In [ ]:

```

```

Decision tree

In [1]:

```
import numpy as np
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\mural\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.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 [4]:

```
df['Marital Status'].value_counts()
```

Out[4]:

```
Marital Status
Single      4
Married     4
Divorced    2
Name: count, dtype: int64
```

In [5]:

```
df['Annual Income'].value_counts()
```

Out[5]:

```
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 [6]:

```
convert={"Home Owner":{"Yes":1,"No":0}}
df=df.replace(convert)
df
```

Out[6]:

	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 [7]:

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

Out[7]:

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

In [8]:

```
x=["Home Owner","Annual Income"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["Defaulted Borrower"]
```

In [9]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.5)
```

In [10]:

```
clf=DecisionTreeClassifier(random_state=0)
```

In [11]:

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

Out[11]:

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [12]:

```
score=clf.score(x_test,y_test)
print(score)
```

```
1.0
```

DRUG DATA

In [1]:

```
import numpy as np
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\mural\Downloads\drug200.csv")
df
```

Out[2]:

	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 [3]:

```
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 [4]:

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

Out[4]:

```
Sex
M    104
F     96
Name: count, dtype: int64
```

In [5]:

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

Out[5]:

```
BP
HIGH      77
LOW       64
NORMAL    59
Name: count, dtype: int64
```

In [6]:

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

Out[6]:

```
Cholesterol
HIGH      103
NORMAL     97
Name: count, dtype: int64
```

In [7]:

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

Out[7]:

```
Drug
drugY    91
drugX    54
drugA    23
drugC    16
drugB    16
Name: count, dtype: int64
```

In [8]:

```
convert={"Sex":{"M":1,"F":0}}
df=df.replace(convert)
df
```

Out[8]:

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

200 rows × 6 columns

In [9]:

```
convert={"BP":{"HIGH":2,"LOW":0,"NORMAL":1}}
df=df.replace(convert)
df
```

Out[9]:

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

200 rows × 6 columns

In [10]:

```
convert={"Cholesterol":{"HIGH":2,"NORMAL":1}}
df=df.replace(convert)
df
```

Out[10]:

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

200 rows × 6 columns

In [11]:

```
x=["Sex","BP","Cholesterol"]
y=["DrugY","DrugX","DrugA","DrugC","DrugB"]
all_inputs=df[x]
all_classes=df["Drug"]
```

In [12]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.3)
clf=DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
```

Out[12]:

```
▾       DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [13]:

```
score=clf.score(x_test,y_test)
print(score)
```

```
0.36666666666666664
```