

```
import sys #access to system parameters https://docs.python.org/3/library/sys.html
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib # collection of functions for scientific and publication-ready visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings # ignore warnings
warnings.filterwarnings('ignore')
import os
```

```
df = pd.read_csv(r'/content/heart.csv')
df.shape
```

```
(1025, 14)
```

```
df.head(3)
```

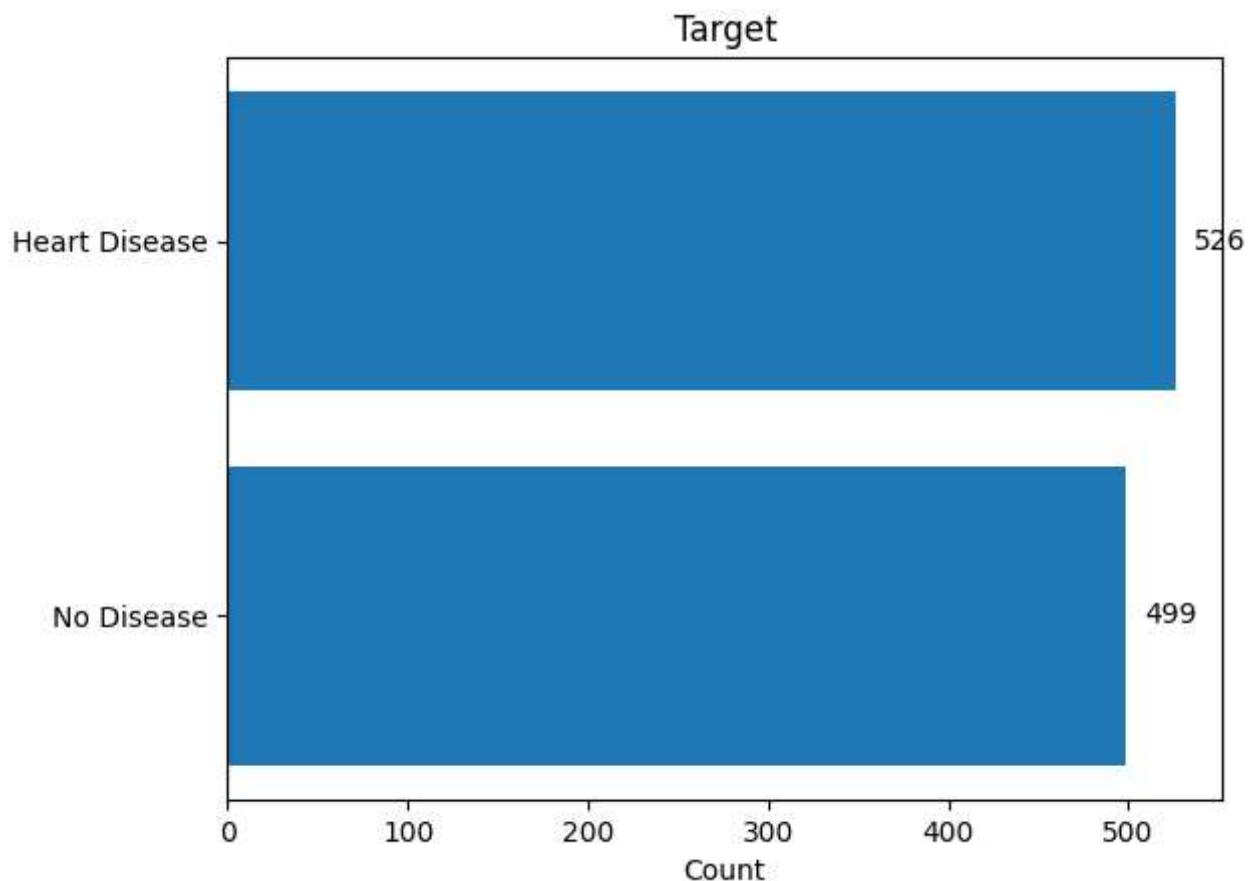
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
0	52	1	0	125	212	0	1	168	0	1.0	2	2	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	

```
df.target.value_counts()
```

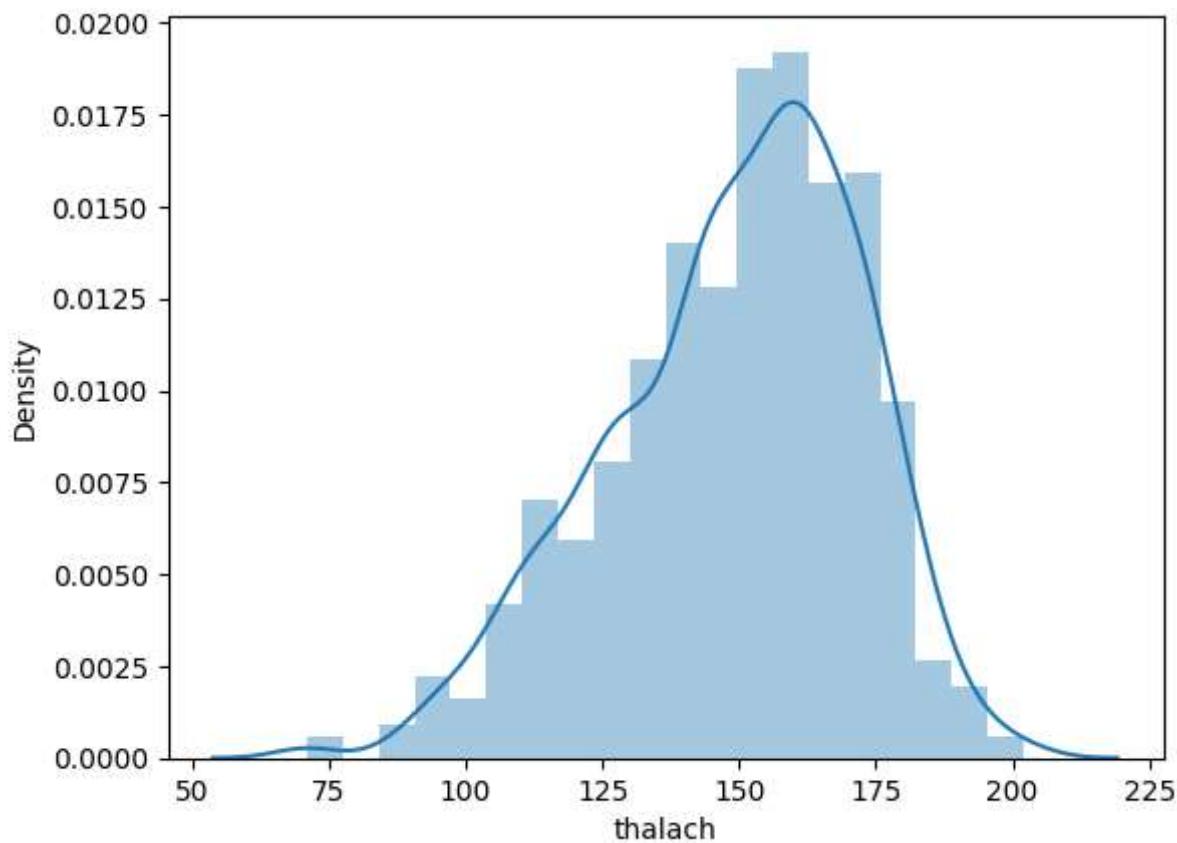
```
1    526
0    499
Name: target, dtype: int64
```

```
disease = len(df[df['target'] == 1])
no_disease = len(df[df['target']== 0])
```

```
import matplotlib.pyplot as plt
plt.rcParams()
fig, ax = plt.subplots()
y = ('Heart Disease', 'No Disease')
y_pos = np.arange(len(y))
x = (disease, no_disease)
ax.barh(y_pos, x, align='center')
ax.set_yticks(y_pos)
ax.set_yticklabels(y)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Count')
ax.set_title('Target')
for i, v in enumerate(x):
    ax.text(v + 10, i, str(v), color='black', va='center', fontweight='normal')
plt.show()
```



```
ax = sns.distplot(df['thalach'])
```



```
X = df.drop('target',1)
y = df['target']
```

```
print('shape of X and y respectively :', X.shape, y.shape)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
print('shape of X and y respectively (train) :', X_train.shape, y_train.shape)
print('shape of X and y respectively (test) :', X_test.shape, y_test.shape)

shape of X and y respectively : (1025, 13) (1025,)
shape of X and y respectively (train) : (820, 13) (820,)
shape of X and y respectively (test) : (205, 13) (205,)
```

```
print('RandomForestClassifier')
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100) # , max_depth=5, random_state=1
model.fit(X_train, y_train)
Y_pred = model.predict(X_test)
score = model.score(X_train, y_train)
print('Training Score:', score)
score = model.score(X_test, y_test)
print('Testing Score:', score)
output = pd.DataFrame({'Predicted':Y_pred}) # Heart-Disease yes or no? 1/0
print("-----")
print(output.head())
print("-----")
people = output.loc[output.Predicted == 1]["Predicted"]
rate_people = 0
if len(people) > 0 :
    rate_people = len(people)/len(output)
print("% of people predicted with heart-disease:", rate_people)
score_rfc = score
out_rfc = output
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred))
```

RandomForestClassifier

Training Score: 1.0

Testing Score: 1.0

 Predicted

0	1
1	0
2	0
3	1
4	0

% of people predicted with heart-disease: 0.5219512195121951

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

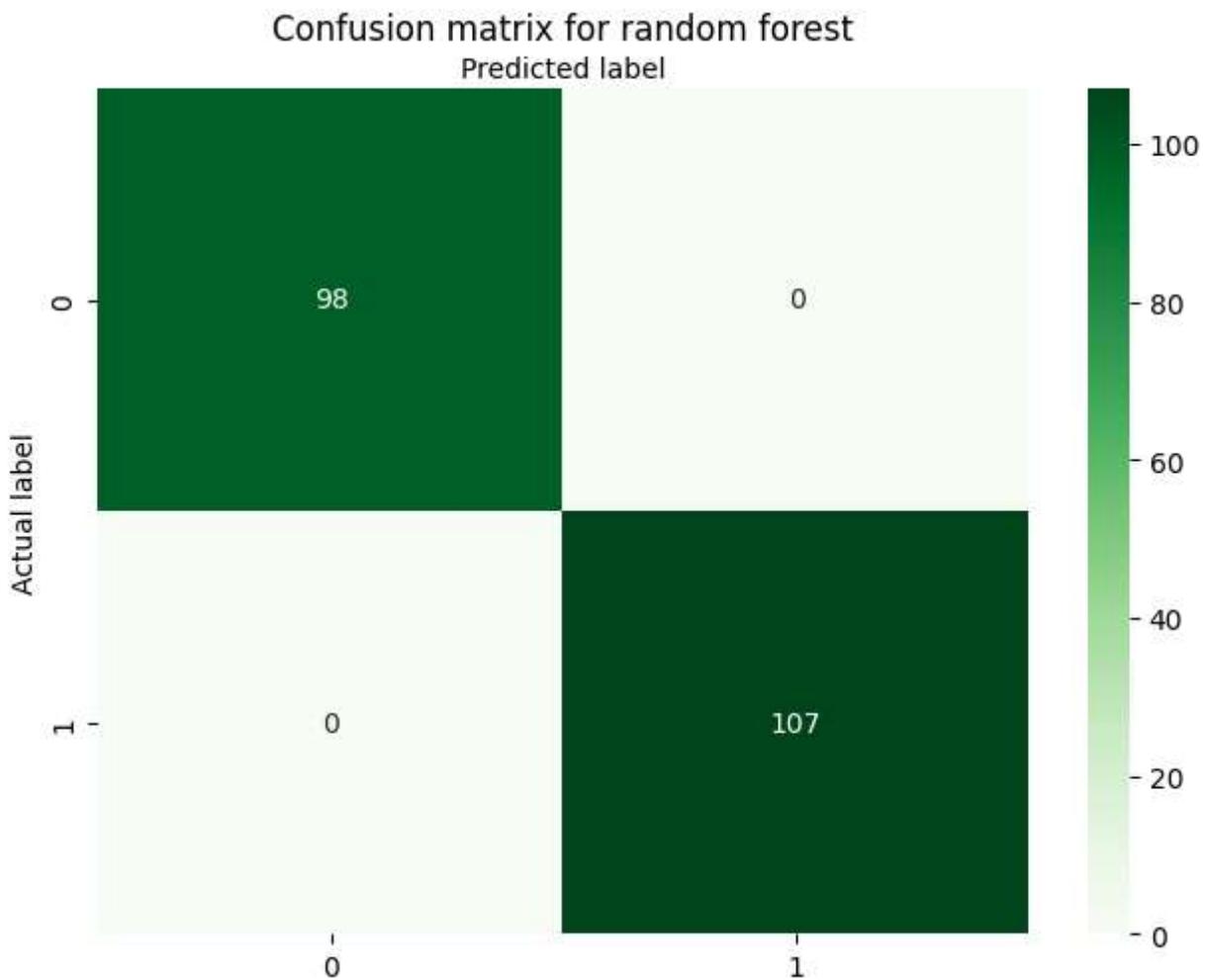
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	107

accuracy			1.00	205
----------	--	--	------	-----

macro avg	1.00	1.00	1.00	205
-----------	------	------	------	-----

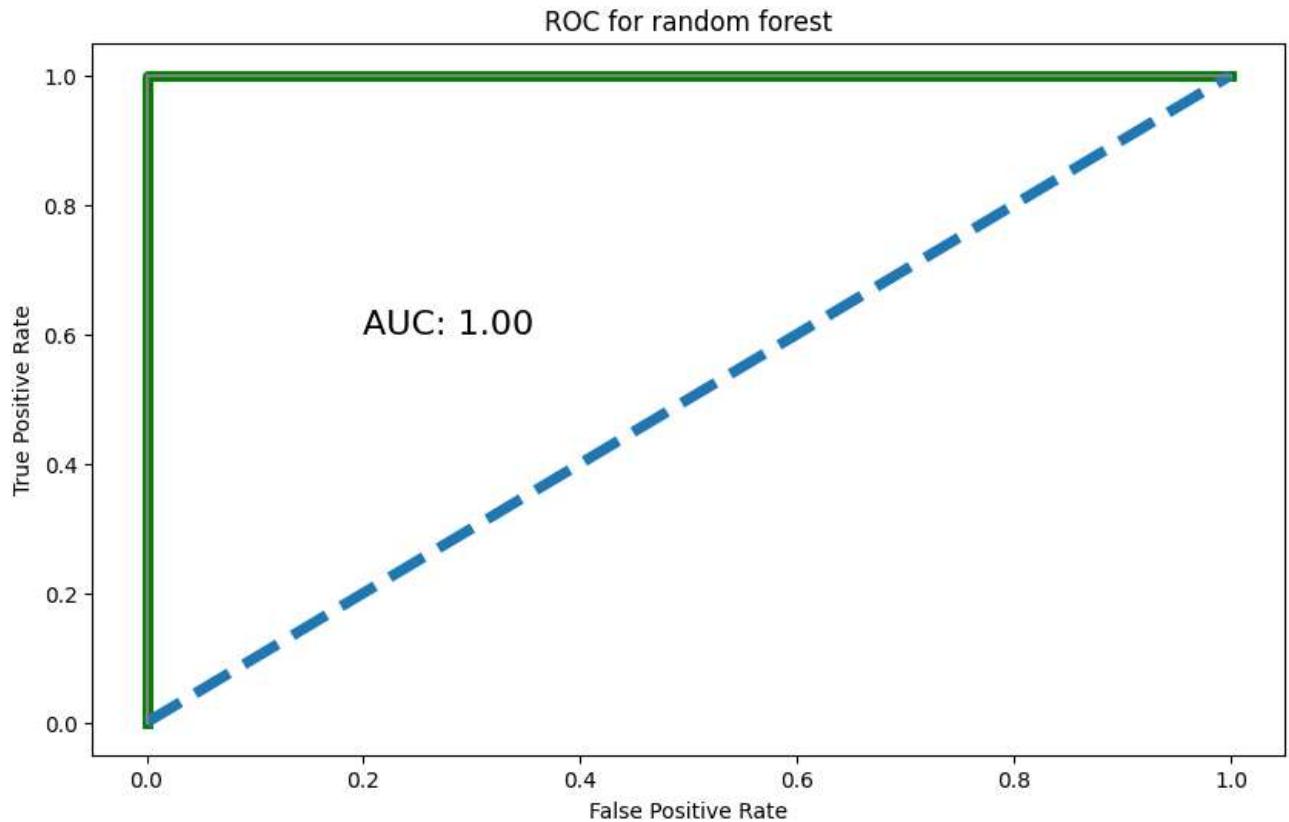
weighted avg	1.00	1.00	1.00	205
--------------	------	------	------	-----

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,Y_pred)
class_names = [0,1]
fig,ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks,class_names)
plt.yticks(tick_marks,class_names)
sns.heatmap(pd.DataFrame(confusion_matrix), annot = True, cmap = 'Greens', fmt = 'g')
ax.xaxis.set_label_position('top')
plt.tight_layout()
plt.title('Confusion matrix for random forest')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```



```
from sklearn.metrics import roc_auc_score,roc_curve
y_probabilities = model.predict_proba(X_test)[:,1]
false_positive_rate, true_positive_rate, threshold_knn = roc_curve(y_test,y_probabilities)
plt.figure(figsize=(10,6))
plt.title('ROC for random forest')
plt.plot(false_positive_rate, true_positive_rate, linewidth=5, color='green')
plt.plot([0,1],ls='--', linewidth=5)
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.text(0.2,0.6,'AUC: {:.2f}'.format(roc_auc_score(y_test,y_probabilities)),size= 16)
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.show()
```



[Colab paid products - Cancel contracts here](#)

✓ 0s completed at 10:14 PM

