Modul 10

I. Naive Bayes Classification untuk Prediksi Penyakit Jantung

Algoritma Naive Bayes merupakan sebuah metoda klasifikasi menggunakan metode probabilitas dan statistik yang dikemukakan oleh ilmuwan Inggris Thomas Bayes. Algoritma Naive Bayes memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai Teorema Bayes. Ciri utama dr Naïve Bayes Classifier ini adalah asumsi yg sangat kuat (naïf) akan independensi dari masing-masing kondisi / kejadian.
Keuntungan penggunan adalah bahwa metoda ini hanya membutuhkan jumlah data pelatihan (training data) yang kecil untuk menentukan estimasi parameter yang diperlukan dalam proses pengklasifikasian.
Tahapan dari proses algoritma Naive Bayes adalah:

1. Menghitung jumlah kelas / label.
2. Menghitung Jumlah Kasus Per Kelas
3. Kalikan Semua Variable Kelas
4. Bandingkan Hasil Per Kelas

Data:
Data yang digunakan adalah **Heart Diseases UCI** yaitu data pasien yang memiliki penyakit jantung atau tidak. Data dapat didownload dari classroom.

Keterangan variabel;
→**age:** umur (tahun)
→**sex:** jenis kelamin (1 = laki-laki; 0 = perempuan)
→**cp:** tipe nyeri dada
→**trestbps**: tekanan darah istirahat (dalam mm Hg saat masuk ke rumah sakit)
→**kol:** serum kolestoral dalam mg / dl
→**fbs** : gula darah puasa> 120 mg / dl → (1 = benar; 0 = salah)
→**restecg:** hasil elektrokardiografi istirahat
→**thalach:** detak jantung maksimum
→**exang:** angina yang diinduksi olahraga (1 = ya; 0 = tidak)
→**oldpeak:** ST depression induced by exercise relative to rest
→**slope:** the slope of the peak exercise ST segment
→**ca:** jumlah pembuluh darah utama (0–3) diwarnai dengan fluoroskopi
→**thal:** 3 = normal; 6 = fixed defect; 7 = reversible defect
→**target:** memiliki penyakit atau tidak (1 = ya, 0 = tidak)

Problem:
Apakah pasien masuk di kategori terkena penyakit jantung atau tidak?

Langkah 1: Memasukkan Packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Langkah 2: Memasukkan Data
untuk memudahkan menemukan data pada direktori, letakan data satu folder dengan file .ipynb

```
heart = pd.read_csv("heart.csv") #data tentang penyakit jantung
pada pasien
heart.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Melihat jenis data masing-masing kolom

```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

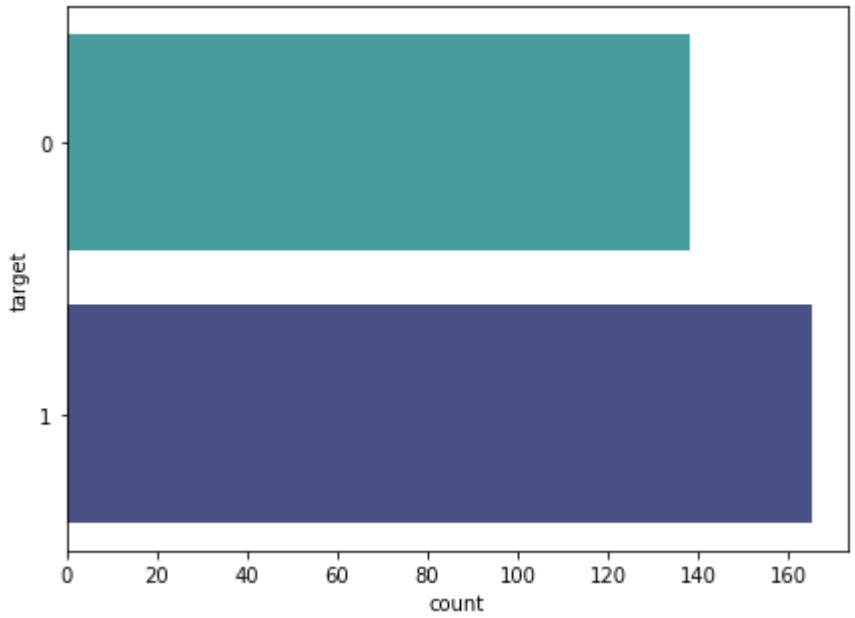
Sebelum melakukan klasifikasi menggunakan naive bayes, lihat terlebih dahulu sekilas statistik deskriptif dari data penyakit jantung.

```
heart.target.value_counts()
```

```
1    165
0    138
Name: target, dtype: int64
```

