

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [2]: df=pd.read_csv(r"C:\Users\DELL E5490\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]: x=df.drop('Defaulted Borrower',axis=1)
y=df['Defaulted Borrower']

In [5]: df['Marital Status'].value_counts()

Out[5]: Single 4
Married 4
Divorced 2
Name: Marital Status, dtype: int64

```
In [6]: HO={"Home Owner":{"Yes":1,'No':0}}
df=df.replace(HO)
print(df)
```

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

	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=df.drop('Defaulted Borrower',axis=1)
y=df['Defaulted Borrower']
```

```
In [9]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=42)
x_train.shape,x_test.shape
```

Out[9]: ((7, 3), (3, 3))

```
In [10]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[10]:

```

  RandomForestClassifier
  RandomForestClassifier()
```

```
In [11]: rf=RandomForestClassifier()
```

```
In [12]: params={'max_depth':[2,3,5,10,20],
               'min_samples_leaf':[5,10,20,50,100,200],
               'n_estimators':[10,25,30,50,100,200]}
```

```
In [16]: from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
grid_search.fit(x_train,y_train)
```

Out[16]:

```

  GridSearchCV
  estimator: RandomForestClassifier
    RandomForestClassifier
```

```
In [17]: grid_search.best_score_
```

Out[17]: 0.5833333333333333

```
In [20]: rf_best=grid_search.best_estimator_  
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

```
In [23]: from sklearn.tree import plot_tree  
plt.figure(figsize=(80,40))  
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'])
```

```
Out[23]: [Text(0.5, 0.5, 'gini = 0.49\nsamples = 5\nvalue = [4, 3]\nnclass = Yes')]
```

gini = 0.49
samples = 5
value = [4, 3]
class = Yes

```
In [25]: from sklearn.tree import plot_tree  
plt.figure(figsize=(80,40))  
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=['Yes','No'])
```

```
Out[25]: [Text(0.5, 0.5, 'gini = 0.49\nsamples = 6\nvalue = [3, 4]\nclass = No')]
```

gini = 0.49
samples = 6
value = [3, 4]
class = No

```
In [26]: rf_best.feature_importances_
```

```
Out[26]: array([0., 0., 0.])
```

```
In [28]: imp_df=pd.DataFrame({'Varname':x_train.columns,"Imp":rf_best.feature_importances_})  
imp_df.sort_values(by="Imp",ascending=False)
```

Out[28]:

	Varname	Imp
0	Home Owner	0.0
1	Marital Status	0.0
2	Annual Income	0.0

In []: