Notebook_KLASIFIKASI_A_Heart_RandomForest_VS_LogisticRegression

October 23, 2025

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import time

from sklearn.metrics import (
    confusion_matrix,
    ConfusionMatrixDisplay,
    classification_report
)
```

[2]:
```
df = pd.read_csv('heart.csv')


df.head()
```

[2]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

[3]:
```
df["HeartDisease"].unique()
```

[3]:
```
array([0, 1])
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

[5]:
```
df.describe()
```

[5]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

[6]:
```
df.isnull().sum()
```

[6]:
```
Age               0
Sex               0
ChestPainType     0
```

```
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

[7]:
```
numerical_column = df.select_dtypes(include=np.number).columns
categorical_column = df.select_dtypes(exclude=np.number).columns

numerical_column
```

[7]:
```
Index(['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak',
       'HeartDisease'],
      dtype='object')
```

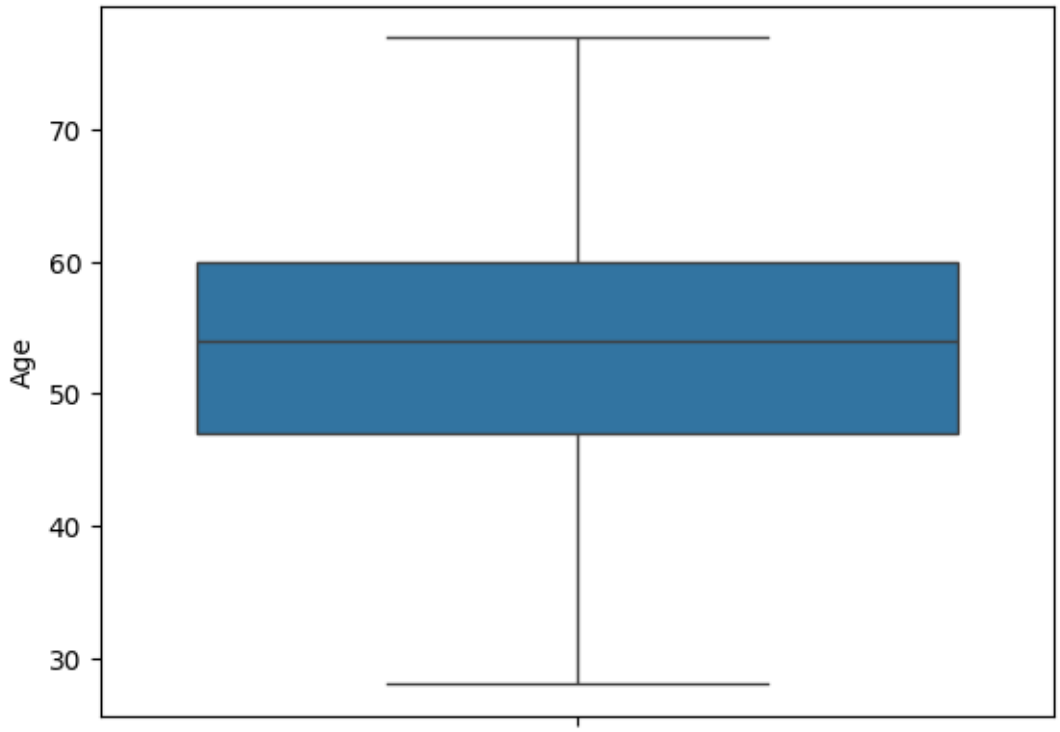
[8]:
```
categorical_column
df["HeartDisease"].unique()
```

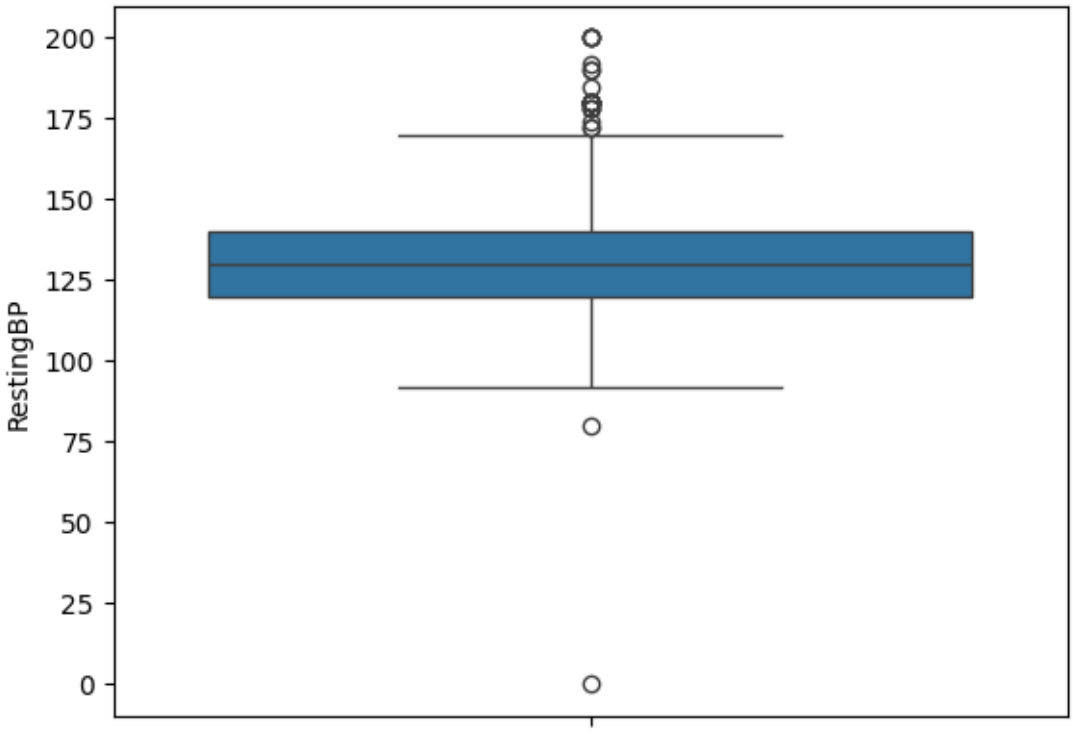
[8]:
```
array([0, 1])
```

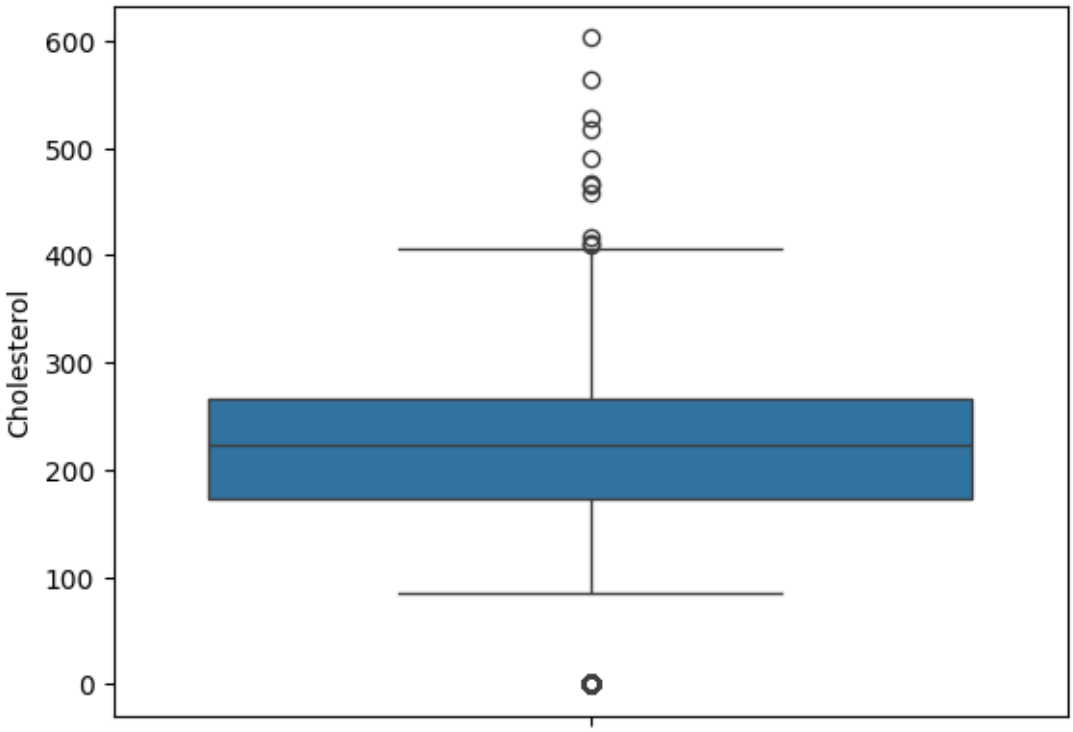
[9]:
```
numerical_column = ['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR',␣
 ↪'Oldpeak']
df["HeartDisease"].unique()
```

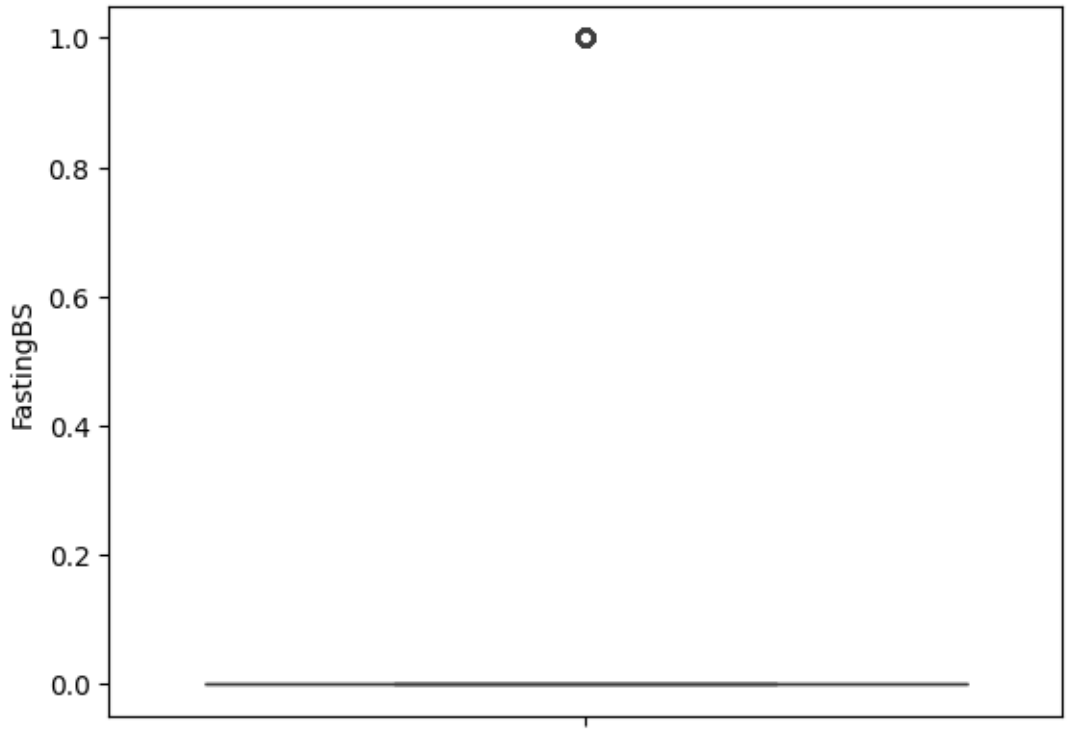
[9]:
```
array([0, 1])
```

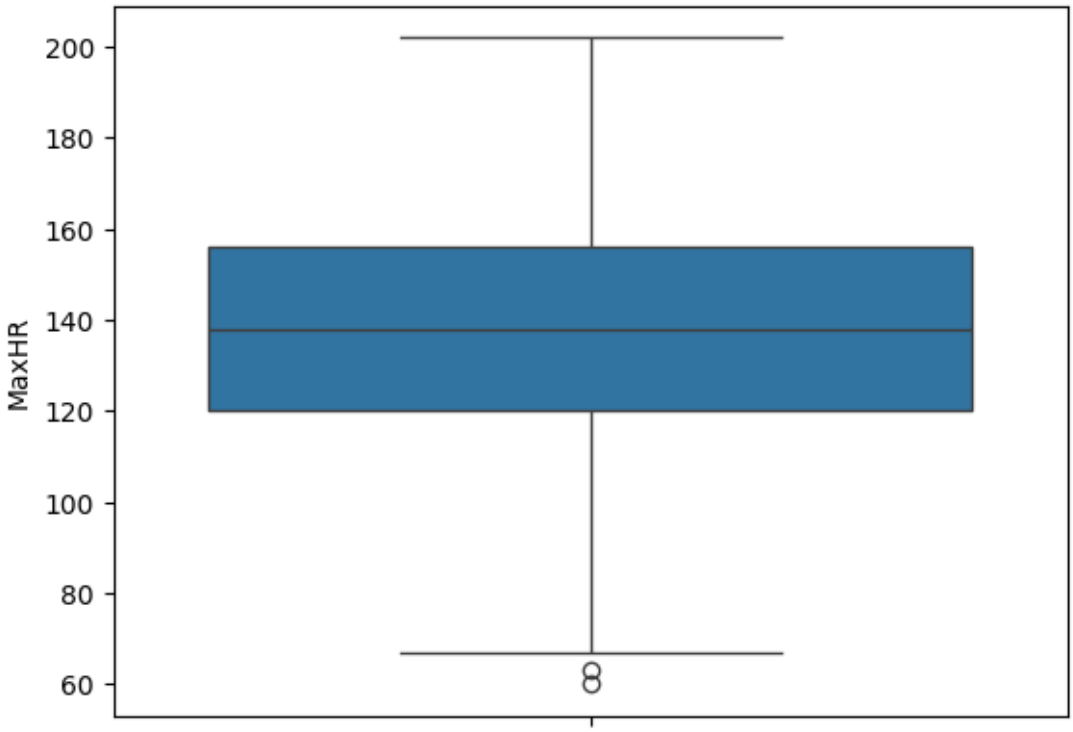
[10]:
```
for column in numerical_column:
  sns.boxplot(df[column])
  plt.show()
```

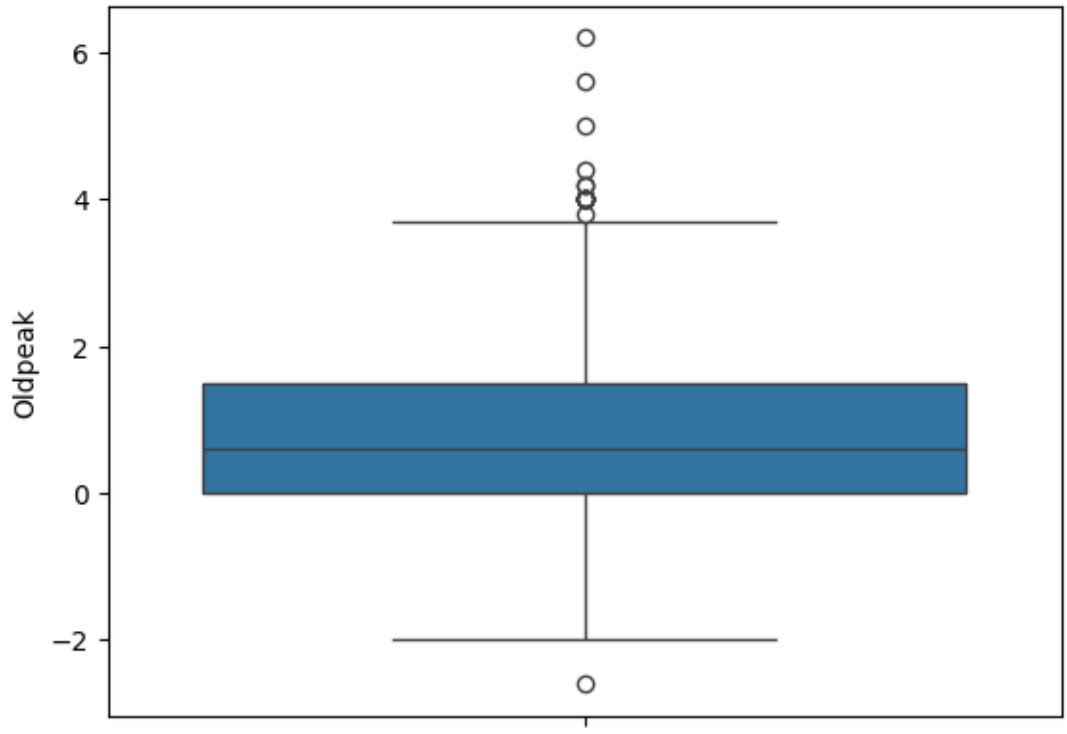

Age
70
60
50
40
30


RestingBP
200
175
150
125
100
75
50
25
0


Cholesterol
600
500
400
300
200
100
0


FastingBS
1.0
0.8
0.6
0.4
0.2
0.0


MaxHR
200
180
160
140
120
100
80
60


Oldpeak
6
4
2
0
−2

[11]:
```
df.dropna(inplace=True)

df.isnull().sum()
```

[11]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

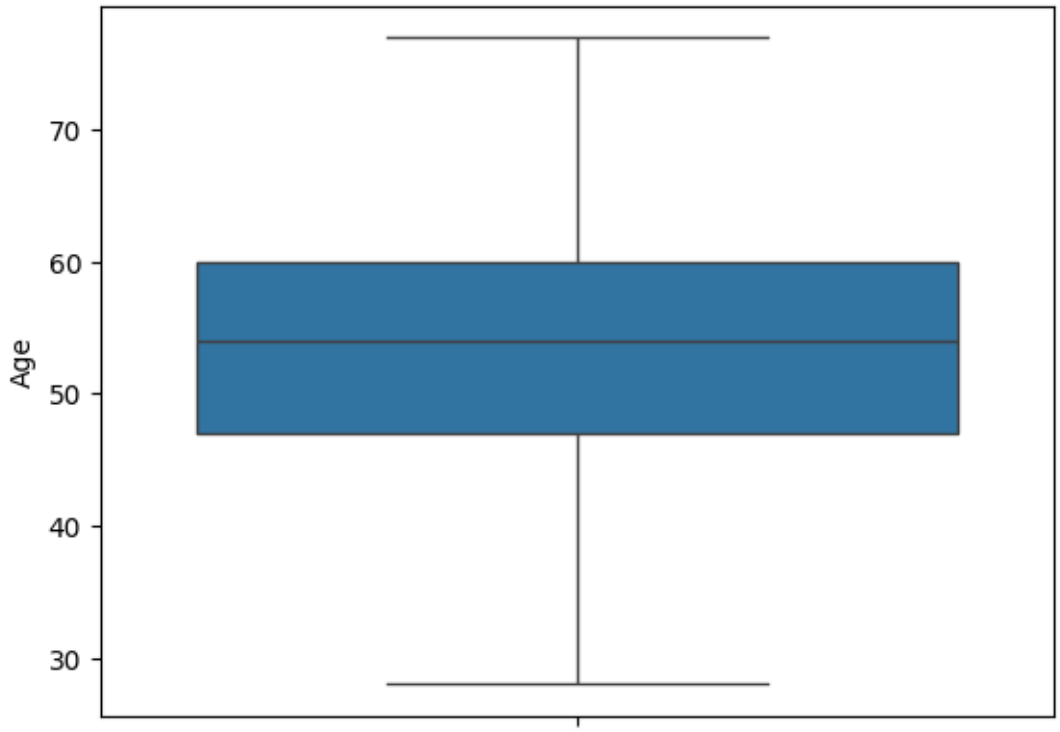
[12]:
```
numerical_column
```

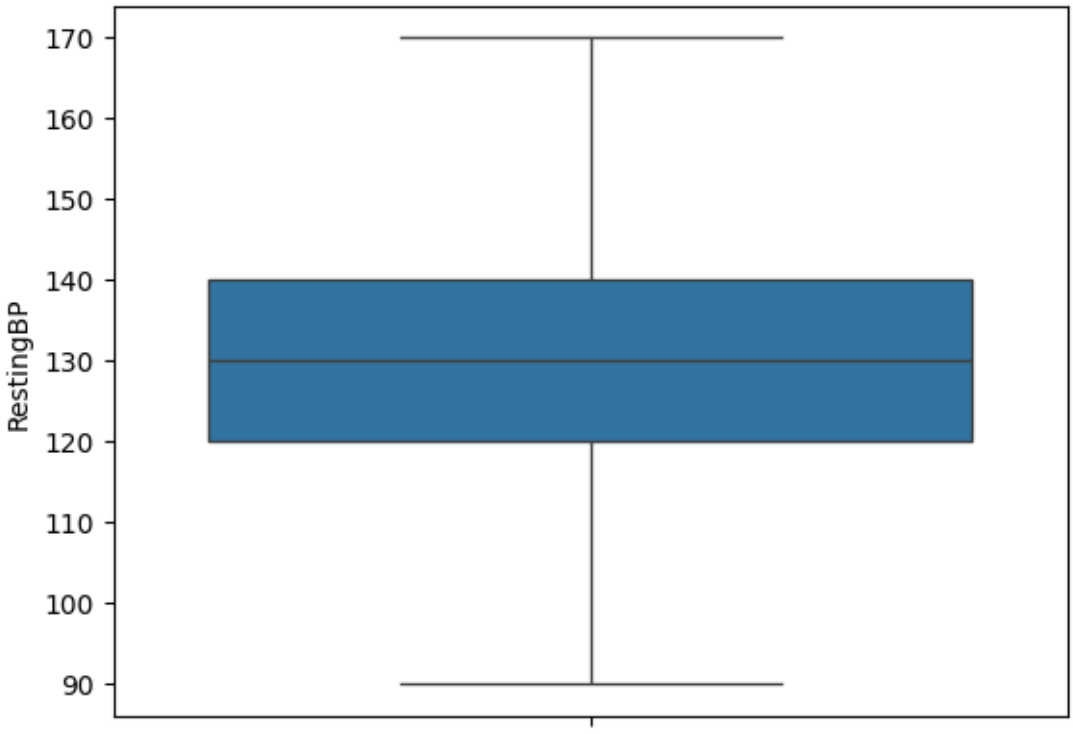
[12]:
```
['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak']
```

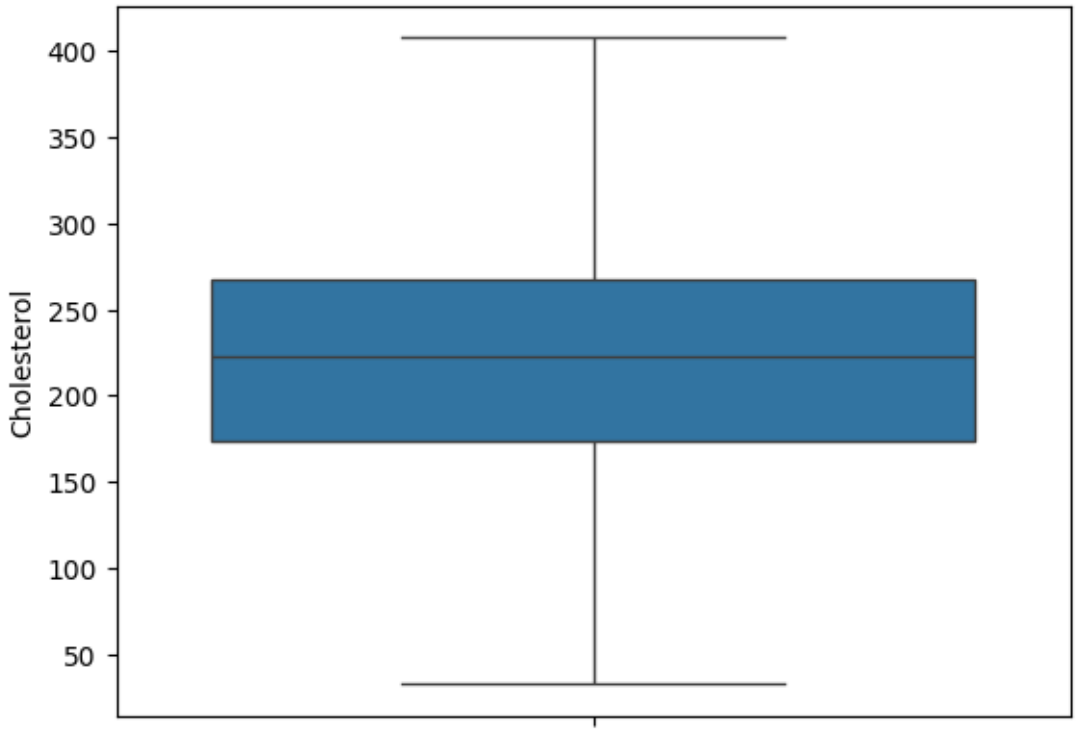
[13]:
```
print(df["HeartDisease"].unique())
for column in numerical_column:          # pastikan 'HeartDisease' tidak ada di␣
 ↪sini
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5*IQR
    upper = Q3 + 1.5*IQR

    df[column] = df[column].clip(lower=lower, upper=upper)

print(df["HeartDisease"].unique())
```

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

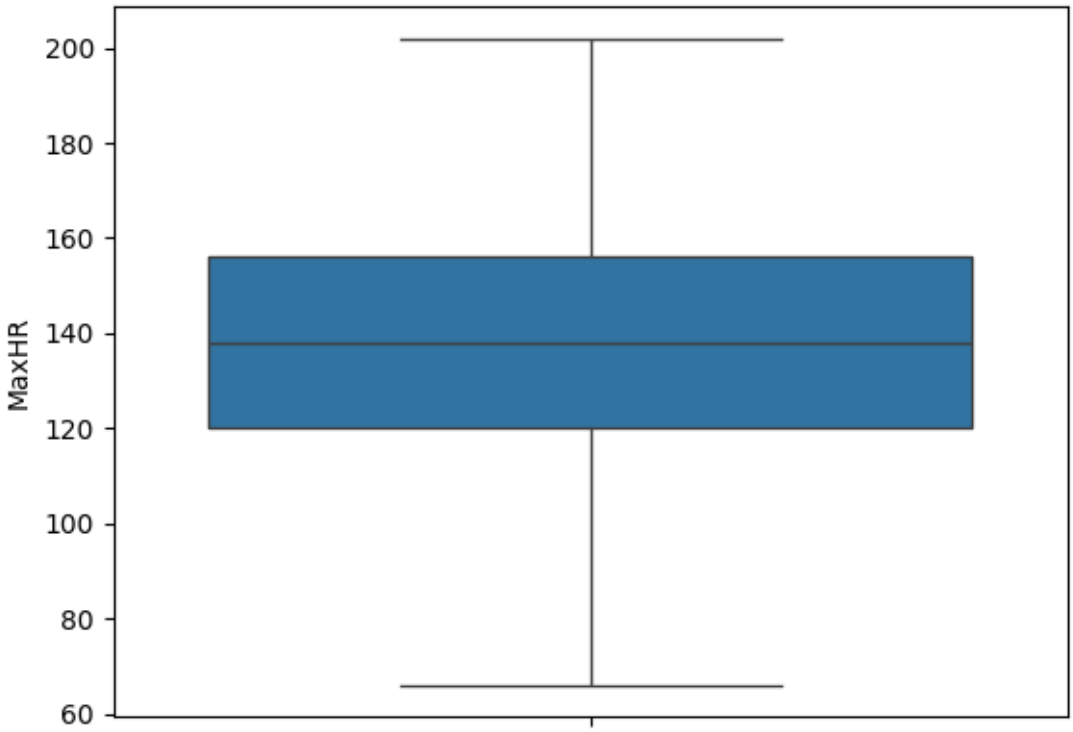
[14]:
```
for column in numerical_column:
  sns.boxplot(df[column])
  plt.show()
```

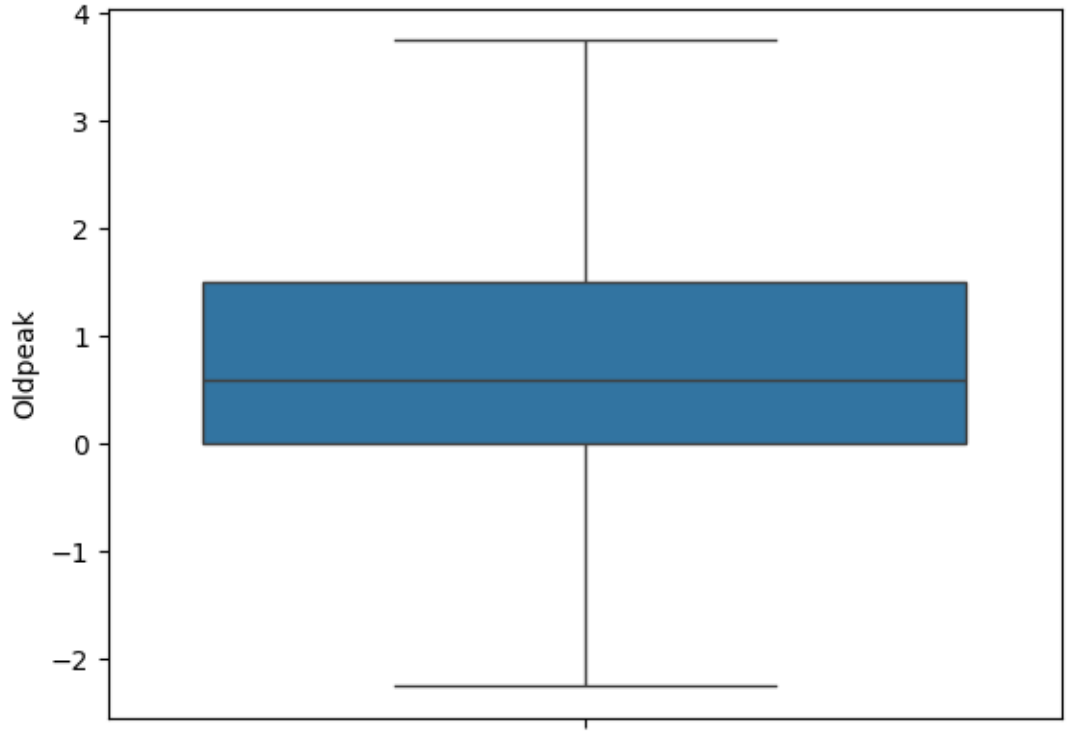

Age
70
60
50
40
30


RestingBP
170
160
150
140
130
120
110
100
90


Cholesterol
400
350
300
250
200
150
100
50


FastingBS
0.04
0.02
0.00
−0.02
−0.04


MaxHR
200
180
160
140
120
100
80
60


Oldpeak
4
3
2
1
0
−1
−2

[15]:
```
df["HeartDisease"].unique()
```

[15]:
```
array([0, 1])
```

[16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    float64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(2), int64(5), object(5)
memory usage: 86.2+ KB
```

[17]:
```
categorical_column
```

[17]:
```
Index(['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope'],
dtype='object')
```

[18]:
```
df[categorical_column].head(10)
```

[18]:

	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope
0	M	ATA	Normal	N	Up
1	F	NAP	Normal	N	Flat
2	M	ATA	ST	N	Up
3	F	ASY	Normal	Y	Flat
4	M	NAP	Normal	N	Up
5	M	NAP	Normal	N	Up
6	F	ATA	Normal	N	Up
7	M	ATA	Normal	N	Up
8	M	ASY	Normal	Y	Flat
9	F	ATA	Normal	N	Up

ChestPainType_ATA | ChestPainType_NAP | ChestPainType_ASY True False False False True False

[19]:
```
from sklearn.preprocessing import LabelEncoder

df[categorical_column]=df[categorical_column].astype(str)
le = LabelEncoder()
for column in categorical_column:
    df[column] = le.fit_transform(df[column])

df[categorical_column]
```

[19]:

	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope
0	1	1	1	0	2
1	0	2	1	0	1
2	1	1	2	0	2
3	0	0	1	1	1
4	1	2	1	0	2
..	…	…	…	…	…
913	1	3	1	0	1
914	1	0	1	0	1
915	1	0	1	1	1
916	0	1	0	0	1
917	1	2	1	0	2

[918 rows x 5 columns]

[20]:
```
df.head()
```

[20]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	\
0	40	1	1	140	289.0	0	1	
1	49	0	2	160	180.0	0	1	
2	37	1	1	130	283.0	0	2	
3	48	0	0	138	214.0	0	1	
4	54	1	2	150	195.0	0	1	

	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	172	0	0.0	2	0
1	156	0	1.0	1	1
2	98	0	0.0	2	0
3	108	1	1.5	1	1
4	122	0	0.0	2	0

[21]:
```
df["HeartDisease"].unique()
```

[21]:
```
array([0, 1])
```

[22]:
```
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.model_selection import train_test_split

X = df.drop(columns="HeartDisease")
y = df["HeartDisease"].astype(int)
```

[23]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[24]:
```
X_scaled
```

[24]:
```
array([[-1.4331398 ,  0.51595242,  0.22903206, …, -0.8235563 ,
        -0.85127647,  1.05211381],
       [-0.47848359, -1.93816322,  1.27505906, …, -0.8235563 ,
         0.11853217, -0.59607813],
       [-1.75135854,  0.51595242,  0.22903206, …, -0.8235563 ,
        -0.85127647,  1.05211381],
       …,
       [ 0.37009972,  0.51595242, -0.81699495, …,  1.21424608,
         0.3124939 , -0.59607813],
       [ 0.37009972, -1.93816322,  0.22903206, …, -0.8235563 ,
        -0.85127647, -0.59607813],
       [-1.64528563,  0.51595242,  1.27505906, …, -0.8235563 ,
        -0.85127647,  1.05211381]])
```

1 CEK TIPE DARI NILAI DI KOLOM Y

[25]:
```
from sklearn.utils.multiclass import type_of_target
print(type_of_target(y))
```

```
binary
```

[26]:
```
y.describe
```

[26]:
```
<bound method NDFrame.describe of 0      0
1      1
2      0
3      1
4      0
      ..
913    1
914    1
915    1
916    1
917    0
```

```
Name: HeartDisease, Length: 918, dtype: int64>
```

1.1 MEMILIH SELECTOR K-BEST

[27]:
```
selector = SelectKBest(score_func=mutual_info_classif)
X_new = selector.fit_transform(X_scaled, y)

X_new
```

[27]:
```
array([[-1.4331398 ,  0.51595242,  0.22903206, …, -0.8235563 ,
        -0.85127647,  1.05211381],
       [-0.47848359, -1.93816322,  1.27505906, …, -0.8235563 ,
         0.11853217, -0.59607813],
       [-1.75135854,  0.51595242,  0.22903206, …, -0.8235563 ,
        -0.85127647,  1.05211381],
       …,
       [ 0.37009972,  0.51595242, -0.81699495, …,  1.21424608,
         0.3124939 , -0.59607813],
       [ 0.37009972, -1.93816322,  0.22903206, …, -0.8235563 ,
        -0.85127647, -0.59607813],
       [-1.64528563,  0.51595242,  1.27505906, …, -0.8235563 ,
        -0.85127647,  1.05211381]])
```

2 RANDOM FOREST DENGAN PIPELINE

[47]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV,␣
 ↪StratifiedKFold

pipe = Pipeline(steps=[
    ("scaler", StandardScaler()),
    ("feature_selection", SelectKBest(score_func=mutual_info_classif)),
    ("model", RandomForestClassifier())
])

param_grid = {
    "model__max_depth": [5, 10, 14, 20, 30, 35],
    "model__random_state": [42, 52, 68]
}

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.25,␣
 ↪random_state=68)
```

```
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=68)

gs = GridSearchCV(
    estimator=pipe,
    param_grid=param_grid,
    scoring='accuracy',
    cv=cv,
    n_jobs=-1,
    verbose=2
)

gs.fit(X_train, y_train)

print("Best Score:", gs.best_score_)
print("Best Params:", gs.best_params_)
print("Best Estimator:", gs.best_estimator_)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Best Score: 0.8765048132867872
Best Params: {'model__max_depth': 5, 'model__random_state': 68}
Best Estimator: Pipeline(steps=[('scaler', StandardScaler()),
                ('feature_selection',
                 SelectKBest(score_func=<function mutual_info_classif at
0x7fd2855280e0>)),
                ('model',
                 RandomForestClassifier(max_depth=5, random_state=68))])
```

[46]:
```
from sklearn.metrics import classification_report
y_pred = gs.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.71      0.79       110
           1       0.77      0.92      0.84       120

    accuracy                           0.82       230
   macro avg       0.83      0.81      0.81       230
weighted avg       0.83      0.82      0.81       230
```

```
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.4s
```

```
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.7s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.8s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=52; total time=   1.2s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.8s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.6s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.8s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.5s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.9s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.8s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   1.1s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.8s
[CV] END …model__max_depth=10, model__random_state=52; total time=   1.1s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
```

```
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=10, model__random_state=52; total time=   1.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.8s
[CV] END …model__max_depth=14, model__random_state=42; total time=   1.0s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.6s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.8s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.7s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.6s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.6s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.8s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.7s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.6s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.7s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.6s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.6s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
```

```
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=5, model__random_state=68; total time=   1.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   1.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.9s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.6s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.8s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.8s
[CV] END …model__max_depth=14, model__random_state=42; total time=   1.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.6s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.4s
```

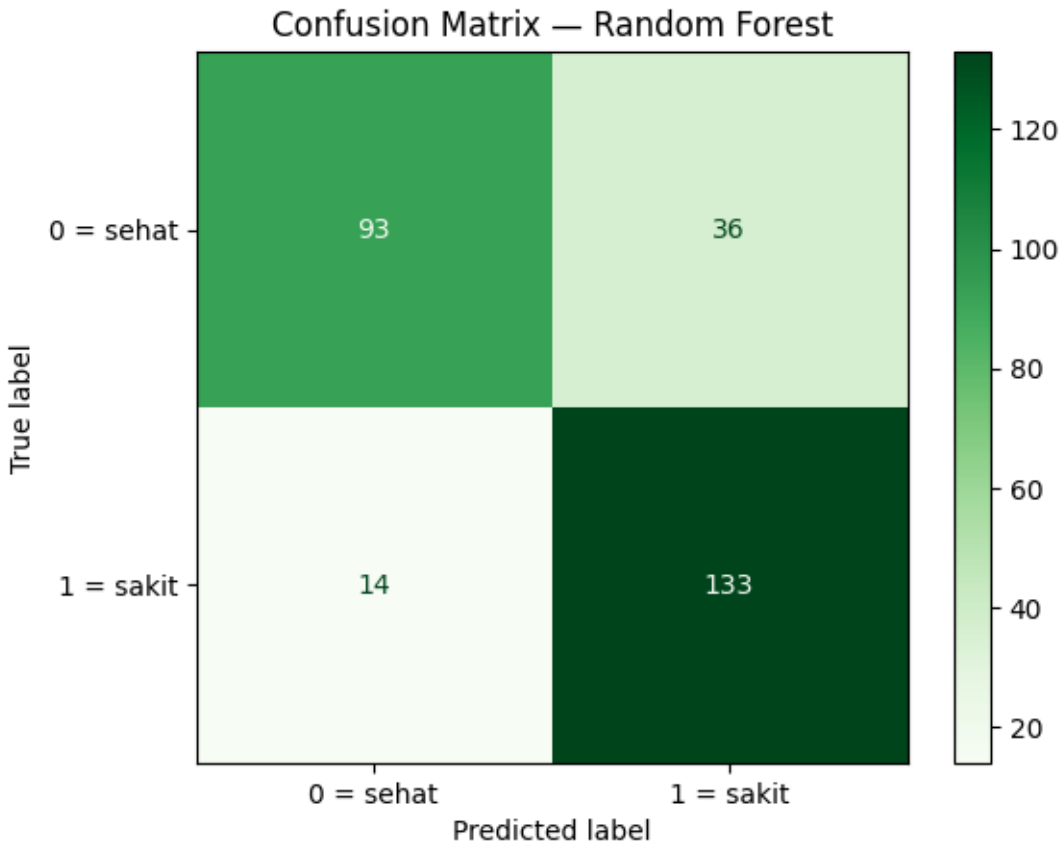
```
[CV] END …model__max_depth=10, model__random_state=68; total time=   1.2s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.8s
[CV] END …model__max_depth=14, model__random_state=42; total time=   1.2s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.8s
[CV] END …model__max_depth=14, model__random_state=42; total time=   1.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   1.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.6s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.5s
[CV] END …model__max_depth=14, model__random_state=68; total time=   1.2s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.8s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.6s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.5s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.7s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.2s
```

```
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.8s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.6s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.5s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.7s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.5s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=14, model__random_state=68; total time=   1.2s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.6s
[CV] END …model__max_depth=14, model__random_state=68; total time=   1.2s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.6s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.8s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.6s
[CV] END …model__max_depth=5, model__random_state=42; total time=   1.0s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.7s
```

```
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   1.0s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.7s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.7s
[CV] END …model__max_depth=5, model__random_state=68; total time=   1.1s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   1.2s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.5s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=42; total time=   1.1s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.6s
```

[40]:
```
print("CV Score (F1) terbaik:", gs.best_score_)
print("Kombinasi model terbaik:", gs.best_estimator_)

lr_test_score = gs.best_estimator_.score(X_test,y_test)
print("\nSkor Test (akurasi) Random Forest:", lr_test_score)

selector = gs.best_estimator_.named_steps['scaler']
if hasattr(selector, 'get_support'):
    mask = selector.get_support()
    selected = np.array(X.columns)
    print("\nFitur terbaik (terpilih):", selected)

lr_pred = gs.predict(X_test)
cm_lr = confusion_matrix(y_test, lr_pred)
disp_lr = ConfusionMatrixDisplay(confusion_matrix=cm_lr, display_labels=['0 =␣
 ↪sehat','1 = sakit'])
disp_lr.plot(cmap=plt.cm.Greens)
plt.title("Confusion Matrix - Random Forest")
plt.show()

best_model = gs.best_estimator_
y_pred = best_model.predict(X_test)
print("\nClassification Report - Random Forest:\n",␣
 ↪classification_report(y_test, lr_pred))
```

```
CV Score (F1) terbaik: 0.8769016472868216
Kombinasi model terbaik: Pipeline(steps=[('scaler', StandardScaler()),
                ('feature_selection',
                 SelectKBest(score_func=<function mutual_info_classif at
0x7fd2855280e0>)),
                ('model',
                 RandomForestClassifier(max_depth=5, random_state=42))])

Skor Test (akurasi) Random Forest: 0.8188405797101449
```




```
Classification Report - Random Forest:
              precision    recall  f1-score   support

           0       0.87      0.72      0.79       129
           1       0.79      0.90      0.84       147

    accuracy                           0.82       276
   macro avg       0.83      0.81      0.81       276
weighted avg       0.83      0.82      0.82       276
```

```
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.2s
```

```
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
```

```
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.2s
```

```
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.4s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=20, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=52; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=5, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=42; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.4s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=10, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=14, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
[CV] END …model__max_depth=10, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=30, model__random_state=68; total time=   0.2s
[CV] END …model__max_depth=5, model__random_state=52; total time=   0.4s
[CV] END …model__max_depth=35, model__random_state=68; total time=   0.3s
```

[30]:
```
import pickle
with open('modelRandomForest.pkl', 'wb') as file:
    pickle.dump(pipe, file)
```

[31]:
```
y_pred = gs.predict(X_test)
```

[32]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support
```

```
              0       0.87      0.72      0.79       129
              1       0.79      0.90      0.84       147

    accuracy                            0.82       276
   macro avg       0.83      0.81      0.81       276
weighted avg       0.83      0.82      0.82       276
```

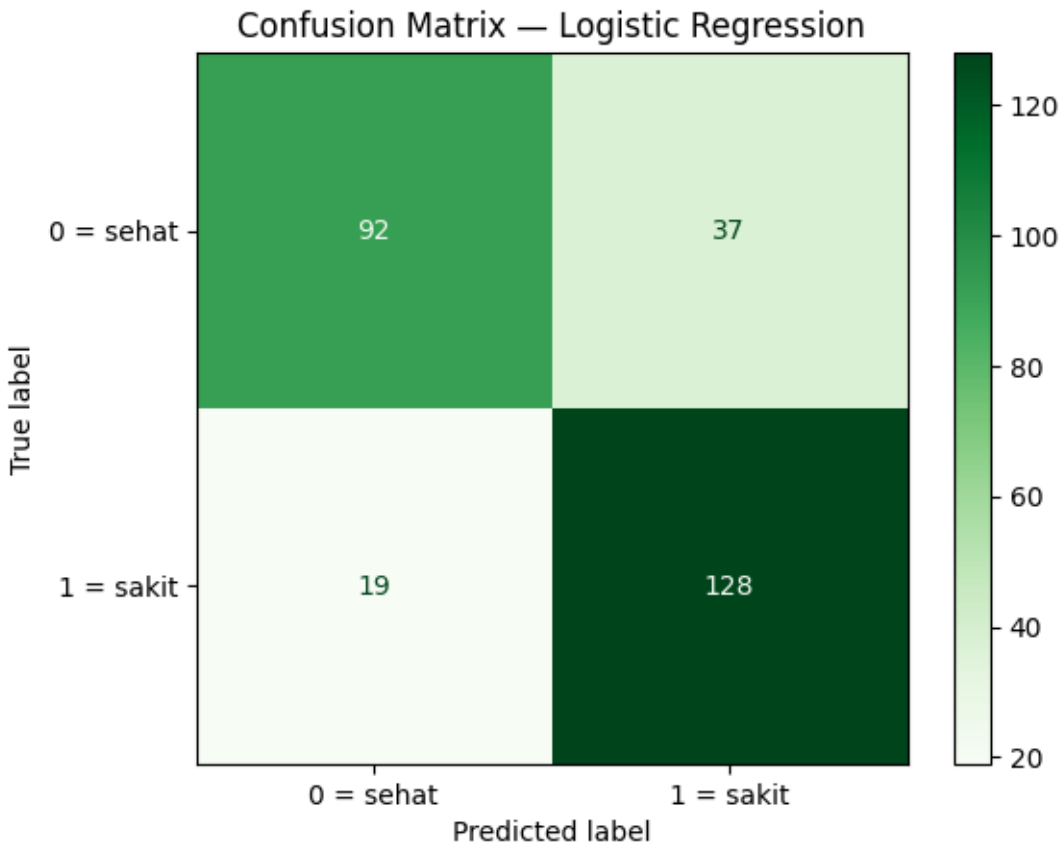
3 LOGISTIC REGRESSION DENGAN PIPELINE STANDARD SCALER

[55]:
```
from sklearn.model_selection import train_test_split, GridSearchCV,␣
 ↪StratifiedKFold
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest, SelectPercentile, f_classif,␣
 ↪mutual_info_classif
from sklearn.metrics import classification_report
pipe = Pipeline(steps=[
    ("scaler", StandardScaler()),
    ("selector", SelectKBest()),
    ("model", LogisticRegression(max_iter=1000))
])

param_grid = [
    {
        "scaler": [StandardScaler(), MinMaxScaler()],
        "selector": [
            SelectKBest(score_func=f_classif),
            SelectKBest(score_func=mutual_info_classif)
        ],
        "selector__k": [5, 10, 20, 'all'],
        "model__C": [0.01, 0.1, 1, 10],
        "model__class_weight": [None, "balanced"],
        "model__solver": ["lbfgs"]
    },
    {
        "scaler": [StandardScaler(), MinMaxScaler()],
        "selector": [
            SelectPercentile(score_func=f_classif),
            SelectPercentile(score_func=mutual_info_classif)
        ],
        "selector__percentile": [10, 20, 30, 50, 100],
        "model__C": [0.01, 0.1, 1, 10],
```

```
packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning:
invalid value encountered in divide
  f = msb / msw
```

[56]:
```
print("CV Score (F1) terbaik:", gs.best_score_)
print("Kombinasi model terbaik:", gs.best_estimator_)

lr_test_score = gs.best_estimator_.score(X_test,y_test)
print("\nSkor Test (akurasi) Logistic Regression:", lr_test_score)

selector = gs.best_estimator_.named_steps['selector']
if hasattr(selector, 'get_support'):
    mask = selector.get_support()
    selected = np.array(X.columns)
    print("\nFitur terbaik (terpilih):", selected)

lr_pred = gs.predict(X_test)
cm_lr = confusion_matrix(y_test, lr_pred)
disp_lr = ConfusionMatrixDisplay(confusion_matrix=cm_lr, display_labels=['0 =␣
 ↪sehat','1 = sakit'])
disp_lr.plot(cmap=plt.cm.Greens)
plt.title("Confusion Matrix - Logistic Regression")
plt.show()

best_model = gs.best_estimator_
y_pred = best_model.predict(X_test)
print("\nClassification Report - Logistic Regression:\n",␣
 ↪classification_report(y_test, lr_pred))
```

```
CV Score (F1) terbaik: 0.8623948305991634
Kombinasi model terbaik: Pipeline(steps=[('scaler', StandardScaler()),
('selector', SelectKBest()),
                ('model', LogisticRegression(C=0.1, max_iter=1000))])

Skor Test (akurasi) Logistic Regression: 0.7971014492753623

Fitur terbaik (terpilih): ['Age' 'Sex' 'ChestPainType' 'RestingBP' 'Cholesterol'
'FastingBS'
 'RestingECG' 'MaxHR' 'ExerciseAngina' 'Oldpeak' 'ST_Slope']
```




```
Classification Report - Logistic Regression:
              precision    recall  f1-score   support

           0       0.83      0.71      0.77       129
           1       0.78      0.87      0.82       147

    accuracy                           0.80       276
   macro avg       0.80      0.79      0.79       276
weighted avg       0.80      0.80      0.80       276
```

4 YANG TERBAIK ADALAH RANDOM FOREST DENGAN PERSENTASI 82%

[35]:
```
import pickle
with open('modelLogisticRegression.pkl', 'wb') as file:
    pickle.dump(pipe, file)
```

Notebook_KLASIFIKASI_A_Heart_GradientBoosting_VS_SVM

October 23, 2025

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

[2]:
```
df = pd.read_csv('heart.csv')

df.head()
```

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

[3]:
```
df["HeartDisease"].unique()
```

[3]:
```
array([0, 1])
```

[4]:
```
df.describe()
```

[4]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

[5]:
```
df.isnull().sum()
```

[5]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

[6]:
```
numerical_column = df.select_dtypes(include=np.number).columns
categorical_column = df.select_dtypes(exclude=np.number).columns

numerical_column
df["HeartDisease"].unique()
```

[6]:
```
array([0, 1])
```

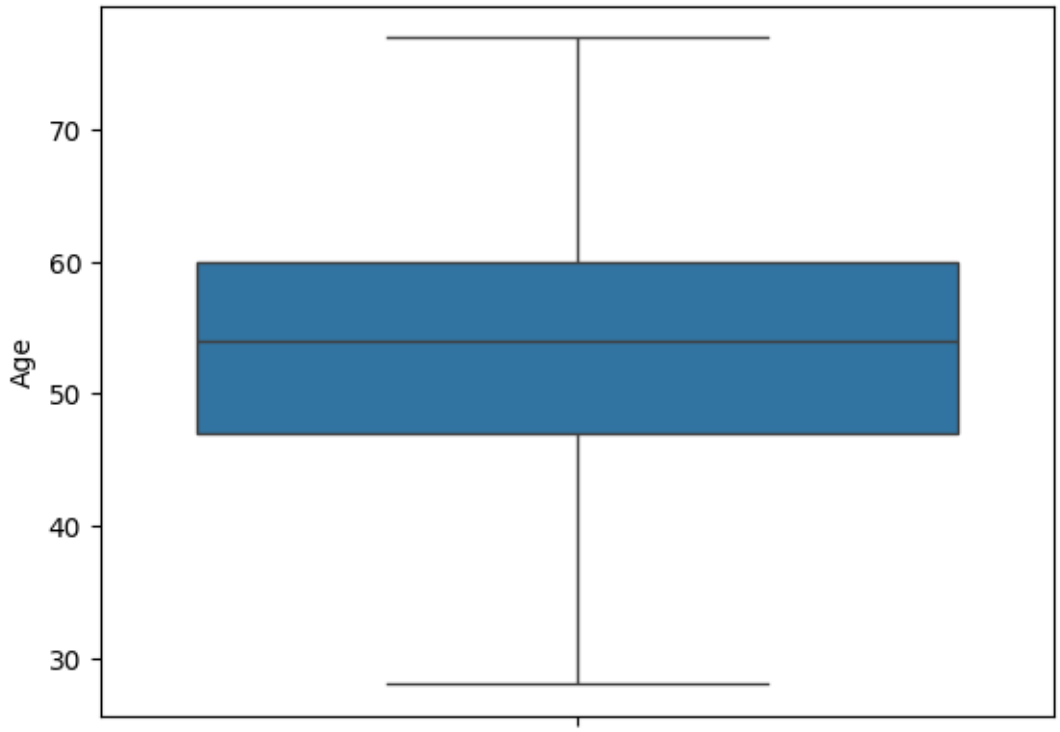
[7]:
```
categorical_column
```

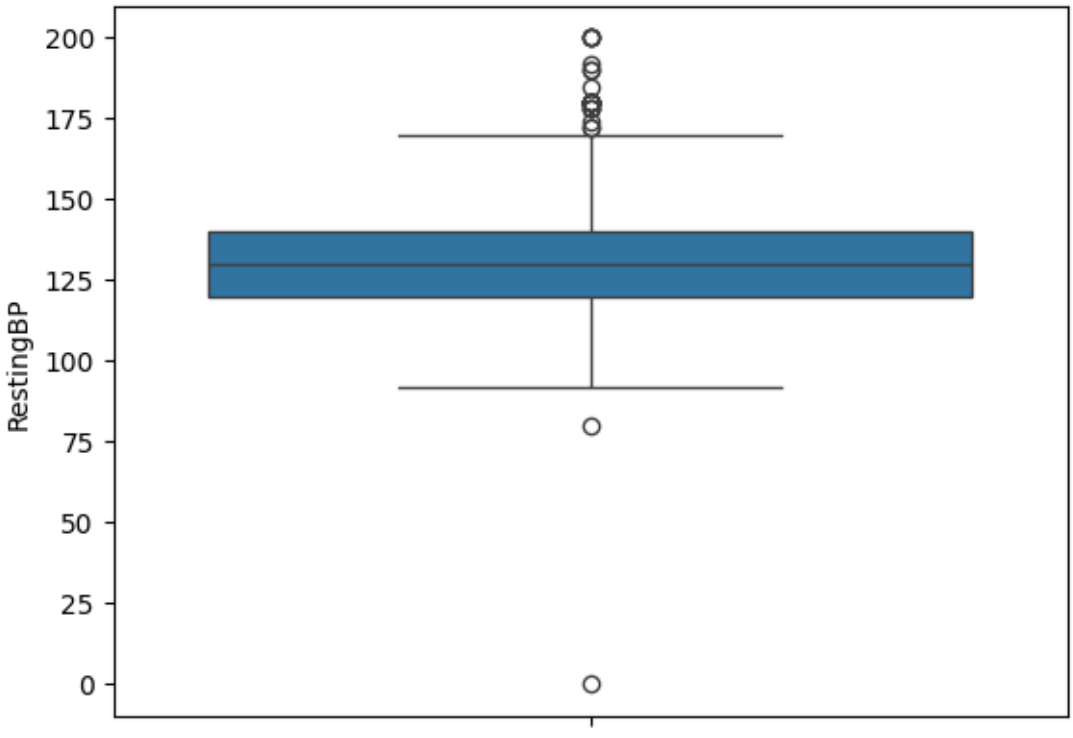
[7]:
```
Index(['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope'],
dtype='object')
```

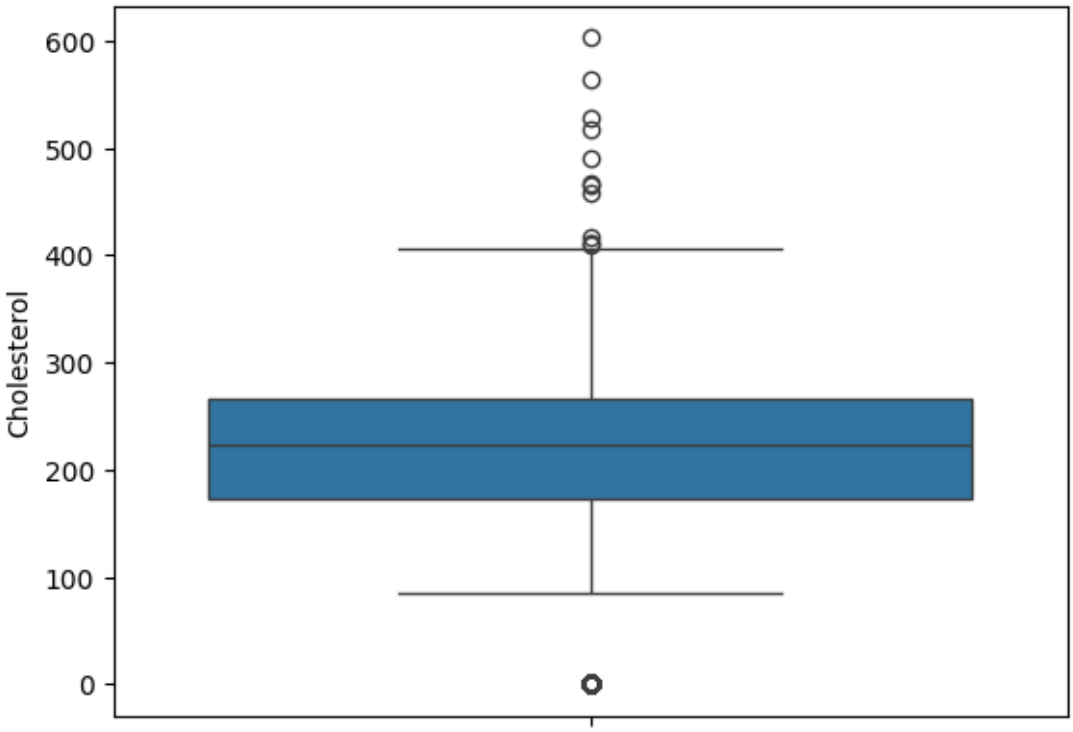
[8]:
```
numerical_column = ['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR',␣
 ↪'Oldpeak']
df["HeartDisease"].unique()
```

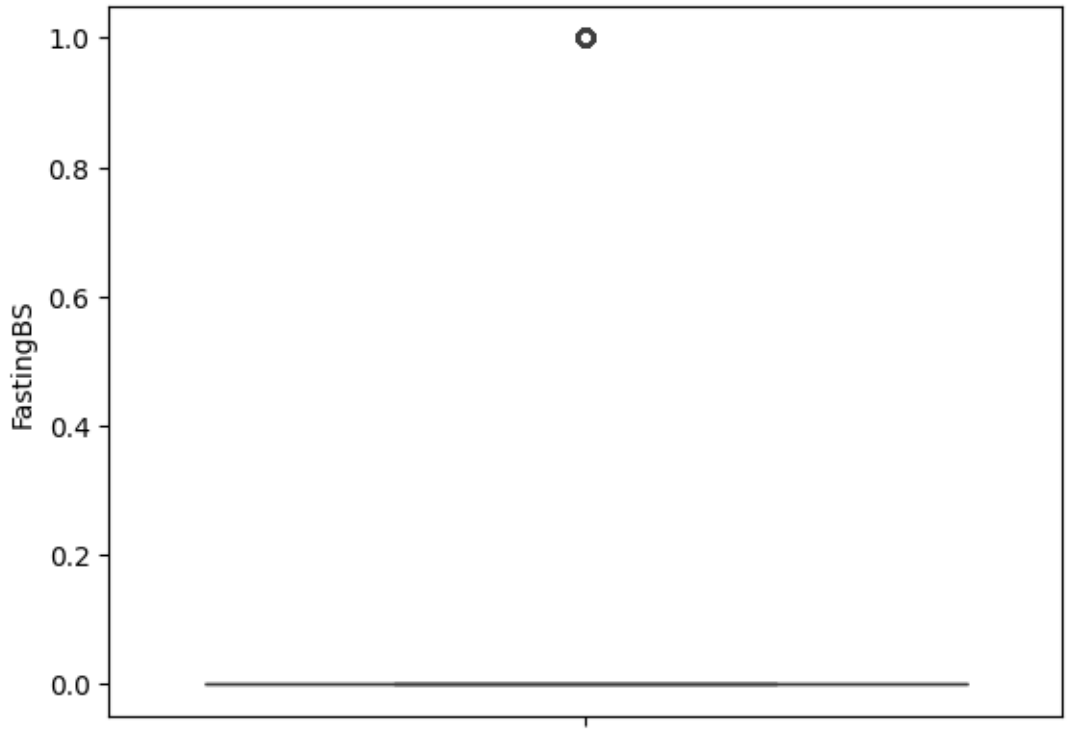
[8]:
```
array([0, 1])
```

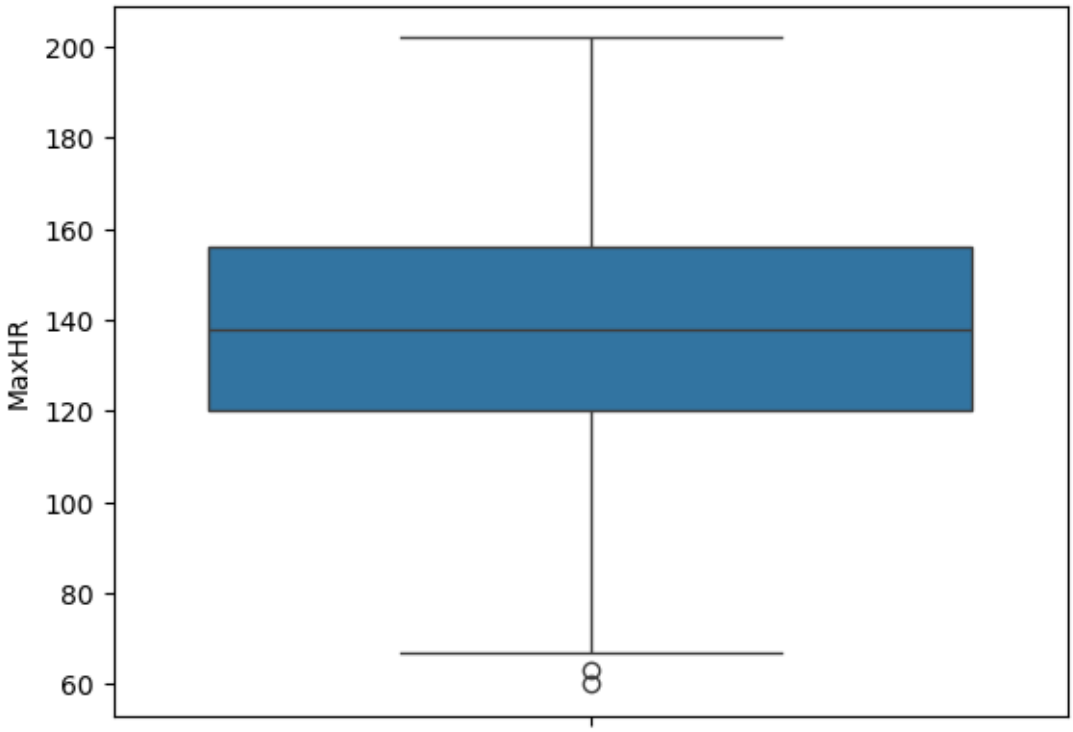
[9]:
```
for column in numerical_column:
  sns.boxplot(df[column])
  plt.show()
```

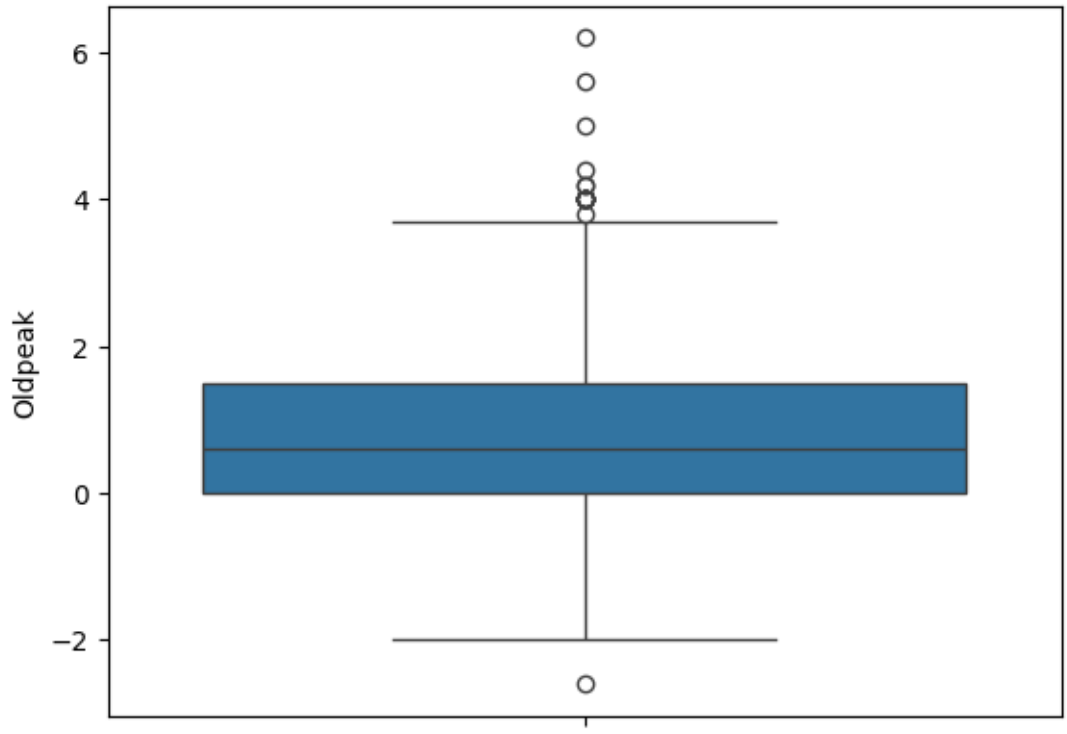

Age
70
60
50
40
30


RestingBP
200
175
150
125
100
75
50
25
0


Cholesterol
600
500
400
300
200
100
0


FastingBS
1.0
0.8
0.6
0.4
0.2
0.0


MaxHR
200
180
160
140
120
100
80
60


Oldpeak
6
4
2
0
−2

[10]:
```
df.dropna(inplace=True)

df.isnull().sum()
df["HeartDisease"].unique()
```

[10]:
```
array([0, 1])
```

[11]:
```
numerical_column
```

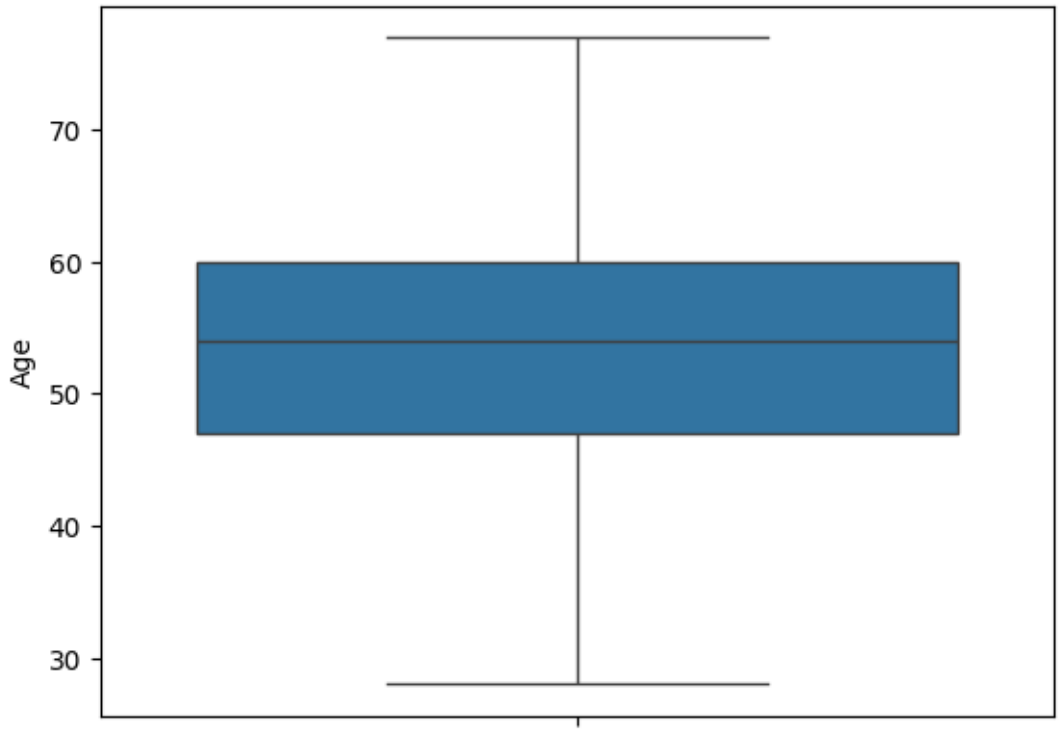
[11]:
```
['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak']
```

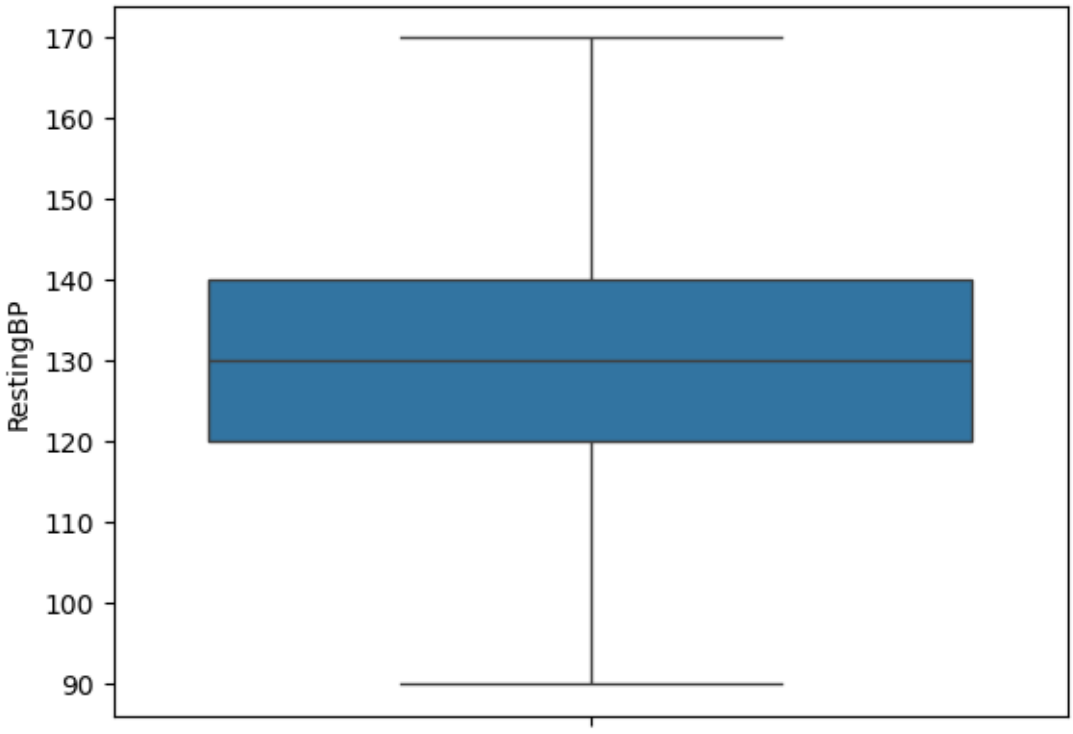
0.1 Penanganan Outliers dengan cara mengganti nilai outliers dengan nilai bawah atau atas terdekat dengan tujuan untuk m

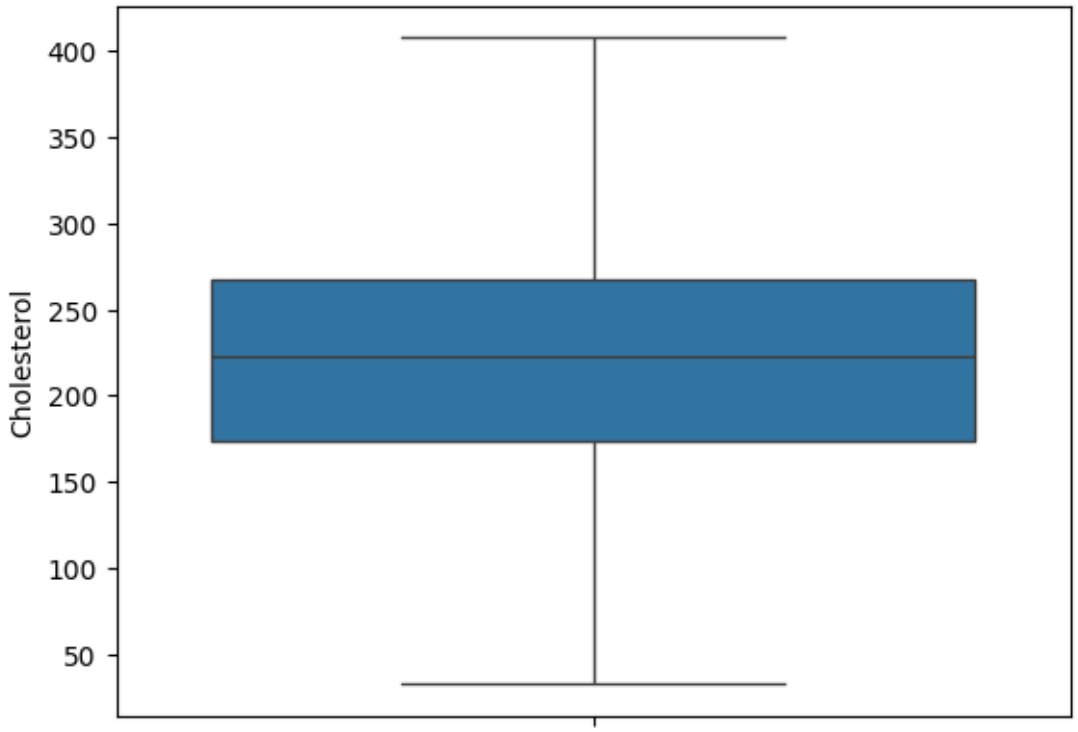
[12]:
```
for column in numerical_column:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5*IQR
    upper = Q3 + 1.5*IQR

    df[column] = df[column].clip(lower=lower, upper=upper)

df["HeartDisease"].unique()
```

[12]:
```
array([0, 1])
```

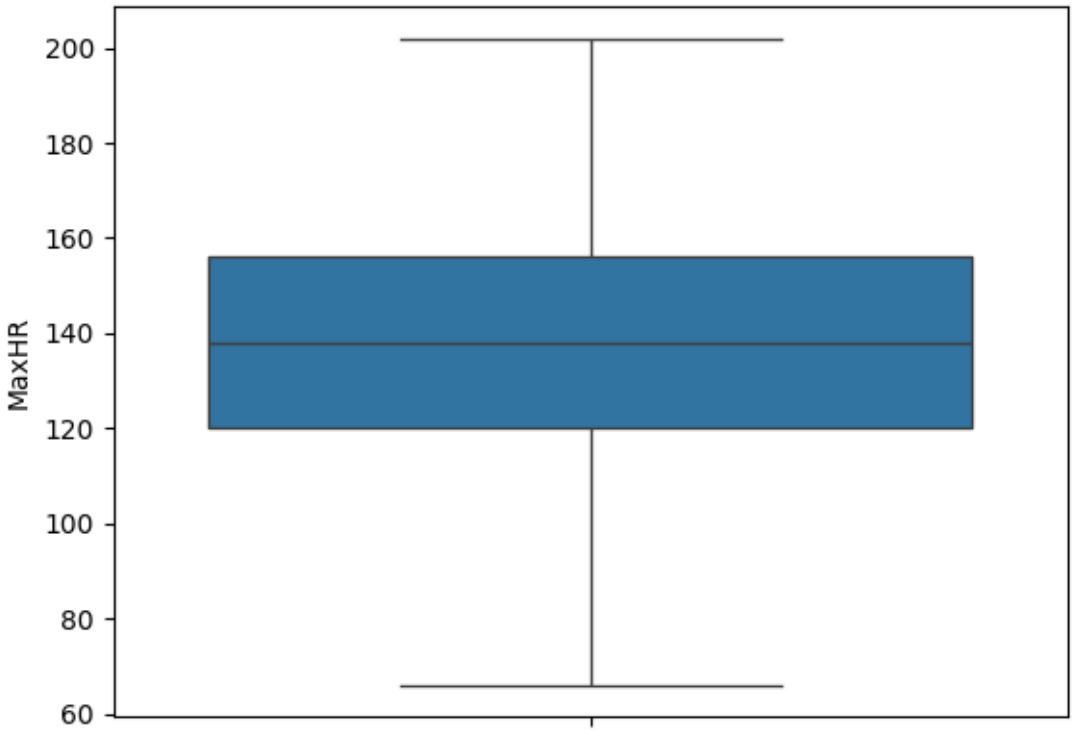
[13]:
```
for column in numerical_column:
  sns.boxplot(df[column])
  plt.show()
```

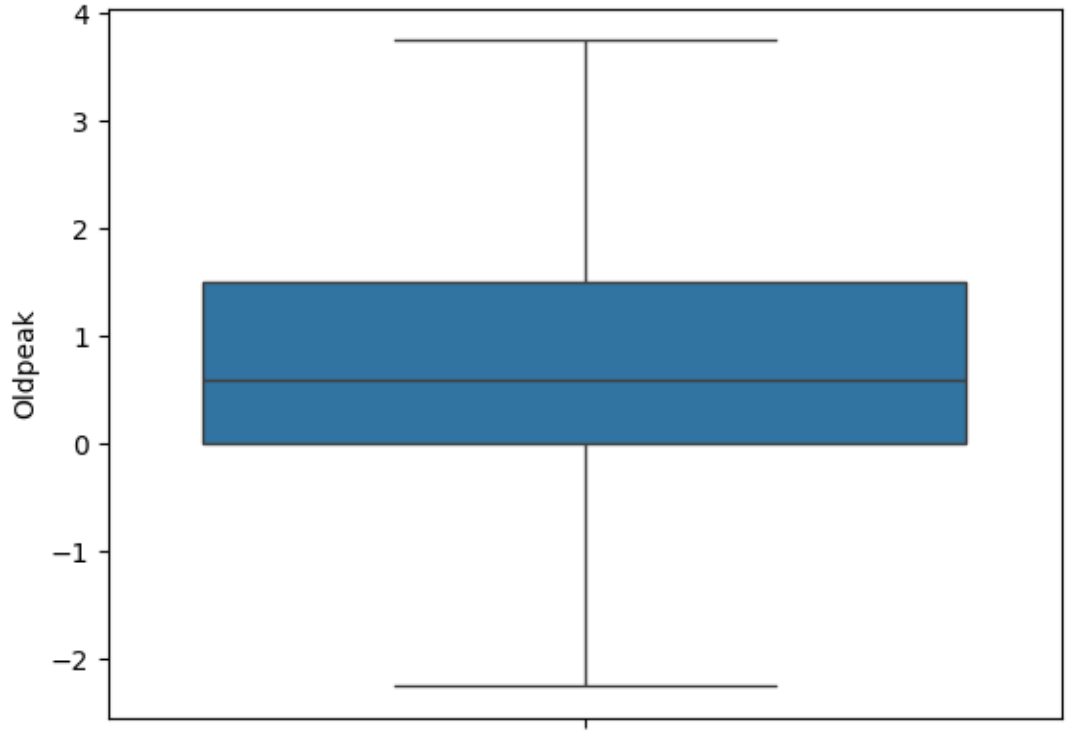

Age
70
60
50
40
30


RestingBP
170
160
150
140
130
120
110
100
90


Cholesterol
400
350
300
250
200
150
100
50


FastingBS
0.04
0.02
0.00
−0.02
−0.04


MaxHR
200
180
160
140
120
100
80
60


Oldpeak
4
3
2
1
0
−1
−2

[14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    float64
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 86.2+ KB
```

[15]:
```
categorical_column
```

[15]:
```
Index(['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope'],
dtype='object')
```

[16]:
```
df[categorical_column].head(10)
```

[16]:

	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope
0	M	ATA	Normal	N	Up
1	F	NAP	Normal	N	Flat
2	M	ATA	ST	N	Up
3	F	ASY	Normal	Y	Flat
4	M	NAP	Normal	N	Up
5	M	NAP	Normal	N	Up
6	F	ATA	Normal	N	Up
7	M	ATA	Normal	N	Up
8	M	ASY	Normal	Y	Flat
9	F	ATA	Normal	N	Up

[17]:
```
from sklearn.preprocessing import LabelEncoder

df[categorical_column]=df[categorical_column].astype(str)
le = LabelEncoder()
for column in categorical_column:
    df[column] = le.fit_transform(df[column])
```

