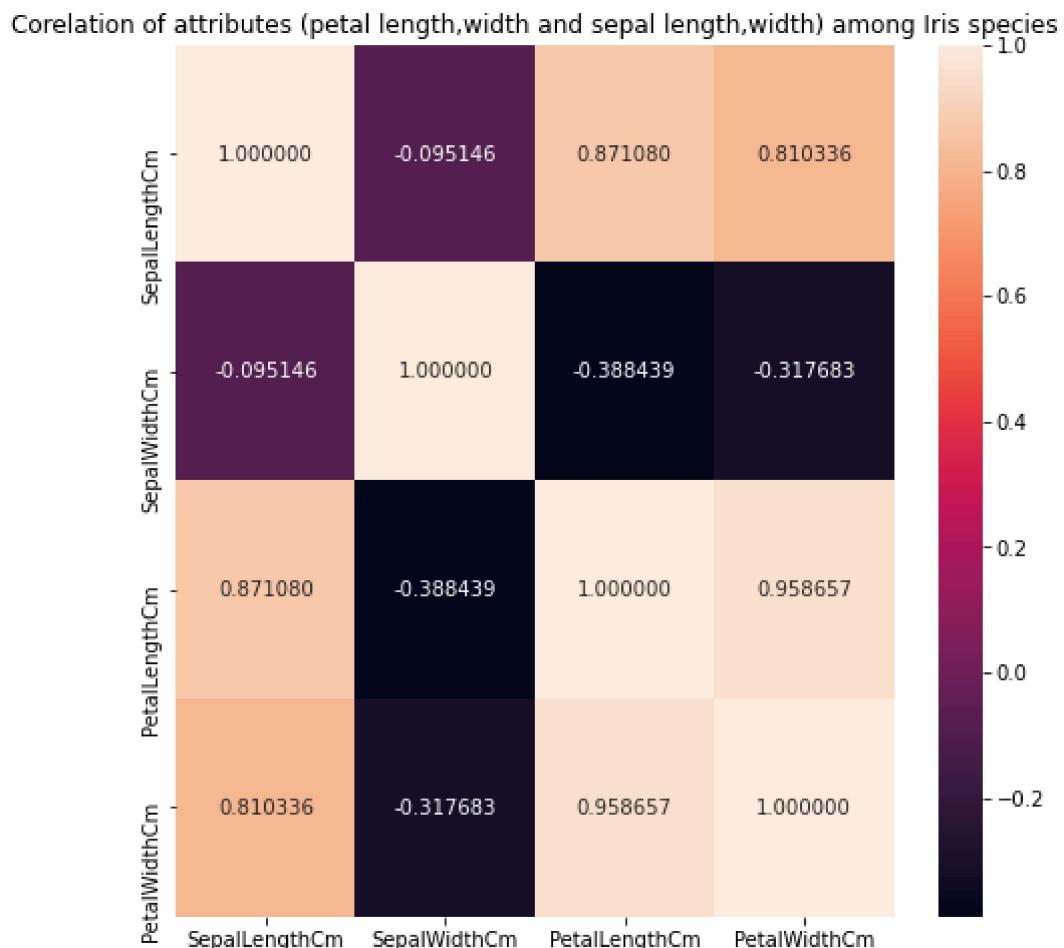
lihat korelasi dataset dan visualisasi dengan heatmap pada feature kolumn

```
In [137]: # Lihat korelasi dengan function corr()  
dataset.corr()
```

Out[137]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.095146	0.871080	0.810336
SepalWidthCm	-0.095146	1.000000	-0.388439	-0.317683
PetalLengthCm	0.871080	-0.388439	1.000000	0.958657
PetalWidthCm	0.810336	-0.317683	0.958657	1.000000

```
In [138]: # Lihat korelasi dengan visualisasi heatmap  
plt.figure(figsize = (8,8))  
sns.heatmap(dataset.corr(), annot=True, fmt="f").set_title("Corelation of attributes")  
plt.show()
```



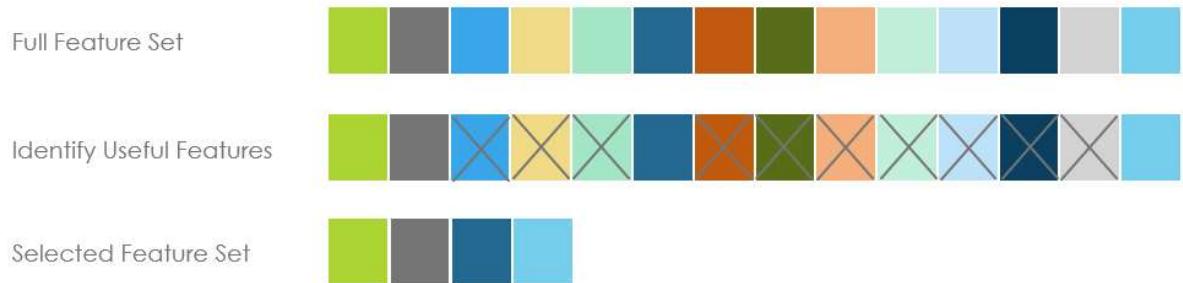
Observasi :

Sepal Width dan Sepal Length tidak berkorelasi || Petal Width and Petal Length sangat berkorelasi

Kami akan menggunakan semua fitur untuk melatih algoritme dan memeriksa keakuratannya.

Dividing data into features and labels

Feature Selection



Seperti yang kita lihat, dataset berisi lima kolom: SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm dan Species. Fitur yang sebenarnya dijelaskan oleh kolom 1-4. Kolom terakhir berisi label sampel. Pertama kita perlu membagi data menjadi dua array: X (fitur) dan y (label).

