In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [3]:

```
traindf=pd.read_csv(r"C:\Users\shaik\Downloads\Mobile_Price_Classification_train.csv")
traindf
```

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	..
0	842	0	2.2	0	1	0	7	0.6	188	2	..
1	1021	1	0.5	1	0	1	53	0.7	136	3	..
2	563	1	0.5	1	2	1	41	0.9	145	5	..
3	615	1	2.5	0	0	0	10	0.8	131	6	..
4	1821	1	1.2	0	13	1	44	0.6	141	2	..
...	...	...	...	...	...	...	...	...	...	...	..
1995	794	1	0.5	1	0	1	2	0.8	106	6	..
1996	1965	1	2.6	1	0	0	39	0.2	187	4	..
1997	1911	0	0.9	1	1	1	36	0.7	108	8	..
1998	1512	0	0.9	0	4	1	46	0.1	145	5	..
1999	510	1	2.0	1	5	1	45	0.9	168	6	..

2000 rows × 21 columns

In [4]:

```
testdf=pd.read_csv(r"C:\Users\shaik\Downloads\Mobile_Price_Classification_test.csv")
testdf
```

Out[4]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	p
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	1
1	2	841	1	0.5	1	4	1	61	0.8	191	...	1
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	2
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	1
...	...	...	...	...	...	...	...	...	...	...	...	..
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	1
996	997	609	0	1.8	1	0	0	13	0.9	186	...	
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	1
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	1
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	1

1000 rows × 21 columns

In [5]:

```
traindf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

In [6]:

```
testdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             1000 non-null   int64  
 1   battery_power  1000 non-null   int64  
 2   blue           1000 non-null   int64  
 3   clock_speed    1000 non-null   float64
 4   dual_sim       1000 non-null   int64  
 5   fc             1000 non-null   int64  
 6   four_g         1000 non-null   int64  
 7   int_memory     1000 non-null   int64  
 8   m_dep          1000 non-null   float64
 9   mobile_wt      1000 non-null   int64  
 10  n_cores        1000 non-null   int64  
 11  pc             1000 non-null   int64  
 12  px_height      1000 non-null   int64  
 13  px_width       1000 non-null   int64  
 14  ram            1000 non-null   int64  
 15  sc_h           1000 non-null   int64  
 16  sc_w           1000 non-null   int64  
 17  talk_time      1000 non-null   int64  
 18  three_g        1000 non-null   int64  
 19  touch_screen   1000 non-null   int64  
 20  wifi           1000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 164.2 KB
```

In [7]:

```
traindf.shape,testdf.shape
```

Out[7]:

```
((2000, 21), (1000, 21))
```

In [8]:

```
traindf=traindf.head(1000)
traindf
```

Out[8]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...
0	842	0	2.2	0	1	0	7	0.6	188	2	...
1	1021	1	0.5	1	0	1	53	0.7	136	3	...
2	563	1	0.5	1	2	1	41	0.9	145	5	...
3	615	1	2.5	0	0	0	10	0.8	131	6	...
4	1821	1	1.2	0	13	1	44	0.6	141	2	...
...	...	...	...	...	...	...	...	...	...	...	...
995	1456	0	1.6	1	5	0	49	0.2	193	3	...
996	774	0	0.5	1	2	1	10	0.5	188	2	...
997	1068	0	0.5	1	0	1	19	0.9	197	8	...
998	1373	1	1.9	1	1	1	29	0.9	141	6	...
999	1777	1	3.0	0	3	0	20	0.6	188	6	...

1000 rows × 21 columns

In [9]:

```
traindf.shape,testdf.shape
```

Out[9]:

```
((1000, 21), (1000, 21))
```

In [10]:

```
X=testdf
y=traindf['price_range']
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7,random_state=42)
```

In [11]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
```

Out[11]:

```
RandomForestClassifier()
```

In [12]:

```
rf=RandomForestClassifier()
params={'max_depth':[2,3,5,10,20],
        'min_samples_leaf':[5,10,20,50,100,200],
        'n_estimators':[10,25,30,50,100,200]}
```

In [13]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
```

In [14]:

```
grid_search.fit(X_train,y_train)
```

Out[14]:

```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

In [15]:

```
grid_search.best_score_
```

Out[15]:

```
0.2914285714285714
```

In [16]:

```
rf_best=grid_search.best_estimator_
rf_best
```

Out[16]:

```
RandomForestClassifier(max_depth=20, min_samples_leaf=20, n_estimators=25)
```

In [17]:

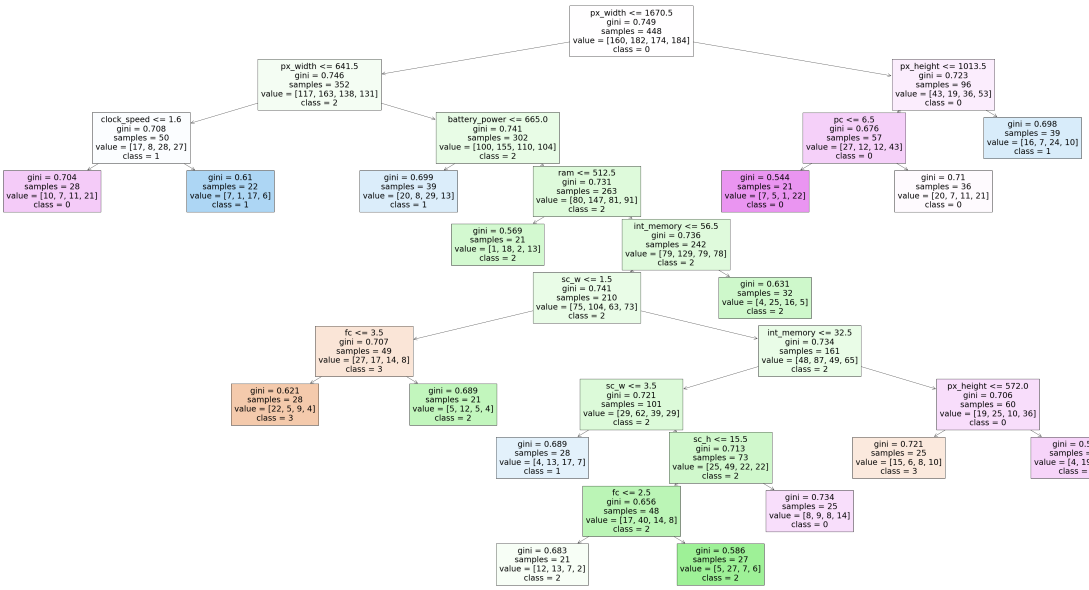
```
traindf['price_range'].value_counts()
```

Out[17]:

```
3    276
2    248
0    242
1    234
Name: price_range, dtype: int64
```

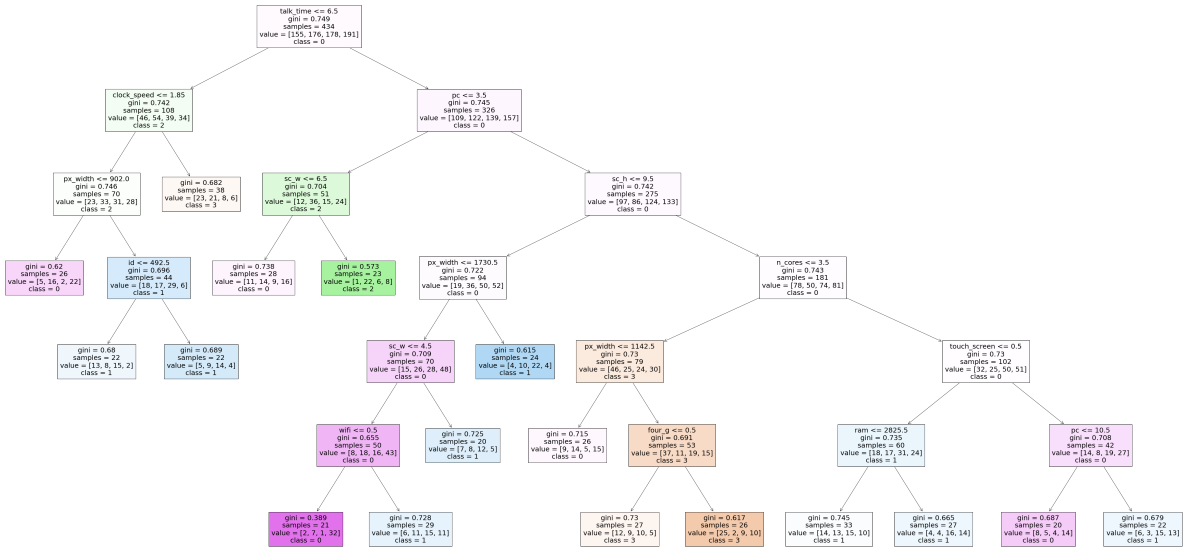
In [18]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],feature_names=X.columns,class_names=['3','2','1','0'],filled=True)
```



In [19]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=X.columns,class_names=['3','2','1','0'],filled=True)
```



In [20]:

```
rf_best.feature_importances_
```

Out[20]:

```
array([0.06683607, 0.06320588, 0.00550118, 0.07435044, 0.        ,
       0.04132326, 0.01354109, 0.0692836 , 0.04022201, 0.05693953,
       0.06601389, 0.06681763, 0.10930151, 0.08267723, 0.07008001,
       0.04050325, 0.05253657, 0.03777228, 0.0053751 , 0.01481684,
       0.02290263])
```

In [21]:

```
imp_df=pd.DataFrame({"Varname":X_train.columns,"Imp":rf_best.feature_importances_})
```

In [22]:

```
imp_df.sort_values(by="Imp",ascending=False)
```

Out[22]:

	Varname	Imp
12	px_height	0.109302
13	px_width	0.082677
3	clock_speed	0.074350
14	ram	0.070080
7	int_memory	0.069284
0	id	0.066836
11	pc	0.066818
10	n_cores	0.066014
1	battery_power	0.063206
9	mobile_wt	0.056940
16	sc_w	0.052537
5	fc	0.041323
15	sc_h	0.040503
8	m_dep	0.040222
17	talk_time	0.037772
20	wifi	0.022903
19	touch_screen	0.014817
6	four_g	0.013541
2	blue	0.005501
18	three_g	0.005375
4	dual_sim	0.000000