```
df[categorical_column]
```

[17]:
```
     Sex  ChestPainType  RestingECG  ExerciseAngina  ST_Slope
0      1              1           1               0         2
1      0              2           1               0         1
2      1              1           2               0         2
3      0              0           1               1         1
4      1              2           1               0         2
..   …            …         …             …       …
913    1              3           1               0         1
914    1              0           1               0         1
915    1              0           1               1         1
916    0              1           0               0         1
917    1              2           1               0         2

[918 rows x 5 columns]
```

[18]:
```
df.head()
```

[18]:
```
   Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  RestingECG  \
0   40    1              1        140        289.0          0           1
1   49    0              2        160        180.0          0           1
2   37    1              1        130        283.0          0           2
3   48    0              0        138        214.0          0           1
4   54    1              2        150        195.0          0           1

   MaxHR  ExerciseAngina  Oldpeak  ST_Slope  HeartDisease
0    172               0      0.0         2             0
1    156               0      1.0         1             1
2     98               0      0.0         2             0
3    108               1      1.5         1             1
4    122               0      0.0         2             0
```

[19]:
```
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


X = df.drop(columns="HeartDisease")
y = df["HeartDisease"].astype(int)

y.unique()
```

[19]:
```
array([0, 1])
```

[20]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[21]:
```
X_scaled
```

[21]:
```
array([[-1.4331398 ,  0.51595242,  0.22903206, …, -0.8235563 ,
        -0.85127647,  1.05211381],
       [-0.47848359, -1.93816322,  1.27505906, …, -0.8235563 ,
         0.11853217, -0.59607813],
       [-1.75135854,  0.51595242,  0.22903206, …, -0.8235563 ,
        -0.85127647,  1.05211381],
       …,
       [ 0.37009972,  0.51595242, -0.81699495, …,  1.21424608,
         0.3124939 , -0.59607813],
       [ 0.37009972, -1.93816322,  0.22903206, …, -0.8235563 ,
        -0.85127647, -0.59607813],
       [-1.64528563,  0.51595242,  1.27505906, …, -0.8235563 ,
        -0.85127647,  1.05211381]])
```

[22]:
```
y.unique()
```

[22]:
```
array([0, 1])
```

1 CEK TIPE DARI NILAI DI KOLOM Y

[23]:
```
from sklearn.utils.multiclass import type_of_target
print(type_of_target(y))
```

```
binary
```

1.1 MEMILIH SELECTOR K-BEST

[24]:
```
selector = SelectKBest(score_func=mutual_info_classif)
X_new = selector.fit_transform(X_scaled, y)

X_new
```

[24]:
```
array([[-1.4331398 ,  0.51595242,  0.22903206, …, -0.8235563 ,
        -0.85127647,  1.05211381],
       [-0.47848359, -1.93816322,  1.27505906, …, -0.8235563 ,
         0.11853217, -0.59607813],
       [-1.75135854,  0.51595242,  0.22903206, …, -0.8235563 ,
        -0.85127647,  1.05211381],
       …,
```

```
       [ 0.37009972,  0.51595242, -0.81699495, …,  1.21424608,
         0.3124939 , -0.59607813],
       [ 0.37009972, -1.93816322,  0.22903206, …, -0.8235563 ,
        -0.85127647, -0.59607813],
       [-1.64528563,  0.51595242,  1.27505906, …, -0.8235563 ,
        -0.85127647,  1.05211381]])
```

2 MEMBAGI DATA SET TESTING DAN TRAINING

[25]:
```
X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2,␣
 ↪random_state=68)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(734, 10)
(184, 10)
(734,)
(184,)
```

3 SVM DENGAN PIPELINE

[26]:
```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV,␣
 ↪StratifiedKFold
from sklearn.feature_selection import SelectKBest, SelectPercentile, f_classif,␣
 ↪mutual_info_classif
from sklearn.preprocessing import StandardScaler, MinMaxScaler

pipe = Pipeline(steps=[
    ("scaler", StandardScaler()),
    ("feature_selection", SelectKBest(score_func=mutual_info_classif)),
    ("model", SVC())
])

max_depth = [5, 10, 15, 20]
random_state = [42, 52, 68]

param_grid = [
    {
        'scaler': [StandardScaler()],
        'feature_selection': [SelectKBest(score_func=mutual_info_classif)],
        'feature_selection__k': [5, 10, 20],
        'model__kernel': ['linear', 'rbf'],
```

```
        'model__C': [0.1, 1, 10],
        'model__gamma': ['scale', 'auto']
    },
    {
        'scaler': [MinMaxScaler()],
        'feature_selection': [SelectPercentile(score_func=mutual_info_classif)],
        'feature_selection__percentile': [10, 20, 50],
        'model__kernel': ['linear', 'rbf'],
        'model__C': [0.1, 1, 10],
        'model__gamma': ['scale', 'auto']
    }
]

# Grid Search dengan CV
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=68)

gs = GridSearchCV(
    estimator=pipe,
    param_grid=param_grid,
    cv=cv,
    n_jobs=-1,
    scoring="f1_macro"
)

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.3,␣
 ↪random_state=68)
gs.fit(X_train, y_train)
```

```
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
```

```
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
/opt/tljh/user/envs/ml/lib/python3.12/site-
packages/sklearn/feature_selection/_univariate_selection.py:783: UserWarning:
k=20 is greater than n_features=10. All the features will be returned.
  warnings.warn(
```

[26]:
```
GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=68, shuffle=True),
             estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                       ('feature_selection',
                                        SelectKBest(score_func=<function
mutual_info_classif at 0x7f34db9504a0>)),
                                       ('model', SVC())]),
             n_jobs=-1,
             param_grid=[{'feature_selection': [SelectKBest(score_func=<function
mutual_info_classif at 0x7f34db9504a0>)…
                          'model__gamma': ['scale', 'auto'],
                          'model__kernel': ['linear', 'rbf'],
                          'scaler': [StandardScaler()]},
                         {'feature_selection':
[SelectPercentile(score_func=<function mutual_info_classif at 0x7f34db9504a0>)],
                          'feature_selection__percentile': [10, 20, 50],
                          'model__C': [0.1, 1, 10],
                          'model__gamma': ['scale', 'auto'],
```

```
                        'model__kernel': ['linear', 'rbf'],
                        'scaler': [MinMaxScaler()]}],
              scoring='f1_macro')
```

[27]:
```
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report

print("Best parameters:", gs.best_params_)
print("Best CV accuracy:", gs.best_score_)

best_model = gs.best_estimator_
y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

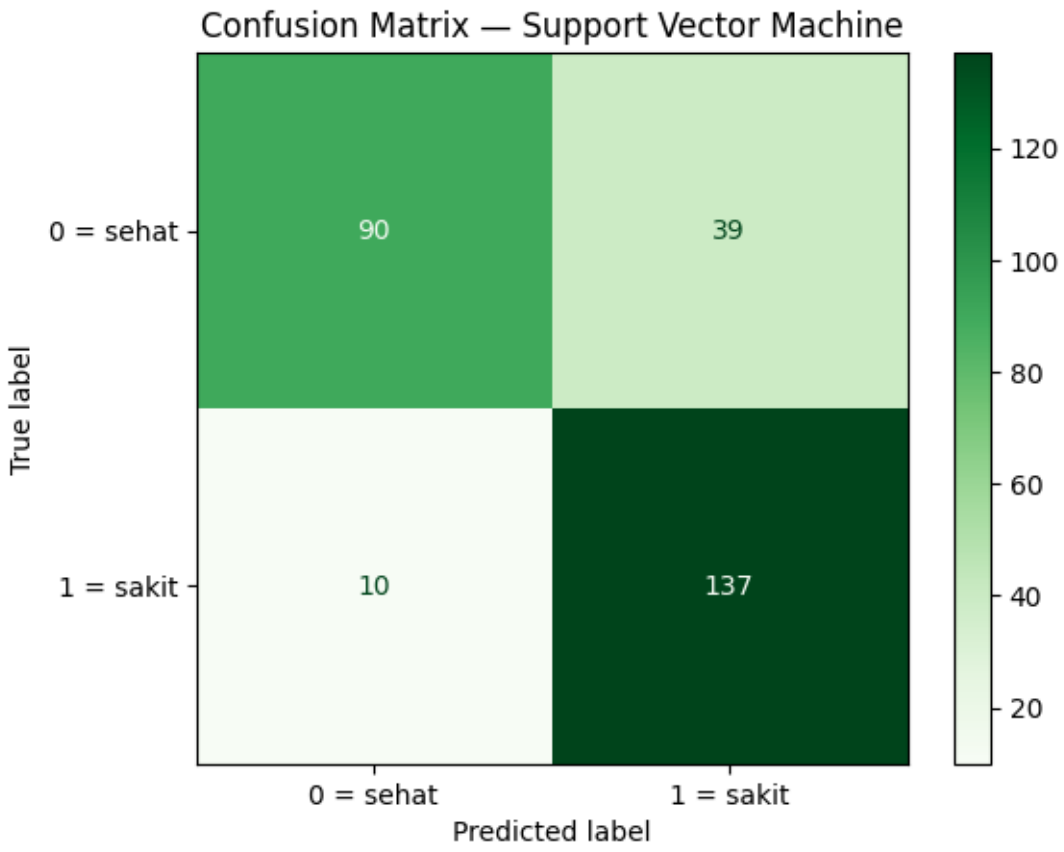
```
Best parameters: {'feature_selection': SelectKBest(score_func=<function
mutual_info_classif at 0x7f34db9504a0>), 'feature_selection__k': 10, 'model__C':
1, 'model__gamma': 'scale', 'model__kernel': 'rbf', 'scaler': StandardScaler()}
Best CV accuracy: 0.8736905958188735
              precision    recall  f1-score   support

           0       0.90      0.70      0.79       129
           1       0.78      0.93      0.85       147

    accuracy                           0.82       276
   macro avg       0.84      0.81      0.82       276
weighted avg       0.84      0.82      0.82       276
```

[28]:
```
import pickle
with open('modelSVM.pkl', 'wb') as file:
    pickle.dump(pipe, file)
```

[29]:
```
from sklearn.metrics import (
    confusion_matrix,
    ConfusionMatrixDisplay,
    classification_report
)

print("CV Score (F1) terbaik:", gs.best_score_)
print("Kombinasi model terbaik:", gs.best_estimator_)

lr_test_score = gs.best_estimator_.score(X_test,y_test)
print("\nSkor Test (akurasi) Support Vector Machine:", lr_test_score)

selector = gs.best_estimator_.named_steps['feature_selection']
if hasattr(selector, 'get_support'):
    mask = selector.get_support()
    selected = np.array(X.columns)
```

```
    print("\nFitur terbaik (terpilih):", selected)

#  Confusion Matrix & Classification Report
lr_pred = gs.predict(X_test)
cm_lr = confusion_matrix(y_test, lr_pred)
disp_lr = ConfusionMatrixDisplay(confusion_matrix=cm_lr, display_labels=['0 =␣
 ↪sehat','1 = sakit'])
disp_lr.plot(cmap=plt.cm.Greens)
plt.title("Confusion Matrix - Support Vector Machine")
plt.show()

best_model = gs.best_estimator_
y_pred = best_model.predict(X_test)
print("\nClassification Report - Support Vector Machine:\n",␣
 ↪classification_report(y_test, lr_pred))
```

```
CV Score (F1) terbaik: 0.8736905958188735
Kombinasi model terbaik: Pipeline(steps=[('scaler', StandardScaler()),
                ('feature_selection',
                 SelectKBest(score_func=<function mutual_info_classif at
0x7f34db9504a0>)),
                ('model', SVC(C=1))])

Skor Test (akurasi) Support Vector Machine: 0.822463768115942

Fitur terbaik (terpilih): ['Age' 'Sex' 'ChestPainType' 'RestingBP' 'Cholesterol'
'FastingBS'
 'RestingECG' 'MaxHR' 'ExerciseAngina' 'Oldpeak' 'ST_Slope']
```




```
Classification Report - Support Vector Machine:
              precision    recall  f1-score   support

           0       0.90      0.70      0.79       129
           1       0.78      0.93      0.85       147

    accuracy                           0.82       276
   macro avg       0.84      0.81      0.82       276
weighted avg       0.84      0.82      0.82       276
```

[ ]:
```python
# ==== Gradient Boosting  ====
import time
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, StratifiedKFold,␣
↪GridSearchCV
from sklearn.compose import ColumnTransformer
```

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler, MinMaxScaler
from sklearn.feature_selection import VarianceThreshold, SelectKBest,␣
 ↪SelectPercentile, f_classif, mutual_info_classif
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report

X_train, X_test, y_train, y_test = train_test_split(
    X, y.astype(int), test_size=0.3, random_state=68, stratify=y
)

num_cols = X_train.select_dtypes(include=[np.number]).columns.tolist()
cat_cols = [c for c in X_train.columns if c not in num_cols]

num_pipe = Pipeline(steps=[
    ("impute", SimpleImputer(strategy="median")),
    ("scale", "passthrough"),
])

cat_pipe = Pipeline(steps=[
    ("impute", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore", drop="if_binary",␣
 ↪min_frequency=0.02)),
])

preproc = ColumnTransformer(
    transformers=[
        ("num", num_pipe, num_cols),
        ("cat", cat_pipe, cat_cols),
    ],
    remainder="drop"
)

pipe = Pipeline(steps=[
    ("prep", preproc),
    ("vt", VarianceThreshold(threshold=0.0)),
    ("selector", SelectKBest(score_func=mutual_info_classif, k="all")),
    ("clf", GradientBoostingClassifier(random_state=42)),
])

is_binary = (np.unique(y_train).size == 2)
scoring = "roc_auc" if is_binary else "roc_auc_ovr"
n_splits = max(3, min(5, np.bincount(y_train).min()))
cv = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)

p = X_train.shape[1]
```

```
k_opts = ["all"] + [k for k in [5, 10, 15, 20, 30] if k <= p]
percentile_opts = [10, 20, 30, 50, 80, 100]

param_grid = [
    {
        "prep__num__scale": ["passthrough", StandardScaler(), MinMaxScaler()],
        "selector": [SelectKBest(score_func=f_classif),␣
 ↪SelectKBest(score_func=mutual_info_classif)],
        "selector__k": k_opts,
        "clf__n_estimators": [200, 400],
        "clf__learning_rate": [0.05, 0.1],
        "clf__max_depth": [2, 3],
        "clf__subsample": [0.8, 1.0],
        "clf__max_features": ["sqrt", None],
    },
    {
        "prep__num__scale": ["passthrough", StandardScaler(), MinMaxScaler()],
        "selector": [SelectPercentile(score_func=f_classif),␣
 ↪SelectPercentile(score_func=mutual_info_classif)],
        "selector__percentile": percentile_opts,
        "clf__n_estimators": [200, 400],
        "clf__learning_rate": [0.05, 0.1],
        "clf__max_depth": [2, 3],
        "clf__subsample": [0.8, 1.0],
        "clf__max_features": ["sqrt", None],
    },
]

gs = GridSearchCV(
    estimator=pipe,
    param_grid=param_grid,
    scoring=scoring,
    cv=cv,
    n_jobs=-1,
    verbose=1,
    error_score=np.nan
)

start = time.time()
gs.fit(X_train, y_train)
print("Best CV score :", gs.best_score_)
print("Best params   :", gs.best_params_)

best_pipe = gs.best_estimator_
y_pred  = best_pipe.predict(X_test)
proba   = best_pipe.predict_proba(X_test)
```

```
auc = roc_auc_score(y_test, proba[:, 1]) if is_binary else␣
 ↪roc_auc_score(y_test, proba, multi_class="ovr", average="macro")
acc = accuracy_score(y_test, y_pred)

print(f"\nTest Accuracy : {acc:.4f}")
print(f"Test ROC AUC  : {auc:.4f}\n")
print("Classification report:\n", classification_report(y_test, y_pred,␣
 ↪zero_division=0))
print(f"Training selesai dalam {time.time() - start:.2f} detik")
```

```
Fitting 5 folds for each of 1728 candidates, totalling 8640 fits
```

[31]:
```
import pickle

filename = "gradientBest.pkl"
with open(filename, "wb") as f:
    pickle.dump(best_pipe, f)

print(f"Model disimpan ke: {filename}")
```

```
Model disimpan ke: gradient_boosting_best_model.pkl
```

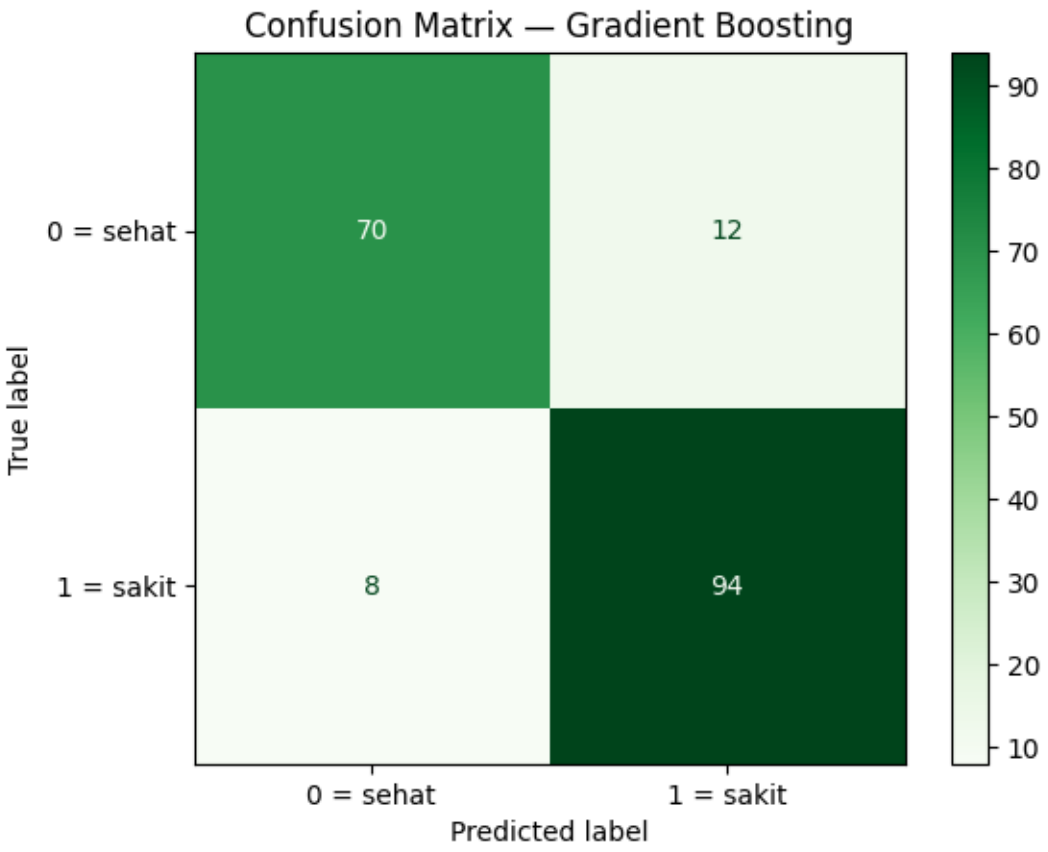
[32]:
```
y.unique()
```

[32]:
```
array([0, 1])
```

[33]:
```
print("CV Score (F1) terbaik:", gs.best_score_)
print("Kombinasi model terbaik:", gs.best_estimator_)

lr_test_score = gs.best_estimator_.score(X_test,y_test)
print("\nSkor Test (akurasi) Gradient Boosting:", lr_test_score)

selector = gs.best_estimator_.named_steps['selector']
if hasattr(selector, 'get_support'):
    mask = selector.get_support()
    selected = np.array(X.columns)
    print("\nFitur terbaik (terpilih):", selected)

lr_pred = gs.predict(X_test)
cm_lr = confusion_matrix(y_test, lr_pred)
disp_lr = ConfusionMatrixDisplay(confusion_matrix=cm_lr, display_labels=['0 =␣
 ↪sehat','1 = sakit'])
disp_lr.plot(cmap=plt.cm.Greens)
plt.title("Confusion Matrix - Gradient Boosting")
plt.show()

best_model = gs.best_estimator_
```

```
y_pred = best_model.predict(X_test)
print("\nClassification Report - Gradient Boosting:\n",␣
 ↪classification_report(y_test, lr_pred))
```

```
CV Score (F1) terbaik: 0.9202156229798506
Kombinasi model terbaik: Pipeline(steps=[('prep',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('impute',
SimpleImputer(strategy='median')),
                                                                  ('scale',
MinMaxScaler())]),
                                                  ['Age', 'Sex',
                                                   'ChestPainType', 'RestingBP',
                                                   'Cholesterol', 'FastingBS',
                                                   'RestingECG', 'MaxHR',
                                                   'ExerciseAngina', 'Oldpeak',
                                                   'ST_Slope']),
                                                 ('cat',
                                                  Pipeline(steps=[('impute',
SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
OneHotEncoder(drop='if_binary',
 handle_unknown='ignore',
 min_frequency=0.02))]),
                                                  [])])),
                ('vt', VarianceThreshold()),
                ('selector',
                 SelectPercentile(percentile=80,
                                  score_func=<function mutual_info_classif at
0x7f34db9504a0>)),
                ('clf',
                 GradientBoostingClassifier(learning_rate=0.05, max_depth=2,
                                            max_features='sqrt',
                                            n_estimators=200,
                                            random_state=42))])

Skor Test (akurasi) Gradient Boosting: 0.8913043478260869

Fitur terbaik (terpilih): ['Age' 'Sex' 'ChestPainType' 'RestingBP' 'Cholesterol'
'FastingBS'
 'RestingECG' 'MaxHR' 'ExerciseAngina' 'Oldpeak' 'ST_Slope']
```




```
Classification Report - Gradient Boosting:
              precision    recall  f1-score   support

           0       0.90      0.85      0.88        82
           1       0.89      0.92      0.90       102

    accuracy                           0.89       184
   macro avg       0.89      0.89      0.89       184
weighted avg       0.89      0.89      0.89       184
```

4 MODEL TERBAIK ADALAH GRADIENT BOOSTING DENGAN AKURASI 89%

[ ]:

Python Code

```python
import streamlit as st
import pandas as pd
import pickle

st.set_page_config(page_title="Heart Disease Predictor", page_icon="❤️", layout="centered")

@st.cache_resource
def load_model(path: str):
    with open(path, "rb") as f:
        return pickle.load(f)
model = load_model("BestModel_CLF_GradientBoosting_Heart.pkl")

st.title("🫀 Heart Disease Prediction🫀 ")

col1, col2 = st.columns(2)
with col1:
    Age = st.number_input("Age (tahun)", min_value=1, max_value=120, value=50)
    Sex = st.selectbox("Sex", ["Male", "Female"])
    ChestPainType = st.selectbox("Chest Pain Type", ["ATA", "NAP", "ASY", "TA"])
    RestingBP = st.number_input("Resting Blood Pressure (mmHg)", min_value=0, max_value=250, value=120)
    Cholesterol = st.number_input("Cholesterol (mg/dl)", min_value=0, max_value=600, value=200)
    FastingBS = st.selectbox("Fasting Blood Sugar > 120 mg/dl", ["No", "Yes"])
with col2:
    RestingECG = st.selectbox("Resting ECG Result", ["Normal", "ST", "LVH"])
    MaxHR = st.number_input("Maximum Heart Rate Achieved", min_value=50, max_value=250, value=150)
    ExerciseAngina = st.selectbox("Exercise Induced Angina", ["No", "Yes"])
    Oldpeak = st.number_input("Oldpeak (ST Depression)", min_value=0.0, max_value=10.0, value=1.0, step=0.1)
    ST_Slope = st.selectbox("ST Slope", ["Up", "Flat", "Down"])

submitted = st.button("Prediksi")

MAPS = {
    "Sex": {"Female": 0, "Male": 1},
    "ExerciseAngina": {"No": 0, "Yes": 1},
    "FastingBS": {"No": 0, "Yes": 1},
    "ChestPainType": {"ASY": 0, "ATA": 1, "NAP": 2, "TA": 3},
    "RestingECG": {"LVH": 0, "Normal": 1, "ST": 2},
    "ST_Slope": {"Down": 0, "Flat": 1, "Up": 2},
}

NUMERIC_COLS = [
```

```
    "Age","RestingBP","Cholesterol","MaxHR","Oldpeak",
    "Sex","ExerciseAngina","FastingBS","ChestPainType","RestingECG","ST_Slop
e"
]

def encode_input(df: pd.DataFrame) -> pd.DataFrame:
    df = df.copy()
    for col, m in MAPS.items():
        if col in df.columns and isinstance(df.at[0, col], str):
            if df.at[0, col] not in m:
                raise ValueError(f"Nilai '{df.at[0, col]}' pada kolom '{col}'
tidak dikenal. Pilihan: {list(m.keys())}")
            df.at[0, col] = m[df.at[0, col]]
    for col in NUMERIC_COLS:
        if col in df.columns:
            df[col] = pd.to_numeric(df[col], errors="raise")
    return df

if submitted:
    raw_input = pd.DataFrame({
        "Age": [Age],
        "Sex": [Sex],
        "ChestPainType": [ChestPainType],
        "RestingBP": [RestingBP],
        "Cholesterol": [Cholesterol],
        "FastingBS": [FastingBS],
        "RestingECG": [RestingECG],
        "MaxHR": [MaxHR],
        "ExerciseAngina": [ExerciseAngina],
        "Oldpeak": [Oldpeak],
        "ST_Slope": [ST_Slope],
    })

    used_encoded = False
    try:
        yhat = model.predict(raw_input)[0]
        proba = model.predict_proba(raw_input)[0][1] if hasattr(model, "predi
ct_proba") else None
    except Exception:
        input_num = encode_input(raw_input)
        yhat = model.predict(input_num)[0]
        proba = model.predict_proba(input_num)[0][1] if hasattr(model, "predi
ct_proba") else None
        used_encoded = True

    st.subheader(" Hasil Prediksi 📊 ")
    if yhat == 1:
        st.error("🚨 Pasien berisiko memiliki penyakit jantung 🚨")
    else:
        st.success("✅ Pasien kemungkinan tidak memiliki penyakit jantung
✅")

```

```
89      if proba is not None:
90          st.write(f"Probabilitas positif: **{proba:.2%}**")
91
92  st.markdown("------------")
93
```