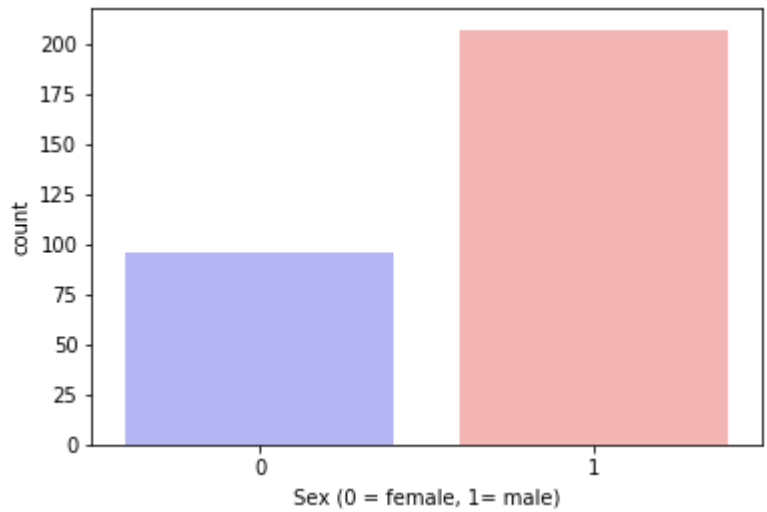
Diketauhi pasien yang terkena penyakit jantung sebanyak 165 dari 303 pasien. Berikut untuk script visualisasinya.

```
f, ax = plt.subplots(figsize=(7, 5))
sns.countplot(y="target", data=heart, palette="mako_r");
```



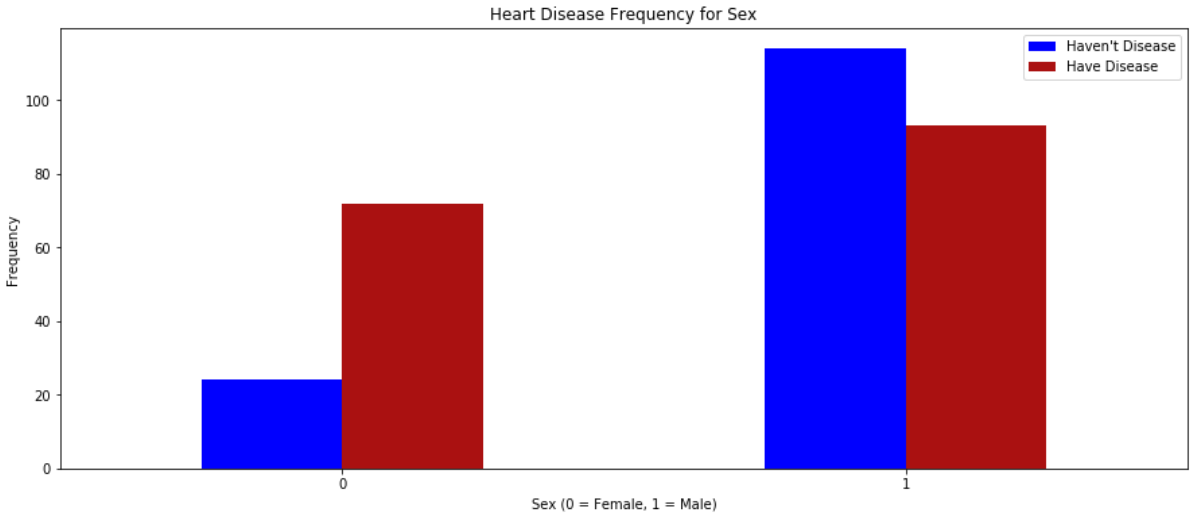
Visualisasi untuk pasien pengidap penyakit jantung berdasarkan jenis kelamin.

```
pd.crosstab(heart.sex,heart.target).plot(kind="bar",figsize=(15,
6),color=['blue','#AA1111' ])
plt.title('Heart Disease Frequency for Sex')
plt.xlabel('Sex (0 = Female, 1 = Male)')
plt.xticks(rotation=0)
plt.legend(["Haven't Disease", "Have Disease"])
plt.ylabel('Frequency')
plt.show()
```



