Rekomendasi Makanan Pendamping ASI Bayi Menggunakan Naive Bayes

Importing the libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

Import dataset digunakan untuk memanggil data kedalam program atau biasa disebut dengan pembacaan data

In [2]:
```
df = pd.read_csv('finaldata.csv')
df.head(100)
```

Out[2]:

	umur	jk	bb	alergi	kebutuhankalori	kalori_asi	jml_kebutuhankalori	rekomendasimakanan
0	6	P	5.7	Beras	682	486	196	Bubur Kentang
1	6	P	6.5	Gabus	682	486	196	Bubur Kentang
2	6	P	8.2	Semangka	682	486	196	Bubur Kentang
3	7	P	6.0	Ayam	682	486	196	Bubur Kentang
4	7	L	6.8	Tahu	682	486	196	Bubur Kentang
5	7	P	8.6	Pisang Raja	682	486	196	Bubur Kentang
6	8	P	6.3	Ikan layang	682	486	196	Bubur Kentang
7	8	P	7.0	Tempe	682	486	196	Bubur Kentang
8	8	P	9.0	Pepaya	682	486	196	Bubur Kentang
9	9	P	6.5	Bayam	830	375	455	Bubur Beras
10	9	P	7.3	Kembang kol	830	375	455	Bubur Beras
11	9	P	9.3	Alpukat	830	375	455	Bubur Beras
12	10	P	6.7	Sawi	830	375	455	Bubur Beras
13	10	L	7.5	Bubur beras	830	375	455	Bubur Beras
14	10	P	9.6	Apel	830	375	455	Bubur Beras
15	11	P	6.9	Telur ayam	830	375	455	Bubur Beras
16	11	L	7.7	Bubur kentang	830	375	455	Bubur Beras
17	11	P	9.9	Pisang	830	375	455	Bubur Beras
18	12	P	7.0	Telur bebek	1092	313	779	Ayam
19	12	P	7.9	Bubur wortel	1092	313	779	Ayam
20	12	P	10.1	Jagung Bubur	1092	313	779	Ayam
21	13	P	7.2	Kangkung	1092	313	779	Ayam
22	13	L	8.1	Kentang hitam	1092	313	779	Ayam

	umur	jk	bb	alergi	kebutuhankalori	kalori_asi	jml_kebutuhankalori	rekomendasimakanan
23	13	P	10.4	Beras Tumbuk	1092	313	779	Ayam
24	14	P	7.4	Udang	1092	313	779	Ayam
25	14	L	8.3	Ketela pohon (singkong)	1092	313	779	Ayam
26	14	P	10.6	Roti Putih	1092	313	779	Ayam
27	15	P	7.6	Teri	1092	313	779	Ayam
28	15	P	8.5	Ubi jalar putih	1092	313	779	Ayam
29	15	P	10.9	Beras giling	1092	313	779	Ayam
30	16	P	7.7	Kacang panjang	1092	313	779	Ayam
31	16	P	8.7	Ubi jalar merah	1092	313	779	Ayam
32	16	P	11.1	Jawawut	1092	313	779	Ayam
33	17	P	7.9	Ketimun	1092	313	779	Ayam
34	17	L	8.9	Biskuit	1092	313	779	Ayam
35	17	P	11.4	Sosis Daging	1092	313	779	Ayam
36	18	P	8.1	Labu kuning	1092	313	779	Ayam
37	18	L	9.1	Promina	1092	313	779	Ayam
38	18	P	11.6	Jagung giling putih	1092	313	779	Ayam
39	19	L	8.2	Labu siam	1092	313	779	Ayam
40	19	P	9.2	Pudding	1092	313	779	Ayam
41	19	P	11.8	Ikan Cakalang	1092	313	779	Ayam
42	20	P	8.4	Terong	1092	313	779	Ayam
43	20	P	9.4	Ikan Mas	1092	313	779	Ayam
44	21	P	8.6	Wortel	1092	313	779	Ayam
45	21	P	9.6	Ikan Mujair	1092	313	779	Ayam
46	22	P	8.7	Jagung kuning	1092	313	779	Ayam
47	22	L	9.8	Ikan Kakap	1092	313	779	Ayam
48	23	P	8.9	Kentang	1092	313	779	Ayam

	umur	jk	bb	alergi	kebutuhankalori	kalori_asi	jml_kebutuhankalori	rekomendasimakanan
49	23	L	10.0	Ikan Tongkol	1092	313	779	Ayam
50	24	P	9.0	Tahu	1092	313	779	Ayam
51	24	P	10.2	Keju	1092	313	779	Ayam

Label Encoder

Pada data diatas masih terdapat data yang berupa kategorikal, sehingga di butuhkan proses untuk mengubah data Kategorikal menjadi Numerik atau angka sehingga bisa di proses dengan menggunakan algoritma Naive Bayes.

Label encoder sendiri mempunyai fungsi untuk mengubah nilai dari Kategorikal menjadi numerik berdasarkan kesamaan dari banyaknya data yang digunakan pada dataset yang kita gunakan.

Penerapannya: Pada data kita dapat melihat bahwa terdapat 3 Atribut/variabel yang mempunyai type data Kategorikal, yaitu Jenis Kelamin atau disingkat jk, Alergi, dan rekomendasimakanan. Dari ketiga data tersebut belum bisa diproses karena masih bersifat kategorikal data sehingga digunakanlah metode Label Encoder untuk mengubah value/nilai dari atribut-atribut yang digunakan, khususnya pada data yang bertipe Kategorikal.

In [3]:
```
# Fungsi untuk melakukan label encoder
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()
df['jk'] = labelencoder.fit_transform(df['jk'])
df['alergi'] = labelencoder.fit_transform(df['alergi'])
df['rekomendasimakanan'] = labelencoder.fit_transform(df['rekomendasimakanan'])

# Fungsi Untuk Menampilkan Data
df.head(51)
```

Out[3]:

	umur	jk	bb	alergi	kebutuhankalori	kalori_asi	jml_kebutuhankalori	rekomendasimakanan
0	6	1	5.7	4	682	486	196	2
1	6	1	6.5	11	682	486	196	2
2	6	1	8.2	39	682	486	196	2
3	7	1	6.0	2	682	486	196	2
4	7	0	6.8	41	682	486	196	2
5	7	1	8.6	34	682	486	196	2
6	8	1	6.3	17	682	486	196	2
7	8	1	7.0	44	682	486	196	2
8	8	1	9.0	32	682	486	196	2
9	9	1	6.5	3	830	375	455	1
10	9	1	7.3	25	830	375	455	1
11	9	1	9.3	0	830	375	455	1
12	10	1	6.7	38	830	375	455	1
13	10	0	7.5	8	830	375	455	1
14	10	1	9.6	1	830	375	455	1
15	11	1	6.9	42	830	375	455	1
16	11	0	7.7	9	830	375	455	1
17	11	1	9.9	33	830	375	455	1

	umur	jk	bb	alergi	kebutuhankalori	kalori_asi	jml_kebutuhankalori	rekomendasimakanan
18	12	1	7.0	43	1092	313	779	0
19	12	1	7.9	10	1092	313	779	0
20	12	1	10.1	18	1092	313	779	0
21	13	1	7.2	23	1092	313	779	0
22	13	0	8.1	27	1092	313	779	0
23	13	1	10.4	5	1092	313	779	0
24	14	1	7.4	49	1092	313	779	0
25	14	0	8.3	28	1092	313	779	0
26	14	1	10.6	37	1092	313	779	0
27	15	1	7.6	45	1092	313	779	0
28	15	1	8.5	48	1092	313	779	0
29	15	1	10.9	6	1092	313	779	0
30	16	1	7.7	22	1092	313	779	0
31	16	1	8.7	47	1092	313	779	0
32	16	1	11.1	21	1092	313	779	0
33	17	1	7.9	29	1092	313	779	0
34	17	0	8.9	7	1092	313	779	0
35	17	1	11.4	40	1092	313	779	0
36	18	1	8.1	30	1092	313	779	0
37	18	0	9.1	35	1092	313	779	0
38	18	1	11.6	19	1092	313	779	0
39	19	0	8.2	31	1092	313	779	0
40	19	1	9.2	36	1092	313	779	0
41	19	1	11.8	12	1092	313	779	0
42	20	1	8.4	46	1092	313	779	0
43	20	1	9.4	14	1092	313	779	0

	umur	jk	bb	alergi	kebutuhankalori	kalori_asi	jml_kebutuhankalori	rekomendasimakanan
44	21	1	8.6	50	1092	313	779	0
45	21	1	9.6	15	1092	313	779	0
46	22	1	8.7	20	1092	313	779	0
47	22	0	9.8	13	1092	313	779	0
48	23	1	8.9	26	1092	313	779	0
49	23	0	10.0	16	1092	313	779	0
50	24	1	9.0	41	1092	313	779	0

Memisahkan Data Label dengan Atribut

Label ialah hasil rekomendasi makanan, sedangkan Atribut nilai-nilai yang mempengaruhi rekomendasi makanan yang direkomendasikan. Pemisahan ini dilakukan untuk memberikan pengujian di naive bayes dalam hal mengenali/memprediksi asupan makanan yang direkomendasikan.

In [4]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

Spiliting dataset adalah proses pemisalahan data Training dan data Testing, dimana untuk pembagian ini data yang digunakan dibagi kedalam 20% untuk testing data dan untuk training sebesar 80%.

In [41]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

In [42]:
```
# Cetak data kelompok data X Train
print(X_train)
```

```
[[1.500e+01 1.000e+00 1.090e+01 6.000e+00 1.092e+03 3.130e+02 7.790e+02]
 [2.000e+01 1.000e+00 8.400e+00 4.600e+01 1.092e+03 3.130e+02 7.790e+02]
 [2.000e+01 1.000e+00 9.400e+00 1.400e+01 1.092e+03 3.130e+02 7.790e+02]
 [8.000e+00 1.000e+00 7.000e+00 4.400e+01 6.820e+02 4.860e+02 1.960e+02]
 [1.000e+01 1.000e+00 9.600e+00 1.000e+00 8.300e+02 3.750e+02 4.550e+02]
 [1.700e+01 1.000e+00 7.900e+00 2.900e+01 1.092e+03 3.130e+02 7.790e+02]
 [2.400e+01 1.000e+00 9.000e+00 4.100e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.200e+01 1.000e+00 7.000e+00 4.300e+01 1.092e+03 3.130e+02 7.790e+02]
 [2.300e+01 0.000e+00 1.000e+01 1.600e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.700e+01 1.000e+00 1.140e+01 4.000e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.100e+01 1.000e+00 6.900e+00 4.200e+01 8.300e+02 3.750e+02 4.550e+02]
 [7.000e+00 1.000e+00 8.600e+00 3.400e+01 6.820e+02 4.860e+02 1.960e+02]
 [1.500e+01 1.000e+00 7.600e+00 4.500e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.100e+01 0.000e+00 7.700e+00 9.000e+00 8.300e+02 3.750e+02 4.550e+02]
 [2.400e+01 1.000e+00 1.020e+01 2.400e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.200e+01 1.000e+00 1.010e+01 1.800e+01 1.092e+03 3.130e+02 7.790e+02]
 [2.100e+01 1.000e+00 9.600e+00 1.500e+01 1.092e+03 3.130e+02 7.790e+02]
 [8.000e+00 1.000e+00 9.000e+00 3.200e+01 6.820e+02 4.860e+02 1.960e+02]
 [1.000e+01 0.000e+00 7.500e+00 8.000e+00 8.300e+02 3.750e+02 4.550e+02]
 [1.400e+01 0.000e+00 8.300e+00 2.800e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.100e+01 1.000e+00 9.900e+00 3.300e+01 8.300e+02 3.750e+02 4.550e+02]
 [1.800e+01 1.000e+00 1.160e+01 1.900e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.900e+01 1.000e+00 1.180e+01 1.200e+01 1.092e+03 3.130e+02 7.790e+02]
 [6.000e+00 1.000e+00 6.500e+00 1.100e+01 6.820e+02 4.860e+02 1.960e+02]
 [1.000e+01 1.000e+00 6.700e+00 3.800e+01 8.300e+02 3.750e+02 4.550e+02]
 [2.200e+01 1.000e+00 8.700e+00 2.000e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.400e+01 1.000e+00 7.400e+00 4.900e+01 1.092e+03 3.130e+02 7.790e+02]
 [8.000e+00 1.000e+00 6.300e+00 1.700e+01 6.820e+02 4.860e+02 1.960e+02]
 [1.300e+01 1.000e+00 1.040e+01 5.000e+00 1.092e+03 3.130e+02 7.790e+02]
 [1.800e+01 1.000e+00 8.100e+00 3.000e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.300e+01 1.000e+00 7.200e+00 2.300e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.200e+01 1.000e+00 7.900e+00 1.000e+01 1.092e+03 3.130e+02 7.790e+02]
 [9.000e+00 1.000e+00 6.500e+00 3.000e+00 8.300e+02 3.750e+02 4.550e+02]
 [1.900e+01 0.000e+00 8.200e+00 3.100e+01 1.092e+03 3.130e+02 7.790e+02]
 [2.300e+01 1.000e+00 8.900e+00 2.600e+01 1.092e+03 3.130e+02 7.790e+02]
 [7.000e+00 1.000e+00 6.000e+00 2.000e+00 6.820e+02 4.860e+02 1.960e+02]
 [6.000e+00 1.000e+00 5.700e+00 4.000e+00 6.820e+02 4.860e+02 1.960e+02]
```

```
 [2.200e+01 0.000e+00 9.800e+00 1.300e+01 1.092e+03 3.130e+02 7.790e+02]
 [2.100e+01 1.000e+00 8.600e+00 5.000e+01 1.092e+03 3.130e+02 7.790e+02]]
```