Latihan (12)

definisi variabel X(feature kolom) dan y(species/label):

```
In [139]: dataset.columns
```

```
Out[139]: Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

```
In [140]: X = dataset[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']] .  
X[0:5]
```

```
Out[140]: array([[4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2]])
```

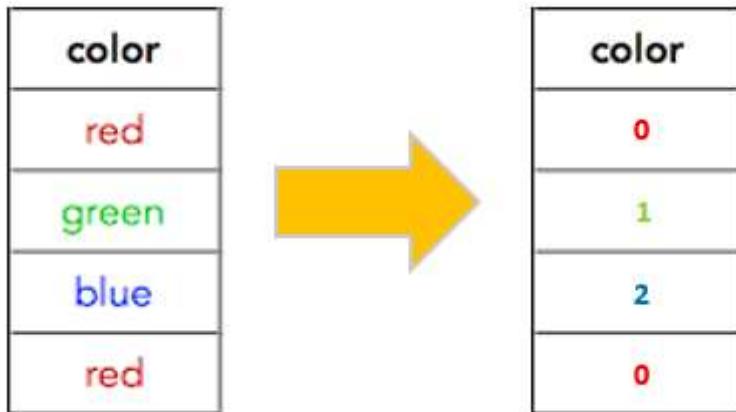
```
In [151]: y = dataset['Species']  
y[0:4]
```

```
Out[151]: 0    Iris-setosa  
1    Iris-setosa  
2    Iris-setosa  
3    Iris-setosa  
Name: Species, dtype: object
```

```
In [142]: X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X[0:5]
```

```
Out[142]: array([[-1.59579136,  0.17975613, -1.40630921, -1.42827825],
 [-1.10585542,  1.43804903, -1.464645 , -1.42827825],
 [-0.61591947,  2.19302477, -1.28963763, -1.15951622],
 [-1.22833941,  0.17975613, -1.40630921, -1.56265927],
 [-0.61591947,  1.68970761, -1.40630921, -1.42827825]])
```

Label encoding



Seperti yang kita lihat, label bersifat kategoris. KNeighborsClassifier tidak menerima label string. Kita perlu menggunakan LabelEncoder untuk mengubahnya menjadi angka. Iris-setosa sesuai dengan 0, Iris-versicolor sesuai dengan 1 dan Iris-virginica sesuai dengan 2.

Latihan (13)

transform label data species dengan menggunakan library LabelEncoder

```
In [152]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

3. Building Machine Learning Models

Latihan (14)

import library dalam kebutuhan membangun model

In [184]:

```
#Metrics
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import classification_report

# Import Libaray confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score

#Model Select
from sklearn.model_selection import KFold,train_test_split,cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Import Libaray Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier

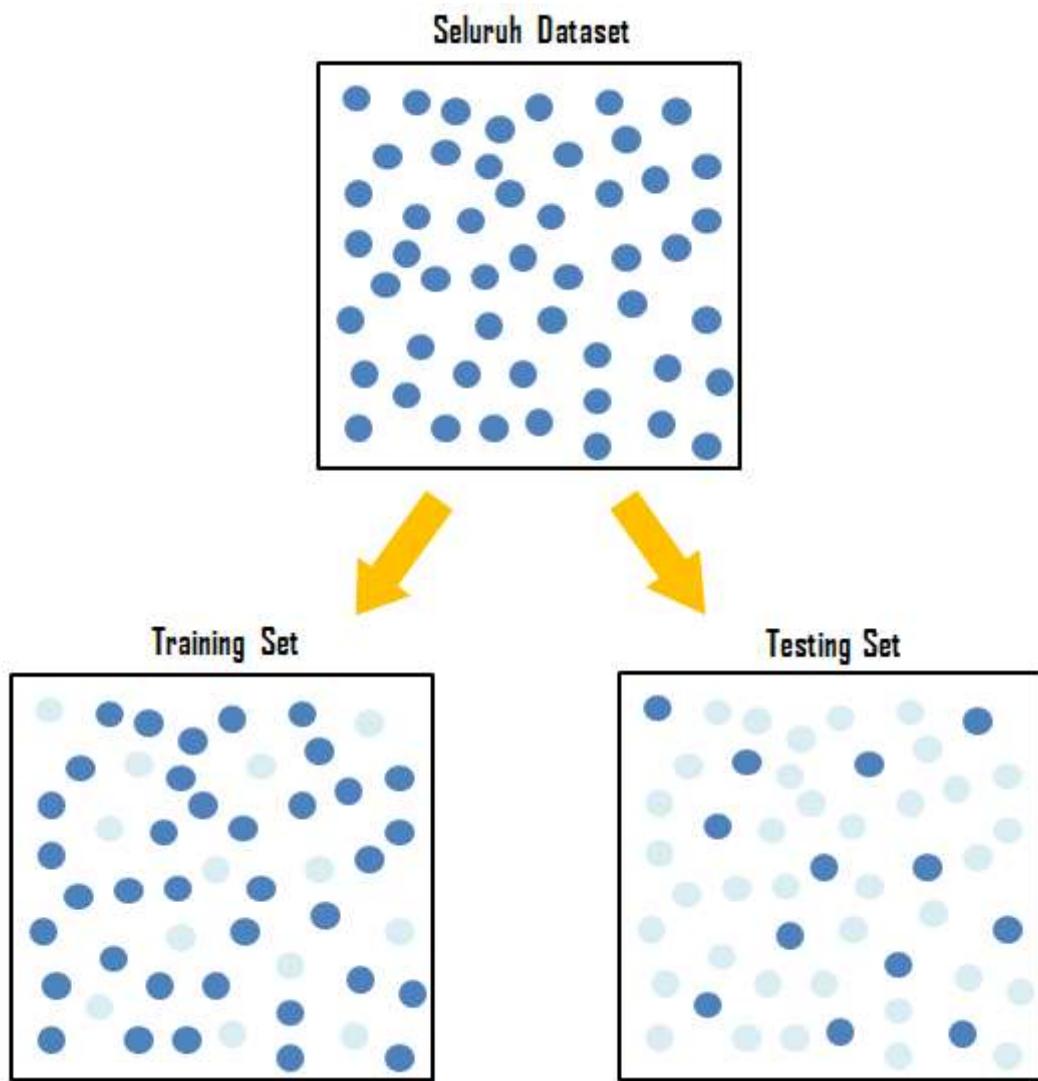
# Import Libaray KNN
from sklearn.neighbors import KNeighborsClassifier

# Import Libaray Support Vector Machines dan Linier Support Vector Machines
from sklearn.svm import SVC, LinearSVC

# Import Libaray Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
```

Splitting The Data into Training And Testing Dataset

Train/test split adalah salah satu metode yang dapat digunakan untuk mengevaluasi performa model machine learning. Metode evaluasi model ini membagi dataset menjadi dua bagian yakni bagian yang digunakan untuk training data dan untuk testing data dengan proporsi tertentu. Train data digunakan untuk fit model machine learning, sedangkan test data digunakan untuk mengevaluasi hasil fit model tersebut.



Python memiliki library yang dapat mengimplementasikan train/test split dengan mudah yaitu Scikit-Learn. Untuk menggunakannya, kita perlu mengimport Scikit-Learn terlebih dahulu, kemudian setelah itu kita dapat menggunakan fungsi `train_test_split()`.

Latihan (15)

split data train dan test dengan function `train_test_split()` dengan `train_size=0.7, test_size=0.3` dan `random_state=0`

```
In [185]: 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_s
```

In [186]: X

```
Out[186]: array([[-1.59579136,  0.17975613, -1.40630921, -1.42827825],
 [-1.10585542,  1.43804903, -1.464645 , -1.42827825],
 [-0.61591947,  2.19302477, -1.28963763, -1.15951622],
 [-1.22833941,  0.17975613, -1.40630921, -1.56265927],
 [-0.61591947,  1.68970761, -1.40630921, -1.42827825],
 [-1.35082339,  0.93473187, -1.34797342, -1.42827825],
 [-1.35082339, -0.07190245, -1.464645 , -1.56265927],
 [-1.96324332, -0.07190245, -1.63965237, -1.56265927],
 [-0.12598353,  2.44468335, -1.58131658, -1.42827825],
 [-0.61591947,  2.19302477, -1.52298079, -1.15951622],
 [-0.98337143,  1.18639045, -1.464645 , -1.29389724],
 [-0.24846751,  1.94136619, -1.28963763, -1.29389724],
 [-0.98337143,  1.94136619, -1.40630921, -1.29389724],
 [-0.61591947,  0.93473187, -1.28963763, -1.42827825],
 [-0.98337143,  1.68970761, -1.40630921, -1.15951622],
 [-1.59579136,  1.43804903, -1.69798815, -1.42827825],
 [-0.98337143,  0.68307329, -1.28963763, -1.0251352 ],
 [-1.35082339,  0.93473187, -1.17296605, -1.42827825],
 [-1.10585542, -0.07190245, -1.34797342, -1.42827825],
 [-1.10585542,  0.93473187, -1.34797342, -1.15951622],
 [-0.86088745,  1.18639045, -1.40630921, -1.42827825],
 [-0.86088745,  0.93473187, -1.464645 , -1.42827825],
 [-1.47330738,  0.43141471, -1.34797342, -1.42827825],
 [-1.35082339,  0.17975613, -1.34797342, -1.42827825],
 [-0.61591947,  0.93473187, -1.40630921, -1.15951622],
 [-1.22833941,  0.17975613, -1.40630921, -1.56265927],
 [-1.10585542,  0.43141471, -1.58131658, -1.42827825],
 [-0.49343549,  1.18639045, -1.52298079, -1.42827825],
 [-1.22833941,  0.17975613, -1.40630921, -1.56265927],
 [-1.84075934, -0.07190245, -1.52298079, -1.42827825],
 [-0.98337143,  0.93473187, -1.40630921, -1.42827825],
 [-1.10585542,  1.18639045, -1.52298079, -1.29389724],
 [-1.71827535, -1.83351251, -1.52298079, -1.29389724],
 [-1.84075934,  0.43141471, -1.52298079, -1.42827825],
 [-1.10585542,  1.18639045, -1.34797342, -0.89075418],
 [-0.98337143,  1.94136619, -1.17296605, -1.15951622],
 [-1.35082339, -0.07190245, -1.464645 , -1.29389724],
 [-0.98337143,  1.94136619, -1.34797342, -1.42827825],
 [-1.59579136,  0.43141471, -1.464645 , -1.42827825],
 [-0.73840346,  1.68970761, -1.40630921, -1.42827825],
 [-1.10585542,  0.68307329, -1.464645 , -1.42827825],
 [ 1.34382431,  0.43141471,  0.46043605,  0.18429397],
 [ 0.60892039,  0.43141471,  0.34376447,  0.31867499],
 [ 1.22134032,  0.17975613,  0.57710763,  0.31867499],
 [-0.49343549, -1.83351251,  0.05208553,  0.04991295],
 [ 0.73140438, -0.57521961,  0.40210026,  0.31867499],
 [-0.24846751, -0.57521961,  0.34376447,  0.04991295],
 [ 0.4864364 ,  0.68307329,  0.46043605,  0.45305601],
 [-1.22833941, -1.58185393, -0.356265 , -0.35323011],
 [ 0.85388836, -0.32356103,  0.40210026,  0.04991295],
 [-0.86088745, -0.82687819, -0.00625026,  0.18429397],
 [-0.00349954, -0.07190245,  0.1687571 ,  0.31867499],
 [ 0.11898444, -2.08517109,  0.05208553, -0.35323011],
 [ 0.24146843, -0.32356103,  0.46043605,  0.18429397],
```

[-0.3709515 , -0.32356103, -0.18125763, 0.04991295],
 [0.97637235, 0.17975613, 0.28542868, 0.18429397],
 [-0.3709515 , -0.07190245, 0.34376447, 0.31867499],
 [-0.12598353, -0.82687819, 0.11042132, -0.35323011],
 [0.36395242, -2.08517109, 0.34376447, 0.31867499],
 [-0.3709515 , -1.33019535, -0.00625026, -0.21884909],
 [-0.00349954, 0.43141471, 0.51877184, 0.72181804],
 [0.24146843, -0.57521961, 0.05208553, 0.04991295],
 [0.4864364 , -1.33019535, 0.57710763, 0.31867499],
 [0.24146843, -0.57521961, 0.46043605, -0.08446807],
 [0.60892039, -0.32356103, 0.22709289, 0.04991295],
 [0.85388836, -0.07190245, 0.28542868, 0.18429397],
 [1.09885633, -0.57521961, 0.51877184, 0.18429397],
 [0.97637235, -0.07190245, 0.63544342, 0.58743702],
 [0.11898444, -0.32356103, 0.34376447, 0.31867499],
 [-0.24846751, -1.07853677, -0.23959342, -0.35323011],
 [-0.49343549, -1.58185393, -0.06458605, -0.21884909],
 [-0.49343549, -1.58185393, -0.12292184, -0.35323011],
 [-0.12598353, -0.82687819, -0.00625026, -0.08446807],
 [0.11898444, -0.82687819, 0.69377921, 0.45305601],
 [-0.61591947, -0.07190245, 0.34376447, 0.31867499],
 [0.11898444, 0.93473187, 0.34376447, 0.45305601],
 [0.97637235, 0.17975613, 0.46043605, 0.31867499],
 [0.4864364 , -1.83351251, 0.28542868, 0.04991295],
 [-0.3709515 , -0.07190245, 0.11042132, 0.04991295],
 [-0.49343549, -1.33019535, 0.05208553, 0.04991295],
 [-0.49343549, -1.07853677, 0.28542868, -0.08446807],
 [0.24146843, -0.07190245, 0.40210026, 0.18429397],
 [-0.12598353, -1.07853677, 0.05208553, -0.08446807],
 [-1.10585542, -1.83351251, -0.356265 , -0.35323011],
 [-0.3709515 , -0.82687819, 0.1687571 , 0.04991295],
 [-0.24846751, -0.07190245, 0.1687571 , -0.08446807],
 [-0.24846751, -0.32356103, 0.1687571 , 0.04991295],
 [0.36395242, -0.32356103, 0.22709289, 0.04991295],
 [-0.98337143, -1.33019535, -0.53127237, -0.21884909],
 [-0.24846751, -0.57521961, 0.11042132, 0.04991295],
 [0.4864364 , 0.68307329, 1.21880131, 1.66248517],
 [-0.12598353, -0.82687819, 0.69377921, 0.85619906],
 [1.46630829, -0.07190245, 1.16046552, 1.1249611],
 [0.4864364 , -0.32356103, 0.98545816, 0.72181804],
 [0.73140438, -0.07190245, 1.10212973, 1.25934212],
 [2.07872822, -0.07190245, 1.56881605, 1.1249611],
 [-1.22833941, -1.33019535, 0.34376447, 0.58743702],
 [1.71127627, -0.32356103, 1.39380868, 0.72181804],
 [0.97637235, -1.33019535, 1.10212973, 0.72181804],
 [1.58879228, 1.43804903, 1.2771371 , 1.66248517],
 [0.73140438, 0.43141471, 0.69377921, 0.99058008],
 [0.60892039, -0.82687819, 0.81045079, 0.85619906],
 [1.09885633, -0.07190245, 0.92712237, 1.1249611],
 [-0.24846751, -1.33019535, 0.63544342, 0.99058008],
 [-0.12598353, -0.57521961, 0.69377921, 1.52810415],
 [0.60892039, 0.43141471, 0.81045079, 1.39372314],
 [0.73140438, -0.07190245, 0.92712237, 0.72181804],
 [2.20121221, 1.94136619, 1.62715184, 1.25934212],
 [2.20121221, -1.07853677, 1.74382342, 1.39372314],
 [0.11898444, -2.08517109, 0.63544342, 0.31867499],
 [1.22134032, 0.43141471, 1.04379395, 1.39372314],

