In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.tree import plot_tree
```

In [2]:
```
df=pd.read_csv("C8_loan-test - C8_loan-test.csv")
df
```

Out[2]:

| Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term |
|---|---|---|---|---|---|
| Graduate | No | 5720 | 0 | 110.0 | 360.0 |
| Graduate | No | 3076 | 1500 | 126.0 | 360.0 |
| Graduate | No | 5000 | 1800 | 208.0 | 360.0 |
| Graduate | No | 2340 | 2546 | 100.0 | 360.0 |
| Not Graduate | No | 3276 | 0 | 78.0 | 360.0 |
| ... | ... | ... | ... | ... | ... |
| Not Graduate | Yes | 4009 | 1777 | 113.0 | 360.0 |
| Graduate | No | 4158 | 709 | 115.0 | 360.0 |
| Graduate | No | 3250 | 1993 | 126.0 | 360.0 |
| Graduate | No | 5000 | 2393 | 158.0 | 360.0 |
| Graduate | Yes | 9200 | 0 | 98.0 | 180.0 |

In [3]:
```
df1=df.fillna(value=0)
df1
```

Out[3]:

| Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term |
|---|---|---|---|---|---|
| Graduate | No | 5720 | 0 | 110.0 | 360.0 |
| Graduate | No | 3076 | 1500 | 126.0 | 360.0 |
| Graduate | No | 5000 | 1800 | 208.0 | 360.0 |
| Graduate | No | 2340 | 2546 | 100.0 | 360.0 |
| Not Graduate | No | 3276 | 0 | 78.0 | 360.0 |
| ... | ... | ... | ... | ... | ... |
| Not Graduate | Yes | 4009 | 1777 | 113.0 | 360.0 |
| Graduate | No | 4158 | 709 | 115.0 | 360.0 |
| Graduate | No | 3250 | 1993 | 126.0 | 360.0 |
| Graduate | No | 5000 | 2393 | 158.0 | 360.0 |
| Graduate | Yes | 9200 | 0 | 98.0 | 180.0 |

In [4]:
```
df1.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             367 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         367 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      367 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         367 non-null    float64
 9   Loan_Amount_Term   367 non-null    float64
 10  Credit_History     367 non-null    float64
 11  Property_Area      367 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

In [5]:
```
df1.columns
```

Out[5]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [39]:
```
df2=df1[[ 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area']]
df2
```

Out[39]:

| | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Prop |
|---|---|---|---|---|---|---|
| **0** | 5720 | 0 | 110.0 | 360.0 | 1.0 | |
| **1** | 3076 | 1500 | 126.0 | 360.0 | 1.0 | |
| **2** | 5000 | 1800 | 208.0 | 360.0 | 1.0 | |
| **3** | 2340 | 2546 | 100.0 | 360.0 | 0.0 | |
| **4** | 3276 | 0 | 78.0 | 360.0 | 1.0 | |
| **...** | ... | ... | ... | ... | ... | |
| **362** | 4009 | 1777 | 113.0 | 360.0 | 1.0 | |
| **363** | 4158 | 709 | 115.0 | 360.0 | 1.0 | |
| **364** | 3250 | 1993 | 126.0 | 360.0 | 0.0 | |
| **365** | 5000 | 2393 | 158.0 | 360.0 | 1.0 | |
| **366** | 9200 | 0 | 98.0 | 180.0 | 1.0 | |

367 rows × 6 columns

In [40]:
```
df2['Property_Area'].value_counts()
```

Out[40]:
```
Urban        140
Semiurban    116
Rural        111
Name: Property_Area, dtype: int64
```

In [41]:
```
x=df2.drop('Property_Area',axis=1)
y=df2['Property_Area']
```

In [42]:
```
g1={"Property_Area":{'Urban':1,"Rural":2,"Semiurban":3}}
df2=df2.replace(g1)
print(df2)
```

```
     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5720                  0       110.0             360.0
1               3076               1500       126.0             360.0
2               5000               1800       208.0             360.0
3               2340               2546       100.0             360.0
4               3276                  0        78.0             360.0
..               ...                ...         ...               ...
362             4009               1777       113.0             360.0
363             4158                709       115.0             360.0
364             3250               1993       126.0             360.0
365             5000               2393       158.0             360.0
366             9200                  0        98.0             180.0

     Credit_History  Property_Area
0               1.0              1
1               1.0              1
2               1.0              1
3               0.0              1
4               1.0              1
..              ...            ...
362             1.0              1
363             1.0              1
364             0.0              3
365             1.0              2
366             1.0              2

[367 rows x 6 columns]
```

In [43]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.70)
```

In [44]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[44]:
```
RandomForestClassifier()
```

In [45]:
```
parameters = {'max_depth':[1,2,3,4,5],
              'min_samples_leaf':[5,10,15,20,25],
               'n_estimators':[10,20,30,40,50]}
```

In [46]:
```
grid_search = GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring='ac
grid_search.fit(x_train,y_train)
```

Out[46]:
```
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 [47]:
```
grid_search.best_score_
```

Out[47]:
```
0.47272727272727266
```

In [48]:
```
rfc_best =grid_search.best_estimator_
```

In [50]:
```
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

```
 Text(558.0, 815.3999999999999, 'LoanAmount <= 80.5\ngini = 0.577\nsamples
= 18\nvalue = [6, 5, 15]'),
 Text(279.0, 271.7999999999997, 'gini = 0.418\nsamples = 11\nvalue = [1, 3,
11]'),
 Text(837.0, 271.7999999999997, 'gini = 0.628\nsamples = 7\nvalue = [5, 2,
4]'),
 Text(1116.0, 815.3999999999999, 'gini = 0.305\nsamples = 9\nvalue = [0, 1
3, 3]'),
 Text(2232.0, 1359.0, 'CoapplicantIncome <= 146.0\ngini = 0.619\nsamples =
25\nvalue = [14, 6, 17]'),
 Text(1674.0, 815.3999999999999, 'ApplicantIncome <= 4408.0\ngini = 0.571\n
samples = 13\nvalue = [11, 5, 3]'),
 Text(1395.0, 271.7999999999997, 'gini = 0.278\nsamples = 5\nvalue = [5, 0,
1]'),
 Text(1953.0, 271.7999999999997, 'gini = 0.615\nsamples = 8\nvalue = [6, 5,
2]'),
 Text(2790.0, 815.3999999999999, 'ApplicantIncome <= 3287.5\ngini = 0.364\n
samples = 12\nvalue = [3, 1, 14]'),
 Text(2511.0, 271.7999999999997, 'gini = 0.5\nsamples = 5\nvalue = [3, 0,
```

In [ ]:

In [ ]: