In [1]:
```
#Task 1:Use Decision Trees to prepare a model on fraud data treating those who ha
```

1.Nessasry Libraries

In [2]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
import numpy as np   # import all nessasary libraries
```

2.Import dataset

In [3]:
```
data=pd.read_csv('Fraud_check.csv')  #import datasets
data.head()  #Read Only Top 5 Rows.
```

Out[3]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	NO	Single	68833	50047	10	YES
1	YES	Divorced	33700	134075	18	YES
2	NO	Married	36925	160205	30	YES
3	YES	Single	50190	193264	15	YES
4	NO	Married	81002	27533	28	NO

3.Data Undestanding

In [4]:
```
data.describe(include="all")  #describe all Nessasry function.
```

Out[4]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
count	600	600	600.000000	600.000000	600.000000	600
unique	2	3	NaN	NaN	NaN	2
top	YES	Single	NaN	NaN	NaN	YES
freq	312	217	NaN	NaN	NaN	302
mean	NaN	NaN	55208.375000	108747.368333	15.558333	NaN
std	NaN	NaN	26204.827597	49850.075134	8.842147	NaN
min	NaN	NaN	10003.000000	25779.000000	0.000000	NaN
25%	NaN	NaN	32871.500000	66966.750000	8.000000	NaN
50%	NaN	NaN	55074.500000	106493.500000	15.000000	NaN
75%	NaN	NaN	78611.750000	150114.250000	24.000000	NaN
max	NaN	NaN	99619.000000	199778.000000	30.000000	NaN

In [5]:
```
data.info() #infomation of all columns(Memory_usage and null Rows)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Undergrad        600 non-null    object
 1   Marital.Status   600 non-null    object
 2   Taxable.Income   600 non-null    int64 
 3   City.Population  600 non-null    int64 
 4   Work.Experience  600 non-null    int64 
 5   Urban            600 non-null    object
dtypes: int64(3), object(3)
memory usage: 28.2+ KB
```

In [6]:
```
data.isnull().sum()# if there is any null values in data
```

Out[6]:
```
Undergrad          0
Marital.Status     0
Taxable.Income     0
City.Population    0
Work.Experience    0
Urban              0
dtype: int64
```

In [7]:
```
data.shape   #size of Row and Columns
```

Out[7]:
```
(600, 6)
```

In [8]:
```
data.dtypes  #data type of all columns
```

Out[8]:
```
Undergrad          object
Marital.Status     object
Taxable.Income      int64
City.Population     int64
Work.Experience     int64
Urban              object
dtype: object
```

4.Data Preparing

In [9]:
```
data['Taxable.Income']=pd.cut(data['Taxable.Income'],bins=[10000,30000,100000],la
```

In [10]:
```
le=LabelEncoder()
for i in data.columns:
    if data[i].dtypes=="catagory" or "object":
        data[i]=le.fit_transform(data[i])  #apply label encoder to all input cato
```

In [11]:
```
data.head()   #Only Top 5 Rows
```

Out[11]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	0	2	0	84	10	1
1	1	0	0	398	18	1
2	0	1	0	481	30	1
3	1	2	0	574	15	1
4	0	1	0	4	28	0

In [12]:
```
#Rename all columns.Because '.' in between columns create issue for reading data.
data.rename(columns={"Undergrad":"undergrad","Marital.Status":"Marital","Taxable.
```

In [13]:
```
x=data.drop("Income",axis=1)
y=data['Income']  #divide data in input and output data
```

In [14]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,stratify=y,random_state=12,shu
#divide data in train and test data
```

5. Model Building (Gini Criteria)

In [15]:
```
#Building Decision Tree Classifier using Gini Criteria
```

In [16]:
```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(criterion='gini',max_depth=10)
model.fit(x_train,y_train)     #Model Building
```

Out[16]:
```
DecisionTreeClassifier(max_depth=10)
```

In [17]:
```
#Predict On train data
y_train_pred=model.predict(x_train)
```

In [18]:
```
#Find out Accuracy
print(accuracy_score(y_train,y_train_pred))
```

```
0.8777777777777778
```

In [19]:
```
#Predict Data with test data
y_test_pred=model.predict(x_test)
```

In [20]:
```
# Find out Accuracy
print(accuracy_score(y_test,y_test_pred))
```

```
0.7333333333333333
```

In [21]:
```
acc = [['Train_accuracy', '87%'], ['Test_accuracy', '73%']]
acc=pd.DataFrame(acc)
acc
```

Out[21]:

	0	1
0	Train_accuracy	87%
1	Test_accuracy	73%

6.Model Building (entropy Criteria)

In [22]:
```
#Building Decision Tree Classifier using entropy Criteria
```

In [23]:
```
##Model building

model1=DecisionTreeClassifier(criterion='entropy',max_depth=10)
model1.fit(x_train,y_train)
```

Out[23]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=10)
```

In [24]:
```
#Predict On train data
y_train_pred2=model1.predict(x_train)
```

In [25]:
```
#Find out Accuracy
print(accuracy_score(y_train,y_train_pred))
```

```
0.8777777777777778
```

In [26]:
```
#Prediction on test data
y_test_pred1=model1.predict(x_test)
```

In [27]:
```
#find out accuracy
print(accuracy_score(y_test,y_test_pred1))
```

```
0.74
```

In [28]:
```
# confusion_matrix
confusion_matrix(y_train,y_train_pred)
```

Out[28]:
```
array([[355,   2],
       [ 53,  40]], dtype=int64)
```

In [29]:
```
acc = [['Train_accuracy', '87%'], ['Test_accuracy', '74%']]
acc=pd.DataFrame(acc)
acc
```

Out[29]:

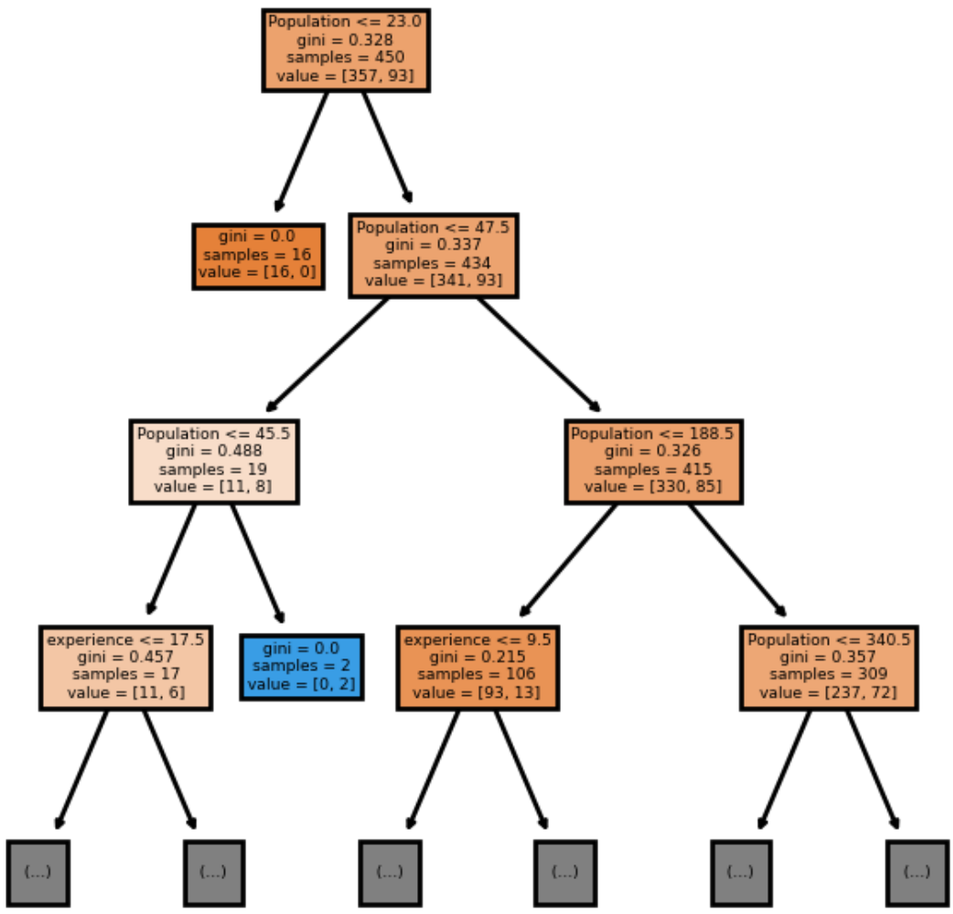
	0	1
0	Train_accuracy	87%
1	Test_accuracy	74%

7.Plot the tree

In [30]:
```
#Plot Tree
from sklearn import tree
```

In [31]:
```
fn=['undergrad', 'Marital', 'Population', 'experience', 'urban']   #columns we wo
cn=['1','0']   #Labels
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=200)   #Size Of

tree.plot_tree(model,max_depth=3,feature_names=fn,filled=True)
plt.show()
```



8. Model Building (RandomForest)

In [32]:
```
#Building Random Forest Classifier using entropy Criteria
```

In [33]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [34]:
```
rf=RandomForestClassifier(n_estimators=12,criterion='entropy',max_depth=10)    #mo
rf.fit(x_train,y_train)
```

Out[34]:
```
RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=12)
```

In [35]:
```
#Prediction on train data
y_pred=rf.predict(x_train)
```

In [36]:
```
#find out accuracy
accuracy_score(y_pred,y_train)
```

Out[36]:
```
0.94
```

In [37]:
```
#Predict on test Data
y_pred2=rf.predict(x_test)
```

In [38]:
```
#Accuracy
accuracy_score(y_pred2,y_test)
```

Out[38]:
```
0.76
```

In [39]:
```
rf.classes_   #No of Classes
rf.n_outputs_  #Prameters
```

Out[39]:
```
1
```

In [40]:
```
acc = [['Train_accuracy', '94%'], ['Test_accuracy', '76%']]
acc=pd.DataFrame(acc)
acc
```

Out[40]:

	0	1
0	Train_accuracy	94%
1	Test_accuracy	76%

9 Model Building(Random Forest)

In [41]:
```
##Building Random Forest Classifier using Gini Criteria.
```

In [42]:
```
rf1=RandomForestClassifier(n_estimators=15,criterion='gini',max_depth=10)
rf1.fit(x_train,y_train)   #fit the model
```

Out[42]:
```
RandomForestClassifier(max_depth=10, n_estimators=15)
```

In [43]:
```
#Predict on train Data
y_pred1=rf1.predict(x_train)
```

In [44]:
```
# Find OAccuracy
accuracy_score(y_pred1,y_train)
```

Out[44]:
```
0.9444444444444444
```

In [45]:
```
#Predict on test Data
y_pred2=rf1.predict(x_test)
```

In [46]:
```
#Accuracy
accuracy_score(y_pred2,y_test)
```

Out[46]:
```
0.7533333333333333
```

In [47]:
```
acc = [['Test_accuracy', '94%'], ['Train_accuracy', '75%']]
acc=pd.DataFrame(acc)
acc
```

Out[47]:

	0	1
0	Test_accuracy	94%
1	Train_accuracy	75%