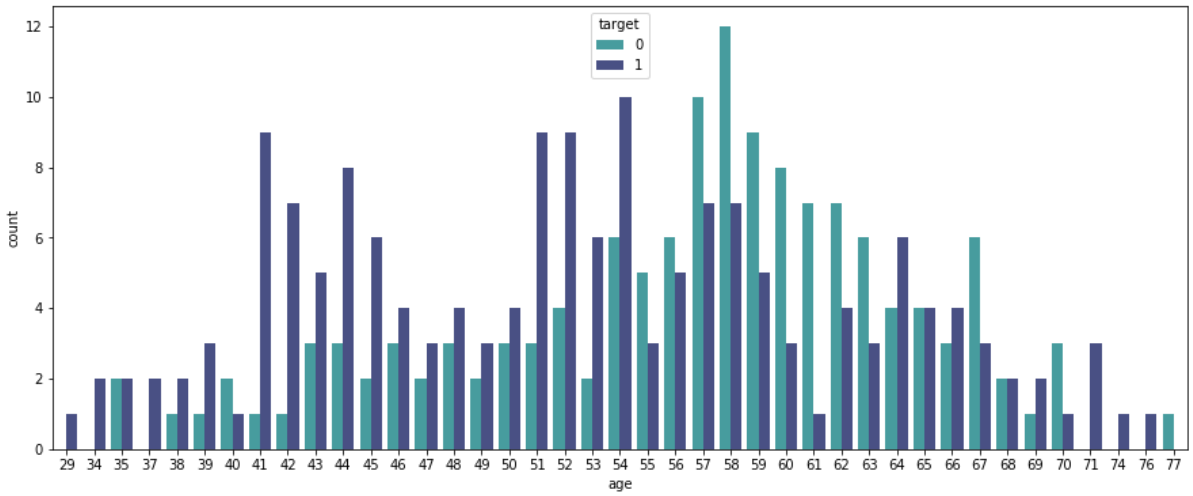
Dapat dilihat pasien berjenis perempuan paling banyak mengidap penyakit jantung.Dari 2 barchat diatas dapat kita gabungkan menjadi 1 grafik. Terlihat berapa pengidap atau tidak dari masing-masing jenis kelamin.

```
sns.countplot(x='sex', data=heart, palette="bwr")
plt.xlabel("Sex (0 = female, 1= male)")
plt.show()
```



selanjutnya melihat pengidap penyakit jantung berdasarkan umur.

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data = heart, hue =
'target',palette='mako_r')
plt.show()
```



Maka diketahui pengidap terbanyak pada umur 54 tahun, dan telihat pola mulai dari umur>40 tahun pengidap penyakit jantung semakin banyak.

Langkah 3: Menentukan variabel dependen dan independen

Variabel **independen** yang digunakan yaitu: **age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca,thal**. Pemilihan variabel dapat menggunakan script berikut.

```
# Variabel independen
x = heart.drop(["target"], axis = 1)
x.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

Selanjutnya, menentukan variabel **dependen**. disini variabel yang digunakan adalah variabel **target**, karena ingin melihat pasien itu dikalsifikasikan terkena penyakit jantung apa tidak.

```
# Variabel dependen
y = heart["target"]
y.head()
```

```
0    1
1    1
2    1
3    1
4    1
Name: target, dtype: int64
```

Langkah 4: Melakukan Training dan Testing

Klasifikasi Naive Bayes terdapat di dalam package Sklearn, maka harus menanggil packages tersebut.

```
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import plot_confusion_matrix
```

kemudian membagi data training dan testing:
→ data training=2%
→ data testing=98%

disini menggunakan random state 123. Nilai random state ini bebas tergantung peneliti, random state menunjukan berapa kali data dilakukan pengacakan. Tapi, untuk kali ini disarankan untuk menggunakan 123 agar hasil random yang didapatkan sama.

