


HEART DISEASE DETECTION using MACHINE LEARNING



```
Generated code may be subject to a license | abondar24/MachineLearnPython |
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
```

```
df=pd.read_csv('/heart_disease_data.csv')
```

```
df.head()
```




	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





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


```
df.tail()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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




```
df.shape
```



```
(303, 14)
```

```
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
```

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
count	303.000000	303.000000	303.000000	303.000000	303.000000	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	0.528053	149.646865	0.326733	1.039604	1.399340
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000

```
df['target'].value_counts()
```

	count
target	
1	165
0	138

df.describe()

0 = Healthy ; 1 = Unhealthy **bold text**

```
x = df.drop(columns='target',axis=1)
x
```

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

303 rows x 13 columns

Next steps: [Generate code with x](#) [View recommended plots](#) [New interactive sheet](#)

```
y = df['target']
y
```

	target
0	1
1	1
2	1
3	1
4	1
...	...
298	0
299	0
300	0
301	0
302	0

303 rows × 1 columns

dtype: int64

## Splitting dataset into Train & Test

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,stratify=y,random_state=23)
```

```
print(x_train.shape,x_test.shape)
```

```
(242, 13) (61, 13)
```

## Model Training

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

```
n_iter_i = _check_optimize_result(
```

```
  LogisticRegression
```

```
LogisticRegression()
```

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

```
accuracy_score(y_pred,y_test)
```

```
0.8524590163934426
```

```
print("The Accuraccy score is ",accuracy_score)
```

```
The Accuraccy score is <function accuracy_score at 0x7ca1f229df80>
```

## Building a Predictive System

```
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,0,2)
```

```
arr = np.array(input_data)
```

```
print(arr)
```

```
↕ [ 62.  0.  0. 140. 268.  0.  0. 160.  0.  3.6  0.  0.
    2. ]
```

```
prediction = model.predict(arr.reshape(1,-1))
```

```
↕ /usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Local
warnings.warn(
```

```
print(prediction)
```

```
↕ [0]
```

```
if prediction == 0:
    print("The person is Healthy")
else:
    print("The person is Unhealthy")
```

```
↕ The person is Healthy
```