ml-project-heart-disease-1

January 22, 2024

1 Data-Driven Heart Attack Risk Detection

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

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

[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
..	…	…	..	…	…	…	…	…	…	…	…	..	…	…
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

[303 rows x 14 columns]

[3]:
```
df.shape
```

[3]:
```
(303, 14)
```

[4]:
```
df.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   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

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

[5]:

	age	sex	cp	trestbps	chol	fbs	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	

	restecg	thalach	exang	oldpeak	slope	ca	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	
std	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	

```
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

             thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

[6]:
```
df.corr()
```

[6]:
```
                age       sex        cp  trestbps      chol       fbs  \
age        1.000000 -0.098447 -0.068653  0.279351  0.213678  0.121308
sex       -0.098447  1.000000 -0.049353 -0.056769 -0.197912  0.045032
cp        -0.068653 -0.049353  1.000000  0.047608 -0.076904  0.094444
trestbps   0.279351 -0.056769  0.047608  1.000000  0.123174  0.177531
chol       0.213678 -0.197912 -0.076904  0.123174  1.000000  0.013294
fbs        0.121308  0.045032  0.094444  0.177531  0.013294  1.000000
restecg   -0.116211 -0.058196  0.044421 -0.114103 -0.151040 -0.084189
thalach   -0.398522 -0.044020  0.295762 -0.046698 -0.009940 -0.008567
exang      0.096801  0.141664 -0.394280  0.067616  0.067023  0.025665
oldpeak    0.210013  0.096093 -0.149230  0.193216  0.053952  0.005747
slope     -0.168814 -0.030711  0.119717 -0.121475 -0.004038 -0.059894
ca         0.276326  0.118261 -0.181053  0.101389  0.070511  0.137979
thal       0.068001  0.210041 -0.161736  0.062210  0.098803 -0.032019
target    -0.225439 -0.280937  0.433798 -0.144931 -0.085239 -0.028046

            restecg   thalach     exang   oldpeak     slope        ca  \
age       -0.116211 -0.398522  0.096801  0.210013 -0.168814  0.276326
sex       -0.058196 -0.044020  0.141664  0.096093 -0.030711  0.118261
cp         0.044421  0.295762 -0.394280 -0.149230  0.119717 -0.181053
trestbps  -0.114103 -0.046698  0.067616  0.193216 -0.121475  0.101389
chol      -0.151040 -0.009940  0.067023  0.053952 -0.004038  0.070511
fbs       -0.084189 -0.008567  0.025665  0.005747 -0.059894  0.137979
restecg    1.000000  0.044123 -0.070733 -0.058770  0.093045 -0.072042
thalach    0.044123  1.000000 -0.378812 -0.344187  0.386784 -0.213177
exang     -0.070733 -0.378812  1.000000  0.288223 -0.257748  0.115739
oldpeak   -0.058770 -0.344187  0.288223  1.000000 -0.577537  0.222682
slope      0.093045  0.386784 -0.257748 -0.577537  1.000000 -0.080155
ca        -0.072042 -0.213177  0.115739  0.222682 -0.080155  1.000000
thal      -0.011981 -0.096439  0.206754  0.210244 -0.104764  0.151832
target     0.137230  0.421741 -0.436757 -0.430696  0.345877 -0.391724
```

```
              thal    target
age       0.068001 -0.225439
sex       0.210041 -0.280937
cp       -0.161736  0.433798
trestbps  0.062210 -0.144931
chol      0.098803 -0.085239
fbs      -0.032019 -0.028046
restecg  -0.011981  0.137230
thalach  -0.096439  0.421741
exang     0.206754 -0.436757
oldpeak   0.210244 -0.430696
slope    -0.104764  0.345877
ca        0.151832 -0.391724
thal      1.000000 -0.344029
target   -0.344029  1.000000
```

[7]:
```
df.isnull().sum()
```

[7]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

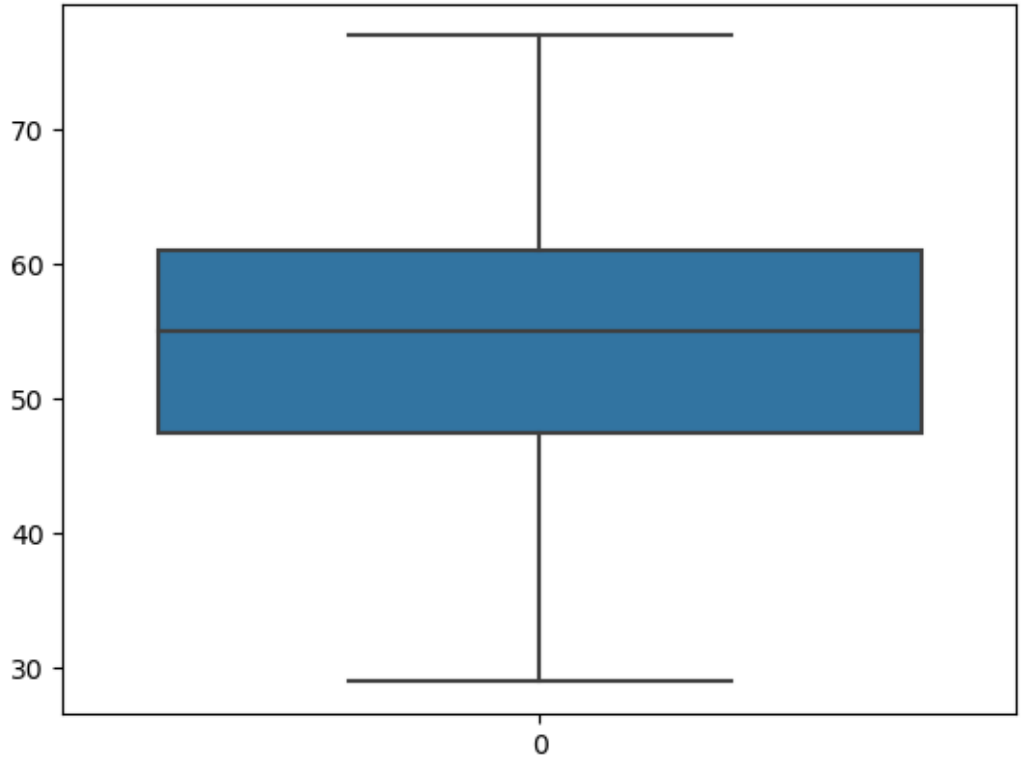
[8]:
```
df.apply(lambda x: x.unique())
```

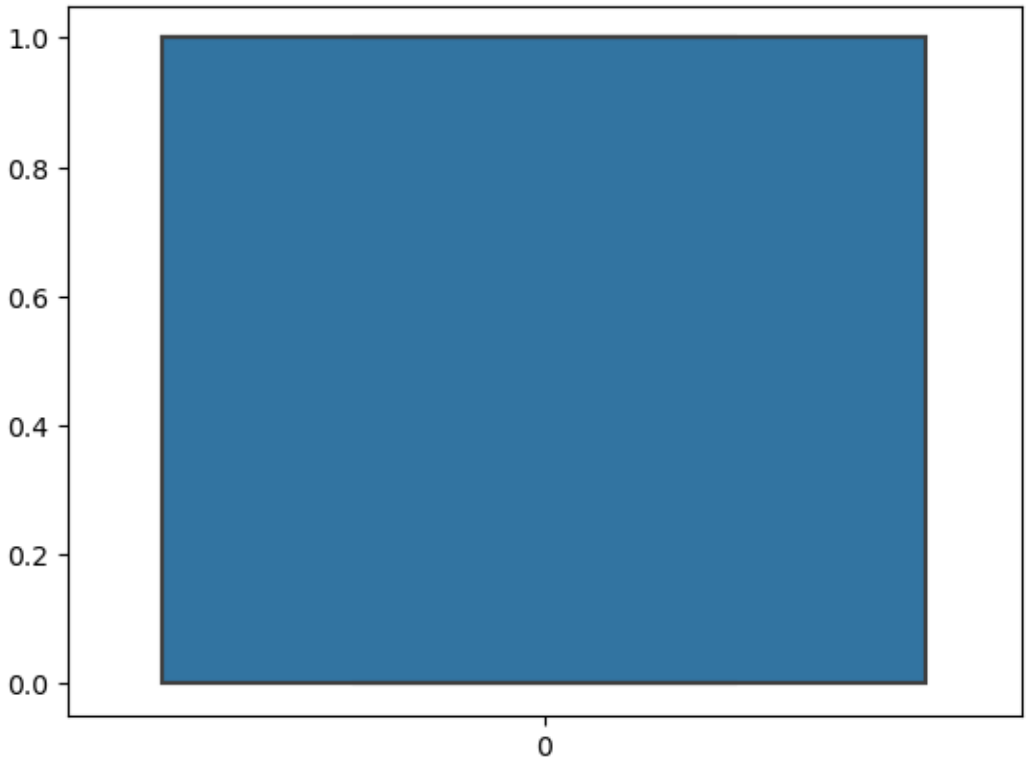
[8]:
```
age         [63, 37, 41, 56, 57, 44, 52, 54, 48, 49, 64, 5…
sex                                                    [1, 0]
cp                                                [3, 2, 1, 0]
trestbps    [145, 130, 120, 140, 172, 150, 110, 135, 160, …
chol        [233, 250, 204, 236, 354, 192, 294, 263, 199, …
fbs                                                    [1, 0]
restecg                                             [0, 1, 2]
thalach     [150, 187, 172, 178, 163, 148, 153, 173, 162, …
exang                                                  [0, 1]
oldpeak     [2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, …
slope                                               [0, 2, 1]
```

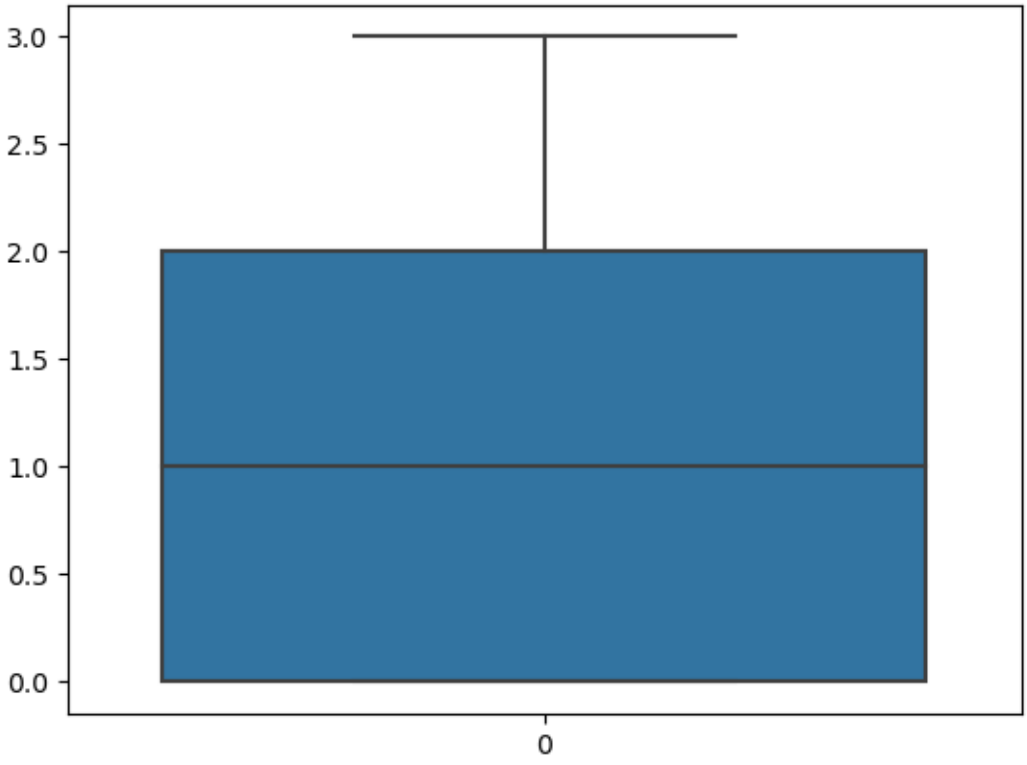
```
ca                                                  [0, 2, 1, 3, 4]
thal                                                   [1, 2, 3, 0]
target                                                       [1, 0]
dtype: object
```

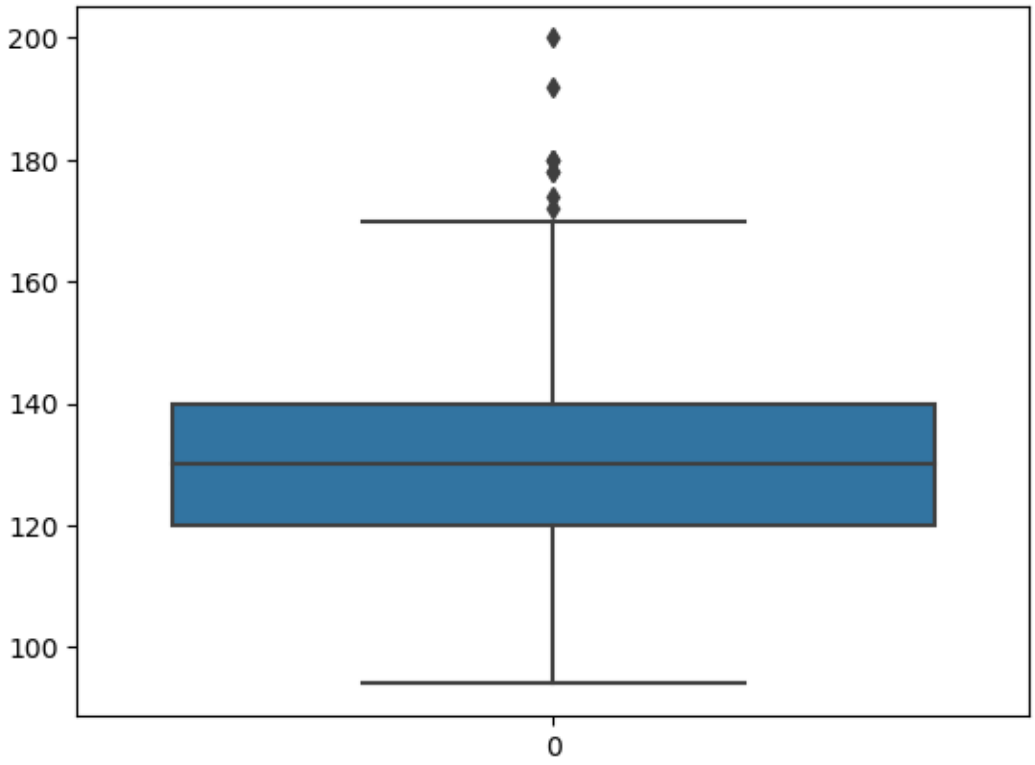
DATA VISUALIZATION

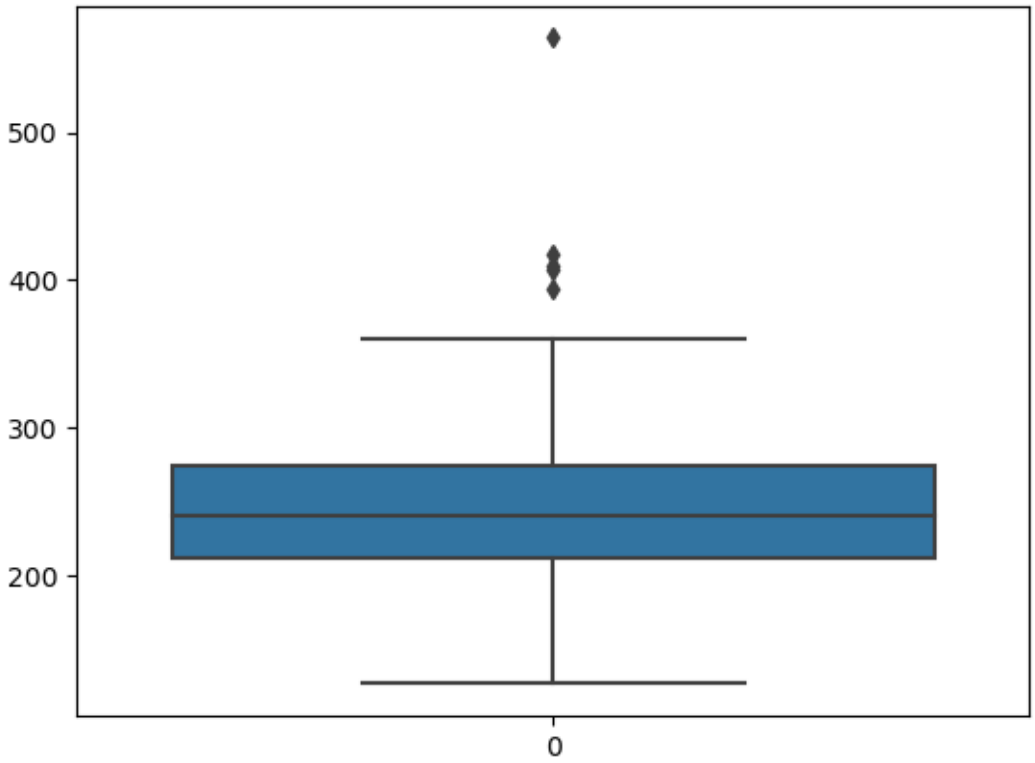
[10]:
```
for i in df.columns:
    if(df[i].dtypes=='int64' or df[i].dtypes=='float64'):
        sns.boxplot(df[i])
        plt.show()
```

