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\C8_loan-test.csv")
df
```

Out[2]:

Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Lc
Male	Yes	0	Graduate	No	5720	0	
Male	Yes	1	Graduate	No	3076	1500	
Male	Yes	2	Graduate	No	5000	1800	
Male	Yes	2	Graduate	No	2340	2546	
Male	No	0	Not Graduate	No	3276	0	
...	...	...	...	...	...	...	
Male	Yes	3+	Not Graduate	Yes	4009	1777	
Male	Yes	0	Graduate	No	4158	709	
Male	No	0	Graduate	No	3250	1993	
Male	Yes	0	Graduate	No	5000	2393	
Male	No	0	Graduate	Yes	9200	0	

columns

In [4]:
```
ge=df[['ApplicantIncome','Property_Area']]
```

In [5]:
```
d=ge.fillna(20)
```

In [6]:
```
ge['Property_Area'].value_counts()
```

Out[6]:
```
Urban        140
Semiurban    116
Rural        111
Name: Property_Area, dtype: int64
```

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

In [8]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [9]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[9]:
```
RandomForestClassifier()
```

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

In [11]:
```
from sklearn.model_selection import GridSearchCV
grid_search= GridSearchCV(estimator = rfc,param_grid=paramets,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[11]:
```
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 [12]:
```
grid_search.best_score_
```

Out[12]:
```
0.37109375
```

In [13]:
```
rfc_best=grid_search.best_estimator_
```

```
In [14]: from sklearn.tree import plot_tree
         plt.figure(figsize=(80,40))
         plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','N
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-14-d9a5e60a8034> in <module>
      1 from sklearn.tree import plot_tree
      2 plt.figure(figsize=(80,40))
----> 3 plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names
=['Yes','No'],filled=True)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inn
er_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_export.py in plot_tr
ee(decision_tree, max_depth, feature_names, class_names, label, filled, impur
ity, node_ids, proportion, rotate, rounded, precision, ax, fontsize)
    192         proportion=proportion, rotate=rotate, rounded=rounded,
    193         precision=precision, fontsize=fontsize)
--> 194     return exporter.export(decision_tree, ax=ax)
    195 
    196 

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_export.py in export
(self, decision_tree, ax)
    582         ax.clear()
    583         ax.set_axis_off()
--> 584         my_tree = self._make_tree(0, decision_tree.tree_,
    585                                   decision_tree.criterion)
    586         draw_tree = buchheim(my_tree)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_export.py in _make_t
ree(self, node_id, et, criterion, depth)
    563         # traverses _tree.Tree recursively, builds intermediate
    564         # "_reingold_tilford.Tree" object
--> 565         name = self.node_to_str(et, node_id, criterion=criterion)
    566         if (et.children_left[node_id] != _tree.TREE_LEAF
    567                 and (self.max_depth is None or depth <= self.max_dept
h)):

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_export.py in node_to
_str(self, tree, node_id, criterion)
    353                 node_string += 'class = '
    354             if self.class_names is not True:
--> 355                 class_name = self.class_names[np.argmax(value)]
    356             else:
    357                 class_name = "y%s%s%s" % (characters[1],

IndexError: list index out of range
```

```
In [ ]:
```