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\HP\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: 448.0+ 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]: c={"Home Owner":{"Yes":1,"No":0}}
df=df.replace(c)
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 [11]:
```
c={"Home Owner":{"Yes":1,"No":0}}
df=df.replace(c)
df
```

Out[11]:

	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 [16]:
```
c={"Martial Status",{"Single":1,"Married":2,"Divorced":3}}
df=df.replace(c)
df
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 1
----> 1 c={"Martial Status",{"Single":1,"Married":2,"Divorced":3}}
      2 df=df.replace(c)
      3 df

TypeError: unhashable type: 'dict'
```

In [14]:
```
x=["Home Owner","Marital Status","Annual Income"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["Defaulted Borrower"]
```

In [15]:
```
x_train,x_test,y_train,y_test=train_test_split(all_inputs,all_classes,test_siz
clf=DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_23240\2072522248.py in ?()
      1 x_train,x_test,y_train,y_test=train_test_split(all_inputs,all_classe
s,test_size=0.25)
      2 clf=DecisionTreeClassifier(random_state=0)
----> 3 clf.fit(x_train,y_train)
      4
      5
      6

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\tree\_cla
sses.py in ?(self, X, y, sample_weight, check_input)
    885         self : DecisionTreeClassifier
    886             Fitted estimator.
    887         """
    888
--> 889         super().fit(
    890             X,
    891             y,
    892             sample_weight=sample_weight,

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\tree\_cla
sses.py in ?(self, X, y, sample_weight, check_input)
    182             # We can't pass multi_output=True because that would allo
w y to be
    183             # csr.
    184             check_X_params = dict(dtype=DTYPE, accept_sparse="csc")
    185             check_y_params = dict(ensure_2d=False, dtype=None)
--> 186             X, y = self._validate_data(
    187                 X, y, validate_separately=(check_X_params, check_y_pa
rams)
    188             )
    189             if issparse(X):

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py i
n ?(self, X, y, reset, validate_separately, **check_params)
    575                 # :(
    576                 check_X_params, check_y_params = validate_separately
    577                 if "estimator" not in check_X_params:
    578                     check_X_params = {**default_check_params, **check
_X_params}
--> 579                 X = check_array(X, input_name="X", **check_X_params)
    580                 if "estimator" not in check_y_params:
    581                     check_y_params = {**default_check_params, **check
_y_params}
    582                 y = check_array(y, input_name="y", **check_y_params)

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\val
idation.py in ?(array, accept_sparse, accept_large_sparse, dtype, order, cop
y, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_feat
ures, estimator, input_name)
    876                     )
    877                     array = xp.astype(array, dtype, copy=False)
    878                 else:
    879                     array = _asarray_with_order(array, order=order, d
type=dtype, xp=xp)
```

```
--> 880                 except ComplexWarning as complex_warning:
    881                     raise ValueError(
    882                         "Complex data not supported\n{}\n".format(array)
    883                     ) from complex_warning

~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\_ar
ray_api.py in ?(array, dtype, order, copy, xp)
    181     if xp is None:
    182         xp, _ = get_namespace(array)
    183     if xp.__name__ in {"numpy", "numpy.array_api"}:
    184         # Use NumPy API to support order
--> 185         array = numpy.asarray(array, order=order, dtype=dtype)
    186         return xp.asarray(array, copy=copy)
    187     else:
    188         return xp.asarray(array, dtype=dtype, copy=copy)

~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\gener
ic.py in ?(self, dtype)
   1996     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.nda
rray:
   1997         values = self._values
-> 1998         arr = np.asarray(values, dtype=dtype)
   1999         if (
   2000             astype_is_view(values.dtype, arr.dtype)
   2001             and using_copy_on_write()

ValueError: could not convert string to float: 'Divorced'
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```