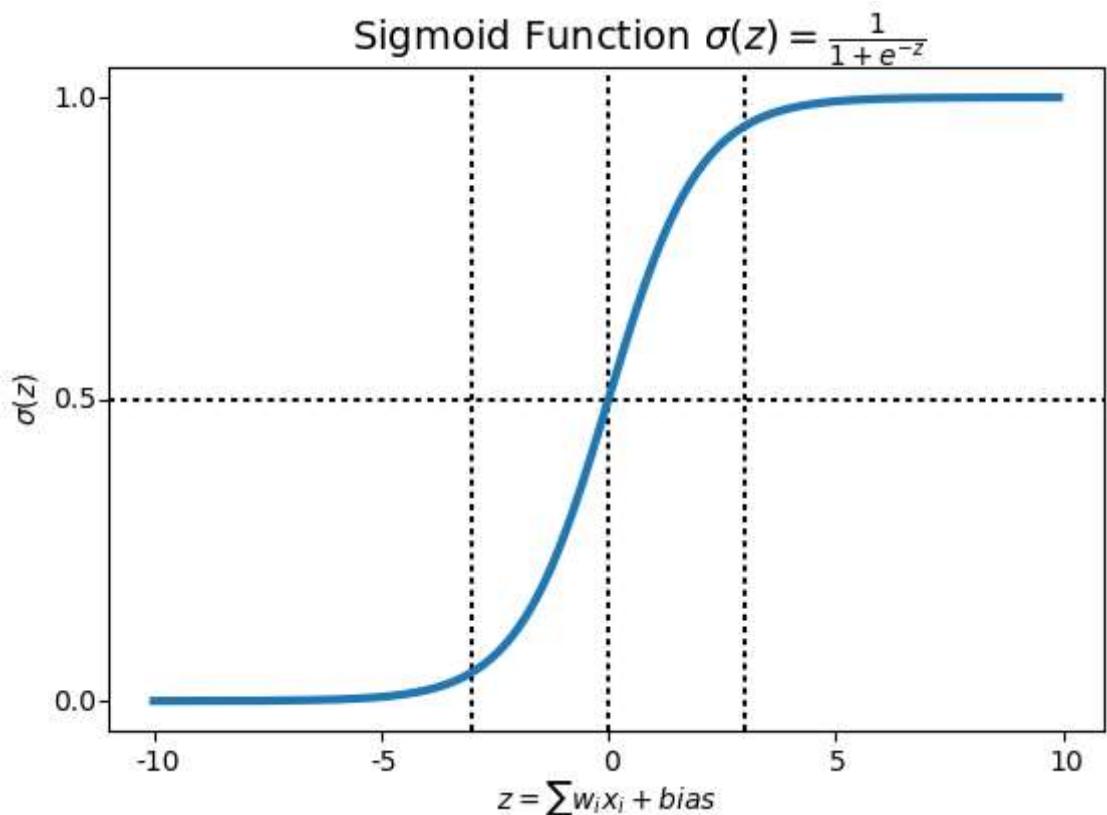
```
[ -0.3709515 , -0.57521961,  0.57710763,  0.99058008],  
[ 2.20121221, -0.57521961,  1.62715184,  0.99058008],  
[ 0.4864364 , -0.82687819,  0.57710763,  0.72181804],  
[ 0.97637235,  0.68307329,  1.04379395,  1.1249611 ],  
[ 1.58879228,  0.43141471,  1.21880131,  0.72181804],  
[ 0.36395242, -0.57521961,  0.51877184,  0.72181804],  
[ 0.24146843, -0.07190245,  0.57710763,  0.72181804],  
[ 0.60892039, -0.57521961,  0.98545816,  1.1249611 ],  
[ 1.58879228, -0.07190245,  1.10212973,  0.45305601],  
[ 1.83376025, -0.57521961,  1.2771371 ,  0.85619906],  
[ 2.44618018,  1.94136619,  1.45214447,  0.99058008],  
[ 0.60892039, -0.57521961,  0.98545816,  1.25934212],  
[ 0.4864364 , -0.57521961,  0.69377921,  0.31867499],  
[ 0.24146843, -1.07853677,  0.98545816,  0.18429397],  
[ 2.20121221, -0.07190245,  1.2771371 ,  1.39372314],  
[ 0.4864364 ,  0.93473187,  0.98545816,  1.52810415],  
[ 0.60892039,  0.17975613,  0.92712237,  0.72181804],  
[ 0.11898444, -0.07190245,  0.51877184,  0.72181804],  
[ 1.22134032,  0.17975613,  0.86878658,  1.1249611 ],  
[ 0.97637235,  0.17975613,  0.98545816,  1.52810415],  
[ 1.22134032,  0.17975613,  0.69377921,  1.39372314],  
[ -0.12598353, -0.82687819,  0.69377921,  0.85619906],  
[ 1.09885633,  0.43141471,  1.16046552,  1.39372314],  
[ 0.97637235,  0.68307329,  1.04379395,  1.66248517],  
[ 0.97637235, -0.07190245,  0.752115 ,  1.39372314],  
[ 0.4864364 , -1.33019535,  0.63544342,  0.85619906],  
[ 0.73140438, -0.07190245,  0.752115 ,  0.99058008],  
[ 0.36395242,  0.93473187,  0.86878658,  1.39372314],  
[ -0.00349954, -0.07190245,  0.69377921,  0.72181804]])
```

Sekarang kita akan melatih beberapa model Machine Learning dan membandingkan hasilnya. Perhatikan bahwa karena set data tidak memberikan label untuk set pengujinya, kita perlu menggunakan prediksi pada set pelatihan untuk membandingkan algoritme satu sama lain.

3.2 Logistic Regression:

Logistic Regression adalah algoritma Machine Learning yang digunakan untuk masalah klasifikasi, ini adalah algoritma analisis prediktif dan berdasarkan konsep probabilitas.

Kita dapat menyebut Logistic Regression sebagai model Regresi Linier tetapi Regresi Logistik menggunakan fungsi biaya yang lebih kompleks, fungsi biaya ini dapat didefinisikan sebagai 'fungsi Sigmoid' atau juga dikenal sebagai 'fungsi logistik' daripada fungsi linier.



Bangun model LogisticRegression dan akurasi nya

```
In [187]: logreg = LogisticRegression(solver= 'lbfgs',max_iter=400)
logreg.fit(X_train, y_train)
Y_pred = logreg.predict(X_test)
accuracy_lr=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_log = round(logreg.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test,Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Logistic Regression\n',cm)
print('accuracy_Logistic Regression : %.3f' %accuracy)
print('precision_Logistic Regression : %.3f' %precision)
print('recall_Logistic Regression: %.3f' %recall )
print('f1-score_Logistic Regression : %.3f' %f1)
```

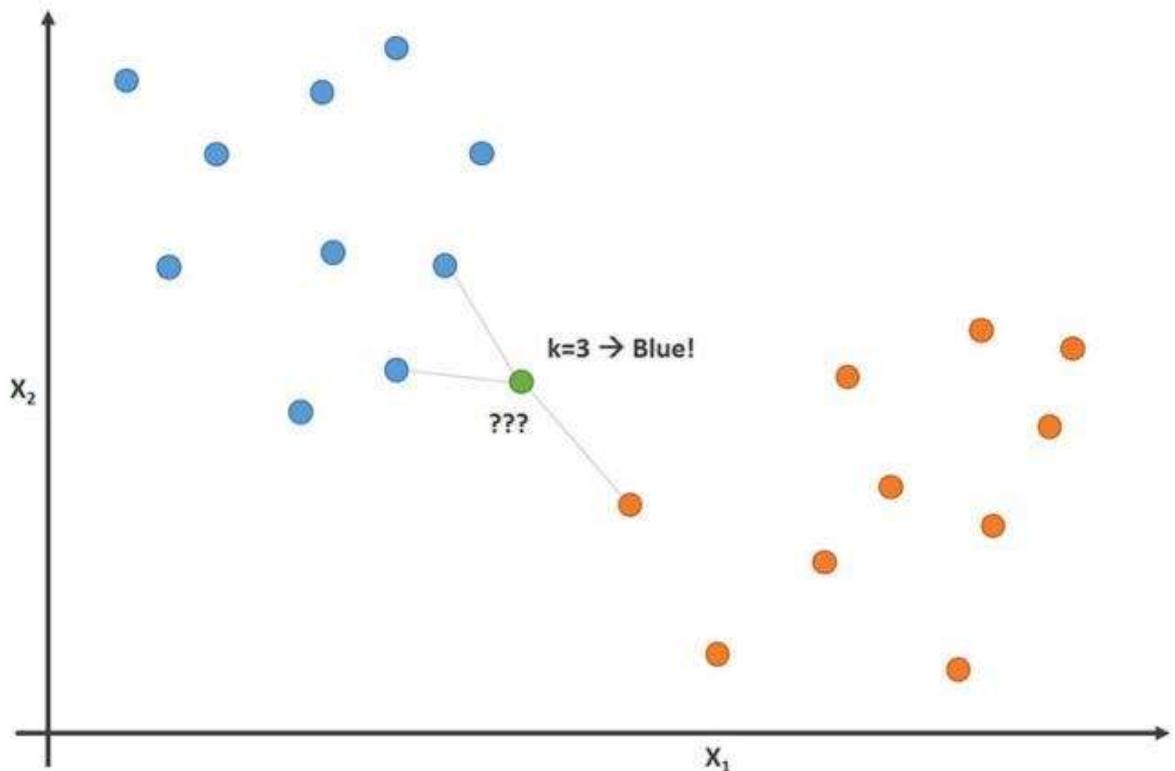
```
Confusion matrix for Logistic Regression
[[12  0  0]
 [ 0 14  1]
 [ 0  2 13]]
accuracy_Logistic Regression : 0.929
precision_Logistic Regression : 0.929
recall_Logistic Regression: 0.929
f1-score_Logistic Regression : 0.929
```

3.3 K Nearest Neighbor:

K-Nearest Neighbor adalah salah satu algoritma Machine Learning yang paling sederhana berdasarkan teknik Supervised Learning.

Algoritma K-NN mengasumsikan kesamaan antara kasus/data baru dengan kasus yang tersedia dan memasukkan kasus baru ke dalam kategori yang paling mirip dengan kategori yang tersedia.

Algoritma K-NN menyimpan semua data yang tersedia dan mengklasifikasikan titik data baru berdasarkan kesamaan. Artinya ketika data baru muncul maka dapat dengan mudah diklasifikasikan ke dalam kategori well suite dengan menggunakan algoritma K-NN.



Latihan (16)

Bangun model KNN dan akurasi nya

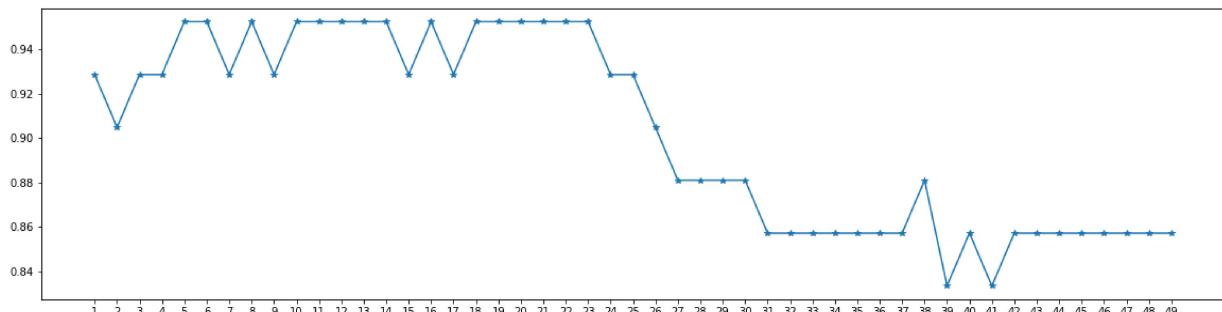
```
In [188]: knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
Y_pred = knn.predict(X_test)
accuracy_knn=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_knn = round(knn.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test,Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 =f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for KNN\n',cm)
print('accuracy_KNN : %.3f' %accuracy)
print('precision_KNN : %.3f' %precision)
print('recall_KNN: %.3f' %recall)
print('f1-score_KNN : %.3f' %f1)
```

```
Confusion matrix for KNN
[[12  0  0]
 [ 0 14  1]
 [ 0  2 13]]
accuracy_KNN : 0.929
precision_KNN : 0.929
recall_KNN: 0.929
f1-score_KNN : 0.929
```

Mari kita periksa akurasi untuk berbagai nilai n untuk Model KNN

```
In [189]: plt.subplots(figsize=(20,5))
a_index=list(range(1,50))
a=pd.Series()
x=range(1,50)
#x=[1,2,3,4,5,6,7,8,9,10]
for i in list(range(1,50)):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(X_train, y_train)
    prediction=model.predict(X_test)
    a=a.append(pd.Series(accuracy_score(y_test,prediction)))
plt.plot(a_index, a,marker="*")
plt.xticks(x)
plt.show()
```



Di atas adalah grafik yang menunjukkan akurasi untuk model KNN menggunakan nilai n yang

berbeda.

3.4 Gaussian Naive Bayes:

Naive Bayes adalah algoritma klasifikasi untuk masalah klasifikasi biner (dua kelas) dan multi kelas. Teknik ini paling mudah dipahami ketika dijelaskan menggunakan nilai input biner atau kategoris.

Disebut naive bayes atau idiot bayes karena perhitungan probabilitas untuk setiap hipotesis disederhanakan untuk membuat perhitungannya dapat dilakukan. Daripada mencoba menghitung nilai dari setiap nilai atribut $P(d1, d2, d3|h)$, mereka diasumsikan independen bersyarat dengan nilai target dan dihitung sebagai $P(d1|h) * P(d2|H)$ dan seterusnya.

Ini adalah asumsi yang sangat kuat yang paling tidak mungkin dalam data nyata, yaitu bahwa atribut tidak berinteraksi. Namun demikian, pendekatan ini bekerja dengan sangat baik pada data di mana asumsi ini tidak berlaku.

GAUSSIAN
NAIVE BAYES
CLASSIFIER

"Gaussian" because this is a normal distribution

This is our prior belief

We don't calculate this in naive bayes classifiers

$$P(\text{class} \mid \text{data}) = \frac{P(\text{data} \mid \text{class}) \times p(\text{class})}{P(\text{data})}$$

ChrisAlbon

Latihan (17)

Bangun model gaussian Naive Bayes dan akurasi nya

```
In [190]: gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_gaussian = round(gaussian.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 =f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
```

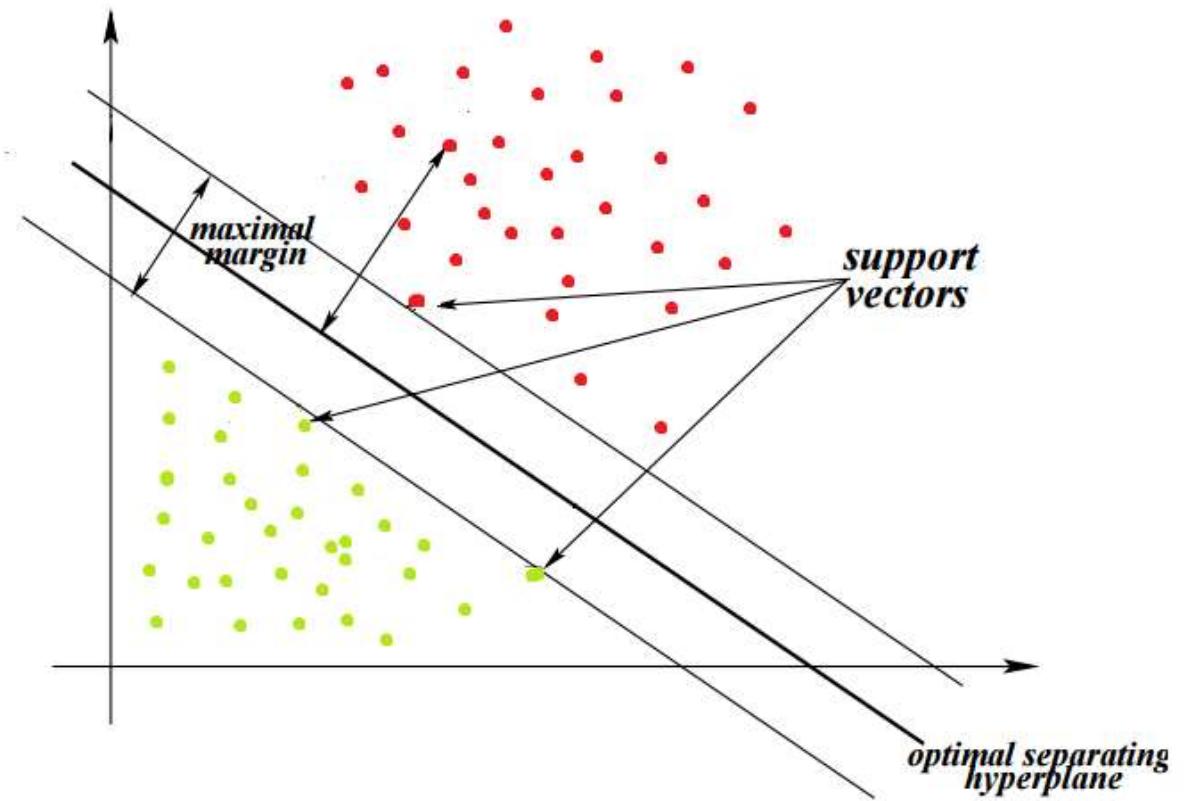
```
Confusion matrix for Naive Bayes
[[12  0  0]
 [ 0 14  1]
 [ 0  2 13]]
accuracy_Naive Bayes: 0.929
precision_Naive Bayes: 0.929
recall_Naive Bayes: 0.929
f1-score_Naive Bayes : 0.929
```

Latihan (18)

Bangun model gaussian Naive Bayes dan akurasi nya

3.5 Linear Support Vector Machine:

Support Vector Machine” (SVM) adalah algoritma pembelajaran mesin terawasi yang dapat digunakan untuk klasifikasi atau regresi. Namun, sebagian besar digunakan dalam masalah klasifikasi. Dalam algoritma SVM, kami memplot setiap item data sebagai titik dalam ruang n-dimensi (di mana n adalah jumlah fitur yang Anda miliki) dengan nilai setiap fitur menjadi nilai koordinat tertentu. Kemudian, kami melakukan klasifikasi dengan menemukan hyper-plane yang membedakan kedua kelas dengan sangat baik



Latihan (19)

Bangun model Linear Support Vector Machines dan akurasi nya

```
In [192]: from sklearn import svm
linear_svc = LinearSVC(max_iter=4000)
linear_svc.fit(X_train, y_train)
Y_pred = linear_svc.predict(X_test)
accuracy_svc=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_linear_svc = round(linear_svc.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision = precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for SVC\n',cm)
print('accuracy_SVC: %.3f' %accuracy)
print('precision_SVC: %.3f' %precision)
print('recall_SVC: %.3f' %recall)
print('f1-score_SVC : %.3f' %f1)
```

```
Confusion matrix for SVC
[[12  0  0]
 [ 0 14  1]
 [ 0  2 13]]
accuracy_SVC: 0.929
precision_SVC: 0.929
recall_SVC: 0.929
f1-score_SVC : 0.929
```

Latihan (20)

Model mana yang terbaik ?

```
In [198]: results = pd.DataFrame({
    'Model': [ 'KNN',
        'Logistic Regression',
        'Naive Bayes',
        ' Support Vector Machine'],
    'Score': [ acc_knn,
        acc_log,
        acc_gaussian,
        acc_linear_svc],
    "Accuracy_score": [accuracy_knn,
        accuracy_lr,
        accuracy_nb,
        accuracy_svc
    ]})
result_df = results.sort_values(by='Accuracy_score', ascending=False)
result_df = result_df.reset_index(drop=True)
result_df.head(9)
```

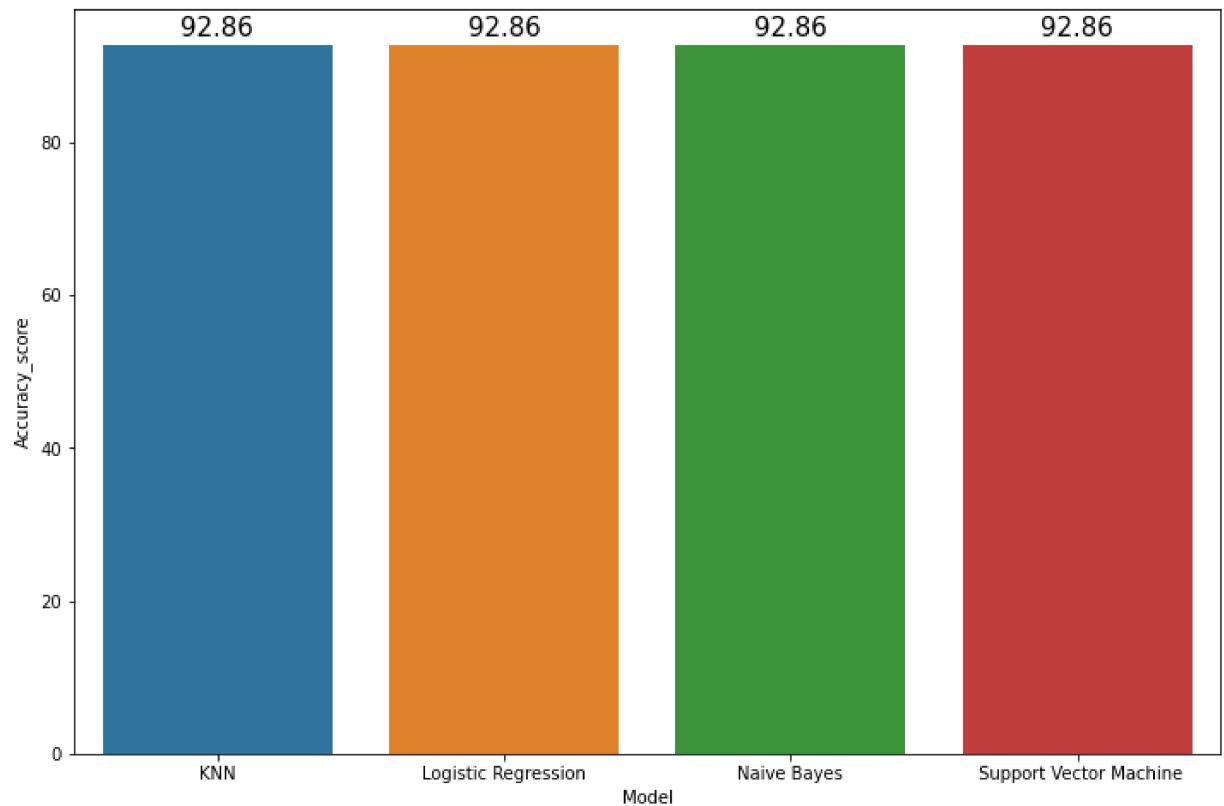
Out[198]:

	Model	Score	Accuracy_score
0	KNN	95.92	92.86
1	Logistic Regression	96.94	92.86
2	Naive Bayes	96.94	92.86
3	Support Vector Machine	94.90	92.86

Seperti yang kita lihat Model terbaik diberikan oleh Logistic Regression & Naive Bayes (Akurasi 96%).

Visualisasikan Hasil Akurasi

```
In [199]: plt.figure(figsize=(12,8))
ax=sns.barplot(x='Model',y="Accuracy_score",data=result_df)
labels = (result_df["Accuracy_score"])
# add result numbers on barchart
for i, v in enumerate(labels):
    ax.text(i, v+1, str(v), horizontalalignment = 'center', size = 15, color = 'black')
```



Hasil Observasi:

Hal ini seperti yang diharapkan dapat terlihat pada heatmap di atas bahwa korelasi antara Sepal Width dan Sepal Length sangat rendah sedangkan korelasi antara Petal Width and Petal Lengthl sangat tinggi. Dengan model terbaik diberikan oleh Logistic Regression dengan akurasi 96%

Thank you!!