Importing Libraries and dataset

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

```
df=pd.read_csv('heart.csv')
df
```

	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 × 14 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

age: the age of the patient in years.

sex: the sex of the patient (1 = male, 0 = female).

cp: the type of chest pain the patient experienced (1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic).

trestbps: the resting blood pressure of the patient in mm Hg.

chol: the serum cholesterol level of the patient in mg/dl.

fbs: the fasting blood sugar level of the patient, measured in mg/dl (1 = high, 0 = low).

restecg: the resting electrocardiographic results of the patient (0 = normal, 1 = ST-T wave abnormality, 2 = left ventricular hypertrophy).

(Resting electrocardiographic (ECG or EKG) is a non-invasive diagnostic test that records the electrical activity of the heart while the patient is at rest. The test is performed using an electrocardiogram machine, which records the electrical signals produced by the heart through electrodes placed on the chest, arms, and legs.)

thalach: the maximum heart rate achieved by the patient during exercise. exang: whether the patient experienced exercise-induced angina (1 = yes, 0 = no).

oldpeak: the ST depression induced by exercise relative to rest. slope: the slope of the ST segment during peak exercise (1 = upsloping, 2 = flat, 3 = downsloping).

(ST depression induced by exercise relative to rest Oldpeak, also known as ST depression, is a common parameter measured during an exercise stress test to evaluate the presence and severity of coronary artery disease. It represents the amount of ST segment depression that occurs on an electrocardiogram (ECG) during exercise compared to rest.)

ca: the number of major vessels colored by fluoroscopy (0-3).

(he number of major vessels (0-3) colored by fluoroscopy is a parameter that is used to assess the severity of coronary artery disease (CAD) in patients who undergo coronary angiography)

thal: the type of thallium scan performed on the patient (1 = fixed defect, 2 = reversible defect, 3 = normal).

target: the presence of heart disease in the patient (0 = no disease, 1 = disease present).

EDA

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

```
np.int64(0)
```

```
df.duplicated().sum() #print the total number of duplicate rows in the data
```

```
np.int64(1)
```

```
df[df.duplicated()]  #print all the duplicate rows
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

```
df.drop_duplicates(inplace=True)
df.duplicated().sum()
```

```
np.int64(0)
```

```
#we should not worry about ouliers in the DT model as they get ignored whil taking decision
#label encoding ==> no object col are there in df
```

```
# model building
# 1.split the data in terms of x and y
# 2.split in terms of train and test
# 3.model initialization
# 4.train the model
# 5.prediction by model
# 6.evaluate , accuracy
# 7.hyperparameter tuning
# 8.visualize the tree
```

Machine learning Process

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
```

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

```
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
302 rows × 13 columns
```

Next steps: Generate code with x | View recommended plots | New interactive sheet

```
y
```

```
     target
  0       1
  1       1
  2       1
  3       1
  4       1
...     ...
298       0
299       0
300       0
301       0
302       0
302 rows × 1 columns

dtype: int64
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=25)
```

```
x_train
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
263	63	0	0	108	269	0	1	169	1	1.8	1	2	2
123	54	0	2	108	267	0	0	167	0	0.0	2	0	2
37	54	1	2	150	232	0	0	165	0	1.6	2	0	3
118	46	0	1	105	204	0	1	172	0	0.0	2	0	2
197	67	1	0	125	254	1	1	163	0	0.2	1	2	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
254	59	1	3	160	273	0	0	125	0	0.0	2	0	2
151	71	0	0	112	149	0	1	125	0	1.6	1	0	2
256	58	1	0	128	259	0	0	130	1	3.0	1	2	3
143	67	0	0	106	223	0	1	142	0	0.3	2	2	2
132	42	1	1	120	295	0	1	162	0	0.0	2	0	2

241 rows × 13 columns

Next steps: Generate code with x_train View recommended plots New interactive sheet

```
y_train
```

	target
263	0
123	1
37	1
118	1
197	0
...	...
254	0
151	1
256	0
143	1
132	1

241 rows × 1 columns

dtype: int64

```
x_test
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
58	34	1	3	118	182	0	0	174	0	0.0	2	0	2
48	53	0	2	128	216	0	0	115	0	0.0	2	0	0
40	51	0	2	140	308	0	0	142	0	1.5	2	1	2
104	50	1	2	129	196	0	1	163	0	0.0	2	0	2
68	44	1	1	120	220	0	1	170	0	0.0	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
154	39	0	2	138	220	0	1	152	0	0.0	1	0	2
247	66	1	1	160	246	0	1	120	1	0.0	1	3	1
194	60	1	2	140	185	0	0	155	0	3.0	1	0	2
73	51	1	0	140	261	0	0	186	1	0.0	2	0	2
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2

61 rows × 13 columns

Next steps: Generate code with x_test   View recommended plots   New interactive sheet

```
y_test
```

	target
58	1
48	1
40	1
104	1
68	1
...	...
154	1
247	0
194	0
73	1
1	1

61 rows × 1 columns

dtype: int64

Applying Decision Tree on Dataset

```
from sklearn.tree import DecisionTreeClassifier
```

```
model=DecisionTreeClassifier()
```

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

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

```
array([1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1])
```

```
accuracy_score(y_test,model_pred)*100
```

```
73.77049180327869
```

```
confusion_matrix(y_test,model_pred)
```

```
array([[17,  9],
       [ 7, 28]])
```

```
depth=[1,2,3,4,5,6,7,8,9,10]
for i in depth:
  model=DecisionTreeClassifier(max_depth=i)
  model.fit(x_train,y_train)
  model_pred=model.predict(x_test)
  acc=accuracy_score(y_test,model_pred)*100
  print(f"for the max depth {i} the accuracy score is: {acc}")
```

```
for the max depth 1 the accuracy score is: 67.21311475409836
for the max depth 2 the accuracy score is: 67.21311475409836
for the max depth 3 the accuracy score is: 75.40983606557377
for the max depth 4 the accuracy score is: 78.68852459016394
for the max depth 5 the accuracy score is: 78.68852459016394
for the max depth 6 the accuracy score is: 78.68852459016394
for the max depth 7 the accuracy score is: 78.68852459016394
for the max depth 8 the accuracy score is: 80.32786885245902
for the max depth 9 the accuracy score is: 77.04918032786885
for the max depth 10 the accuracy score is: 75.40983606557377
```

Final Decision Tree Model

```
final_model=DecisionTreeClassifier(max_depth=5)
final_model.fit(x_train,y_train)
```

```
▾   DecisionTreeClassifier   ⓘ ?

DecisionTreeClassifier(max_depth=5)
```

```
final_model_pred=final_model.predict(x_test)
final_model_pred
```

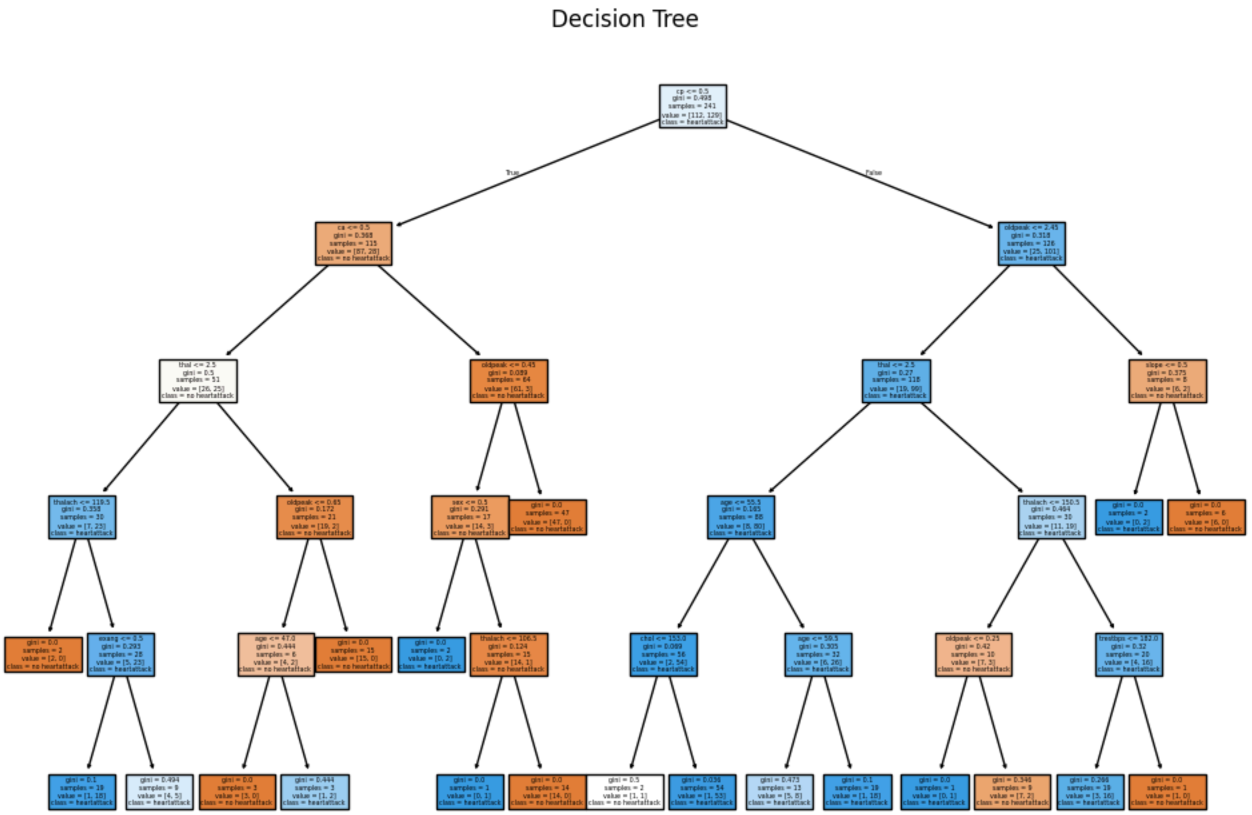
```
array([1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1])
```

```
accuracy_score(y_test,final_model_pred)*100
```

```
80.32786885245902
```

```
from sklearn.tree import plot_tree

plt.figure(figsize=(12,8))
plot_tree(final_model,filled=True,feature_names=x.columns,class_names=['no heartattack','heartattack'])
plt.title("Decision Tree")
plt.show()
```