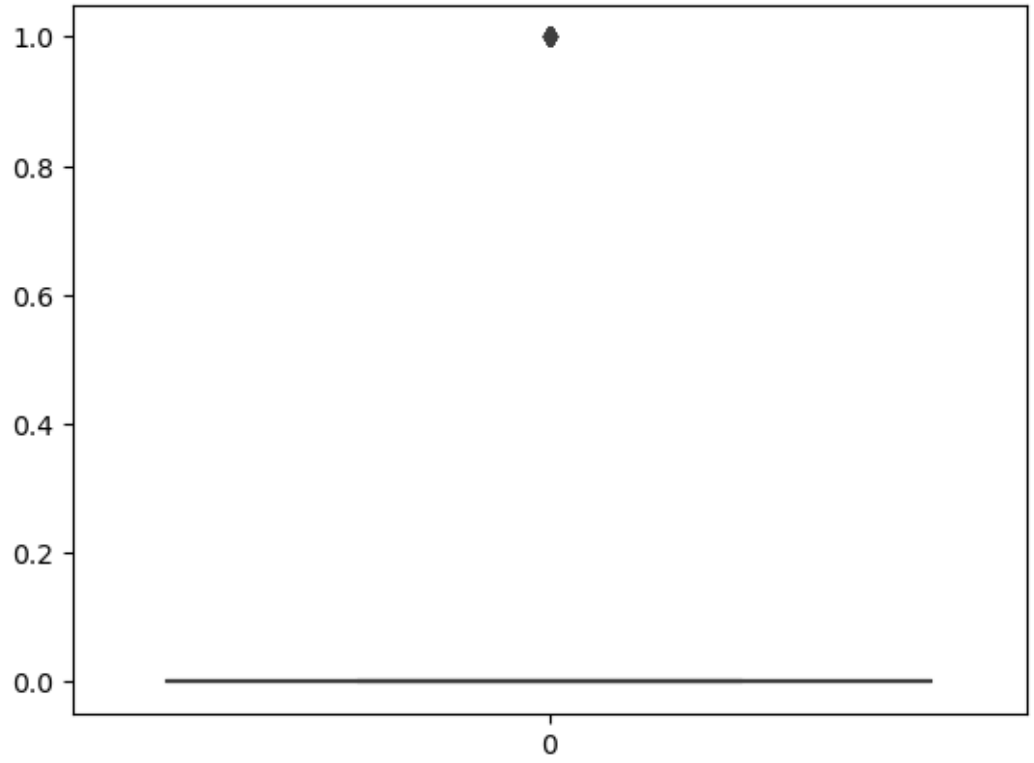


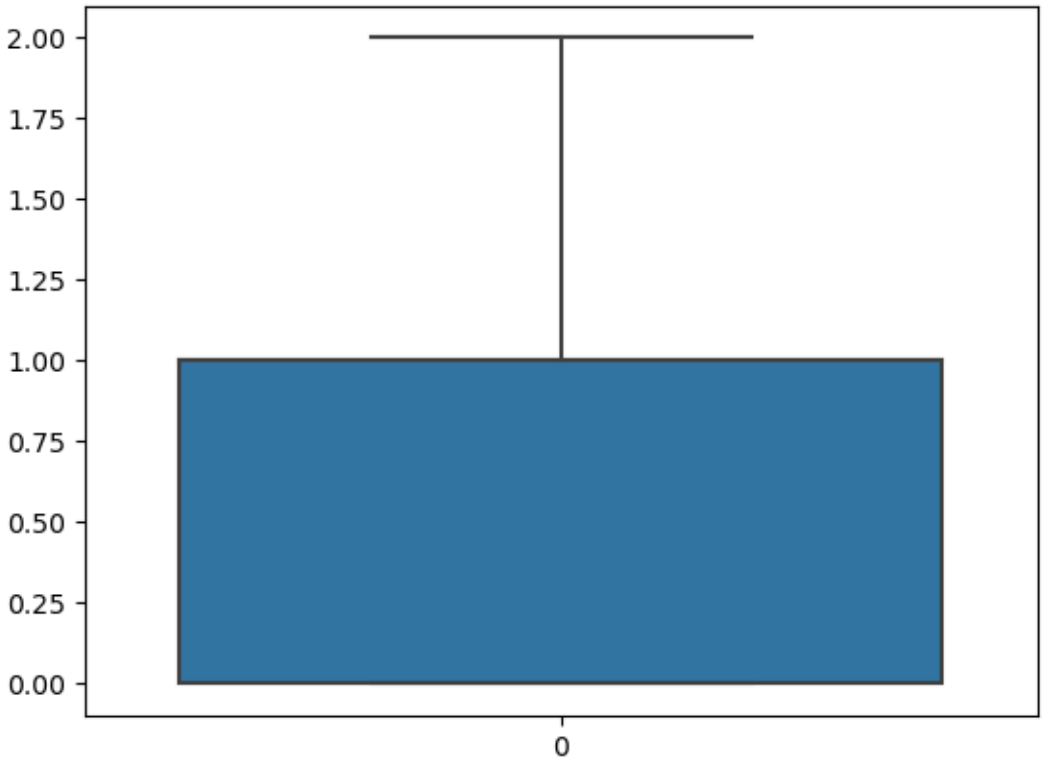

1.0
0.8
0.6
0.4
0.2
0.0
0

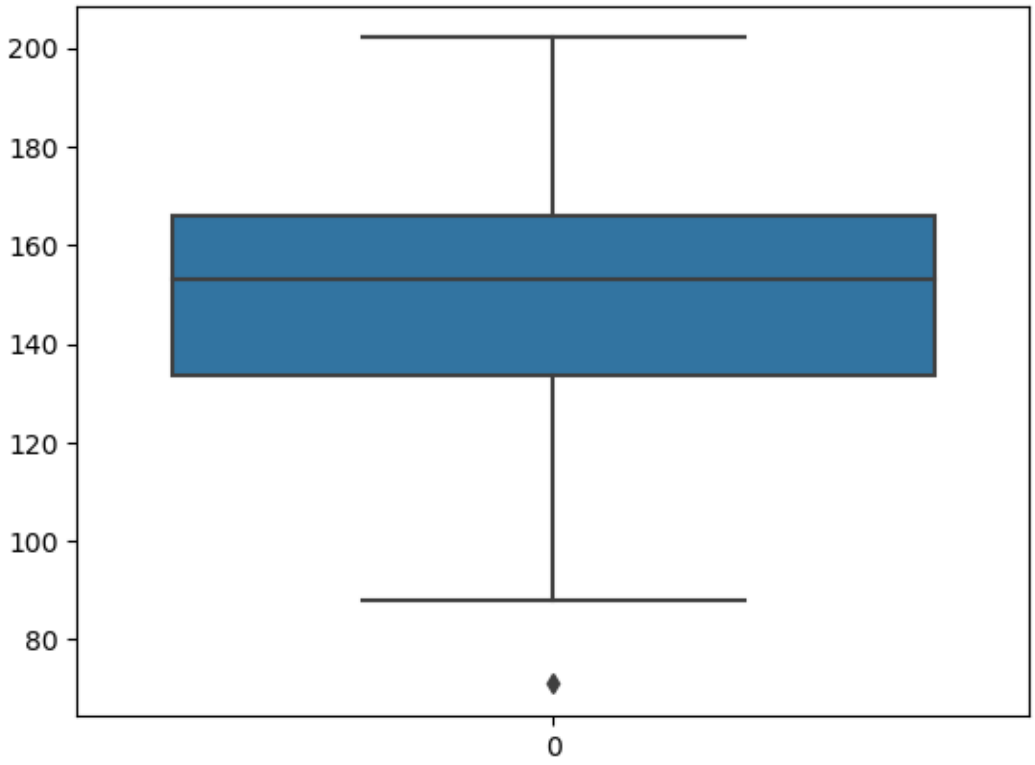

3.0
2.5
2.0
1.5
1.0
0.5
0.0
0

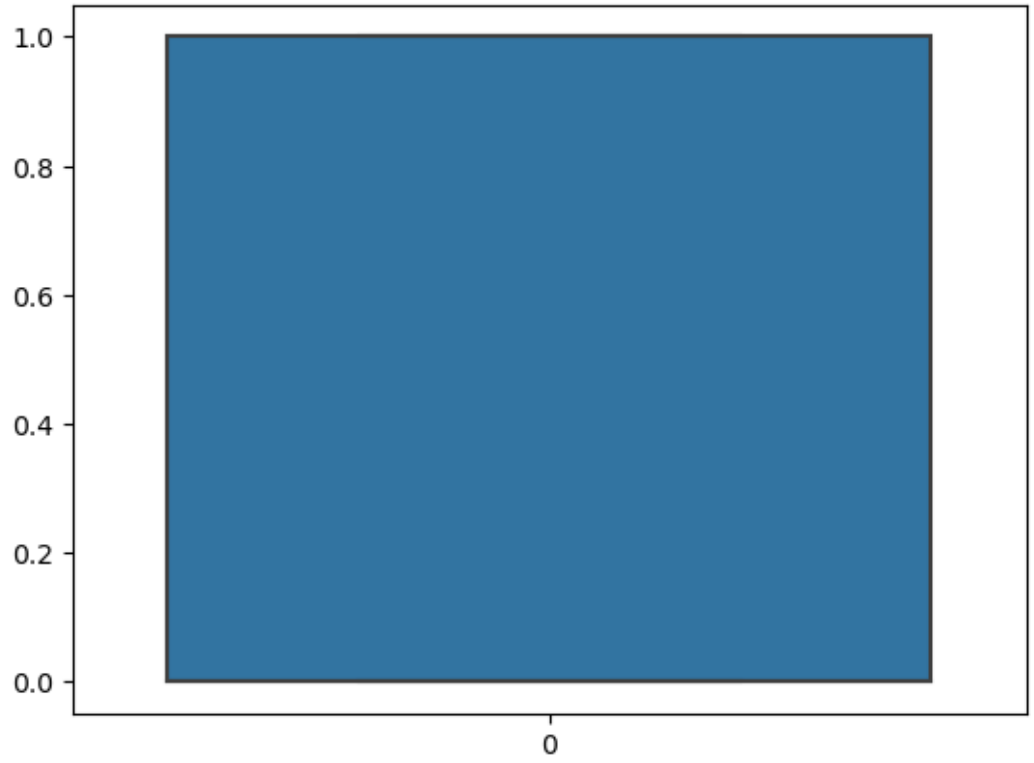

200
180
160
140
120
100
0

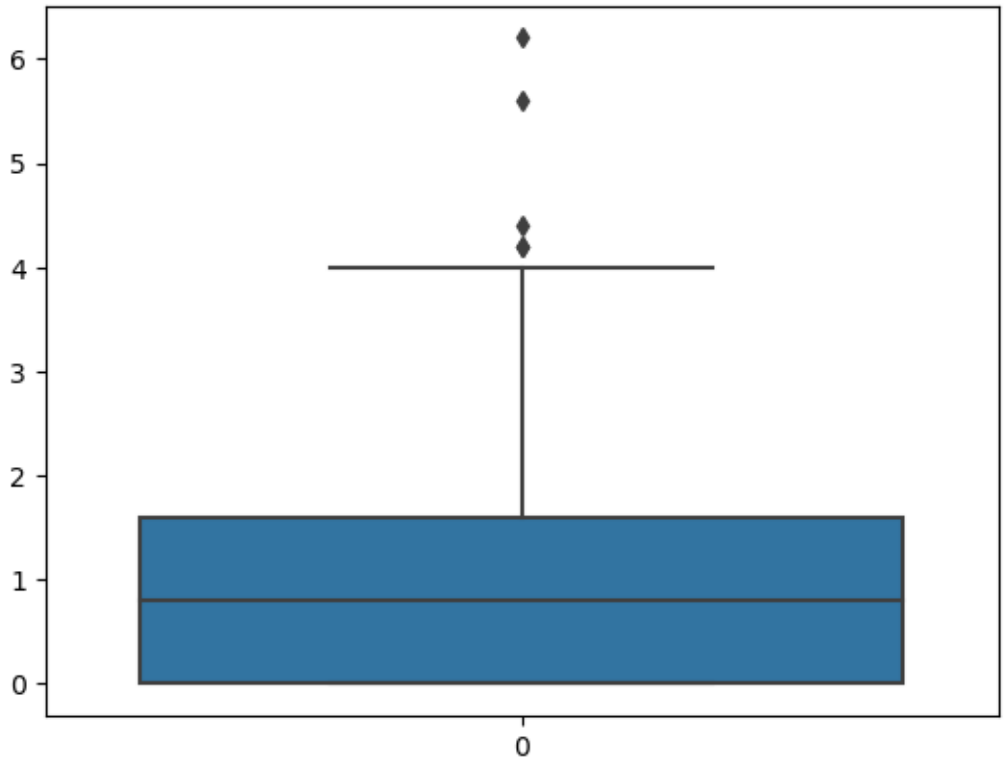

500
400
300
200
0

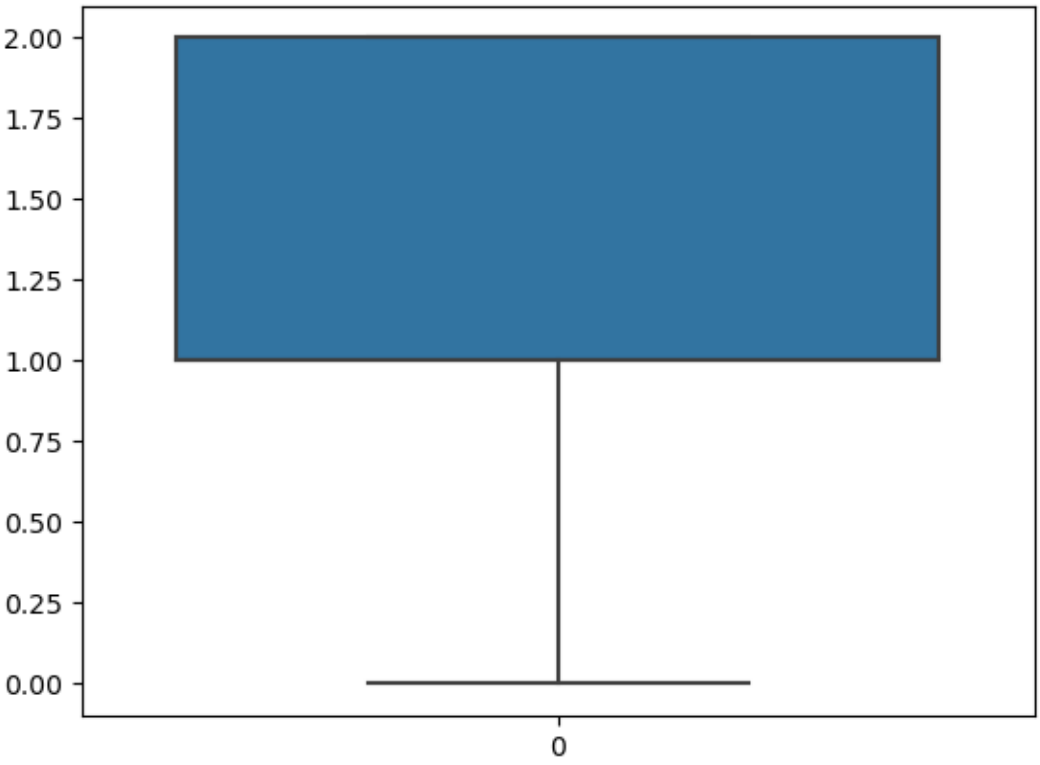

1.0
0.8
0.6
0.4
0.2
0.0
0

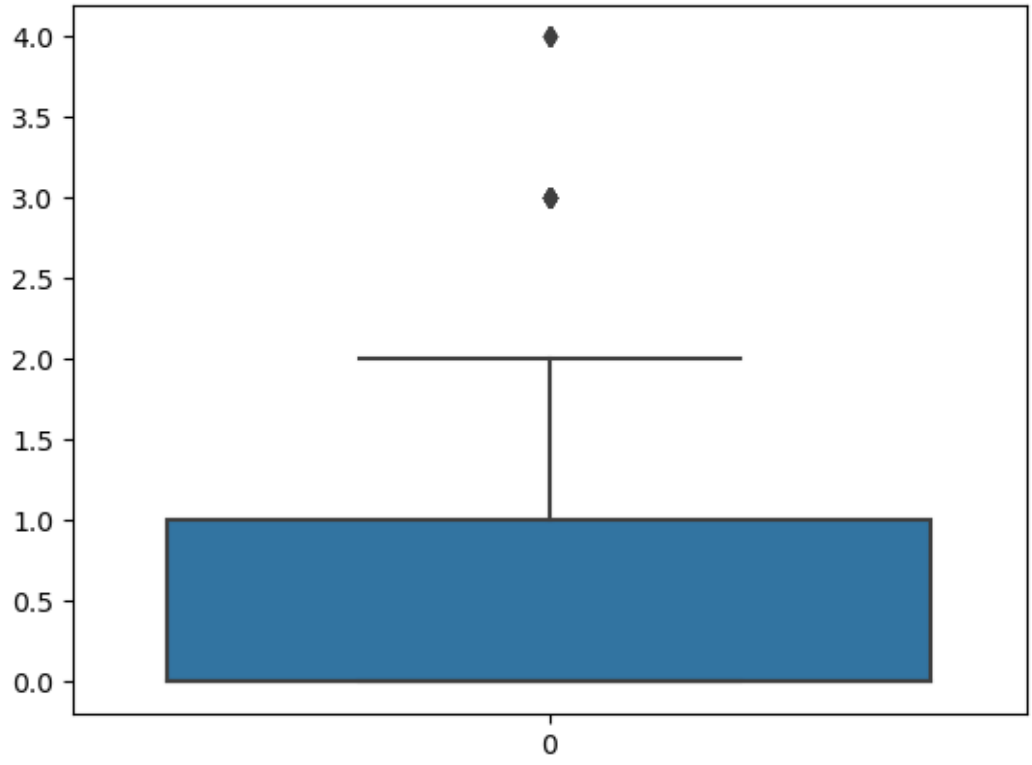

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0

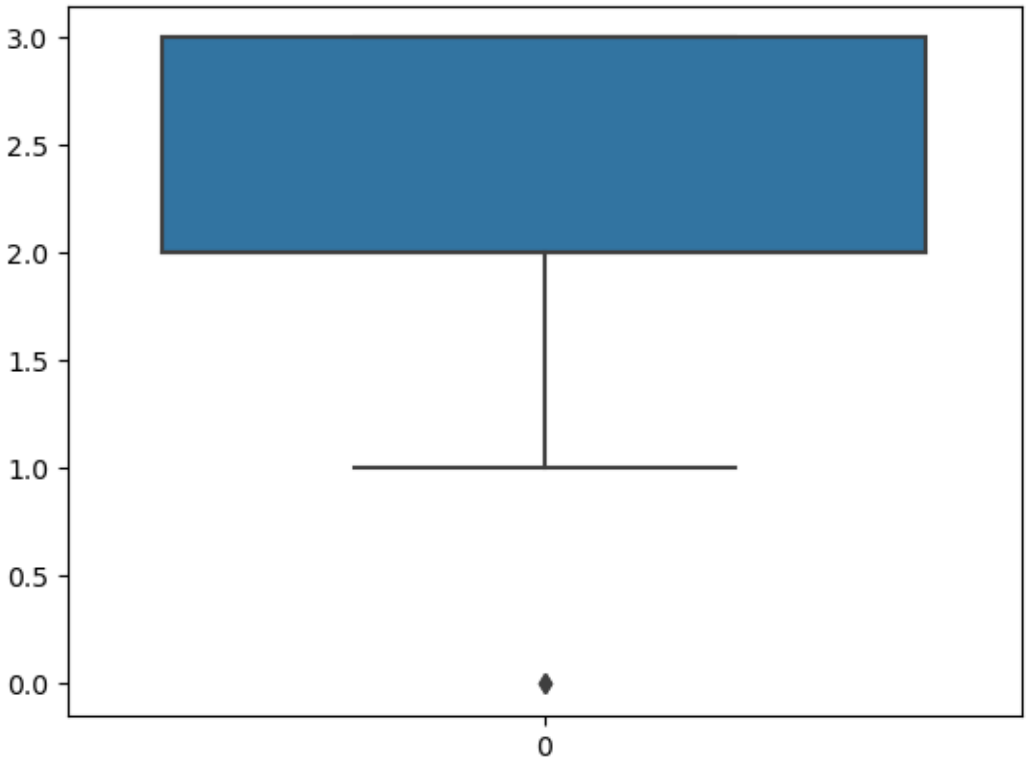

200
180
160
140
120
100
80
0

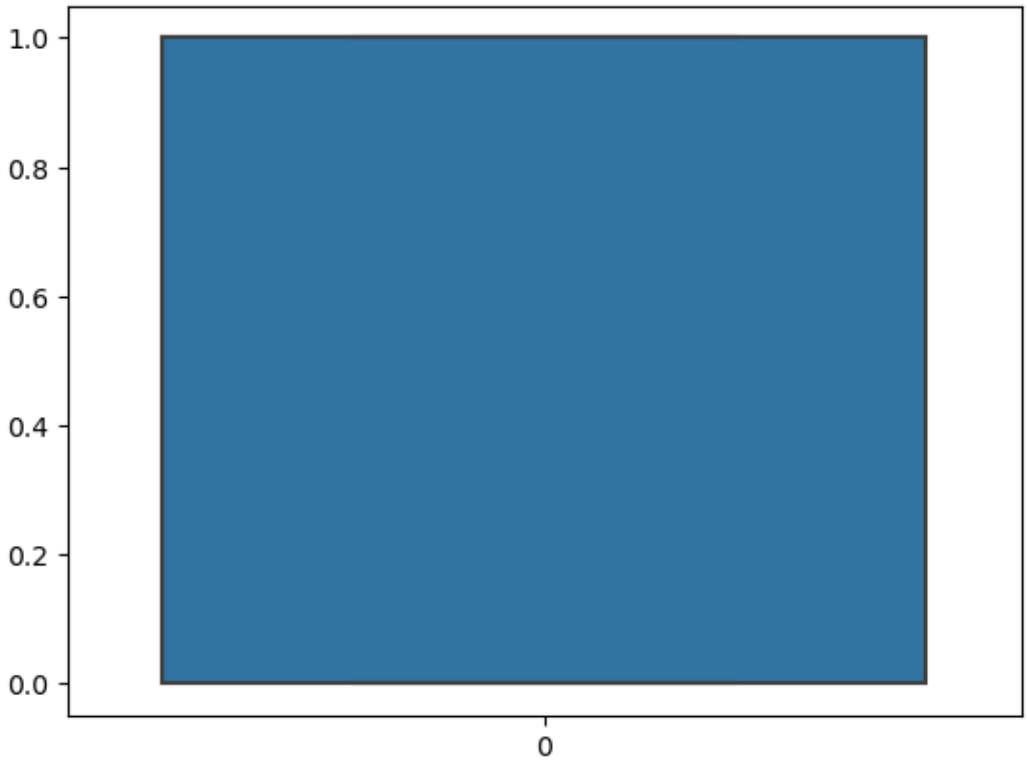

1.0
0.8
0.6