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size = 0.2, random_state = 123)
```

kemudian mengaktifkan/memanggil/membuat fungsi klasifikasi Naive bayes.

```
modelnb = GaussianNB()
```

Kemudian memasukkan data training pada fungsi klasifikasi Naive Bayes.

```
nbtrain = modelnb.fit(x_train, y_train)
```

menghitung kelas.

```
nbtrain.class_count_
```

```
array([108., 134.])
```

kemudian menentukan hasil prediksi dati data testing.

```
# Menentukan hasil prediksi dari x_test
y_pred = nbtrain.predict(x_test)
y_pred
```

```
array([1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0], dtype=int64)
```

Hasil prediksi data testing

kemudian mari lihat lagi nilai sebenarnya sebelum diprediksi.

```
nbtrain.predict_proba(x_test)
```

```
array([1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1], dtype=int64)
```

Data testing

dapat dilihat bahwa ada 11 data yang berbeda prediksi. Terlihat berapa probabilitas hasil prediksi dari data testing menggnunaan script berikut.

```
nbtrain.predict_proba(x_test)
```

Langkah 6: Confussion Matrix

Confusion matrix merupakan matrik yang berisi ketepatan prediksi. Untuk melihat Confusion matrix diperlukan packages confusion matrix, maka dapat dipanggil menggunakan script berikut,

```
from sklearn.metrics import confusion_matrix
```

setelah dipanggil, lalu menghitung nilai confusion matrixnya.

```
confusion_matrix(y_test, y_pred)
```

```
array([[27,  3],
       [ 7, 24]], dtype=int64)
```

Hasil confusion matrix

Agar tampilannya confusion matrix lebih rapi dapat menggunakan cara berikut,

```
y_actual = pd.Series([1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1,
1, 1, 0, 1, 1], name = "actual")

y_pred = pd.Series([1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 1, 1, 0,1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1,1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0,
1, 0], name = "prediction")

df_confusion = pd.crosstab(y_actual, y_pred)
df_confusion
```

prediction	0	1
actual		
0	27	3
1	7	24

Hasil confusion matrix

Langkah 7: Nilai akurasi

Kemudian menghitung nilai akurasi. Untuk melihat nilai akurasi cukup lengkap dapat menggunakan packages **clasication_report**, silahkan menggunakan script berikut,

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.90      0.84        30
           1       0.89      0.77      0.83        31

   micro avg       0.84      0.84      0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61
```

Didapatkan dari nilai **micro avg,** nilai akurasinya sebesar 0,84 atau 84%, yang artinya hasil prediksinya sudah sangat bagus. Berdasarakan hasil diatas nilai akurasi dapat dilihat dari nilai **ACC Macro** didapatkan akurasi sebesar 0,08367. Sedangkan untuk melihat kesalahan prediksi dapat melihat nilai **Kappa**.

II. Naïve Bayes Clasifier Untuk Prediksi Penyakit Kutil

Berikut ini akan dilakukan analisis dengan menggunakan *Naive Bayes* pada data perawatan kutil menggunakan *cryotherapy*. Import terlebih dahulu library yang dibutuhkan, untuk sementara import terlebih dahulu library pandas dan numpy.

```
import pandas as pd
import numpy as np
```

Script di atas digunakan untuk mengaktifkan package pandas dan numpy yang akan digunakan pada tahapan analisis. Package pandas sendiri digunakan untuk pengolahan data yang berkaitan dengan data frame, sedangkan package numpy digunakan untuk manipulasi array secara mudah dan cepat.

```
# input data
Cryotherapy=pd.read_excel('Cryotherapy.xlsx')
# Menampilkan data
Cryotherapy.head()
```

Selanjutnya digunakan script untuk menginputkan data dari perangkan komputer ke dalam python.

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
0	1	35	12.00	5	1	100	0
1	1	29	7.00	5	1	96	1
2	1	50	8.00	1	3	132	0
3	1	32	11.75	7	3	750	0
4	1	67	9.25	1	1	42	0

```
# menampilkan informasi data
Cryotherapy.info()
```

Sebelum melakukan analisis, terlebih dahulu digunakan fungsi " *.info* " untuk menampilkan informasi data yang akan dilakukan analisis. Berikut ini *output*-nya.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 7 columns):
sex                 90 non-null int64
age                 90 non-null int64
Time                90 non-null float64
Number_of_Warts     90 non-null int64
Type                90 non-null int64
Area                90 non-null int64
Result_of_Treatment 90 non-null int64
dtypes: float64(1), int64(6)
memory usage: 5.0 KB
```

Data yang akan dianalisis memiliki 7 variabel (kolom) yaitu kolom *sex, age, number of warts, type, area result of treatment* yang memiliki *type* data *integer* dan kolom *time* dengan *type* data *float*. Selanjutnya, digunakan fungsi " *.empty* " untuk melakukan pengecekan apakah terdapat deret data yang kosong.

```
# Mengecek apakah ada deret yang kosong
Cryotherapy.empty
```

```
# Mengecek apakah ada deret yang kosong
Cryotherapy.empty
```

```
False
```

Output menunjukan *False* artinya tidak terdapat deret yang kosong di dalam data yang akan digunakan. Selanjutnya digunakan fungsi " *.size* " untuk melihat ukuran data yang akan digunkaan. Setelah melihat hasilnya, ternyata data yang akan digunakan yaitu sebanyak 630 data.

```
# Melihat ukuran dari data
Cryotherapy.size
```

```
# Melihat ukuran dari data
Cryotherapy.size
```

```
630
```

Tahapan selanjutnya yaitu menentukan variabel independen dan variabel dependen dari data yang akan dianalisis. Berikut *script* yang digunakan.

```
# Variabel independen
x = Cryotherapy.drop(["Result_of_Treatment"], axis = 1)
x.head()
```

	sex	age	Time	Number_of_Warts	Type	Area
0	1	35	12.00	5	1	100
1	1	29	7.00	5	1	96
2	1	50	8.00	1	3	132
3	1	32	11.75	7	3	750
4	1	67	9.25	1	1	42

Kolom *result of treatment* di *drop* atau di hapus dari *data frame* karena akan menjadi variabel dependen.

```
# Variabel dependen
y = Cryotherapy["Result_of_Treatment"]
y.head()
```

```
0    0
1    1
2    0
3    0
4    0
Name: Result_of_Treatment, dtype: int64
```

Setelah menentukan variabel independen dan variabel dependen, selanjutnya kan dilkaukan analisis menggunakan klasifikasi *Naive Bayes*. Pertama dilakukan *Train Test Split untuk membagi dataset menjadi* training set dan test set.

```
# Import train_test_split function
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 = 123)
```

Script di atas membagi dataset menjadi 80% *training* set dan 20% *test* set. Yang artinya dari 630 data, *training* setakan berisi 504 data dan *test* set berisi 120 data. Setalah dilakukan pemisahan, selanjutnya akan dilakukan prediksi pada *training* set dan *test* set.

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

# Mengaktifkan/memanggil/membuat fungsi klasifikasi Naive bayes
modelnb = GaussianNB()

# Memasukkan data training pada fungsi klasifikasi naive bayes
nbtrain = modelnb.fit(x_train, y_train)
nbtrain.class_count_
```

```
nbtrain.class_count_
```

```
array([ 33.,  39.])
```

Selanjutnya, digunakan *script* untuk menentukan hasil prediksi dari *x_test*

```
# Menentukan hasil prediksi dari x_test
y_pred = nbtrain.predict(x_test)
y_pred
```

```
array([1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1], dtype=int64)
```