In [47]:
```
# Cetak data kelompok data X Test
print (X_test)
```

```
[[1.500e+01 1.000e+00 8.500e+00 4.800e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.800e+01 0.000e+00 9.100e+00 3.500e+01 1.092e+03 3.130e+02 7.790e+02]
 [9.000e+00 1.000e+00 9.300e+00 0.000e+00 8.300e+02 3.750e+02 4.550e+02]
 [1.700e+01 0.000e+00 8.900e+00 7.000e+00 1.092e+03 3.130e+02 7.790e+02]
 [6.000e+00 1.000e+00 8.200e+00 3.900e+01 6.820e+02 4.860e+02 1.960e+02]
 [1.600e+01 1.000e+00 7.700e+00 2.200e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.900e+01 1.000e+00 9.200e+00 3.600e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.600e+01 1.000e+00 1.110e+01 2.100e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.400e+01 1.000e+00 1.060e+01 3.700e+01 1.092e+03 3.130e+02 7.790e+02]
 [7.000e+00 0.000e+00 6.800e+00 4.100e+01 6.820e+02 4.860e+02 1.960e+02]
 [9.000e+00 1.000e+00 7.300e+00 2.500e+01 8.300e+02 3.750e+02 4.550e+02]
 [1.300e+01 0.000e+00 8.100e+00 2.700e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.600e+01 1.000e+00 8.700e+00 4.700e+01 1.092e+03 3.130e+02 7.790e+02]]
```

In [54]:
```
# Cetak data kelompok data Y Test
print(y_test)
```

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

In [55]:
```
# Cetak data kelompok data Y Train
print(y_train)
```

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

Feature Scaling

StandardScaler adalah class dari sklearn untuk melakukan normalisasi data agar data yang digunakan tidak memiliki penyimpangan yang besar. Satu hal penting dalam Data Analysis adalah membuat DataFrame dari dataset. Lalu menampilkan 5 data teratas untuk memastikan data seperti apa yang akan di analisis.

