HEART ATTACK CLASSIFICATION USING PYTHON

Python : 1

by : Leonardo FS

Objective

WHO data in 2015 showed that 70% of deaths in the world were caused by non-communicable diseases. Where 45% of them are caused by heart and blood vessel disease. Meanwhile, in 2020, cases of heart attacks in the world have killed nearly 10 million people.

The objective of this study is to analyze the "Heart Attack Classification" dataset to obtain:

1. Insight based on raw data through visualization
2. Factors that influence the cause of heart attacks
3. Classify heart attacks using a decision tree

by : Leonardo FS

Data Understanding

Dataset

- 14 Columns/Features/Variables
- 303 Rows
- Sumber
  (kaggle.com – Heart Attack Classification)

About this dataset

- Age : Age of the patient
- Sex : Sex of the patient
- Exang : exercise induced angina (1 = yes; 0 = no)
- ca : number of major vessels (0-3)
- trtbps : resting blood pressure (in mm Hg)
- chol : cholesterol in mg/dl fetched via BMI sensor
- fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

About this dataset

- cp : Chest Pain type chest pain type
  - -) Value 0: typical angina
  - -) Value 1: atypical angina
  - -) Value 2: non-anginal pain
  - -) Value 3: asymptomatic
- rest_ecg : resting electrocardiographic results
  - -) Value 0: normal
  - -) Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
  - -) Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
- thalach : maximum heart rate achieved
- Slp : ST Slope
- Oldpeak : Previous peak
- target : 0= less chance of heart attack 1= more chance of heart attack

by : Leonardo FS

Libraries

This analysis will use libraries from numpy, pandas, matplotlib, seaborn, and sklearn. Where :

1. Numpy and pandas work for data manipulation
2. Matplotlib and seaborn for visualization (EDA)
3. Sklearn for model building and evaluation.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.tree import plot_tree
import warnings
warnings.filterwarnings("ignore")
```

Load Data

```
df = pd.read_csv("heart.csv")
df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
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

by : Leonardo FS

Exploratory Data Analysis

Shape

```
df.shape
```

```
(303, 14)
```

Drop Duplicate Values

```
df.drop_duplicates(keep=False,inplace=True)
df.shape
```

```
(301, 14)
```

Based on the size of the data after dropping the duplicate values, then there are 2 duplicate values before which there are 2 duplicate values.

Missing Value

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

```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

Based on this result,
there is no missing value

by : Leonardo FS

Exploratory Data Analysis

```
df.describe()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
count	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000
mean	54.475083	0.681063	0.960133	131.581395	246.737542	0.149502	0.524917	149.491694	0.328904	1.046512	1.395349	0.707641	2.315615	0.541528
std	9.013150	0.466841	1.032023	17.588752	51.674503	0.357176	0.526191	22.901618	0.470597	1.161822	0.616316	0.990408	0.613777	0.499102
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	241.000000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	275.000000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

The results above are summary statistics. Based on this result, it is also known that our data needs to be standardized so that all attributes have an even effect on the output.

by : Leonardo FS

Exploratory Data Analysis

First of all, let's define new variables based on the output.

```
df_output0 = df[df["output"]==0]
df_output1 = df[df["output"]==1]
```

Let's visualize the non-category attributes based on the output.

```
column = ['age','trtbps','chol','thalachh','oldpeak']
for i in column:
    sns.displot(x=i,data=df_output0)
    plt.show()
```

```
column = ['age','trtbps','chol','thalachh','oldpeak']
for i in column:
    sns.displot(x=i,data=df_output1)
    plt.show()
```

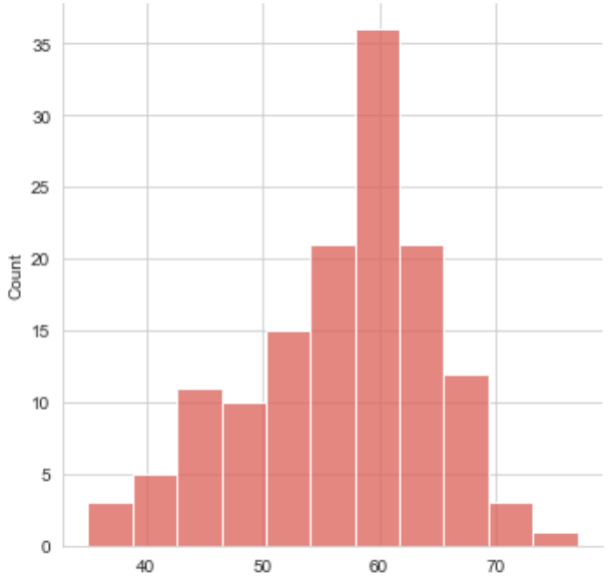
Next, let's do a comparison between the visualizations on each variables based on the output.

by : Leonardo FS

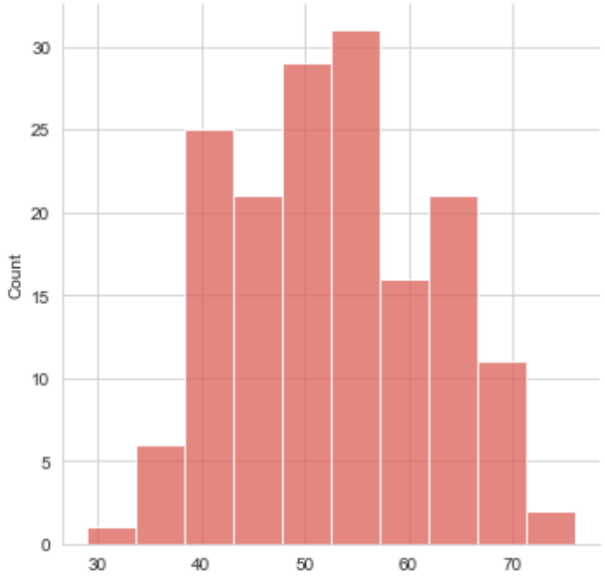
Exploratory Data Analysis

Age Variable

Output = 0



Output = 1



Analysis

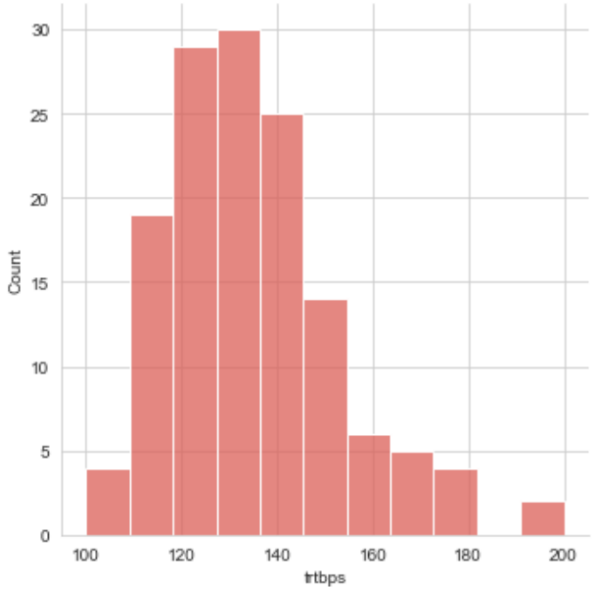
Based on the histogram results on the side, no conclusions can be drawn about the variable age that can cause a person to have a heart attack. This is due to the variation in the distribution. People who do not have a heart attack, most are in the age range of approximately 60 years. Meanwhile, people who had a heart attack were also in the age range of 50 to less than 60 years.

by : Leonardo FS

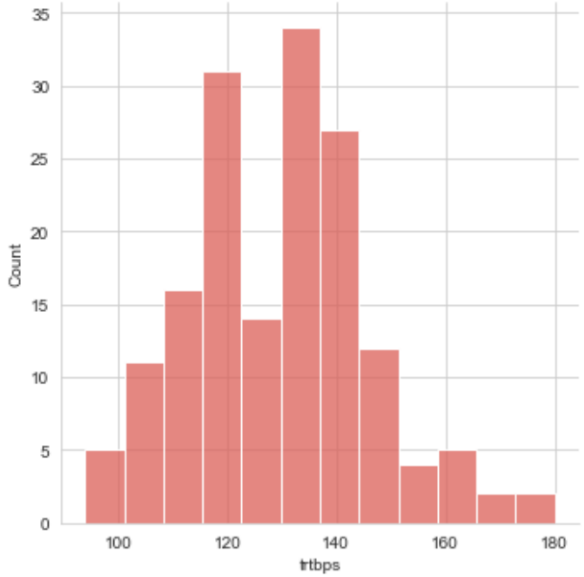
Exploratory Data Analysis

Resting Blood Pressure Variable

Output = 0



Output = 1



Analysis

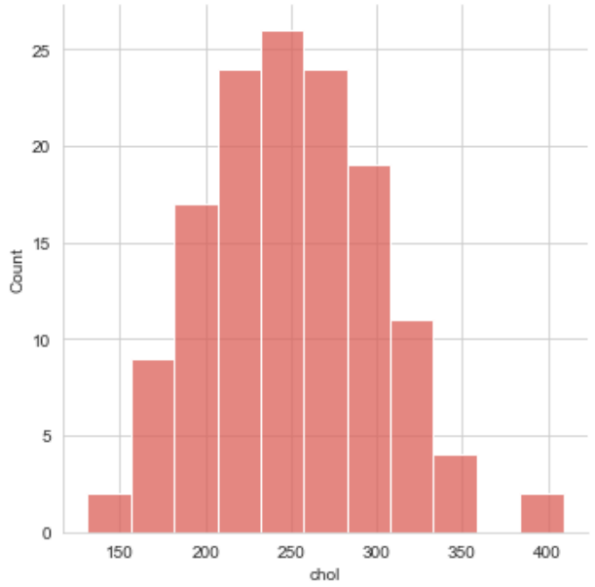
In someone who does not have a heart attack based on the resting blood pressure variable, the most spread is in the range of 120-140, but on the other hand, the range is also the most in people who have a heart attack.

by : Leonardo FS

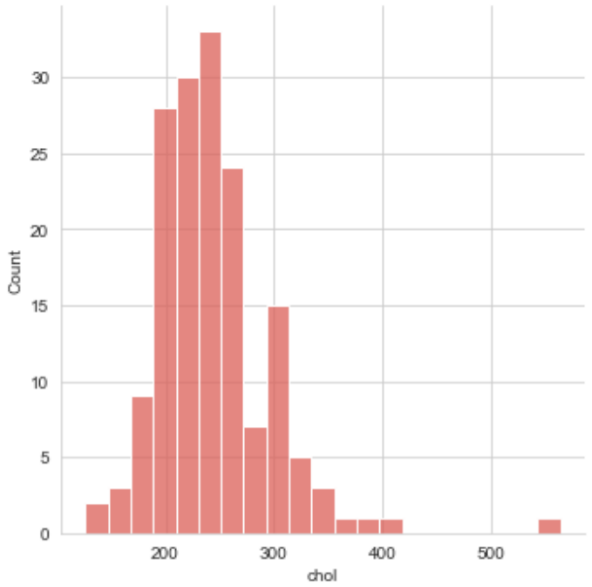
Exploratory Data Analysis

Cholesterol Variable

Output = 0



Output = 1



Analysis

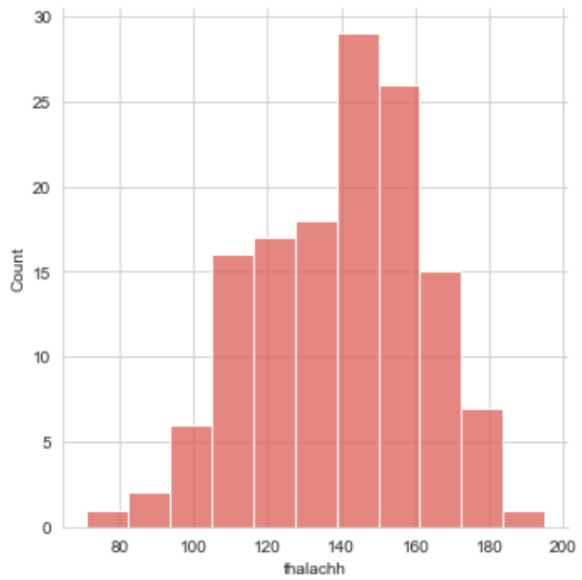
Based on the cholesterol variable, a person who has a heart attack, mostly has cholesterol levels above 200. However, the same goes for someone who doesn't have a heart attack. This identifies that there are other variables that have an influence on heart attacks.

by : Leonardo FS

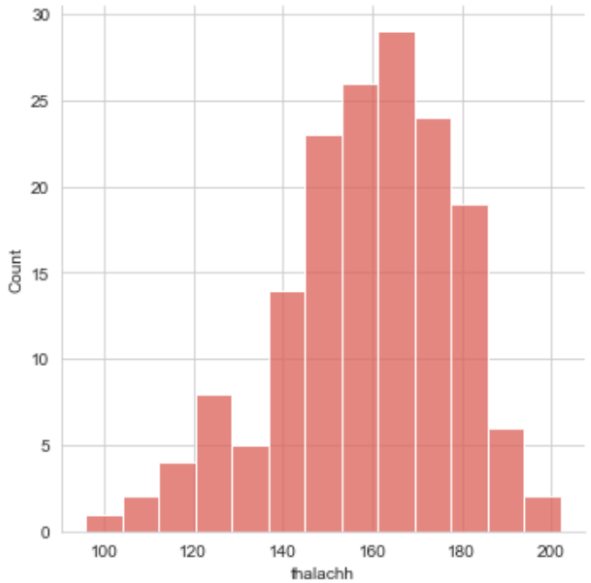
Exploratory Data Analysis

Thalach Variable

Output = 0



Output = 1



Analysis

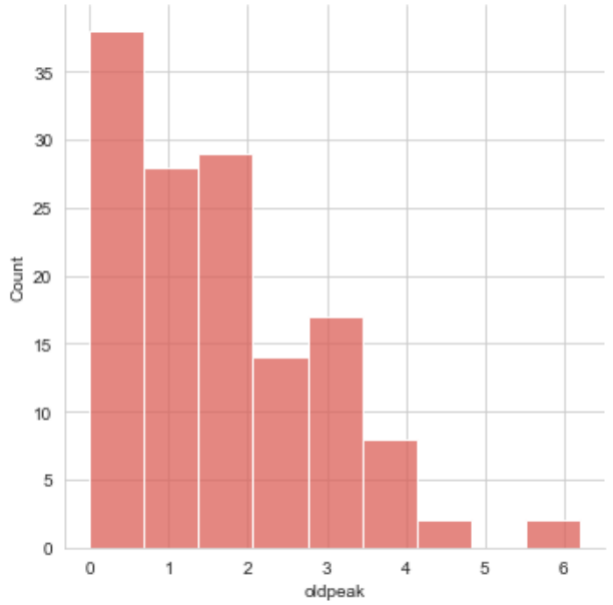
The talach variable is the maximum heart rate achieved. Based on the histogram on the side, a person who has not had a heart attack is at most in the maximum heart rate range of 140-160. While in someone who had a heart attack the most are in the range of 140-180.

by : Leonardo FS

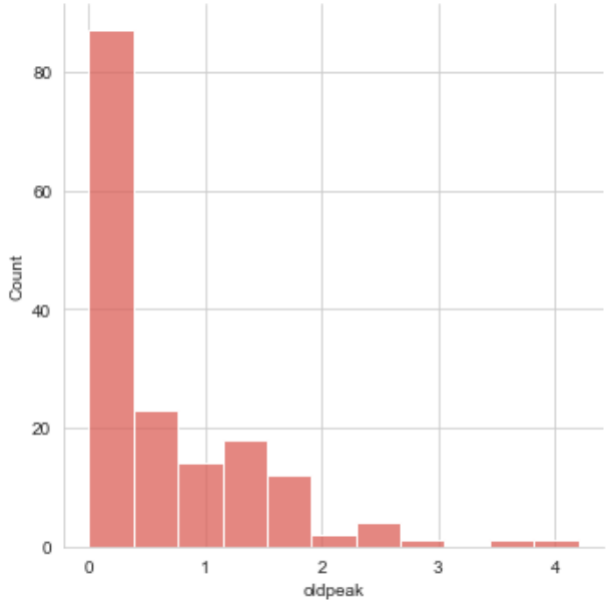
Exploratory Data Analysis

Oldpeak Variable

Output = 0



Output = 1



Analysis

Based on the old peak variable, a person who has a heart attack is mostly around 0 to less than 1.

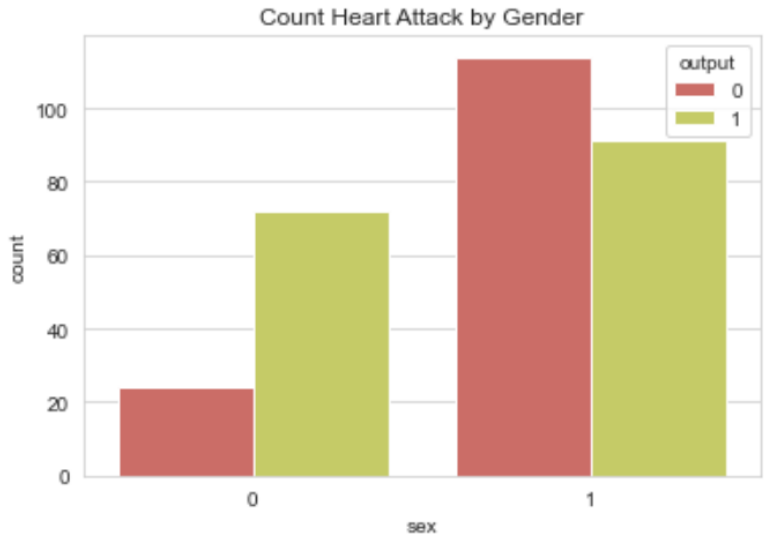
by : Leonardo FS

Exploratory Data Analysis

Sex Variable

```
sns.countplot(x="sex",data=df,hue="output").set_title("Count Heart Attack by Gender")
```

```
Text(0.5, 1.0, 'Count Heart Attack by Gender')
```



Analysis

Based on the visualization on the side, it is known that men have more heart attacks. However, it is not yet certain whether gender affects the cause of heart attacks. This is because in this dataset there are more men than women.

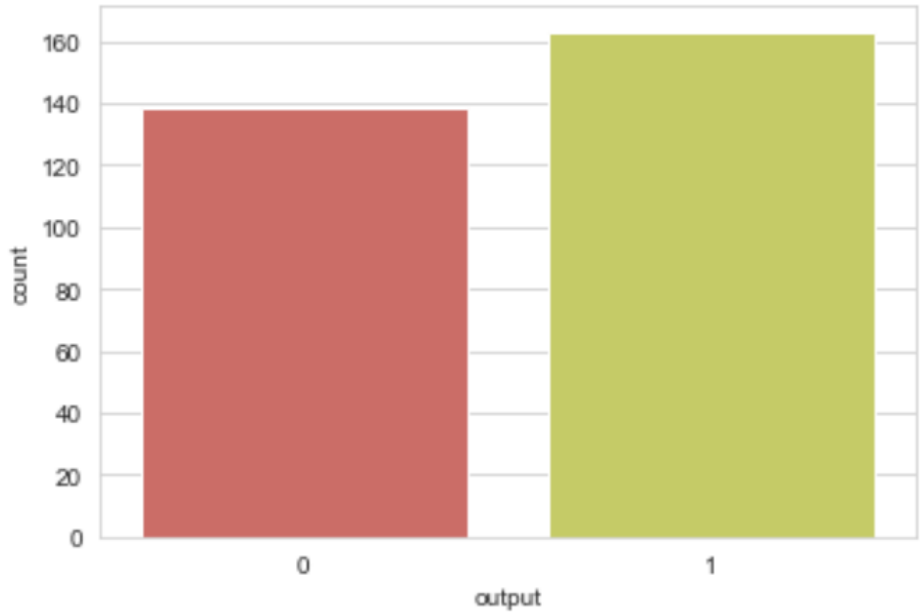
by : Leonardo FS

Exploratory Data Analysis

Output Variable

```
sns.countplot(x="output",data=df)
```

```
<AxesSubplot:xlabel='output', ylabel='count'>
```



Analysis

Based on the visualization on the side, in this dataset, there are more cases of heart attack than those who do not.

by : Leonardo FS

Exploratory Data Analysis

Correlation

```
plt.figure(figsize=(15,7))
sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```


```
df.corr()["output"].sort_values(ascending=False)
```

```
output      1.000000
cp          0.430339
thalachh    0.418146
slp         0.341977
restecg     0.132491
fbs        -0.025594
chol       -0.077575
trtbps     -0.147620
age        -0.217435
sex        -0.286312
thall      -0.342163
caa        -0.427160
oldpeak    -0.427577
exng       -0.434432
Name: output, dtype: float64
```

The information above is a variable that correlates with the output (heart attack) starting from a positive correlation to a negative correlation.

by : Leonardo FS

Data Preprocessing

Independent Variables

```
X = df.drop("output",axis=1)
y = df["output"]
```

```
X.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
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

Dependent Variable

```
y.head()
```

```
0    1
1    1
2    1
3    1
4    1
```

The purpose of defining the independent and dependent variables is to prepare split data to build the model.

by : Leonardo FS

Data Preprocessing

Standardization

```
ss = StandardScaler()
```

```
X = ss.fit(X).transform(X)
```

```
X[0:5]
```

```
array([[ 0.94740608,  0.68431914,  1.97986244,  0.76417876, -0.26629029,
         2.38513918, -0.99923977,  0.02223215, -0.70007072,  1.08069584,
        -2.26778684, -0.71568443, -2.14704327],
       [-1.9420717 ,  0.68431914,  1.00927841, -0.09005919,  0.06323992,
        -0.41926275,  0.90437524,  1.64052897, -0.70007072,  2.11527664,
        -2.26778684, -0.71568443, -0.51507351],
       [-1.49753665, -1.46130649,  0.03869438, -0.09005919, -0.82843006,
        -0.41926275, -0.99923977,  0.98446269, -0.70007072,  0.30476024,
         0.98270763, -0.71568443, -0.51507351],
       [ 0.16946975,  0.68431914,  0.03869438, -0.65955116, -0.2081379 ,
        -0.41926275,  0.90437524,  1.2468892 , -0.70007072, -0.21253016,
         0.98270763, -0.71568443, -0.51507351],
       [ 0.28060352, -1.46130649, -0.93188965, -0.65955116,  2.07918945,
        -0.41926275,  0.90437524,  0.59082292,  1.42842712, -0.3849603 ,
         0.98270763, -0.71568443, -0.51507351]])
```

Split Data

```
X_train, X_test, y_train,y_test = train_test_split(X,y,
                                                   test_size = 0.3,
                                                   random_state=42)
```

The purpose of standardization is so that all independent variables have the same effect when building the model. Then split the data into the training and testing data with a composition of 70% training data and 30% testing data.

by : Leonardo FS

Build Model & Evaluation

Standardization

```
ss = StandardScaler()
```

```
X = ss.fit(X).transform(X)
```

```
X[0:5]
```

```
array([[ 0.94740608,  0.68431914,  1.97986244,  0.76417876, -0.26629029,
         2.38513918, -0.99923977,  0.02223215, -0.70007072,  1.08069584,
        -2.26778684, -0.71568443, -2.14704327],
       [-1.9420717 ,  0.68431914,  1.00927841, -0.09005919,  0.06323992,
        -0.41926275,  0.90437524,  1.64052897, -0.70007072,  2.11527664,
        -2.26778684, -0.71568443, -0.51507351],
       [-1.49753665, -1.46130649,  0.03869438, -0.09005919, -0.82843006,
        -0.41926275, -0.99923977,  0.98446269, -0.70007072,  0.30476024,
         0.98270763, -0.71568443, -0.51507351],
       [ 0.16946975,  0.68431914,  0.03869438, -0.65955116, -0.2081379 ,
        -0.41926275,  0.90437524,  1.2468892 , -0.70007072, -0.21253016,
         0.98270763, -0.71568443, -0.51507351],
       [ 0.28060352, -1.46130649, -0.93188965, -0.65955116,  2.07918945,
        -0.41926275,  0.90437524,  0.59082292,  1.42842712, -0.3849603 ,
         0.98270763, -0.71568443, -0.51507351]])
```

Split Data

```
X_train, X_test, y_train,y_test = train_test_split(X,y,
                                                   test_size = 0.3,
                                                   random_state=42)
```

The purpose of standardization is so that all independent variables have the same effect when building the model. Then split the data into the training and testing data with a composition of 70% training data and 30% testing data.

by : Leonardo FS

Build Model & Evaluation

```
dtree = DecisionTreeClassifier(criterion="entropy", max_depth=9)
model_train = dtree.fit(X_train,y_train)
y_pred = dtree.predict(X_test)
score = accuracy_score(y_test,y_pred)
cm = confusion_matrix(y_test,y_pred)
cr = classification_report(y_test, y_pred)
print("Accuracy Score : ", score, "\n")
print("Classification Report : ","\n",cr, "\n")
cfm = sns.heatmap(pd.DataFrame(cm), annot=True)
plt.title("Confusion matrix for Decision Tree")
plt.ylabel("Actual label")
plt.xlabel("Predicted label")
plt.show()
```

Berdasarkan hasil evaluasi, pada model yang telah dibangun dengan decision tree. Diperoleh akurasi sebesar 78%.

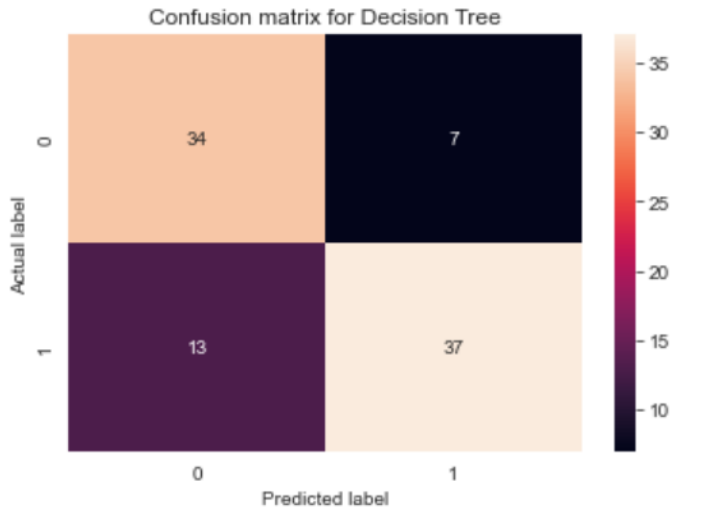
Evaluation

```
Accuracy Score :  0.7802197802197802

Classification Report :
              precision    recall  f1-score   support

           0       0.72      0.83      0.77        41
           1       0.84      0.74      0.79        50

    accuracy                           0.78        91
   macro avg       0.78      0.78      0.78        91
weighted avg       0.79      0.78      0.78        91
```



Conclusion

Based on the visualization of the heatmap or the resulting correlation, the cause of a person having a heart attack can be influenced by:

1. **Chest pain,** with a correlation value of 43%.
2. **Maximum heart rate**, with a correlation value of 41%.
3. **ST slope**, with a correlation value of 34%.
4. **Resting electrocardiograph**, with a correlation value of 13%.

Factors 1 to 3 have a moderate correlation to heart attacks. While 4th factor has a weak correlation to heart attacks.

Note: full code and the decision tree visualization can be seen on my github.

by : Leonardo FS

Thank You!