Untuk menentukan nilai probabilitas dari x_test maka digunakan *script* berikut ini:

```
# Menentukan probabilitas hasil prediksi
nbtrain.predict_proba(x_test)
```

```
array([[  2.34336099e-01,   7.65663901e-01],
       [  1.65201649e-01,   8.34798351e-01],
       [  4.34665582e-01,   5.65334418e-01],
       [  9.28100736e-01,   7.18992637e-02],
       [  1.00002893e-03,   9.98999971e-01],
       [  9.99999912e-01,   8.82148493e-08],
       [  6.08424089e-02,   9.39157591e-01],
       [  8.23832922e-01,   1.76167078e-01],
       [  2.88581316e-03,   9.97114187e-01],
       [  9.98354676e-01,   1.64532426e-03],
       [  1.00000000e+00,   9.56156555e-57],
       [  2.15128221e-01,   7.84871779e-01],
       [  1.79795569e-03,   9.98202044e-01],
       [  9.99985925e-01,   1.40754598e-05],
       [  1.16748529e-02,   9.88325147e-01],
       [  9.85529535e-01,   1.44704650e-02],
       [  8.71986684e-01,   1.28013316e-01],
       [  4.43317684e-02,   9.55668232e-01]])
```

Setelah diperoleh nilai prediksi (*y_pred*), maka tahapan selanjutnya yaitu melakukan *Confussion Matrix*.

```
# import confusion_matrix model
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
array([[7, 2],
       [1, 8]], dtype=int64)
```

Hasil di atas merupakan hasil *confusion matrix*, untuk mempermudah dalam membaca, maka digunakan *script* untuk merapihkan hasil *confusion matrix*.

```
# Merapikan hasil confusion matrix
y_actual1 = pd.Series([1, 0,1,0,1,0,1,0,1,0,0,1,1,0,1,1,0,0],
name = "actual")
y_pred1 = pd.Series([1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
1, 0, 0, 1], name = "prediction")
df_confusion = pd.crosstab(y_actual1, y_pred1)df_confusion
```

prediction	0	1
actual		
0	7	2
1	1	8

Hasil di atas menunjukan bahwa, prediksi *traetment* dinyatkan gagal dan ternyata *traetment* gagal sebanyak 7, prediksi *traetment* gagal dan ternyata *traetment* berhasil 1, prediksi *traetment* berhasil dan ternyata *traetment* gagal sebanyak 2 dan prediksi *traetment* berhasil dan ternyata treatmen berhasil sebanyak 8.
Karena sebagian besar prediksi dan hasil nya sesuai maka dapat di katakan bahwa *tratment cryotherapy* baik digunakan unutk melakukan perawatan karena kutil.
Selanjutnya, akan dilakukan perhitungan nilai akurasi

```
# Menghitung nilai akurasi dari klasifikasi naive bayes
from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.88	0.78	0.82	9
1	0.80	0.89	0.84	9
avg / total	0.84	0.83	0.83	18

SOAL :

1. Download data penyakit Immunotherapy dari https://archive.ics.uci.edu/ml/machine-learning-databases/00428/ atau download dari classroom dengan nama file Immunotherapy.csv Lakukan tahapan-tahapan dalam menentukan klasifikasi untuk memprediksi dengan algoritma Naïve Bayes Classifier.

2. Lakukan analisis dengan menggunakan naïve bayes classifier data mengenai play tenis berikut :

No	Play	Outlook	Temperature	Humidity	Windy
1	No	Sunny	Hot	High	False

2	No	Sunny	Hot	High	True
3	Yes	Cloudy	Hot	High	False
4	Yes	Rainy	Mild	High	False
5	Yes	Rainy	Cool	Normal	False
6	No	Rainy	Cool	Normal	True
7	Yes	Cloudy	Coo	Normal	True
8	No	Sunny	Mild	High	False
9	Yes	Sunny	Coold	Normal	False
10	Yes	Rainy	Mild	Normal	False
11	Yes	Sunny	Mild	Normal	True
12	Yes	Cloudy	Mild	High	True
13	Yes	Cloudy	Hot	Normal	False
14	No	Rainy	Mild	High	True

Data test yang digunakan adalah : **outlook**: rainy, **temperature**: cool, **humidity**: high, **windy**= true