In [10]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [40]:
```
# Hasil StandardScaler untuk data X_train
print (X_train)
```

```
[[ 0.06947572  0.42640143  1.50752283 -1.21426729  0.72153099 -0.64452815
   0.70484278]
 [ 0.97266012  0.42640143 -0.07635083  1.5034615   0.72153099 -0.64452815
   0.70484278]
 [ 0.97266012  0.42640143  0.55719863 -0.67072153  0.72153099 -0.64452815
   0.70484278]
 [-1.19498243  0.42640143 -0.96332008  1.36757507 -1.73112217  1.9990168
  -1.81933588]
 [-0.83370867  0.42640143  0.68390852 -1.55398339 -0.8457742   0.30286946
  -0.69795977]
 [ 0.43074948  0.42640143 -0.39312556  0.34842677  0.72153099 -0.64452815
   0.70484278]
 [ 1.69520763  0.42640143  0.30377885  1.16374541  0.72153099 -0.64452815
   0.70484278]
 [-0.47243491  0.42640143 -0.96332008  1.29963185  0.72153099 -0.64452815
   0.70484278]
 [ 1.51457076 -2.34520788  0.93732831 -0.53483509  0.72153099 -0.64452815
   0.70484278]
 [ 0.43074948  0.42640143  1.82429756  1.09580219  0.72153099 -0.64452815
   0.70484278]
 [-0.65307179  0.42640143 -1.02667503  1.23168863 -0.8457742   0.30286946
  -0.69795977]
 [-1.37561931  0.42640143  0.05035906  0.68814287 -1.73112217  1.9990168
  -1.81933588]
 [ 0.06947572  0.42640143 -0.5831904   1.43551828  0.72153099 -0.64452815
   0.70484278]
 [-0.65307179 -2.34520788 -0.51983546 -1.01043763 -0.8457742   0.30286946
  -0.69795977]
 [ 1.69520763  0.42640143  1.0640382   0.00871067  0.72153099 -0.64452815
   0.70484278]
 [-0.47243491  0.42640143  1.00068326 -0.39894865  0.72153099 -0.64452815
   0.70484278]
 [ 1.153297    0.42640143  0.68390852 -0.60277831  0.72153099 -0.64452815
   0.70484278]
 [-1.19498243  0.42640143  0.30377885  0.55225643 -1.73112217  1.9990168
  -1.81933588]
 [-0.83370867 -2.34520788 -0.64654535 -1.07838085 -0.8457742   0.30286946
  -0.69795977]
```

```
[-0.11116116 -2.34520788 -0.13970578  0.28048355  0.72153099 -0.64452815
  0.70484278]
[-0.65307179  0.42640143  0.87397336  0.62019965 -0.8457742   0.30286946
 -0.69795977]
[ 0.61138636  0.42640143  1.95100745 -0.33100543  0.72153099 -0.64452815
  0.70484278]
[ 0.79202324  0.42640143  2.07771734 -0.80660797  0.72153099 -0.64452815
  0.70484278]
[-1.55625619  0.42640143 -1.28009481 -0.87455119 -1.73112217  1.9990168
 -1.81933588]
[-0.83370867  0.42640143 -1.15338492  0.95991575 -0.8457742   0.30286946
 -0.69795977]
[ 1.33393388  0.42640143  0.11371401 -0.26306221  0.72153099 -0.64452815
  0.70484278]
[-0.11116116  0.42640143 -0.7099003   1.70729116  0.72153099 -0.64452815
  0.70484278]
[-1.19498243  0.42640143 -1.40680471 -0.46689187 -1.73112217  1.9990168
 -1.81933588]
[-0.29179804  0.42640143  1.19074809 -1.28221051  0.72153099 -0.64452815
  0.70484278]
[ 0.61138636  0.42640143 -0.26641567  0.41636999  0.72153099 -0.64452815
  0.70484278]
[-0.29179804  0.42640143 -0.83661019 -0.05923255  0.72153099 -0.64452815
  0.70484278]
[-0.47243491  0.42640143 -0.39312556 -0.94249441  0.72153099 -0.64452815
  0.70484278]
[-1.01434555  0.42640143 -1.28009481 -1.41809695 -0.8457742   0.30286946
 -0.69795977]
[ 0.79202324 -2.34520788 -0.20306073  0.48431321  0.72153099 -0.64452815
  0.70484278]
[ 1.51457076  0.42640143  0.2404239   0.14459711  0.72153099 -0.64452815
  0.70484278]
[-1.37561931  0.42640143 -1.59686955 -1.48604017 -1.73112217  1.9990168
 -1.81933588]
[-1.55625619  0.42640143 -1.78693439 -1.35015373 -1.73112217  1.9990168
 -1.81933588]
[ 1.33393388 -2.34520788  0.81061842 -0.73866475  0.72153099 -0.64452815
  0.70484278]
[ 1.153297    0.42640143  0.05035906  1.77523438  0.72153099 -0.64452815
  0.70484278]]
```

