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\my pc\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 [5]:

```
df.tail()
```

Out[5]:

	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 [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     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: 448.0+ bytes
```

In [7]:
```
df.describe()
```

Out[7]:

	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 [8]:
```
df.isna().any()
```

Out[8]:
```
Home Owner            False
Marital Status        False
Annual Income         False
Defaulted Borrower    False
dtype: bool
```

In [9]:
```
df["Marital Status"].value_counts()
```

Out[9]:
```
Marital Status
Single      4
Married     4
Divorced    2
Name: count, dtype: int64
```

In [10]:
```
df["Defaulted Borrower"].value_counts()
```

Out[10]:
```
Defaulted Borrower
No     7
Yes    3
Name: count, dtype: int64
```

In [11]:
```
df
```

Out[11]:

	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 [12]:
```
c={"Home Owner":{"Yes":1,"No":0}}
df=df.replace(c)
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 [13]:
```
c={'Marital Status':{'Single':1,'Married':2,'Divorced':3}}
df=df.replace(c)
df
```

Out[13]:

	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 [14]:
```
c={'Defaulted Borrower':{'Yes':1,"No":2}}
df=df.replace(c)
df
```

Out[14]:

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

In [15]:
```
x=["Home Owner","Marital Status","Annual Income"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["Defaulted Borrower"]
```

In [16]:
```
#Splitting the data ito train and test data
x_train,x_test,y_train,y_test=train_test_split(all_inputs,all_classes,test_size=0.25)
```

In [17]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [18]:
```
clf.fit(x_train,y_train)                #Fitting training into th model (DecisionTreeClassifie
```

Out[18]:
```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

```
In [19]: score=clf.score(x_test,y_test)    #To find the score for test data
```

```
In [20]: print(score)
```

```
1.0
```

```
In [21]: clf.score(x_train,y_train)        #To find the score for training data
```

```
Out[21]: 1.0
```