In [3]:

```python
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import AdaBoostClassifier

import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```python
fraud_data = pd.read_csv('Fraud_check.csv')
fraud_data
```

Out[4]:

	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
...	...	...	...	...	...	...
595	YES	Divorced	76340	39492	7	YES
596	YES	Divorced	69967	55369	2	YES
597	NO	Divorced	47334	154058	0	YES
598	YES	Married	98592	180083	17	NO
599	NO	Divorced	96519	158137	16	NO

600 rows × 6 columns

In [5]:
```
fraud_data.describe()
```

Out[5]:

	Taxable.Income	City.Population	Work.Experience
count	600.000000	600.000000	600.000000
mean	55208.375000	108747.368333	15.558333
std	26204.827597	49850.075134	8.842147
min	10003.000000	25779.000000	0.000000
25%	32871.500000	66966.750000	8.000000
50%	55074.500000	106493.500000	15.000000
75%	78611.750000	150114.250000	24.000000
max	99619.000000	199778.000000	30.000000

In [6]:
```
fraud_data.shape
```

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

In [7]:
```
fraud_data.isna().sum()
```

Out[7]:
```
Undergrad          0
Marital.Status     0
Taxable.Income     0
City.Population    0
Work.Experience    0
Urban              0
dtype: int64
```

In [8]:
```
fraud_data.dtypes
```

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

In [9]:
```
fraud_data.loc[fraud_data['Taxable.Income']<= 30000,'Taxable_income'] = 'Risky'
fraud_data.loc[fraud_data['Taxable.Income']>30000,'Taxable_income'] = 'Good'
```

In [10]:
```
fraud_data.head()
```

Out[10]:

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

In [11]:
```
fraud_data.drop('Taxable.Income',axis=1,inplace=True)
```

In [12]:
```
fraud_data['Undergrad'].value_counts()
```

Out[12]:
```
YES    312
NO     288
Name: Undergrad, dtype: int64
```

In [13]:
```
le = LabelEncoder()
fraud_data['Undergrad'] = le.fit_transform(fraud_data['Undergrad'])
fraud_data['Undergrad'].unique()
```

Out[13]:
```
array([0, 1])
```

In [14]:
```
fraud_data['Marital.Status'].value_counts()
```

Out[14]:
```
Single      217
Married     194
Divorced    189
Name: Marital.Status, dtype: int64
```

In [15]:
```
fraud_data['Marital.Status'] = le.fit_transform(fraud_data['Marital.Status'])
fraud_data['Marital.Status'].unique()
```

Out[15]:
```
array([2, 0, 1])
```

In [16]:
```
fraud_data['Urban'].value_counts()
```

Out[16]:
```
YES    302
NO     298
Name: Urban, dtype: int64
```

In [17]:
```
fraud_data['Urban'] = le.fit_transform(fraud_data['Urban'])
fraud_data['Urban'].unique()
```

Out[17]:
```
array([1, 0])
```

In [18]:
```
fraud_data.head()
```

Out[18]:

	Undergrad	Marital.Status	City.Population	Work.Experience	Urban	Taxable_income
0	0	2	50047	10	1	Good
1	1	0	134075	18	1	Good
2	0	1	160205	30	1	Good
3	1	2	193264	15	1	Good
4	0	1	27533	28	0	Good

In [19]:
```
x = fraud_data.drop('Taxable_income',axis = 1)
y = fraud_data[['Taxable_income']]
```

In [20]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=
```

In [21]:
```
x_train.shape,y_train.shape
```

Out[21]:
```
((480, 5), (480, 1))
```

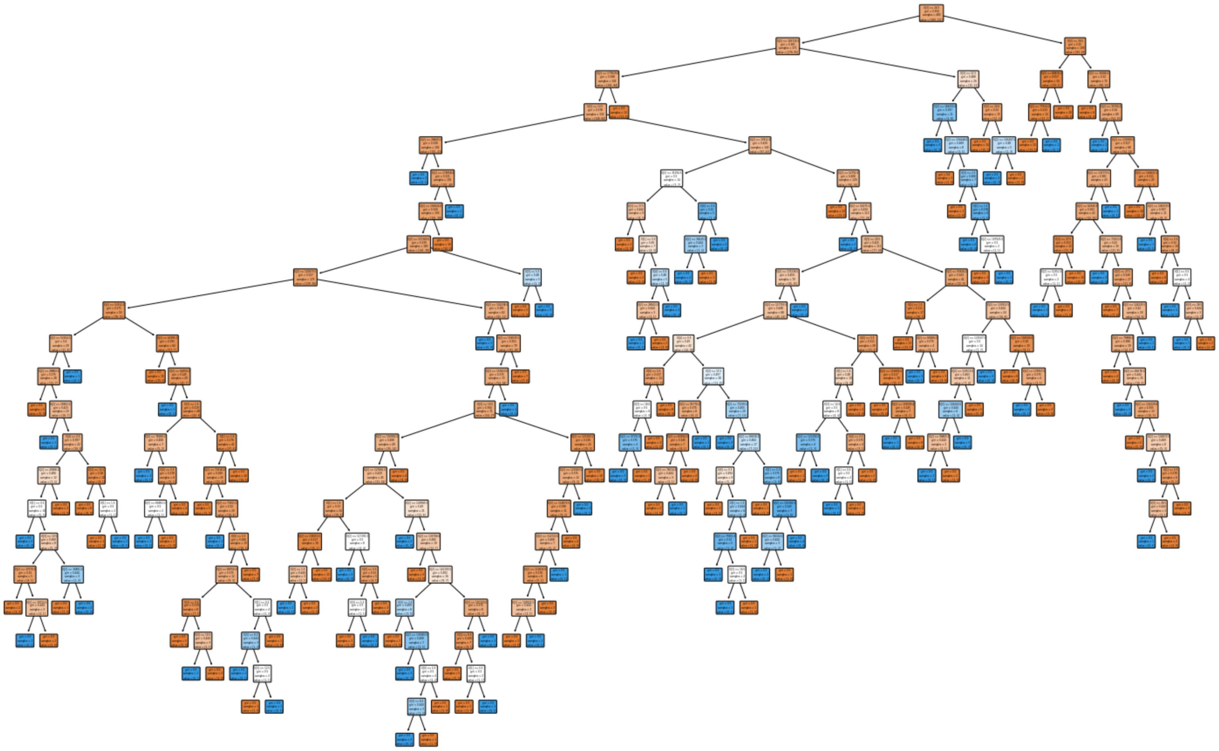
In [22]:
```
x_test.shape,y_test.shape
```

Out[22]:
```
((120, 5), (120, 1))
```

In [23]:
```
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train,y_train)
```

Out[23]:
```
DecisionTreeClassifier()
```

In [24]:
```
plt.figure(figsize=(25,16))
plot_tree(dt_model,filled = True , rounded = True)
plt.show()
```



In [25]:
```
y_pred_train = dt_model.predict(x_train)
```

In [26]:
```
print('Accuracy_score :',accuracy_score(y_train,y_pred_train))
```

```
Accuracy_score : 1.0
```

In [27]:
```
print('Confusion _Matrix :\n',confusion_matrix(y_train,y_pred_train))
```

```
Confusion _Matrix :
 [[369   0]
 [  0 111]]
```

In [28]:
```
y_pred_test = dt_model.predict(x_test)
```

In [29]:
```
print('Accuracy_score :',accuracy_score(y_test,y_pred_test))
```

```
Accuracy_score : 0.65
```

In [30]:
```
print('Confusion _Matrix :\n',confusion_matrix(y_test,y_pred_test))
```

```
Confusion _Matrix :
 [[76 31]
 [11  2]]
```

```
In [31]: abc = AdaBoostClassifier()
         abc.fit(x_train,y_train)

Out[31]: AdaBoostClassifier()
```

```
In [35]: y_pred_tr = abc.predict(x_train)
```

```
In [36]: print('Accuracy_score :',accuracy_score(y_train,y_pred_tr))

         Accuracy_score : 0.7729166666666667
```

```
In [37]: print('Confusion _Matrix :\n',confusion_matrix(y_train,y_pred_tr))

         Confusion _Matrix :
          [[369   0]
          [109   2]]
```

```
In [40]: y_pred_ts = abc.predict(x_test)
```

```
In [41]: print('Accuracy_score :',accuracy_score(y_test,y_pred_ts))

         Accuracy_score : 0.8833333333333333
```

```
In [42]: print('Confusion _Matrix :\n',confusion_matrix(y_test,y_pred_ts))

         Confusion _Matrix :
          [[106   1]
          [ 13   0]]
```

```
In [ ]:
```