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 [2]:

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

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208
...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483

2000 rows × 21 columns

In [3]:

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

Out[3]:

ɔower	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h
1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12
841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6
1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17
1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10
1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1700	1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121	14
609	0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933	8
1185	0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223	5
1533	1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509	15
1270	1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828	9

mns

In [4]:

```
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  
```

In [5]:

```
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 [6]:

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

Out[6]:

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

In [7]:

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

Out[7]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	p
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	
...	...	...	...	...	...	...	...	...	...	...	...	...	
995	1456	0	1.6	1	5	0	49	0.2	193	3	...	1285	
996	774	0	0.5	1	2	1	10	0.5	188	2	...	1480	
997	1068	0	0.5	1	0	1	19	0.9	197	8	...	322	
998	1373	1	1.9	1	1	1	29	0.9	141	6	...	1220	
999	1777	1	3.0	0	3	0	20	0.6	188	6	...	511	

1000 rows × 21 columns

In [8]:

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

Out[8]:

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

In [9]:

```
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 [10]:

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

Out[10]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [11]:

```
rf=RandomForestClassifier()
```

In [12]:

```
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
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [15]:

```
grid_search.best_score_
```

Out[15]:

```
0.28714285714285714
```

In [16]:

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

Out[16]:

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=20, min_samples_leaf=50, n_estimators=10)
```

In [17]:

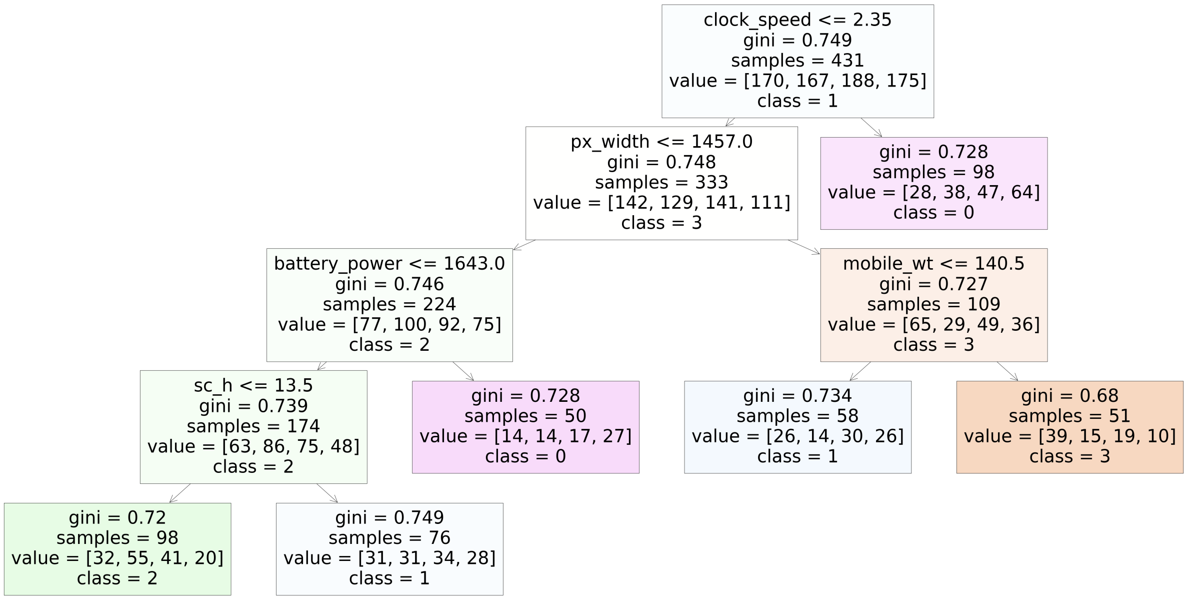
```
traindf['price_range'].value_counts()
```

Out[17]:

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

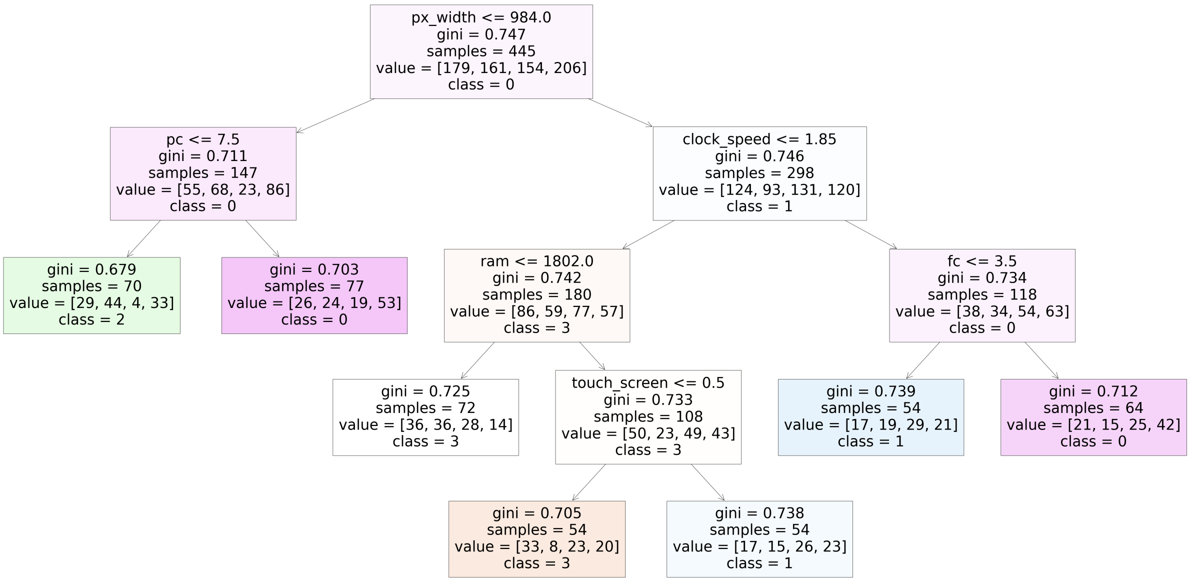
In [18]:

```
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([0.0420621 , 0.04145842, 0.        , 0.13109558, 0.        ,
       0.05955225, 0.03281126, 0.04569771, 0.        , 0.05751693,
       0.07205376, 0.05736547, 0.20151945, 0.12142944, 0.06031637,
       0.01960383, 0.01099444, 0.        , 0.        , 0.03328097,
       0.01324201])
```

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	0.201519
3	clock_speed	0.131096
13	px_width	0.121429
10	n_cores	0.072054
14	ram	0.060316
5	fc	0.059552
9	mobile_wt	0.057517
11	pc	0.057365
7	int_memory	0.045698
0	id	0.042062
1	battery_power	0.041458
19	touch_screen	0.033281
6	four_g	0.032811
15	sc_h	0.019604
20	wifi	0.013242
16	sc_w	0.010994
8	m_dep	0.000000
4	dual_sim	0.000000
2	blue	0.000000
17	talk_time	0.000000
18	three_g	0.000000

In [ ]:

```

```