In [1]:  ▶| 
```python
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]:  ▶| 
```python
df=pd.read_csv(r"C:\Users\chinta pavani\Documents\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]:  ▶| 
```python
df['Marital Status'].value_counts()
df['Annual Income'].value_counts()
```

Out[3]:
```
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 [4]:  ▶| 
```
convert={"Home Owner":{"Yes":1,"No":0}}
df=df.replace(convert)
df
```

Out[4]:

|   | 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 [5]:  ▶| 
```
convert={"Marital Status":{"Single":1,"Married":2,"Divorced":3}}
df=df.replace(convert)
df
```

Out[5]:

|   | 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 [6]: ▶| `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     int64
 1   Marital Status     10 non-null     int64
 2   Annual Income      10 non-null     int64
 3   Defaulted Borrower 10 non-null     object
dtypes: int64(3), object(1)
memory usage: 452.0+ bytes
```

In [7]: ▶| `df.describe()`

Out[7]:

|        | Home Owner | Marital Status | Annual Income |
|--------|-----------|----------------|---------------|
| count  | 10.000000 | 10.000000      | 10.000000     |
| mean   | 0.300000  | 1.800000       | 104.000000    |
| std    | 0.483046  | 0.788811       | 45.631373     |
| min    | 0.000000  | 1.000000       | 60.000000     |
| 25%    | 0.000000  | 1.000000       | 77.500000     |
| 50%    | 0.000000  | 2.000000       | 92.500000     |
| 75%    | 0.750000  | 2.000000       | 115.000000    |
| max    | 1.000000  | 3.000000       | 220.000000    |

In [8]: ▶| `df.head()`

Out[8]:

|   | 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                |

In [9]: ▶|
```python
x=["Home Owner","Marital Status","Annual Income"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["Defaulted Borrower"]
```

In [10]: ▶| `(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,te`

In [11]: ▶| `clf=DecisionTreeClassifier(random_state=0)`

In [12]:  ▶| 
```python
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

Out[12]: 
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [13]:  ▶| 
```python
score=clf.score(x_test,y_test)
print(score)
```

```
0.6666666666666666
```

In [ ]:  ▶|