# DECISION TREE (LOAN)

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\prajapath Arjun\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]:

```python
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]:

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

Out[6]:

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

In [7]:

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

Out[7]:

```
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 [8]:

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

Out[8]:

| | 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 [9]:

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

Out[9]:

| | 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 [10]:

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

In [11]:

```python
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[11]:

```
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 [12]:

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

```
0.3333333333333333
```