Decision Tree
cp <= 0.5 gini = 0.498 samples = 241 value = [112, 129] class = heartattack
True
False
ca <= 0.5 gini = 0.368 samples = 115 value = [87, 28] class = no heartattack
oldpeak <= 2.45 gini = 0.318 samples = 126 value = [25, 101] class = heartattack
thal <= 2.5 gini = 0.5 samples = 51 value = [26, 25] class = no heartattack
oldpeak <= 0.45 gini = 0.089 samples = 64 value = [61, 3] class = no heartattack
thal <= 2.5 gini = 0.27 samples = 118 value = [19, 99] class = heartattack
slope <= 0.5 gini = 0.375 samples = 8 value = [6, 2] class = no heartattack
thalach <= 119.5 gini = 0.358 samples = 30 value = [7, 23] class = heartattack
oldpeak <= 0.85 gini = 0.172 samples = 21 value = [19, 2] class = no heartattack
sex <= 0.5 gini = 0.291 samples = 17 value = [14, 3] class = no heartattack
gini = 0.0 samples = 47 value = [47, 0] class = no heartattack
age <= 55.5 gini = 0.165 samples = 88 value = [8, 80] class = heartattack
thalach <= 150.5 gini = 0.464 samples = 30 value = [11, 19] class = heartattack
gini = 0.0 samples = 2 value = [0, 2] class = heartattack
gini = 0.0 samples = 6 value = [6, 0] class = no heartattack
gini = 0.0 samples = 2 value = [2, 0] class = no heartattack
exang <= 0.5 gini = 0.293 samples = 28 value = [5, 23] class = heartattack
age <= 47.0 gini = 0.444 samples = 6 value = [4, 2] class = no heartattack
gini = 0.0 samples = 15 value = [15, 0] class = no heartattack
gini = 0.0 samples = 2 value = [0, 2] class = heartattack
thalach <= 106.5 gini = 0.124 samples = 15 value = [14, 1] class = no heartattack
chol <= 153.0 gini = 0.069 samples = 56 value = [2, 54] class = heartattack
age <= 59.5 gini = 0.305 samples = 32 value = [6, 26] class = heartattack
oldpeak <= 0.25 gini = 0.42 samples = 10 value = [7, 3] class = no heartattack
trestbps <= 182.0 gini = 0.32 samples = 20 value = [4, 16] class = heartattack
gini = 0.1 samples = 19 value = [1, 18] class = heartattack
gini = 0.494 samples = 9 value = [4, 5] class = heartattack
gini = 0.0 samples = 3 value = [3, 0] class = no heartattack
gini = 0.444 samples = 3 value = [1, 2] class = heartattack
gini = 0.0 samples = 1 value = [0, 1] class = heartattack
gini = 0.0 samples = 14 value = [14, 0] class = no heartattack
gini = 0.5 samples = 2 value = [1, 1] class = no heartattack
gini = 0.036 samples = 54 value = [1, 53] class = heartattack
gini = 0.473 samples = 13 value = [5, 8] class = heartattack
gini = 0.1 samples = 19 value = [1, 18] class = heartattack
gini = 0.0 samples = 1 value = [0, 1] class = heartattack
gini = 0.346 samples = 9 value = [7, 2] class = no heartattack
gini = 0.266 samples = 19 value = [3, 16] class = heartattack
gini = 0.0 samples = 1 value = [1, 0] class = no heartattack