Importing Required Libraries

In [49]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import warnings# used for removing warnings
warnings.filterwarnings('ignore')
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

Impoting the data

In [3]:
```
affair = pd.read_csv("affairs.csv")
affair
```

Out[3]:

	Unnamed: 0	naffairs	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtre
0	1	0	0	0	0	0	1	0	0	0	
1	2	0	0	0	0	0	1	0	0	0	
2	3	3	0	0	0	0	1	0	0	0	
3	4	0	1	0	0	0	1	0	1	0	
4	5	3	1	0	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	
596	597	0	0	0	0	0	1	0	0	0	
597	598	1	1	0	0	0	0	1	0	0	
598	599	0	1	0	1	0	0	0	0	1	
599	600	0	1	0	0	0	1	0	0	0	
600	601	0	1	0	0	0	1	0	0	1	

601 rows × 19 columns

Initial Analysis

In [4]:
```
affair.head()
```

Out[4]:

	Unnamed: 0	naffairs	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel
0	1	0	0	0	0	0	1	0	0	0	1
1	2	0	0	0	0	0	1	0	0	0	0

	Unnamed: 0	naffairs	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel
2	3	3	0	0	0	0	1	0	0	0	1
3	4	0	1	0	0	0	1	0	1	0	0
4	5	3	1	0	0	0	0	1	0	0	1

In [5]:
```
affair.shape
```

Out[5]:
```
(601, 19)
```

In [6]:
```
affair.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 601 entries, 0 to 600
Data columns (total 19 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  601 non-null    int64
 1   naffairs    601 non-null    int64
 2   kids        601 non-null    int64
 3   vryunhap    601 non-null    int64
 4   unhap       601 non-null    int64
 5   avgmarr     601 non-null    int64
 6   hapavg      601 non-null    int64
 7   vryhap      601 non-null    int64
 8   antirel     601 non-null    int64
 9   notrel      601 non-null    int64
 10  slghtrel    601 non-null    int64
 11  smerel      601 non-null    int64
 12  vryrel      601 non-null    int64
 13  yrsmarr1    601 non-null    int64
 14  yrsmarr2    601 non-null    int64
 15  yrsmarr3    601 non-null    int64
 16  yrsmarr4    601 non-null    int64
 17  yrsmarr5    601 non-null    int64
 18  yrsmarr6    601 non-null    int64
dtypes: int64(19)
memory usage: 89.3 KB
```

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

Out[7]:
```
Unnamed: 0    0
naffairs      0
kids          0
vryunhap      0
unhap         0
avgmarr       0
hapavg        0
vryhap        0
antirel       0
notrel        0
slghtrel      0
smerel        0
vryrel        0
yrsmarr1      0
yrsmarr2      0
yrsmarr3      0
yrsmarr4      0
yrsmarr5      0
```

```
yrsmarr6      0
dtype: int64
```

In [8]:
```
affair.describe()
```

Out[8]:

	Unnamed: 0	naffairs	kids	vryunhap	unhap	avgmarr	hapavg	vryha
count	601.000000	601.000000	601.000000	601.000000	601.000000	601.000000	601.000000	601.0000
mean	301.000000	1.455907	0.715474	0.026622	0.109817	0.154742	0.322795	0.3860
std	173.638033	3.298758	0.451564	0.161111	0.312922	0.361960	0.467935	0.4872
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	151.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
50%	301.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.0000
75%	451.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.0000
max	601.000000	12.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

In [10]:
```
affair.dtypes
```

Out[10]:
```
Unnamed: 0    int64
naffairs      int64
kids          int64
vryunhap      int64
unhap         int64
avgmarr       int64
hapavg        int64
vryhap        int64
antirel       int64
notrel        int64
slghtrel      int64
smerel        int64
vryrel        int64
yrsmarr1      int64
yrsmarr2      int64
yrsmarr3      int64
yrsmarr4      int64
yrsmarr5      int64
yrsmarr6      int64
dtype: object
```

In [11]:
```
affair.dropna(axis = 0,inplace = True)
```

In [12]:
```
affair.head(10)
```

Out[12]:

	Unnamed: 0	naffairs	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel
0	1	0	0	0	0	0	1	0	0	0	1
1	2	0	0	0	0	0	1	0	0	0	0
2	3	3	0	0	0	0	1	0	0	0	1
3	4	0	1	0	0	0	1	0	1	0	0

	Unnamed: 0	naffairs	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel
4	5	3	1	0	0	0	0	1	0	0	1
5	6	0	1	0	0	0	0	1	0	0	0
6	7	0	0	0	0	1	0	0	0	1	0
7	8	0	0	0	0	0	0	1	0	1	0
8	9	7	1	0	1	0	0	0	0	0	0
9	10	0	0	0	0	1	0	0	0	1	0

In [13]:
```
affair.shape
```

Out[13]: (601, 19)

In [14]:
```
del affair['Unnamed: 0']
```

In [15]:
```
affair.head()
```

Out[15]:

	naffairs	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel	smerel	vry
0	0	0	0	0	0	1	0	0	0	1	0	
1	0	0	0	0	0	1	0	0	0	0	1	
2	3	0	0	0	0	1	0	0	0	1	0	
3	0	1	0	0	0	1	0	1	0	0	0	
4	3	1	0	0	0	0	1	0	0	1	0	

Model Building

Seperate input features

In [16]:
```
x = affair.drop(labels = 'naffairs' , axis = 1)
y = affair[['naffairs']]
```

In [19]:
```
x
```

Out[19]:

	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel	smerel	vryrel	yr
0	0	0	0	0	1	0	0	0	1	0	0	
1	0	0	0	0	1	0	0	0	0	1	0	
2	0	0	0	0	1	0	0	0	1	0	0	
3	1	0	0	0	1	0	1	0	0	0	0	

	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel	smerel	vryrel	yr
4	1	0	0	0	0	1	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...
596	0	0	0	0	1	0	0	0	0	1	0	
597	1	0	0	0	0	1	0	0	1	0	0	
598	1	0	1	0	0	0	0	1	0	0	0	
599	1	0	0	0	1	0	0	0	1	0	0	
600	1	0	0	0	1	0	0	1	0	0	0	

601 rows × 17 columns

In [20]:
```
y
```

Out[20]:

	naffairs
0	0
1	0
2	3
3	0
4	3
...	...
596	0
597	1
598	0
599	0
600	0

601 rows × 1 columns

In [22]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=12,test_size=0.2)
```

In [23]:
```
x_train
```

Out[23]:

	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel	smerel	vryrel	yr
293	1	0	0	1	0	0	0	0	1	0	0	
377	1	0	0	0	1	0	0	0	0	1	0	
541	0	0	0	0	0	1	1	0	0	0	0	
543	1	0	0	0	1	0	0	0	0	1	0	
5	1	0	0	0	0	1	0	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...

	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel	smerel	vryrel	yr
432	1	0	0	1	0	0	0	0	0	1	0	
259	1	0	0	1	0	0	1	0	0	0	0	
241	0	0	0	0	1	0	0	0	1	0	0	
253	1	0	0	0	0	1	0	1	0	0	0	
390	1	0	0	0	1	0	0	0	1	0	0	

480 rows × 17 columns

In [24]:
```
x_test
```

Out[24]:

	kids	vryunhap	unhap	avgmarr	hapavg	vryhap	antirel	notrel	slghtrel	smerel	vryrel	yr
14	1	0	1	0	0	0	0	0	1	0	0	
262	1	0	0	1	0	0	0	1	0	0	0	
453	1	0	0	1	0	0	0	0	1	0	0	
326	1	0	0	1	0	0	1	0	0	0	0	
210	1	0	0	1	0	0	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...
558	1	0	0	0	0	1	0	0	1	0	0	
307	1	0	0	0	1	0	0	0	0	1	0	
368	1	0	1	0	0	0	0	1	0	0	0	
527	1	0	0	0	0	1	0	0	0	1	0	
553	1	0	0	1	0	0	0	1	0	0	0	

121 rows × 17 columns

In [26]:
```
x_train.shape,y_train.shape
```

Out[26]:
```
((480, 17), (480, 1))
```

In [27]:
```
x_test.shape , y_test.shape
```

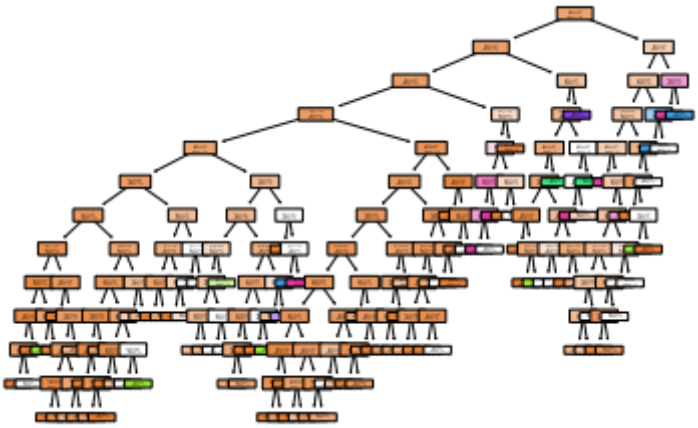
Out[27]:
```
((121, 17), (121, 1))
```

Model Training

In [34]:
```
logic_model  = LogisticRegression()
logic_model.fit(x_train,y_train)

dt_model = DecisionTreeClassifier()
dt_model.fit(x_train,y_train)
```

Out[34]:
```
DecisionTreeClassifier()
```

In [35]:
```
## preparing a plot figure

plot_tree(dt_model,rounded = True,filled  = True)
plt.figure(figsize=(16,10))
plt.show()
```