In [56]:
```
# Hasil StandardScaler untuk data X_test
print (X_test)
```

```
[[1.500e+01 1.000e+00 8.500e+00 4.800e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.800e+01 0.000e+00 9.100e+00 3.500e+01 1.092e+03 3.130e+02 7.790e+02]
 [9.000e+00 1.000e+00 9.300e+00 0.000e+00 8.300e+02 3.750e+02 4.550e+02]
 [1.700e+01 0.000e+00 8.900e+00 7.000e+00 1.092e+03 3.130e+02 7.790e+02]
 [6.000e+00 1.000e+00 8.200e+00 3.900e+01 6.820e+02 4.860e+02 1.960e+02]
 [1.600e+01 1.000e+00 7.700e+00 2.200e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.900e+01 1.000e+00 9.200e+00 3.600e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.600e+01 1.000e+00 1.110e+01 2.100e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.400e+01 1.000e+00 1.060e+01 3.700e+01 1.092e+03 3.130e+02 7.790e+02]
 [7.000e+00 0.000e+00 6.800e+00 4.100e+01 6.820e+02 4.860e+02 1.960e+02]
 [9.000e+00 1.000e+00 7.300e+00 2.500e+01 8.300e+02 3.750e+02 4.550e+02]
 [1.300e+01 0.000e+00 8.100e+00 2.700e+01 1.092e+03 3.130e+02 7.790e+02]
 [1.600e+01 1.000e+00 8.700e+00 4.700e+01 1.092e+03 3.130e+02 7.790e+02]]
```

Prediksi Menggunakan Naive Bayes

Algoritma Naive Bayes merupakan salah satu pengklasifikasi statistik, dimana pengklasifikasi ini dapat memprediksi probabilitas keanggotaan kelas suatu data yang akan masuk ke dalam kelas tertentu, sesuai dengan perhitungan probabilitas.

Pada penelitian ini penulis menggunakan Library dari python, dimana dalam library tersebut telah tersedia Algoritma Naive Bayes di dalamnya yang dapat langsung digunakan untuk studi kasus kita.

In [36]:
```
# Implementasi ALgoritma Naive Bayes menggunakan Library Python
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Pengujian Data
#umur, jk, bb,alergi, kebutuhankalori, kalori_asi, jml_kebutuhankalori
DataTesting = sc.transform([[12, 1,5, 44, 1000, 300, 778]])


# Menampilkan Hasil Prediksi
cl = classifier.predict(DataTesting)

if (cl == 0):
    print ("Makanan yang Bisa di Konsumsi : Bubur Kentang, Biskuit atau Ayam")

if (cl == 1):
    print ("Bubur Wortel, Udang, atau Pisang")

if (cl == 2):
    print ("Bubur Beras, Tahu, Bayam")
```

```
Makanan yang Bisa di Konsumsi : Bubur Kentang, Biskuit atau Ayam
```

Pengujian Matrix Confussinon

Confusion matrix (matriks kebingungan)adalah gambar atau tabel yang digunakan untuk mendeskripsikan kinerja sebuah penggolongan (klasifikasi). Ini biasanya diekstrak dari dataset pengujian yang kebenaran dasarnya diketahui.

Kita bandingkan setiap kelas dengan setiap kelas lainnya dan lihat berapa banyak sampel yang salah diklasifikasikan. Selama pembuatan tabel ini, kami sebenarnya menemukan beberapa metrik utama sangat penting di bidang pembelajaran mesin.

Mari kita pertimbangkan kasus klasifikasi biner dengan output 0 atau 1:

- Positif benar: Ini adalah sampel yang kami prediksi 1 sebagai output dan kebenaran dasarnya adalah 1 juga.
- Negatif benar: Ini adalah sampel yang kami perkirakan 0 sebagai keluarannya dan kebenaran dasarnya adalah 0 juga.
- Positif palsu: Ini adalah sampel yang kami prediksi 1 sebagai output tetapi kebenaran dasarnya adalah 0. Ini juga dikenal sebagai kesalahan Tipe I.

- Negatif palsu: Ini adalah sampel yang kami perkirakan 0 sebagai keluarannya tetapi kebenaran dasarnya adalah 1. Ini juga dikenal sebagai kesalahan Tipe II.

Selesai