PRAKTIKUM DATA WAREHOUSING DAN DATA MINING

MODUL 8

ALGORITMA NAÏVE BAYES

Disusun oleh:

Adinda Aulia Hapsari

L200220037

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA

TAHUN 2024

Setelah kegiatan selesai, lembar kerja ini dicetak (di-print) dan dikumpulkan ke asisten. NIM : L200220037 Nama : Adinda Aulia Hapsari Nama Asisten : Diva Halimah Tanggal Praktikum : 22 November 2024	(Diisi oleh Asisten) Nilai Praktek : Tanda Tangan :

KEGIATAN PRAKTIKUM

Langkah-langkah menggunakan algoritma naïve bayes dengan bahasa pemrograman python adalah sebagai berikut:

1. Bukalah Jupyter notebook yang sudah terinstall pada komputer yang digunakan.
2. Buatlah file baru pada jupyter notebook dengan mengklik menu New-Python 3, secara otomatis akan keluar window baru untuk melakukan coding pada jupyter nootebook. Pastikan file yang dibuat dengan dataset yang digunakan berapa dalam satu folder.
3. Pada baris pertama tuliskan beberapa library yang akan digunakan dalam praktikum. Ada beberapa library seperti numpy, pandas, matplotlib, seaborn dan lain sebagainya.

```
[10]: #import library
      import numpy as np
      import pandas as pd
      import matplotlib.pyplot as plt
      import seaborn as sns

      from sklearn.naive_bayes import GaussianNB
      from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

4. Setelah itu lakukan read dataset menggunakan library pandas, disini kita menggunakan dataset iris dengan ekstensi .csv. Kita akan melakukan beberapa hal untuk mengekplorasi data yang digunakan. Pertama kita akan melihat isi dataset yang dipakai menggunakan fungsi head, fungsi tersebut akan menampilkan isi data beserta atribut yang ada. Terdapat 6 atribut yang digunakan pada dataset tersebut yaitu Id, SepalLengthCm, SepalWidthCm, PetalLengthCM, PetalWidthCm dan Species yang nanti digunakan sebagai label klasifikasi.

```
[11]: iris=pd.read_csv('iris.csv')

[12]: iris.head(5)
```

[12]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

5. Kita juga bisa mengecek bentuk dari data yang digunakan menggunakan fungsi shape, bisa juga melihat value apa saja yang ada pada atribut Species.

```
[15]: #cek jumlah baris dan kolom
      iris.shape

[15]: (150, 6)

[16]: iris['Species'].unique()

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

6. Hal lain yang harus kita lakukan adalah memeriksa apakah dataset memiliki nilai kosong (null) atau tidak, hal ini bisa dilakukan dengan menggunakan fungsi info(). Terlihat tidak ada data yang kosong pada semua atribut, ini menunjukan dataset yang digunakan sangat baik.

```
[17]: iris.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 150 entries, 0 to 149
      Data columns (total 6 columns):
       #   Column         Non-Null Count  Dtype
      ---  ------         --------------  -----
       0   Id             150 non-null    int64
       1   SepalLengthCm  150 non-null    float64
       2   SepalWidthCm   150 non-null    float64
       3   PetalLengthCm  150 non-null    float64
       4   PetalWidthCm   150 non-null    float64
       5   Species        150 non-null    object
      dtypes: float64(4), int64(1), object(1)
      memory usage: 7.2+ KB
```

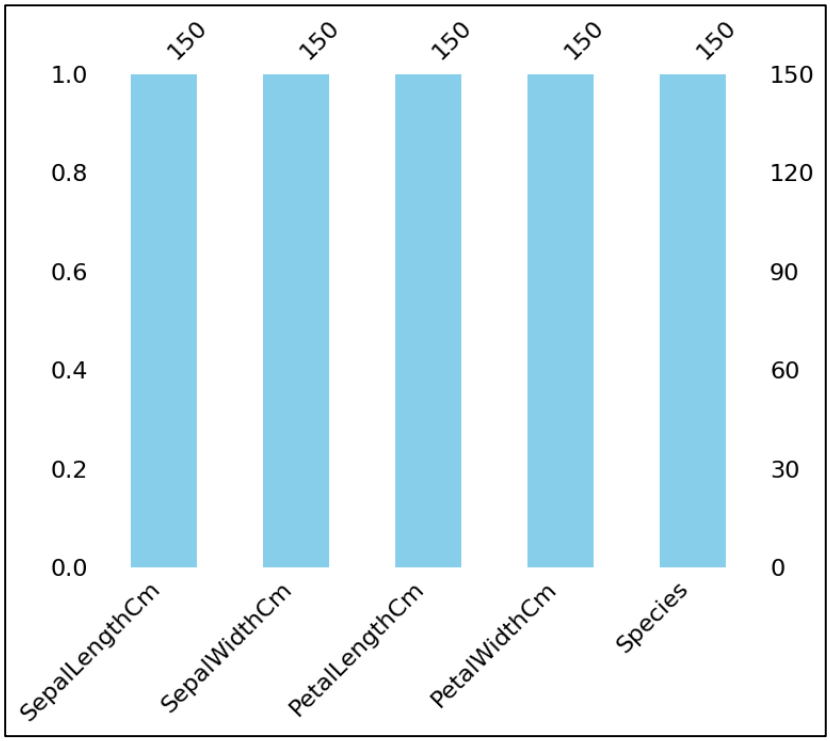
7. Dari semua atribut yang ada kita akan memakai lima fitur saja yaitu SepalLengthCm, SepalWidthCm, PetalLengthCM, PetalWidthCm dan Species. Sehingga atribut Id akan dihapus karena fitur tersebut tidak memiliki pengaruh dalam proses klasifikasi, penghapusan atribut bisa menggunakan fungsi drop.

```
[19]: iris.drop(columns="Id",inplace=True)
      iris.isnull().sum()

[19]: SepalLengthCm    0
      SepalWidthCm     0
      PetalLengthCm    0
      PetalWidthCm     0
      Species          0
      dtype: int64
```

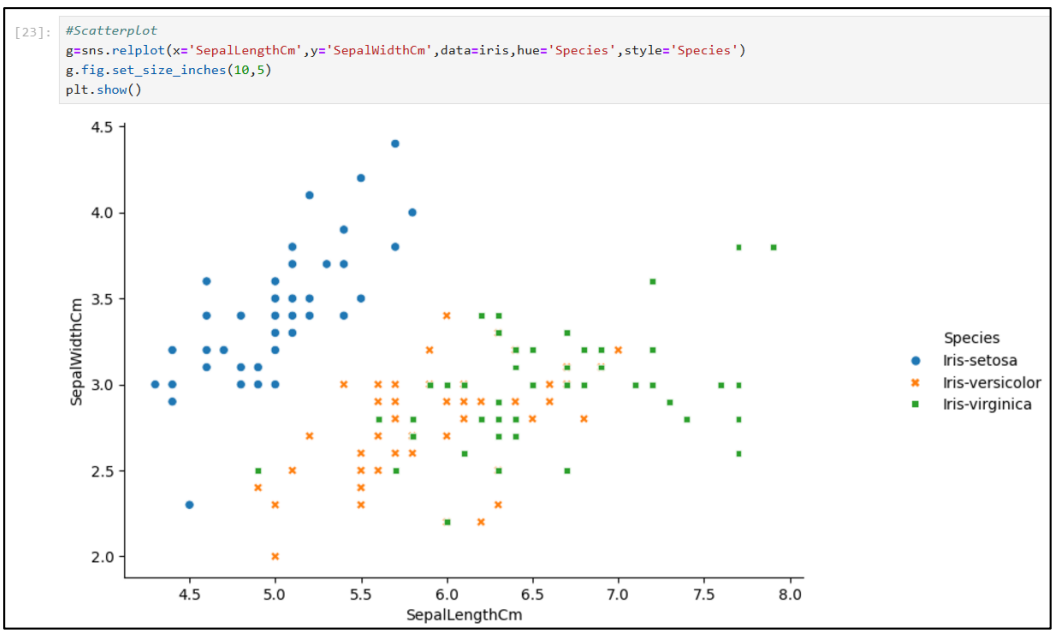
8. Kita juga bisa mengecek data yang kosong menggunakan grafik.

```
[21]: import missingno as msno
      msno.bar(iris,figsize=(8,6),color='skyblue')
      plt.show()
```



