Loan

Random forest

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

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\loan.csv")[0:500]
df
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loan/
0	LP001015	Male	Yes	0	Graduate	No	5720	0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	
...	...	...	...	...	...	...	...	...	
362	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	1777	
363	LP002975	Male	Yes	0	Graduate	No	4158	709	
364	LP002980	Male	No	0	Graduate	No	3250	1993	
365	LP002986	Male	Yes	0	Graduate	No	5000	2393	
366	LP002989	Male	No	0	Graduate	Yes	9200	0	

367 rows × 12 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            367 non-null    object 
 1   Gender             356 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         357 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      344 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [5]:
```
d=df[[ 'ApplicantIncome', 'CoapplicantIncome']]
```

In [6]:
```
d['CoapplicantIncome'].value_counts()
```

Out[6]:
```
0       156
2000      3
700       3
1083      2
1517      2
       ...
4309      1
2774      1
3803      1
2845      1
3333      1
Name: CoapplicantIncome, Length: 194, dtype: int64
```

In [7]:
```
x=d.drop('CoapplicantIncome',axis=1)
y=d['CoapplicantIncome']
```

In [8]:
```
g1={"CoapplicantIncome":{"":0,0:1}}
d=d.replace(g1)
print(d)
```

```
     ApplicantIncome  CoapplicantIncome
0               5720                  1
1               3076               1500
2               5000               1800
3               2340               2546
4               3276                  1
..               ...                ...
362             4009               1777
363             4158                709
364             3250               1993
365             5000               2393
366             9200                  1

[367 rows x 2 columns]
```

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [10]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [11]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [12]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[12]:
```
RandomForestClassifier()
```

In [13]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [14]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:666: UserWarnin
g: The least populated class in y has only 1 members, which is less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[14]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [15]:
```
grid_search.best_score_
```

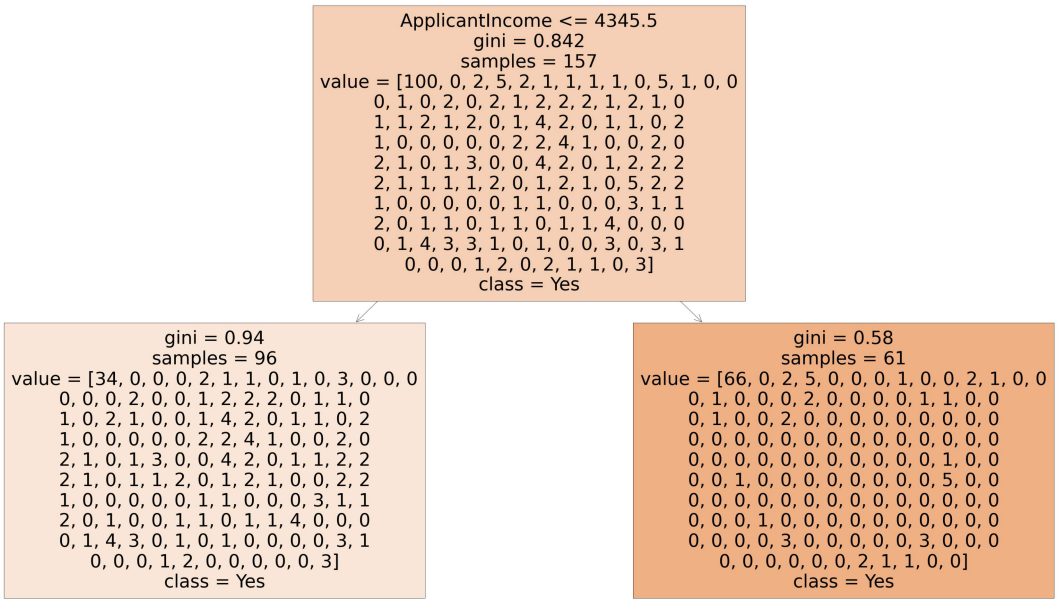
Out[15]:
```
0.4296875
```

In [16]:
```
from sklearn.tree import plot_tree
```

In [17]:
```
rfc_best=grid_search.best_estimator_
```

In [18]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True
```

Out[18]:
```
[Text(2232.0, 1630.8000000000002, 'ApplicantIncome <= 4345.5\ngini = 0.842\nsamples = 157\nva
lue = [100, 0, 2, 5, 2, 1, 1, 1, 1, 0, 5, 1, 0, 0\n0, 1, 0, 2, 0, 2, 1, 2, 2, 2, 1, 2, 1, 0\n
1, 1, 2, 1, 2, 0, 1, 4, 2, 0, 1, 1, 0, 2\n1, 0, 0, 0, 0, 0, 2, 2, 4, 1, 0, 0, 2, 0\n2, 1, 0,
1, 3, 0, 0, 4, 2, 0, 1, 2, 2, 2\n2, 1, 1, 1, 1, 2, 0, 1, 2, 1, 0, 5, 2, 2\n1, 0, 0, 0, 0, 0,
1, 1, 0, 0, 0, 3, 1, 1\n2, 0, 1, 1, 0, 1, 1, 0, 1, 1, 4, 0, 0, 0\n0, 1, 4, 3, 3, 1, 0, 1, 0,
0, 3, 0, 3, 1\n0, 0, 0, 1, 2, 0, 2, 1, 1, 0, 3]\nclass = Yes'),
 Text(1116.0, 543.5999999999999, 'gini = 0.94\nsamples = 96\nvalue = [34, 0, 0, 0, 2, 1, 1,
0, 1, 0, 3, 0, 0, 0\n0, 0, 0, 2, 0, 0, 1, 2, 2, 2, 0, 1, 1, 0\n1, 0, 2, 1, 0, 0, 1, 4, 2, 0,
1, 1, 0, 2\n1, 0, 0, 0, 0, 0, 2, 2, 4, 1, 0, 0, 2, 0\n2, 1, 0, 1, 3, 0, 0, 4, 2, 0, 1, 1, 2,
2\n2, 1, 0, 1, 1, 2, 0, 1, 2, 1, 0, 0, 2, 2\n1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 3, 1, 1\n2, 0,
1, 0, 0, 1, 1, 0, 1, 1, 4, 0, 0, 0\n0, 1, 4, 3, 0, 1, 0, 1, 0, 0, 0, 0, 3, 1\n0, 0, 0, 1, 2,
0, 0, 0, 0, 0, 3]\nclass = Yes'),
 Text(3348.0, 543.5999999999999, 'gini = 0.58\nsamples = 61\nvalue = [66, 0, 2, 5, 0, 0, 0,
1, 0, 0, 2, 1, 0, 0\n0, 1, 0, 0, 0, 2, 0, 0, 0, 0, 1, 1, 0, 0\n0, 1, 0, 0, 2, 0, 0, 0, 0, 0,
0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0\n0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 3, 0, 0, 0\n0, 0, 0, 0, 0,
0, 2, 1, 1, 0, 0]\nclass = Yes')]
```



In [ ]:
```
```