

```

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

```

```

df = pd.read_csv('/content/heart_disease.csv')
df.head()

```

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	target	grid icon
0	40	1	2	140	289	0	0	172	0	0.0	1	0	bar chart icon
1	49	0	3	160	180	0	0	156	0	1.0	2	1	line chart icon
2	37	1	2	130	283	0	1	98	0	0.0	1	0	line chart icon
3	48	0	4	138	214	0	0	108	1	1.5	2	1	line chart icon
..	..	..	..	..	..	..	..	..	..	..	..	..	..

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   age              1190 non-null   int64  
 1   sex              1190 non-null   int64  
 2   chest pain type 1190 non-null   int64  
 3   resting bp s    1190 non-null   int64  
 4   cholesterol      1190 non-null   int64  
 5   fasting blood sugar 1190 non-null   int64  
 6   resting ecg      1190 non-null   int64  
 7   max heart rate  1190 non-null   int64  
 8   exercise angina 1190 non-null   int64  
 9   oldpeak          1190 non-null   float64 
 10  ST slope         1190 non-null   int64  
 11  target            1190 non-null   int64  
dtypes: float64(1), int64(11)
memory usage: 111.7 KB

```

```
df.describe()
```

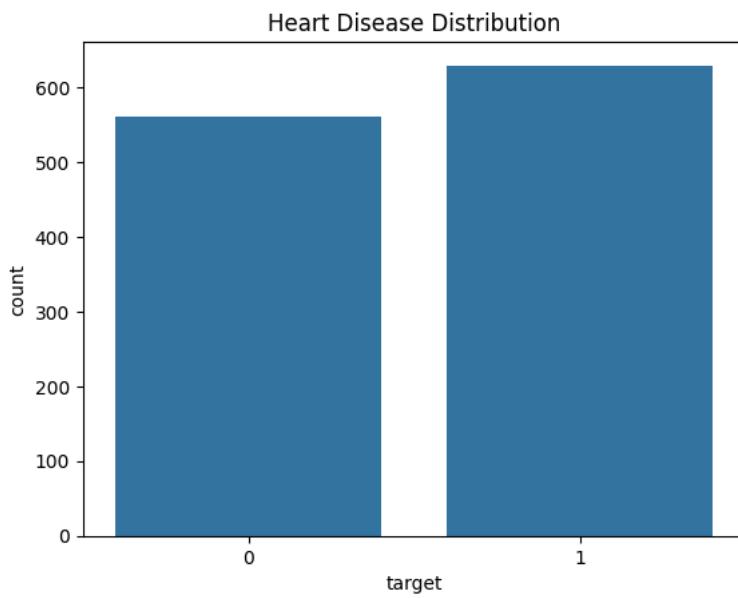
	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	o
count	1190.000000	1190.000000	1190.000000	1190.000000	1190.000000	1190.000000	1190.000000	1190.000000	1190.000000	1190.
mean	53.720168	0.763866	3.232773	132.153782	210.363866	0.213445	0.698319	139.732773	0.387395	0.
std	9.358203	0.424884	0.935480	18.368823	101.420489	0.409912	0.870359	25.517636	0.487360	1.
min	28.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	60.000000	0.000000	-2.
25%	47.000000	1.000000	3.000000	120.000000	188.000000	0.000000	0.000000	121.000000	0.000000	0.
50%	54.000000	1.000000	4.000000	130.000000	229.000000	0.000000	0.000000	140.500000	0.000000	0.
75%	60.000000	1.000000	4.000000	140.000000	269.750000	0.000000	2.000000	160.000000	1.000000	1.
max	77.000000	1.000000	4.000000	200.000000	603.000000	1.000000	2.000000	202.000000	1.000000	6.

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

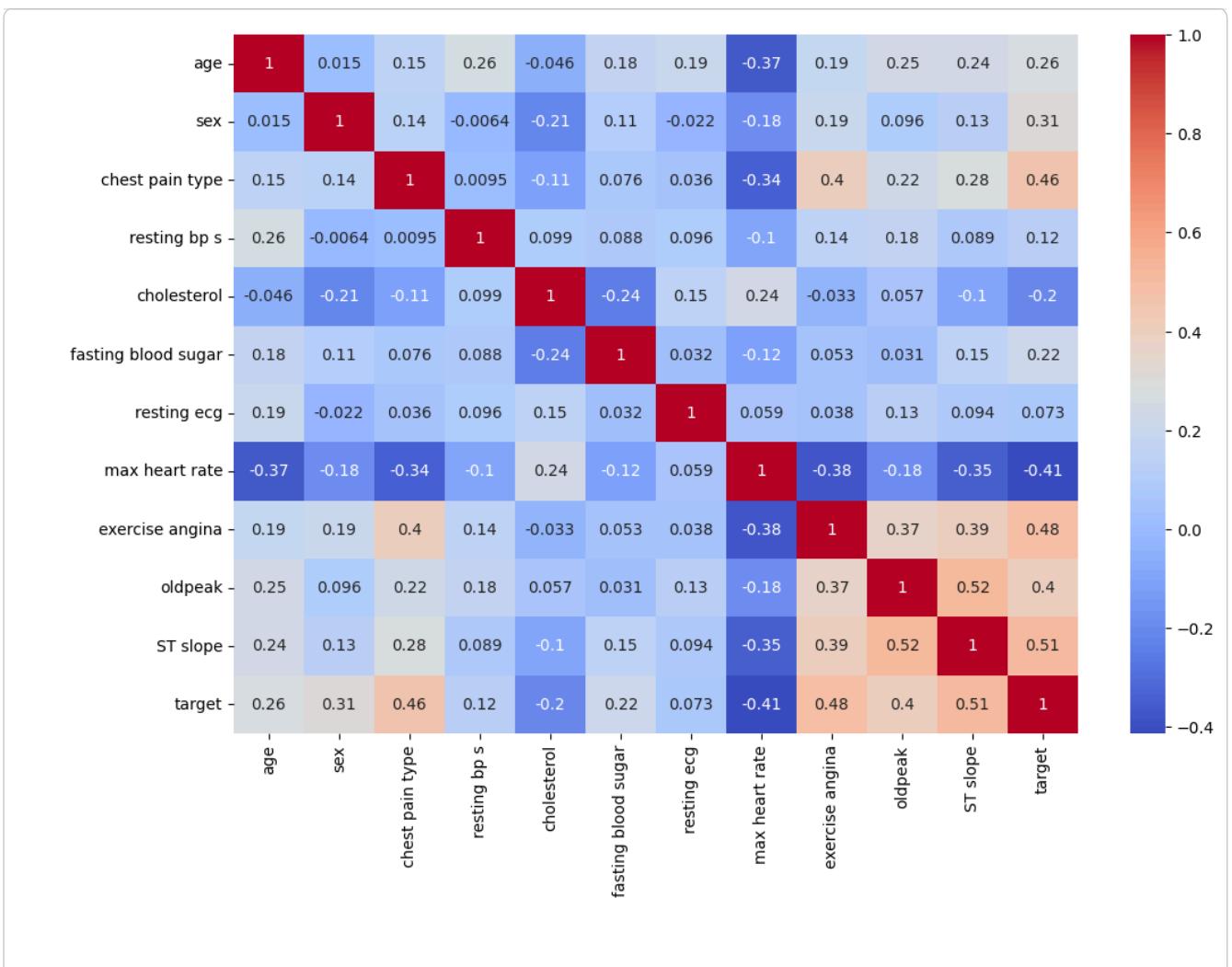
```
      0  
age    0  
sex    0  
chest pain type 0  
resting bp s    0  
cholesterol     0  
fasting blood sugar 0  
resting ecg      0  
max heart rate   0  
exercise angina   0  
oldpeak        0  
ST slope       0  
target         0
```

**dtype:** int64

```
sns.countplot(x='target', data=df)  
plt.title("Heart Disease Distribution")  
plt.show()
```



```
plt.figure(figsize=(12,8))  
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')  
plt.show()
```



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

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

▼ LogisticRegression ⓘ ?

```
LogisticRegression()
```

```
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
```

```
Model Accuracy: 0.8613445378151261
```

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

```
array([[ 90,  17],  
       [ 16, 115]])
```

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

	precision	recall	f1-score	support
0	0.85	0.84	0.85	107
1	0.87	0.88	0.87	131
accuracy			0.86	238
macro avg	0.86	0.86	0.86	238
weighted avg	0.86	0.86	0.86	238

```
new_patient = np.array([[52,1,2,140,240,1,1,160,0,1.2,2]])  
new_patient = scaler.transform(new_patient)
```

```
prediction = model.predict(new_patient)
```

```
if prediction[0] == 1:  
    print(" Heart Disease Detected")  
else:  
    print(" No Heart Disease")
```

```
Heart Disease Detected  
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names,  
warnings.warn(
```