0.4
0.2
0.0
0


6
5
4
3
2
1
0
0

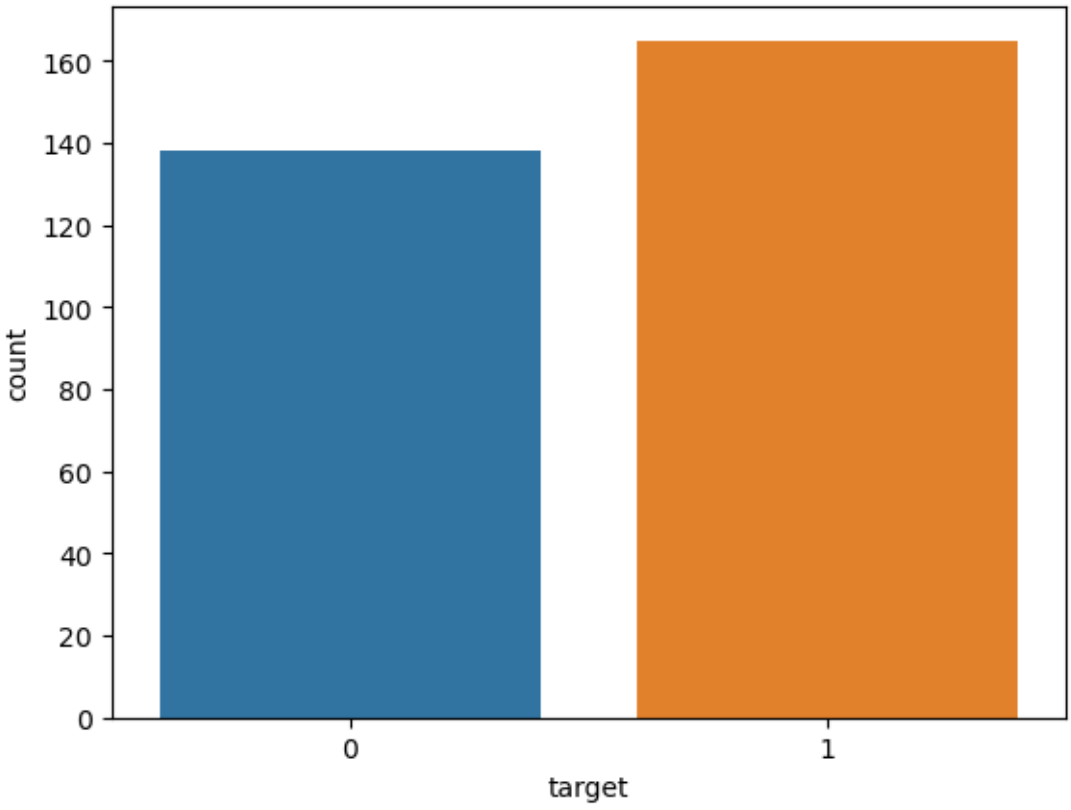

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0

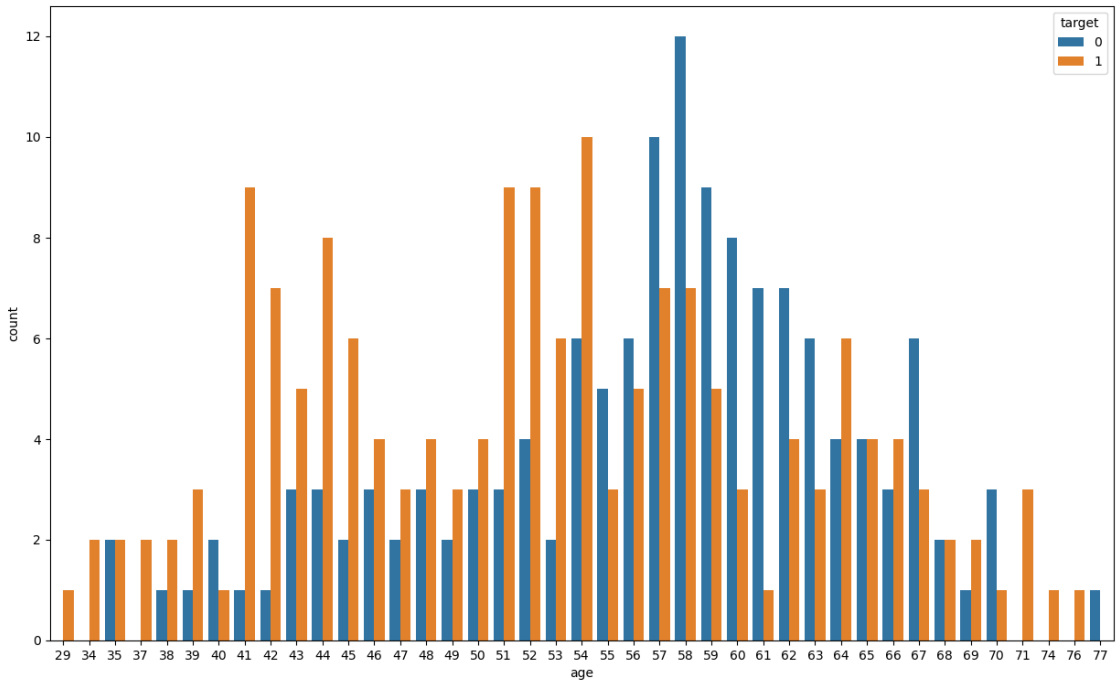

3.0
2.5
2.0
1.5
1.0
0.5
0.0
0



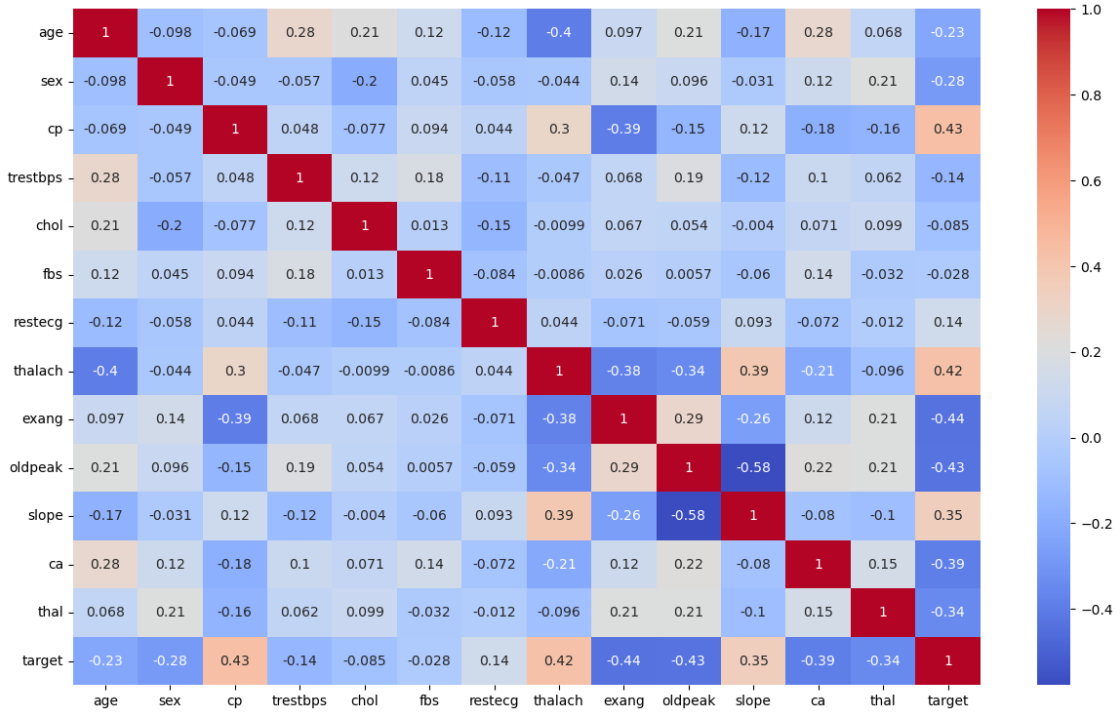
[11]:
```
sns.countplot(x=df['target'])
plt.show()
```



