In [1]:

```
# nama : donny pratama
# kelas : 5p42
# nim : 20.240.0116
```

In [3]:

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

In [5]:

```
heart = pd.read_csv("D:\heart.csv")
heart.head()
```

Out[5]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	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

In [6]:

```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [7]:

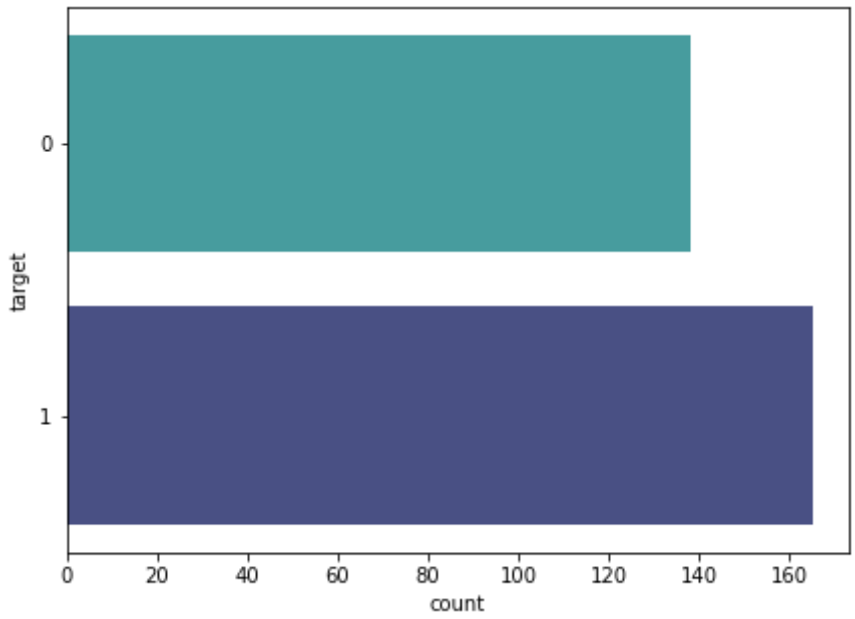
```
heart.target.value_counts()
```

Out[7]:

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

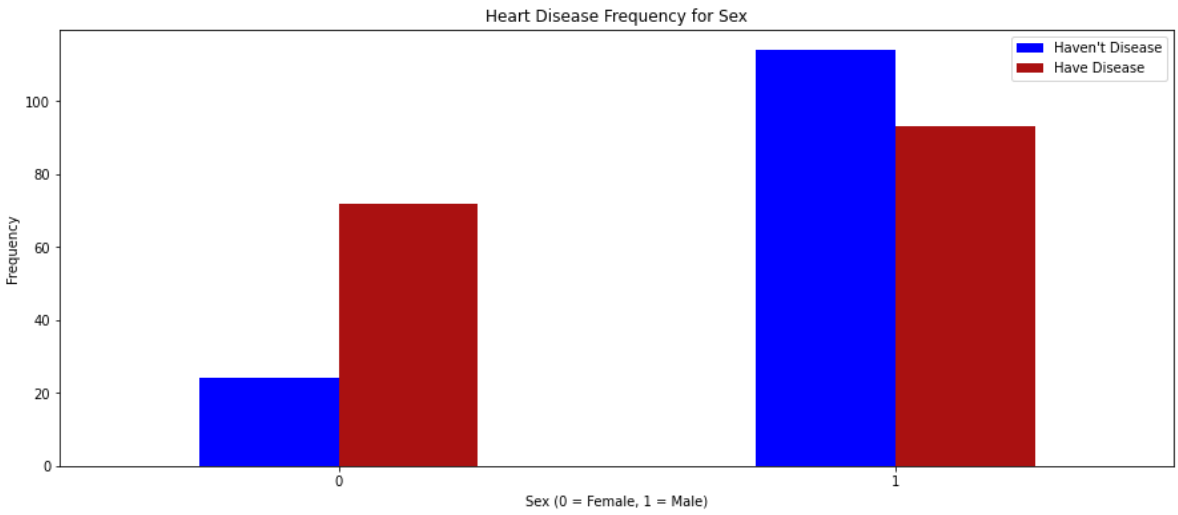
In [9]:

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



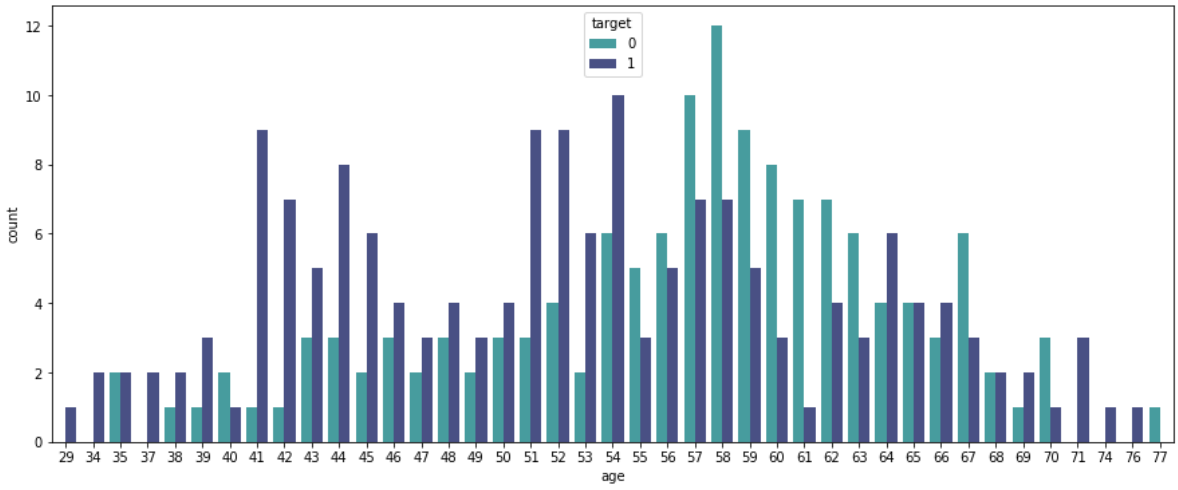
In [14]:

```
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()
```



In [15]:

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



In [16]:

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

Out[16]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
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

In [17]:

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

Out[17]:

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

In [18]:

```
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
```

In [19]:

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

In [21]:

```
modelnb = GaussianNB()
```

In [22]:

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

In [24]:

```
nbtrain.class_count_
```

Out[24]:

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

In [25]:

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

Out[25]:

```
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)
```

In [26]:

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

Out[26]:

```
array([[5.82571613e-04, 9.99417428e-01],
       [9.99240610e-01, 7.59390366e-04],
       [9.99938869e-01, 6.11309321e-05],
       [9.99992658e-01, 7.34177465e-06],
       [3.63366729e-03, 9.96366333e-01],
       [9.99951069e-01, 4.89308903e-05],
       [8.58720034e-03, 9.91412800e-01],
       [5.98819913e-01, 4.01180087e-01],
       [5.96228807e-04, 9.99403771e-01],
       [9.33511439e-01, 6.64885614e-02],
       [9.97942318e-01, 2.05768198e-03],
       [4.57117589e-01, 5.42882411e-01],
       [9.99999986e-01, 1.36224796e-08],
       [9.99995730e-01, 4.26988712e-06],
       [5.16991611e-01, 4.83008389e-01],
       [1.00000000e+00, 5.08353833e-11],
       [9.10174530e-01, 8.98254697e-02],
       [9.86808541e-01, 1.31914589e-02],
       [5.72076923e-01, 4.27923077e-01],
       [2.24348439e-01, 7.75651561e-01],
       [1.35283843e-01, 8.64716157e-01],
       [9.99163302e-01, 8.36698360e-04],
       [4.11827050e-05, 9.99958817e-01],
       [9.82142245e-01, 1.78577552e-02],
       [9.87487847e-01, 1.25121528e-02],
       [6.12756815e-01, 3.87243185e-01],
       [9.98445679e-01, 1.55432096e-03],
       [5.27229563e-01, 4.72770437e-01],
       [9.99999798e-01, 2.01877057e-07],
       [8.92625300e-01, 1.07374700e-01],
       [9.99977531e-01, 2.24690154e-05],
       [2.83019736e-01, 7.16980264e-01],
       [6.30801388e-02, 9.36919861e-01],
       [9.99898787e-01, 1.01213253e-04],
       [3.84301563e-01, 6.15698437e-01],
       [2.48969111e-04, 9.99751031e-01],
       [3.27891000e-02, 9.67210900e-01],
       [1.25745349e-01, 8.74254651e-01],
       [2.46760601e-04, 9.99753239e-01],
       [2.15045133e-02, 9.78495487e-01],
       [6.63584159e-04, 9.99336416e-01],
       [4.14585955e-01, 5.85414045e-01],
       [9.99647022e-01, 3.52978207e-04],
       [5.65078130e-04, 9.99434922e-01],
       [9.19585446e-03, 9.90804146e-01],
       [9.87381063e-01, 1.26189369e-02],
       [9.99990910e-01, 9.09037606e-06],
       [7.91427151e-01, 2.08572849e-01],
       [1.38836067e-02, 9.86116393e-01],
       [9.89442682e-03, 9.90105573e-01],
       [9.96646980e-01, 3.35301971e-03],
       [9.98756337e-01, 1.24366336e-03],
       [6.90049722e-04, 9.99309950e-01],
       [5.95604279e-01, 4.04395721e-01],
       [8.21909495e-02, 9.17809050e-01],
```

```
       [1.52842321e-01, 8.47157679e-01],
       [9.95476149e-01, 4.52385096e-03],
       [2.45417179e-02, 9.75458282e-01],
       [9.92418418e-01, 7.58158236e-03],
       [8.00379195e-02, 9.19962080e-01],
       [9.97808961e-01, 2.19103887e-03]])
```

In [27]:

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

In [28]:

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

Out[28]:

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

In [29]:

```
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
```

Out[29]:

prediction	0	1
actual		
0	27	3
1	7	24

In [30]:

```
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

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61
```

In [31]:

```
# naive bayes untuk prediksi penyakit kutil
import pandas as pd
import numpy as np
```

In [33]:

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

Out[33]:

	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

In [34]:

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

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

In [35]:

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

Out[35]:

```
False
```

In [36]:

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

Out[36]:

```
630
```

In [38]:

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

Out[38]:

	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

In [39]:

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

Out[39]:

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

In [40]:

```
# 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)
```

In [41]:

```
# 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_
```

Out[41]:

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

In [42]:

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

Out[42]:

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

In [43]:

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

Out[43]:

```
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]])
```

In [44]:

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

Out[44]:

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

In [47]:

```
# 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 = "predict
df_confusion = pd.crosstab(y_actual1, y_pred1)df_confusion
```

```
  Input In [47]
    df_confusion = pd.crosstab(y_actual1, y_pred1)df_confusion
                                                  ^
SyntaxError: invalid syntax
```

In [ ]:

```

```