9. Proses selanjutnya adalah melakukan visualisasi data, proses ini sangat penting dilakukan untuk mengetahui sebaran data yang ada dalam dataset.

```
[23]: #Scatterplot
      g=sns.relplot(x='SepalLengthCm',y='SepalWidthCm',data=iris,hue='Species',style='Species')
      g.fig.set_size_inches(10,5)
      plt.show()
```

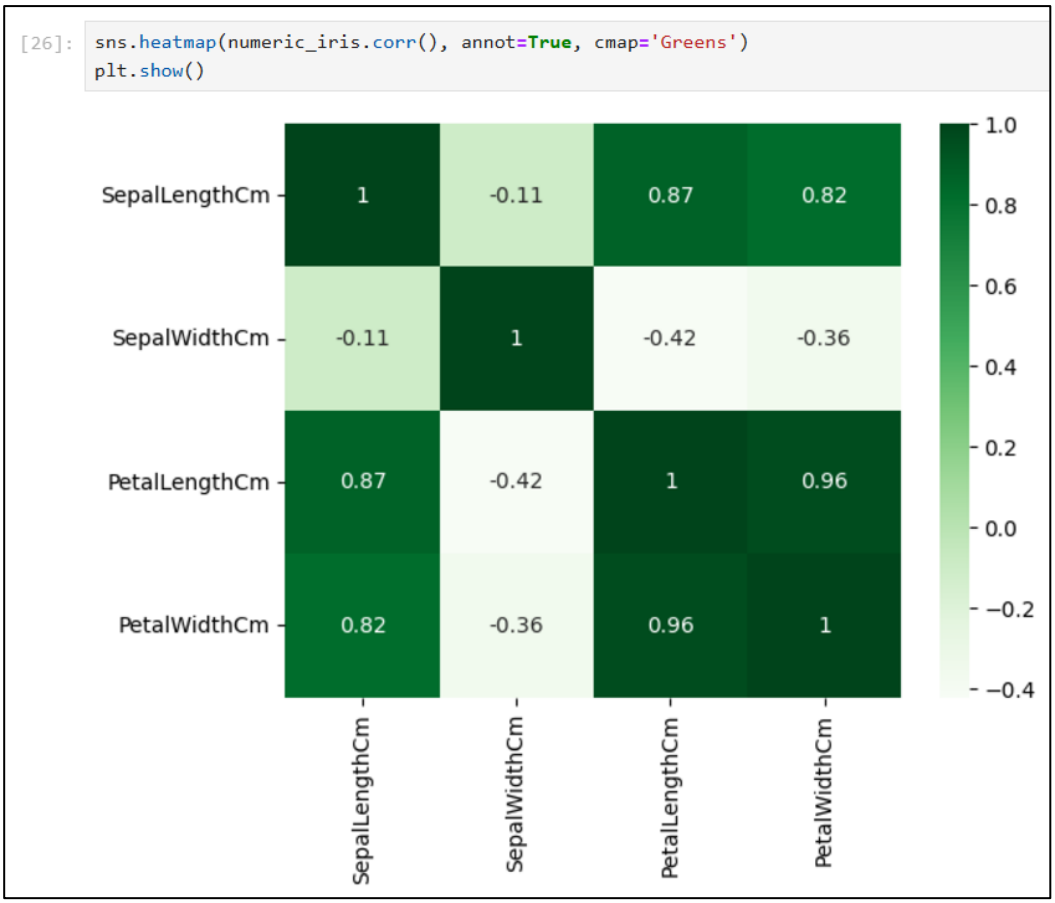


10. Kemudian kita bisa mencari korelasi pada setiap fitur yang digunakan untuk melakukan klasifikasi.

```
[24]: numeric_iris = iris.select_dtypes(include=['float64', 'int64'])
      correlation_matrix = numeric_iris.corr()
      print(correlation_matrix)
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

```
[26]: sns.heatmap(numeric_iris.corr(), annot=True, cmap='Greens')
      plt.show()
```



11. Hal penting lainnya yang harus dilakukan adalah memisahkan antara fitur dan label, atribut Species akan digunakan sebagai label dan atribut lainnya digunakna sebagai fitur pada proses klasifikasi.

Data Preprocessing

```
[27]: #pisahkan antara fitur dan label
      x = iris.drop('Species', axis = 1)
      y = iris['Species']
```

12. Kita bisa mengecek atribut yang sudah terpisah dengan menampilkan variabel X yang merupakan fitur dan variabel y sebagai label.

```
[28]: x
```

[28]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
[29]: y
```

```
[29]: 0         Iris-setosa
      1         Iris-setosa
      2         Iris-setosa
      3         Iris-setosa
      4         Iris-setosa
                   ...
      145    Iris-virginica
      146    Iris-virginica
      147    Iris-virginica
      148    Iris-virginica
      149    Iris-virginica
      Name: Species, Length: 150, dtype: object
```

13. Pada dataset iris, label yang digunakan bukan merupakan data numerical, padahal mesin hanya bisa memahami angka, sehingga kita harus merubah isi dari label menjadi angka, proses ini dinamakan dengan encoding.

```
[31]: #Label encoding
      from sklearn.preprocessing import LabelEncoder
      le = LabelEncoder()
      y = le.fit_transform(y)
      y
```

```
[31]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

14. Sebelum memasuki proses klasifikasi data perlu dibagi menjadi dua bagian yaitu data training dan data testing. Data training akan digunakan untuk pembuatan model, sedangkan data testing digunakan untuk melakukan evaluasi pada model yang sudah dibuat, proses pembagian dataset ini menggunakan library sklearn. Terlihat pada gambar dataset terbagi menjadi 120 untuk data training dan 30 untuk data testing.

```
[32]: #Spliting data menjadi train data dan test data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=0)

print('The shape of x_train is: {}'.format(x_train.shape))
print('The shape of x_test is: {}'.format(x_test.shape))
print('The shape of y_train is: {}'.format(y_train.shape))
print('The shape of y_test is: {}'.format(y_test.shape))

The shape of x_train is: (120, 4)
The shape of x_test is: (30, 4)
The shape of y_train is: (120,)
The shape of y_test is: (30,)
```

15. Tahap selanjutnya adalah pembuatan model menggunakan algoritma Naive Bayes. Kita bisa memanggilnya dari library sklearn dan menaruhnya pada variabel model.

Membuat Model

```
[34]: #Model training
model = GaussianNB()
model.fit(x_train,y_train)

[34]: GaussianNB
      GaussianNB()
```

16. Kita lakukan training model menggunakan algoritma naive bayes pada data training. Disini kita mendapatkan akurasi pada saat training memperoleh skor 95%.

```
[37]: #Prediksi pada data train
pred_train = model.predict(x_train)
cm = confusion_matrix(y_train, pred_train)

#confusion matrix
print('Confusion matrix Naive Bayes\n',cm)
print('')

#akurasi
print('Akurasi pada saat training: {}' .format(accuracy_score(y_train,pred_train))) #confusion matrix

Confusion matrix Naive Bayes
 [[39  0  0]
 [ 0 34  3]
 [ 0  3 41]]

Akurasi pada saat training: 0.95
```

17. Langkah terakhir adalah menguji model yang dibuat dengan data testing yang sudah disiapkan. Pada saat evaluasi model kita mendapatkan nilai yang sama pada akurasi, presisi, recall dan f1 score, yaitu sebesar 96,7%.

[38]:
```
#prediksi pada data test
pred_test = model.predict(x_test)
```

[40]:
```
cm = confusion_matrix(y_test, pred_test)
accuracy = accuracy_score(y_test, pred_test)
precision = precision_score(y_test, pred_test, average='micro')
recall = recall_score(y_test, pred_test, average='micro')
f1 = f1_score(y_test, pred_test, average='micro')
print('Confusion matrix Naive Bayes\n',cm)
print('')
print('Akurasi pada data test: %.3f' %accuracy)
print('precision: %.3f' %precision)
print('recall: %.3f' %recall)
print('f1-score: %.3f' %f1)
```

```
Confusion matrix Naive Bayes
 [[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]

Akurasi pada data test: 0.967
precision: 0.967
recall: 0.967
f1-score: 0.967
```

TUGAS

1. Buatlah ekplorasi data dari dataset Breast-Cancer.csv!

```
[43]: BreastCancer=pd.read_csv('Breast-Cancer.csv')
```

2. Cari 4 fitur yang memiliki pengaruh paling besar terhadap proses klasifikasi!

```
[44]: BreastCancer.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 116 entries, 0 to 115
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             116 non-null    int64
 1   BMI             116 non-null    float64
 2   Glucose         116 non-null    int64
 3   Insulin         116 non-null    float64
 4   HOMA            116 non-null    float64
 5   Leptin          116 non-null    float64
 6   Adiponectin     116 non-null    float64
 7   Resistin        116 non-null    float64
 8   MCP.1           116 non-null    float64
 9   Classification  116 non-null    int64
dtypes: float64(7), int64(3)
memory usage: 9.2 KB
```

```
[48]: # Hitung korelasi
correlation_matrix = BreastCancer.corr()
print(correlation_matrix['Classification'].sort_values(ascending=False))

Classification    1.000000
Glucose           0.384315
HOMA              0.284012
Insulin           0.276804
Resistin          0.227310
MCP.1             0.091381
Leptin           -0.001078
Adiponectin      -0.019490
Age              -0.043555
BMI              -0.132586
Name: Classification, dtype: float64
```

```
[50]: BreastCancer.drop(columns=["MCP.1", "Leptin", "Adiponectin", "Age", "BMI"] ,inplace=True)
BreastCancer.isnull().sum()

[50]: Glucose          0
Insulin          0
HOMA             0
Resistin         0
Classification   0
dtype: int64
```

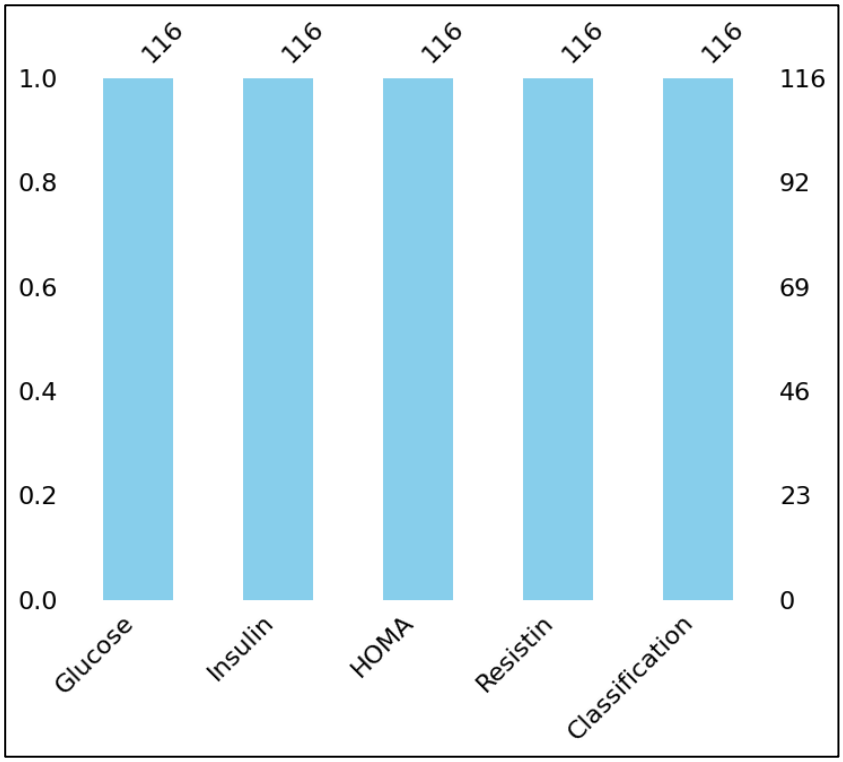
3. Gunakan 4 fitur yang sudah dipilih untuk melakukan klasifikasi!

```
[34]: BreastCancer.head(5)
```

[34]:

	Glucose	Insulin	HOMA	Resistin	Classification
0	70	2.707	0.467409	7.99585	1
1	92	3.115	0.706897	4.06405	1
2	91	4.498	1.009651	9.27715	1
3	77	3.226	0.612725	12.76600	1
4	92	3.549	0.805386	10.57635	1

```
[36]: import missingno as msno
      msno.bar(BreastCancer,figsize=(8,6),color='skyblue')
      plt.show()
```




```
[37]: #pisahkan antara fitur dan label
      x = BreastCancer.drop('Classification', axis = 1)
      y = BreastCancer['Classification']
```

[38]:
```
x
```

[38]:

	Glucose	Insulin	HOMA	Resistin
0	70	2.707	0.467409	7.99585
1	92	3.115	0.706897	4.06405
2	91	4.498	1.009651	9.27715
3	77	3.226	0.612725	12.76600
4	92	3.549	0.805386	10.57635
...	...	...	...	...
111	92	3.330	0.755688	10.96000
112	100	4.530	1.117400	7.32000
113	97	5.730	1.370998	10.33000
114	82	2.820	0.570392	3.27000
115	138	19.910	6.777364	4.35000

116 rows × 4 columns

[39]:
```
y
```

[39]:
```
0      1
1      1
2      1
3      1
4      1
      ..
111    2
112    2
113    2
114    2
115    2
Name: Classification, Length: 116, dtype: int64
```

[40]:
```
#Spliting data menjadi train data dan test data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=0)

print('The shape of x_train is: {}'.format(x_train.shape))
print('The shape of x_test is: {}'.format(x_test.shape))
print('The shape of y_train is: {}'.format(y_train.shape))
print('The shape of y_test is: {}'.format(y_test.shape))
```

```
The shape of x_train is: (92, 4)
The shape of x_test is: (24, 4)
The shape of y_train is: (92,)
The shape of y_test is: (24,)
```

4. Buatlah model untuk algoritma Naive Bayes dan hitunglah nilai akurasi, presisi, recall dan F1-score!

```
[41]: #Model training
      model = GaussianNB()
      model.fit(x_train,y_train)

[41]: GaussianNB
      GaussianNB()
```

```
[42]: #Prediksi pada data train
      pred_train = model.predict(x_train)
      cm = confusion_matrix(y_train, pred_train)

      #confusion matrix
      print('Confusion matrix Naive Bayes\n',cm)
      print('')

      #akurasi
      print('Akurasi pada saat training: {}' .format(accuracy_score(y_train,pred_train))) #confusion matrix

Confusion matrix Naive Bayes
 [[37  4]
 [29 22]]

Akurasi pada saat training: 0.6413043478260869
```

```
[43]: #prediksi pada data test
      pred_test = model.predict(x_test)

[44]: cm = confusion_matrix(y_test, pred_test)
      accuracy = accuracy_score(y_test, pred_test)
      precision = precision_score(y_test, pred_test, average='micro')
      recall = recall_score(y_test, pred_test, average='micro')
      f1 = f1_score(y_test, pred_test, average='micro')
      print('Confusion matrix Naive Bayes\n',cm)
      print('')
      print('Akurasi pada data test: %.3f' %accuracy)
      print('precision: %.3f' %precision)
      print('recall: %.3f' %recall)
      print('f1-score: %.3f' %f1)

Confusion matrix Naive Bayes
 [[9 2]
 [8 5]]

Akurasi pada data test: 0.583
precision: 0.583
recall: 0.583
f1-score: 0.583
```