[12]:
```
plt.figure(figsize=(15,9))
sns.countplot(x=df['age'],hue=df['target'])
plt.show()
```



[13]:
```
plt.figure(figsize=(15,9))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
plt.show()
```



Logistic Regression

[14]:
```
X = df.iloc[:, :-1]
y = df.iloc[:, -1].values
X
```

[14]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      63    1   3       145   233    1        0      150      0      2.3
1      37    1   2       130   250    0        1      187      0      3.5
2      41    0   1       130   204    0        0      172      0      1.4
3      56    1   1       120   236    0        1      178      0      0.8
4      57    0   0       120   354    0        1      163      1      0.6
..    ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298    57    0   0       140   241    0        1      123      1      0.2
299    45    1   3       110   264    0        1      132      0      1.2
300    68    1   0       144   193    1        1      141      0      3.4
301    57    1   0       130   131    0        1      115      1      1.2
302    57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

[15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

```
y_pred = lr.predict(X_test)

pd.DataFrame({"actaul_value":y_test,"predicted_value":y_pred})
```

[15]:

	actaul_value	predicted_value
0	0	0
1	0	1
2	1	1
3	0	0
4	1	1
..	…	…
86	0	0
87	1	1
88	1	1
89	1	0
90	1	1

[91 rows x 2 columns]

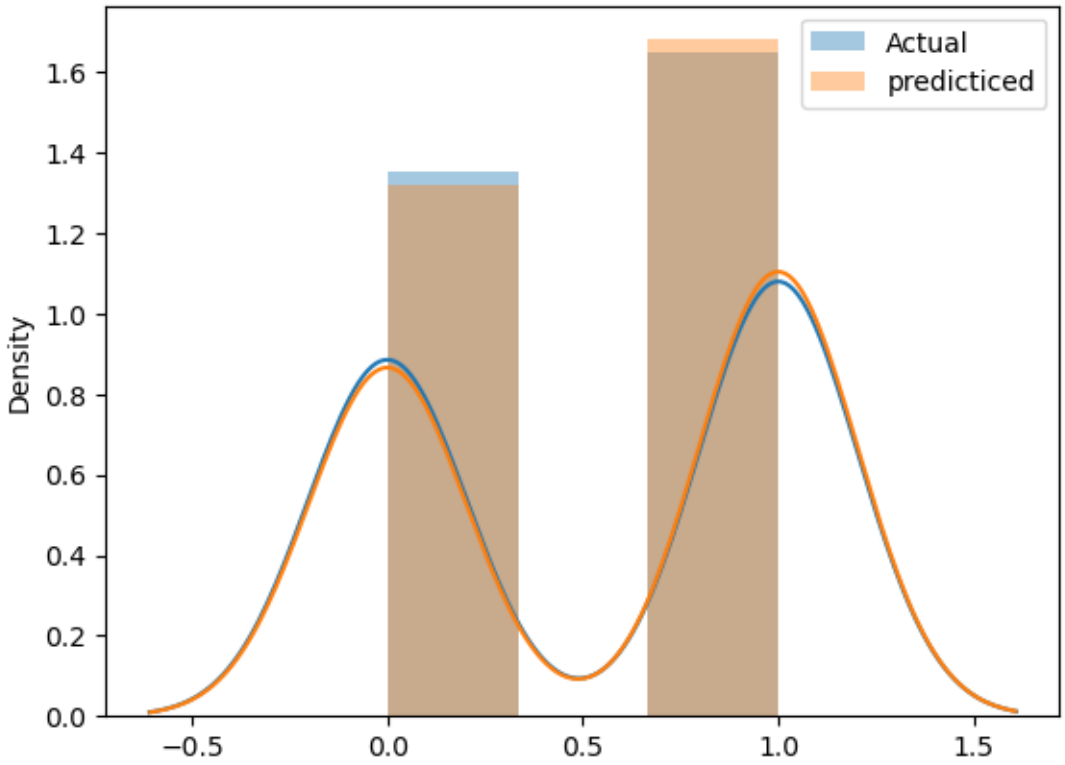
[16]:
```
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.81

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.78      0.79        41
           1       0.82      0.84      0.83        50

    accuracy                           0.81        91
   macro avg       0.81      0.81      0.81        91
weighted avg       0.81      0.81      0.81        91


Confusion Matrix:
[[32  9]
```

```
 [ 8 42]]
```

[17]:
```
sns.distplot(y_test,label='Actual')
sns.distplot(y_pred,label='predicticed')
plt.legend()
plt.show()
```

```
C:\Users\SIMRAN BANSAL\AppData\Local\Temp\ipykernel_5588\3633856401.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_test,label='Actual')
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\SIMRAN BANSAL\AppData\Local\Temp\ipykernel_5588\3633856401.py:2:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,label='predicticed')
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Decision Tree Classifier

[18]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

y_pred = dt.predict(X_test)
```

[19]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

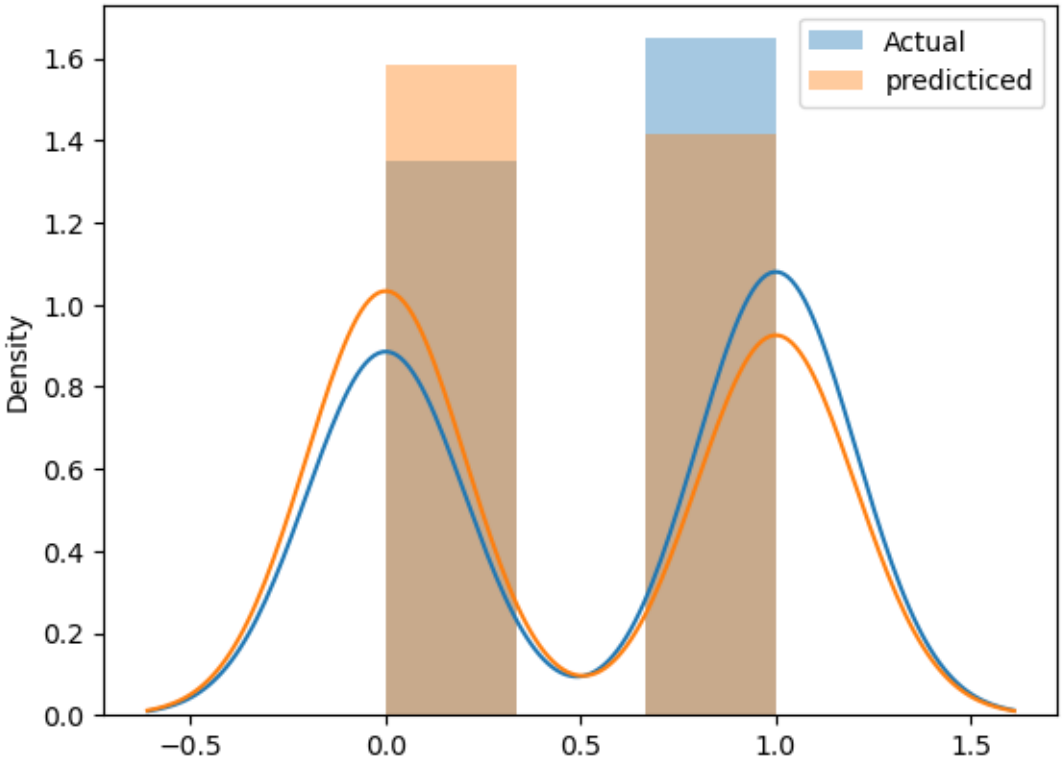
```
Accuracy: 0.75

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

```
              0       0.69      0.80      0.74        41
              1       0.81      0.70      0.75        50

       accuracy                           0.75        91
      macro avg       0.75      0.75      0.75        91
   weighted avg       0.76      0.75      0.75        91


Confusion Matrix:
[[33  8]
 [15 35]]
```

[20]:
```
sns.distplot(y_test,label='Actual')
sns.distplot(y_pred,label='predicticed')
plt.legend()
plt.show()
```

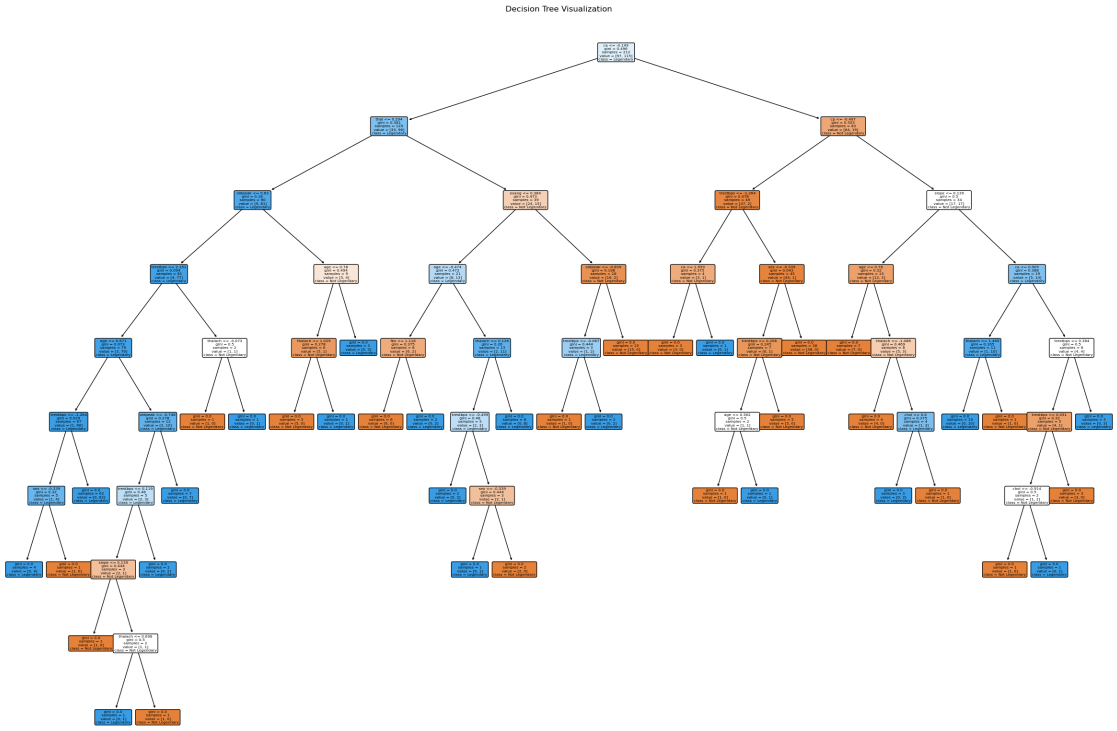
```
C:\Users\SIMRAN BANSAL\AppData\Local\Temp\ipykernel_5588\3633856401.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_test,label='Actual')
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\SIMRAN BANSAL\AppData\Local\Temp\ipykernel_5588\3633856401.py:2:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,label='predicticed')
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
```

```
with pd.option_context('mode.use_inf_as_na', True):
```



Visualize Decision Tree

[21]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(30, 20))
features = X.columns.tolist()
plot_tree(dt, feature_names=features, class_names=['Not Legendary',␣
 ↪'Legendary'], filled=True, rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
```



Visualize the Decision Tree using Graphviz

[22]:
```
# pip install graphviz
```

[23]:
```
from sklearn.tree import export_graphviz
import graphviz
```

[24]:
```
import os
graphviz_path = r'C:\Program Files\Graphviz\bin'
os.environ["PATH"] = f'{graphviz_path};{os.environ["PATH"]}'
```

[25]:
```
dot_data = export_graphviz(dt, out_file=None,
                           feature_names=X.columns,
                           class_names=np.unique(y).astype(str),
                           filled=True, rounded=True, special_characters=True)
```

[26]:
```
graph = graphviz.Source(dot_data)
graph.render("decision_tree", format="png", cleanup=True)
```

[26]:
```
'decision_tree.png'
```

[27]:
```
# Open the visualization in the default viewer
graph.view("decision_tree")
```

[27]:
```
'decision_tree.pdf'
```

Random Forest Classifier

[28]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=1000)
rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)
```

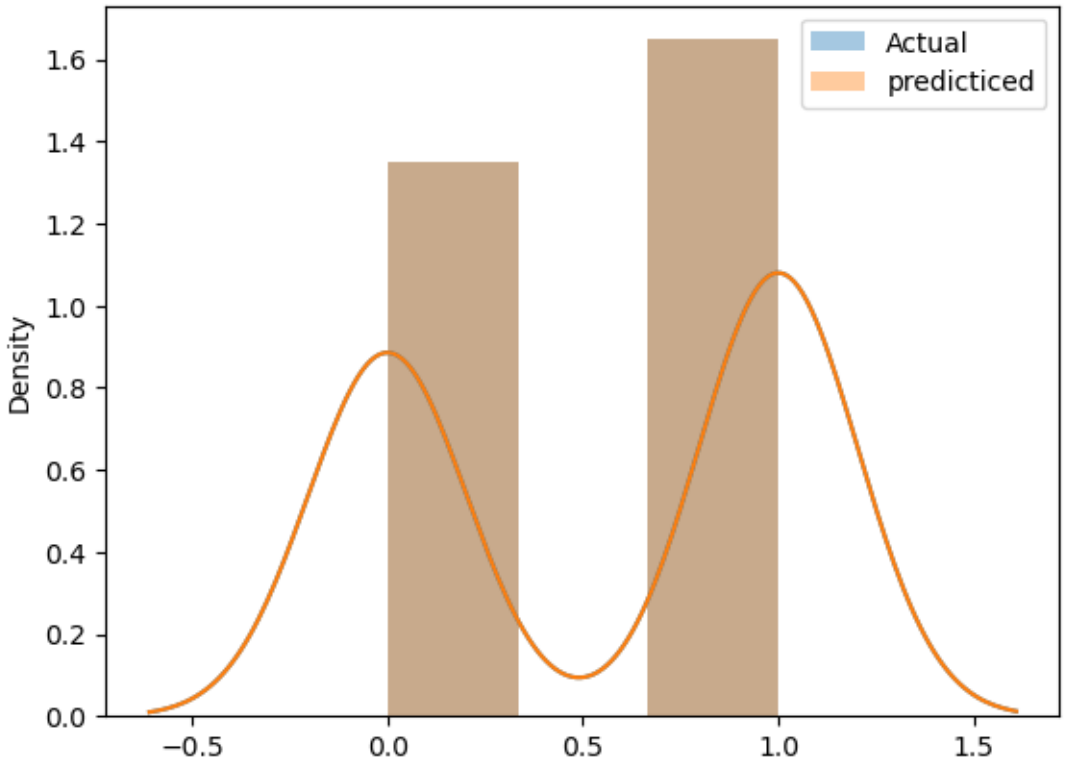
[29]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.80

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.78      0.78        41
           1       0.82      0.82      0.82        50

    accuracy                           0.80        91
   macro avg       0.80      0.80      0.80        91
weighted avg       0.80      0.80      0.80        91


Confusion Matrix:
[[32  9]
 [ 9 41]]
```

[30]:
```
sns.distplot(y_test,label='Actual')
sns.distplot(y_pred,label='predicticed')
plt.legend()
plt.show()
```

```
C:\Users\SIMRAN BANSAL\AppData\Local\Temp\ipykernel_5588\3633856401.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
```

```
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_test,label='Actual')
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\SIMRAN BANSAL\AppData\Local\Temp\ipykernel_5588\3633856401.py:2:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,label='predicticed')
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


Actual
predicticed
Density
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.5
0.0
0.5
1.0
1.5

[ ]:

[ ]:

[ ]: