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	px_
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]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	644	
996	997	609	0	1.8	1	0	0	13	0.9	186	...	2	1152	
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	477	
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	38	
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	457	

1000 rows × 21 columns

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  
 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: 328.3 KB
```

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
```

Out[6]: (2000, 21)

In [7]:
```
testdf.shape
```

Out[7]:
```
(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	...	px_height	px_v
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 [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()
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 [12]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
```

In [13]:
```
grid_search.fit(X_train,y_train)
```

Out[13]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [14]:
```
grid_search.best_score_
```

Out[14]:
```
0.2928571428571428
```

In [15]:
```
rf_best=grid_search.best_estimator_
rf_best
```

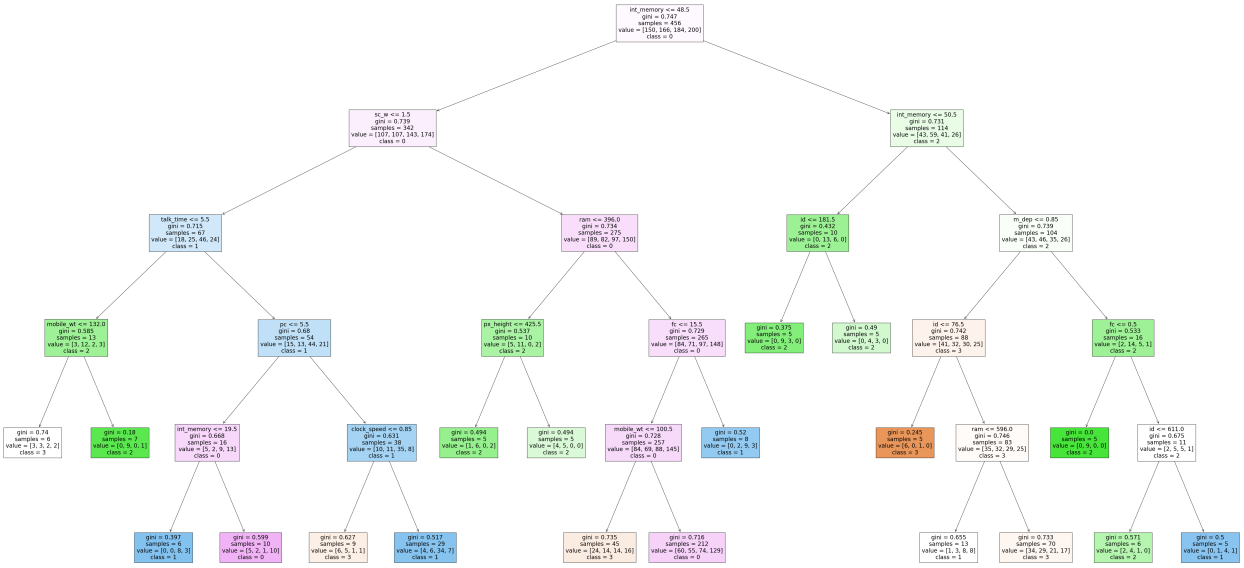
Out[15]:
```
RandomForestClassifier(max_depth=5, min_samples_leaf=5, n_estimators=30)
```

In [17]:
```
traindf['price_range'].value_counts()
```

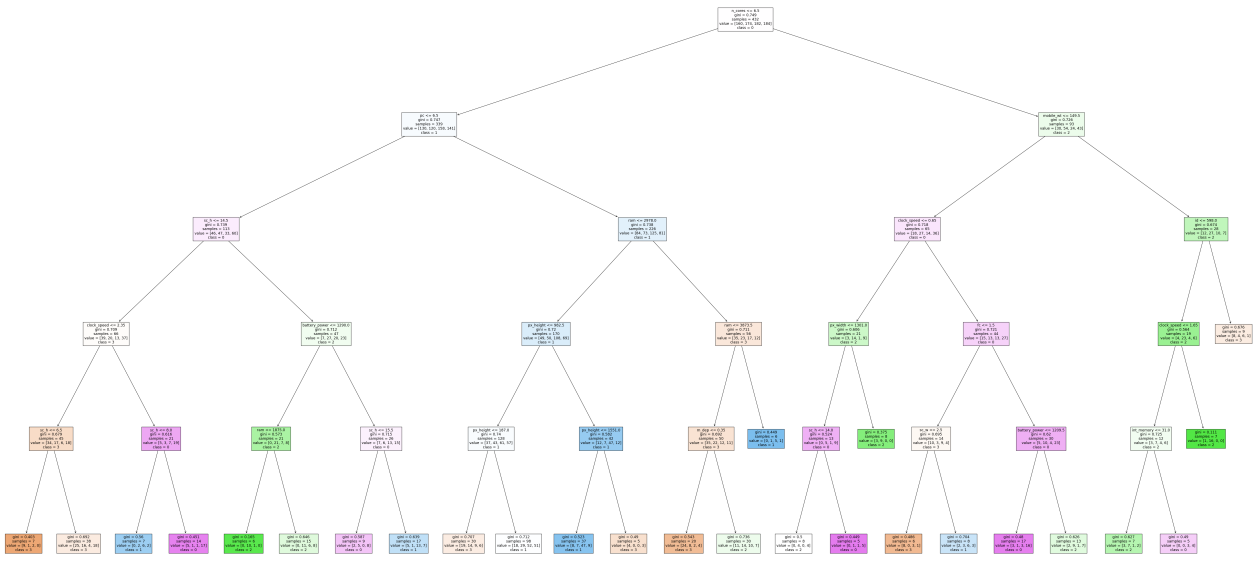
Out[17]:
```
price_range
3    276
2    248
0    242
1    234
Name: count, dtype: int64
```

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)
```

Out[18]:
```
[Text(0.5301724137931034, 0.9166666666666666, 'int_memory <= 48.5\ngini = 0.747\nsamples = 456\nva
lue = [150, 166, 184, 200]\nclass = 0'),
 Text(0.31896551724137934, 0.75, 'sc_w <= 1.5\ngini = 0.739\nsamples = 342\nvalue = [107, 107, 14
3, 174]\nclass = 0'),
 Text(0.15517241379310345, 0.5833333333333334, 'talk_time <= 5.5\ngini = 0.715\nsamples = 67\nvalu
e = [18, 25, 46, 24]\nclass = 1'),
 Text(0.06896551724137931, 0.4166666666666667, 'mobile_wt <= 132.0\ngini = 0.585\nsamples = 13\nva
lue = [3, 12, 2, 3]\nclass = 2'),
 Text(0.034482758620689655, 0.25, 'gini = 0.74\nsamples = 6\nvalue = [3, 3, 2, 2]\nclass = 3'),
 Text(0.10344827586206896, 0.25, 'gini = 0.18\nsamples = 7\nvalue = [0, 9, 0, 1]\nclass = 2'),
 Text(0.2413793103448276, 0.4166666666666667, 'pc <= 5.5\ngini = 0.68\nsamples = 54\nvalue = [15,
13, 44, 21]\nclass = 1'),
 Text(0.1724137931034483, 0.25, 'int_memory <= 19.5\ngini = 0.668\nsamples = 16\nvalue = [5, 2, 9,
13]\nclass = 0'),
 Text(0.13793103448275862, 0.08333333333333333, 'gini = 0.397\nsamples = 6\nvalue = [0, 0, 8, 3]\n
class = 1'),
 Text(0.20689655172413793, 0.08333333333333333, 'gini = 0.599\nsamples = 10\nvalue = [5, 2, 1, 10]
\nclass = 0'),
 Text(0.3103448275862069, 0.25, 'clock_speed <= 0.85\ngini = 0.631\nsamples = 38\nvalue = [10, 11,
35, 8]\nclass = 1'),
 Text(0.27586206896551724, 0.08333333333333333, 'gini = 0.627\nsamples = 9\nvalue = [6, 5, 1, 1]\n
class = 3'),
 Text(0.3448275862068966, 0.08333333333333333, 'gini = 0.517\nsamples = 29\nvalue = [4, 6, 34, 7]
\nclass = 1'),
 Text(0.4827586206896552, 0.5833333333333334, 'ram <= 396.0\ngini = 0.734\nsamples = 275\nvalue =
[89, 82, 97, 150]\nclass = 0'),
 Text(0.41379310344827586, 0.4166666666666667, 'px_height <= 425.5\ngini = 0.537\nsamples = 10\nva
lue = [5, 11, 0, 2]\nclass = 2'),
 Text(0.3793103448275862, 0.25, 'gini = 0.494\nsamples = 5\nvalue = [1, 6, 0, 2]\nclass = 2'),
 Text(0.4482758620689655, 0.25, 'gini = 0.494\nsamples = 5\nvalue = [4, 5, 0, 0]\nclass = 2'),
 Text(0.5517241379310345, 0.4166666666666667, 'fc <= 15.5\ngini = 0.729\nsamples = 265\nvalue = [8
4, 71, 97, 148]\nclass = 0'),
 Text(0.5172413793103449, 0.25, 'mobile_wt <= 100.5\ngini = 0.728\nsamples = 257\nvalue = [84, 69,
88, 145]\nclass = 0'),
 Text(0.4827586206896552, 0.08333333333333333, 'gini = 0.735\nsamples = 45\nvalue = [24, 14, 14, 1
6]\nclass = 3'),
 Text(0.5517241379310345, 0.08333333333333333, 'gini = 0.716\nsamples = 212\nvalue = [60, 55, 74,
129]\nclass = 0'),
 Text(0.5862068965517241, 0.25, 'gini = 0.52\nsamples = 8\nvalue = [0, 2, 9, 3]\nclass = 1'),
 Text(0.7413793103448276, 0.75, 'int_memory <= 50.5\ngini = 0.731\nsamples = 114\nvalue = [43, 59,
41, 26]\nclass = 2'),
 Text(0.6551724137931034, 0.5833333333333334, 'id <= 181.5\ngini = 0.432\nsamples = 10\nvalue =
[0, 13, 6, 0]\nclass = 2'),
 Text(0.6206896551724138, 0.4166666666666667, 'gini = 0.375\nsamples = 5\nvalue = [0, 9, 3, 0]\ncl
ass = 2'),
 Text(0.6896551724137931, 0.4166666666666667, 'gini = 0.49\nsamples = 5\nvalue = [0, 4, 3, 0]\ncla
ss = 2'),
 Text(0.8275862068965517, 0.5833333333333334, 'm_dep <= 0.85\ngini = 0.739\nsamples = 104\nvalue =
[43, 46, 35, 26]\nclass = 2'),
 Text(0.7586206896551724, 0.4166666666666667, 'id <= 76.5\ngini = 0.742\nsamples = 88\nvalue = [4
1, 32, 30, 25]\nclass = 3'),
 Text(0.7241379310344828, 0.25, 'gini = 0.245\nsamples = 5\nvalue = [6, 0, 1, 0]\nclass = 3'),
 Text(0.7931034482758621, 0.25, 'ram <= 596.0\ngini = 0.746\nsamples = 83\nvalue = [35, 32, 29, 2
5]\nclass = 3'),
 Text(0.7586206896551724, 0.08333333333333333, 'gini = 0.655\nsamples = 13\nvalue = [1, 3, 8, 8]\n
class = 1'),
 Text(0.8275862068965517, 0.08333333333333333, 'gini = 0.733\nsamples = 70\nvalue = [34, 29, 21, 1
7]\nclass = 3'),
 Text(0.896551724137931, 0.4166666666666667, 'fc <= 0.5\ngini = 0.533\nsamples = 16\nvalue = [2, 1
4, 5, 1]\nclass = 2'),
 Text(0.8620689655172413, 0.25, 'gini = 0.0\nsamples = 5\nvalue = [0, 9, 0, 0]\nclass = 2'),
 Text(0.9310344827586207, 0.25, 'id <= 611.0\ngini = 0.675\nsamples = 11\nvalue = [2, 5, 5, 1]\ncl
ass = 2'),
 Text(0.896551724137931, 0.08333333333333333, 'gini = 0.571\nsamples = 6\nvalue = [2, 4, 1, 0]\ncl
ass = 2'),
 Text(0.9655172413793104, 0.08333333333333333, 'gini = 0.5\nsamples = 5\nvalue = [0, 1, 4, 1]\ncla
ss = 1')]
```



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 [21]:
```
rf_best.feature_importances_
```

Out[21]:
```
array([0.08446044, 0.06124087, 0.01067031, 0.04798739, 0.0108028 ,
       0.06612092, 0.01732903, 0.06063171, 0.03721904, 0.06339698,
       0.03464304, 0.07058962, 0.11265868, 0.07169324, 0.08589758,
       0.04729788, 0.03523517, 0.04876482, 0.00889158, 0.01232614,
       0.01214276])
```

In [22]:
```
imp_df=pd.DataFrame({"Varname":X_train.columns,"Imp":rf_best.feature_importances_})
```

In [23]:
```
imp_df.sort_values(by="Imp",ascending=False)
```

Out[23]:

	Varname	Imp
12	px_height	0.112659
14	ram	0.085898
0	id	0.084460
13	px_width	0.071693
11	pc	0.070590
5	fc	0.066121
9	mobile_wt	0.063397
1	battery_power	0.061241
7	int_memory	0.060632
17	talk_time	0.048765
3	clock_speed	0.047987
15	sc_h	0.047298
8	m_dep	0.037219
16	sc_w	0.035235
10	n_cores	0.034643
6	four_g	0.017329
19	touch_screen	0.012326
20	wifi	0.012143
4	dual_sim	0.010803
2	blue	0.010670
18	three_g	0.008892

In [ ]:
```

```