```
<Figure size 1152x720 with 0 Axes>
```

Model Testing

training data

In [37]:
```
# y_pred_train = logic_model.predict(x_train)
# y_pred_train
```

In [38]:
```
y_pred_train = dt_model.predict(x_train)
y_pred_train
```

Out[38]:
```
array([ 1,  0, 12,  0,  0,  0,  0,  0,  0,  0, 12,  0,  0,  0,  0,  0,  7,
        0,  0,  0,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  7,  0,  0,  0,  0,  0,  7,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0, 12,  0,  0,  0,  0,  0,  0,  0,  0, 12,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  7,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0, 12,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        3, 12,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0,  0,  0,
        0,  0,  0,  0,  0, 12,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  3,  0,  0,  7,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  1, 12,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0, 12,  3,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  7,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
```

```
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 12,  1,  0,
       7,  0,  0,  0], dtype=int64)
```

In [40]:
```
accuracy_score(y_train,y_pred_train)
```

Out[40]:
```
0.7979166666666667
```

In [41]:
```
confusion_matrix(y_train,y_pred_train)
```

Out[41]:
```
array([[355,   0,   0,   0,   1,   1],
       [ 22,   7,   0,   0,   0,   0],
       [ 11,   1,   3,   0,   0,   0],
       [ 13,   0,   0,   3,   0,   0],
       [ 26,   0,   0,   0,   6,   0],
       [ 22,   0,   0,   0,   0,   9]], dtype=int64)
```

In [42]:
```
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           0       0.79      0.99      0.88       357
           1       0.88      0.24      0.38        29
           2       1.00      0.20      0.33        15
           3       1.00      0.19      0.32        16
           7       0.86      0.19      0.31        32
          12       0.90      0.29      0.44        31

    accuracy                           0.80       480
   macro avg       0.90      0.35      0.44       480
weighted avg       0.82      0.80      0.75       480
```

In [44]:
```
# y_pred_test = logic_model.predict(x_test)
# y_pred_test


y_pred_test = dt_model.predict(x_test)
y_pred_test
```

Out[44]:
```
array([12,  0,  1,  7,  1,  0,  0, 12,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  1,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  2,  0,  0,  0,
       0,  0,  0,  7,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0, 12,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  3,
       0,  1], dtype=int64)
```

In [45]:
```
accuracy_score(y_test,y_pred_test)
```

Out[45]:
```
0.71900826446281
```

In [46]:
```
confusion_matrix(y_test,y_pred_test)
```

Out[46]:
```
array([[85,  6,  1,  0,  1,  1],
       [ 5,  0,  0,  0,  0,  0],
       [ 2,  0,  0,  0,  0,  0],
       [ 3,  0,  0,  0,  0,  0],
```

```
       [ 9,  0,  0,  1,  0,  0],
       [ 4,  0,  0,  0,  1,  2]], dtype=int64)
```

In [48]:
```
print(classification_report(y_test,y_pred_test))
```

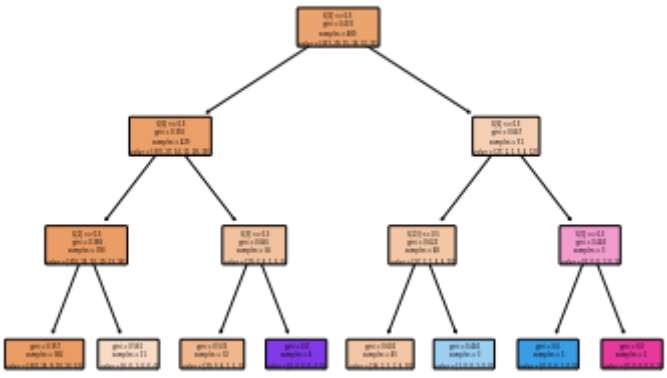
```
              precision    recall  f1-score   support

           0       0.79      0.90      0.84        94
           1       0.00      0.00      0.00         5
           2       0.00      0.00      0.00         2
           3       0.00      0.00      0.00         3
           7       0.00      0.00      0.00        10
          12       0.67      0.29      0.40         7

    accuracy                           0.72       121
   macro avg       0.24      0.20      0.21       121
weighted avg       0.65      0.72      0.68       121
```

Model Improvement with decision Tree

In [51]:
```
dt_model = DecisionTreeClassifier( criterion='gini',max_depth=3)
dt_model.fit(x_train,y_train)
```

Out[51]:
```
DecisionTreeClassifier(max_depth=3)
```

In [62]:
```
plot_tree(dt_model,filled = True,rounded = True)
plt.figure(figsize = (20,12))
plt.show()
```



```
<Figure size 1440x864 with 0 Axes>
```

In [ ]:
```

```