In [1]:
```
import numpy as ny
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
df=pd.read_csv(r"C:\Users\USER\Downloads\loan1.csv")
df
```

Out[2]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

In [3]:
```
df.head()
```

Out[3]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes

In [4]:
```
df.tail()
```

Out[4]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Home Owner          10 non-null     object
 1   Marital Status      10 non-null     object
 2   Annual Income       10 non-null     int64 
 3   Defaulted Borrower  10 non-null     object
dtypes: int64(1), object(3)
memory usage: 452.0+ bytes
```

In [6]:
```
df.describe()
```

Out[6]:

	Annual Income
count	10.000000
mean	104.000000
std	45.631373
min	60.000000
25%	77.500000
50%	92.500000
75%	115.000000
max	220.000000

In [7]:
```
df.isna().any()
```

Out[7]:
```
Home Owner            False
Marital Status        False
Annual Income         False
Defaulted Borrower    False
dtype: bool
```

In [8]:
```
df['Marital Status'].value_counts()
```

Out[8]:
```
Marital Status
Single      4
Married     4
Divorced    2
Name: count, dtype: int64
```

In [9]:
```
df['Annual Income'].value_counts()
```

Out[9]:
```
Annual Income
125    1
100    1
70     1
120    1
95     1
60     1
220    1
85     1
75     1
90     1
Name: count, dtype: int64
```

In [12]:
```
convert={'Home Owner':{'Yes':1,'No':0}}
df = df.replace(convert)
df
```

Out[12]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	1	Single	125	No
1	0	Married	100	No
2	0	Single	70	No
3	1	Married	120	No
4	0	Divorced	95	Yes
5	0	Married	60	No
6	1	Divorced	220	No
7	0	Single	85	Yes
8	0	Married	75	No
9	0	Single	90	Yes

In [31]:
```
cvt={'Defaulted Borrower':{'Yes':1,'No':0}}
df = df.replace(cvt)
print(df)
```

```
   Home Owner Marital Status  Annual Income  Defaulted Borrower
0           1         Single            125                   0
1           0        Married            100                   0
2           0         Single             70                   0
3           1        Married            120                   0
4           0       Divorced             95                   1
5           0        Married             60                   0
6           1       Divorced            220                   0
7           0         Single             85                   1
8           0        Married             75                   0
9           0         Single             90                   1
```

In [ ]:
```

```

In [32]:
```
convert = {'Marital Status': {'Single': 1, 'Married': 2, 'Divorced': 3}}
df = df.replace(convert)
df
```

Out[32]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	1	1	125	0
1	0	2	100	0
2	0	1	70	0
3	1	2	120	0
4	0	3	95	1
5	0	2	60	0
6	1	3	220	0
7	0	1	85	1
8	0	2	75	0
9	0	1	90	1

In [33]:
```
x=['Home Owner','Marital Status','Annual Income']
y=['Yes','No']
all_inputs=df[x]
all_classes=df['Defaulted Borrower']
```

In [34]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.25)
```

In [35]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [36]:
```
clf.fit(x_train,y_train)
```

Out[36]:
```
DecisionTreeClassifier(random_state=0)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [37]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.3333333333333333
```

In [38]:
```
import numpy as ny
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [39]:
```
df=pd.read_csv(r"C:\Users\USER\Downloads\drug200.csv")
df
```

Out[39]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

In [40]:
```
df.head()
```

Out[40]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

In [41]:
```
df.tail()
```

Out[41]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

In [42]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          200 non-null    int64  
 1   Sex          200 non-null    object 
 2   BP           200 non-null    object 
 3   Cholesterol  200 non-null    object 
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object 
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

In [43]:
```
df.describe()
```

Out[43]:

	Age	Na_to_K
count	200.000000	200.000000
mean	44.315000	16.084485
std	16.544315	7.223956
min	15.000000	6.269000
25%	31.000000	10.445500
50%	45.000000	13.936500
75%	58.000000	19.380000
max	74.000000	38.247000

In [44]:
```
df['Cholesterol'].value_counts()
```

Out[44]:
```
Cholesterol
HIGH      103
NORMAL     97
Name: count, dtype: int64
```

In [45]:
```
df['Na_to_K'].value_counts()
```

Out[45]:
```
Na_to_K
12.006    2
18.295    2
25.355    1
11.939    1
16.347    1
         ..
24.658    1
24.276    1
13.967    1
19.675    1
11.349    1
Name: count, Length: 198, dtype: int64
```

In [65]:
```
convert={'Drug':{'drugX':0,'drugY':1,'drugA':2,'drugB':3,'drugC':4}}
df = df.replace(convert)
df
```

Out[65]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	1	HIGH	HIGH	25.355	1
1	47	0	LOW	HIGH	13.093	4
2	47	0	LOW	HIGH	10.114	4
3	28	1	NORMAL	HIGH	7.798	0
4	61	1	LOW	HIGH	18.043	1
...	...	...	...	...	...	...
195	56	1	LOW	HIGH	11.567	4
196	16	0	LOW	HIGH	12.006	4
197	52	0	NORMAL	HIGH	9.894	0
198	23	0	NORMAL	NORMAL	14.020	0
199	40	1	LOW	NORMAL	11.349	0

200 rows × 6 columns

In [66]:
```
cvt={'Sex':{'F':1,'M':0}}
df = df.replace(cvt)
print(df)
```

```
     Age  Sex       BP Cholesterol  Na_to_K  Drug
0     23    1     HIGH        HIGH   25.355     1
1     47    0      LOW        HIGH   13.093     4
2     47    0      LOW        HIGH   10.114     4
3     28    1   NORMAL        HIGH    7.798     0
4     61    1      LOW        HIGH   18.043     1
..   ...  ...      ...         ...      ...   ...
195   56    1      LOW        HIGH   11.567     4
196   16    0      LOW        HIGH   12.006     4
197   52    0   NORMAL        HIGH    9.894     0
198   23    0   NORMAL      NORMAL   14.020     0
199   40    1      LOW      NORMAL   11.349     0

[200 rows x 6 columns]
```

In [83]:
```
convert = {'BP': {'HIGH':1,'NORMAL':2,'LOW':3}}
df = df.replace(convert)
df
```

Out[83]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	1	1	HIGH	25.355	1
1	47	0	3	HIGH	13.093	4
2	47	0	3	HIGH	10.114	4
3	28	1	2	HIGH	7.798	0
4	61	1	3	HIGH	18.043	1
...	...	...	...	...	...	...
195	56	1	3	HIGH	11.567	4
196	16	0	3	HIGH	12.006	4
197	52	0	2	HIGH	9.894	0
198	23	0	2	NORMAL	14.020	0
199	40	1	3	NORMAL	11.349	0

200 rows × 6 columns

In [88]:
```
x=['Drug','Sex','BP']
y=['HIGH','NORMAL']
all_inputs=df[x]
all_classes=df['Cholesterol']
```

In [89]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.25)
```

In [90]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [91]:
```
clf.fit(x_train,y_train)
```

Out[91]:
```
DecisionTreeClassifier(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:

In [ ]: