1).Test Data

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\HP\Downloads\Mobile_Price_Classification_test.csv")
df
```

Out[2]:

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

1000 rows × 21 columns

In [3]:
```
df.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 [4]:
```
x=df.drop('blue',axis=1)
y=df['blue']
```

In [5]:
```
df['dual_sim'].value_counts()
```

Out[5]:
```
dual_sim
1    517
0    483
Name: count, dtype: int64
```

In [17]:
```
x=df.drop('three_g',axis=1)
y=df['three_g']
```

In [18]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=42)
x_train.shape,x_test.shape
```

Out[18]:
```
((700, 20), (300, 20))
```

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[19]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [20]:
```
rf=RandomForestClassifier()
```

In [21]:
```
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 [22]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[22]:

GridSearchCV

estimator: RandomForestClassifier

RandomForestClassifier

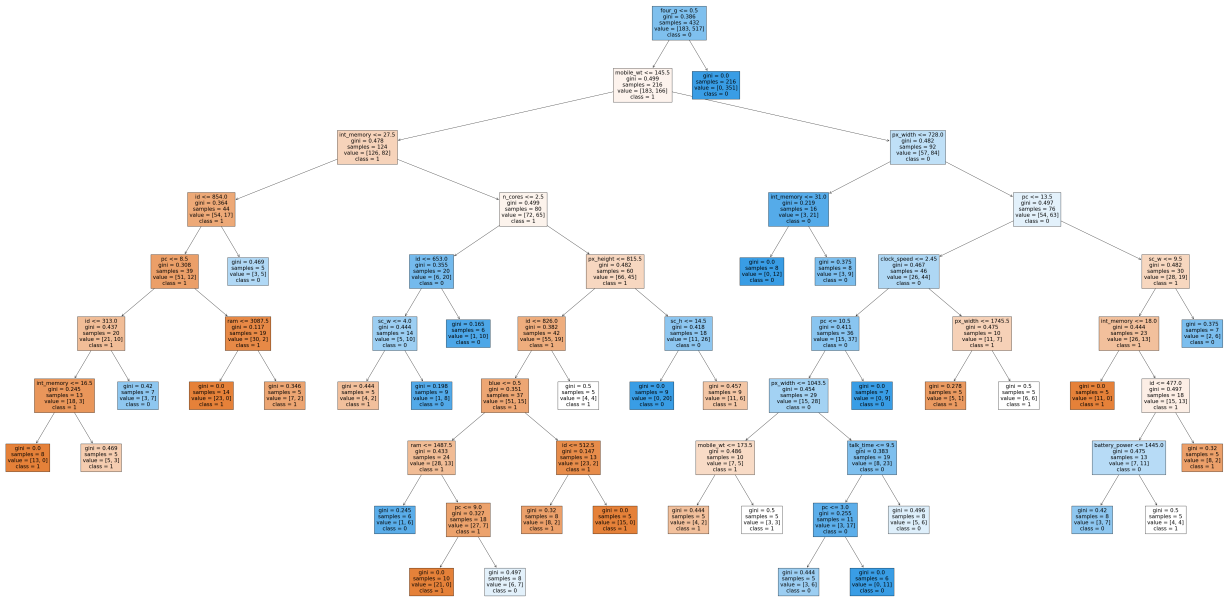
In [23]:
```
grid_search.best_score_
```

Out[23]:
```
0.7728571428571429
```

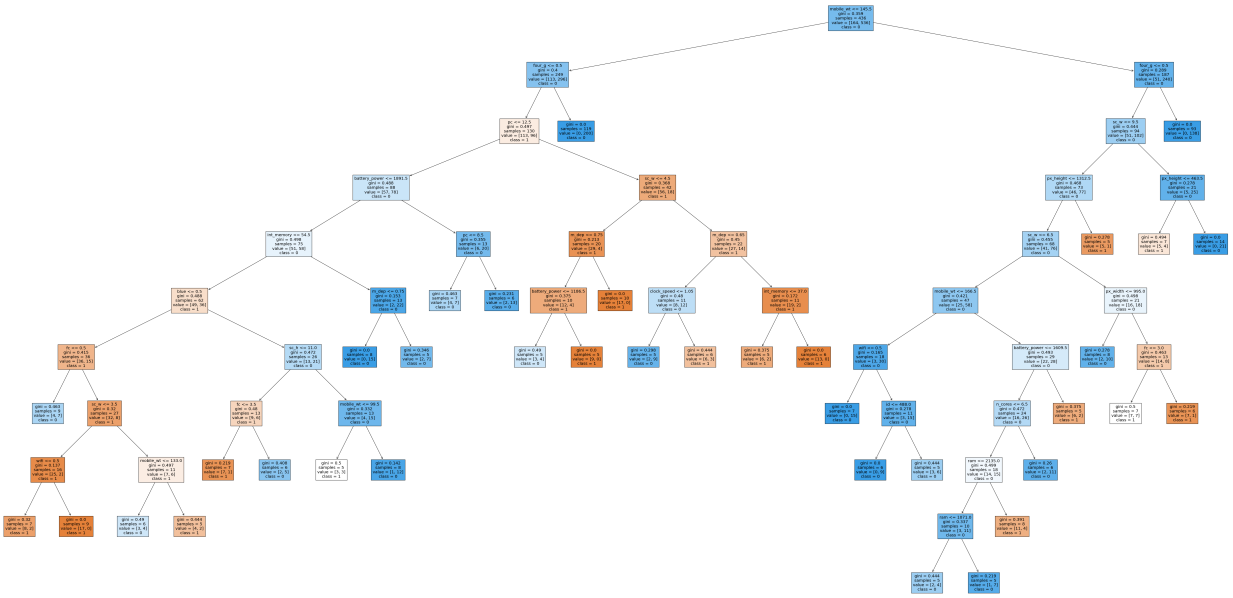
In [24]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=10, min_samples_leaf=5, n_estimators=10)
```

In [25]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=["1","0"],filled=True)
```



In [26]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=["1","0"],filled=True)
```



In [27]:
```
rf_best.feature_importances_
```

Out[27]:
```
array([0.06804712, 0.04019622, 0.00934414, 0.02592258, 0.00095101,
       0.03320852, 0.43192313, 0.0494484 , 0.02144998, 0.03610472,
       0.02400898, 0.03997531, 0.03466396, 0.03256085, 0.05666537,
       0.02471161, 0.03574372, 0.02094047, 0.00916053, 0.00497337])
```

In [28]:
```
imp_df=pd.DataFrame({"Varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[28]:

	Varname	Imp
6	four_g	0.431923
0	id	0.068047
14	ram	0.056665
7	int_memory	0.049448
1	battery_power	0.040196
11	pc	0.039975
9	mobile_wt	0.036105
16	sc_w	0.035744
12	px_height	0.034664
5	fc	0.033209
13	px_width	0.032561
3	clock_speed	0.025923
15	sc_h	0.024712
10	n_cores	0.024009
8	m_dep	0.021450
17	talk_time	0.020940
2	blue	0.009344
18	touch_screen	0.009161
19	wifi	0.004973
4	dual_sim	0.000951

2).Train Data

In [29]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [30]:
```
df=pd.read_csv(r"C:\Users\HP\Downloads\Mobile_Price_Classification_train.csv")
df
```

Out[30]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_l
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 [31]:
```
df.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 [32]:
```
x=df.drop('blue',axis=1)
y=df['blue']
```

In [33]:
```
df['dual_sim'].value_counts()
```

Out[33]:
```
dual_sim
1    1019
0     981
Name: count, dtype: int64
```

In [34]:
```
x=df.drop('three_g',axis=1)
y=df['three_g']
```

In [35]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=42)
x_train.shape,x_test.shape
```

Out[35]:
```
((1400, 20), (600, 20))
```

In [36]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[36]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [37]:
```
rf=RandomForestClassifier()
```

In [38]:
```
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 [39]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[39]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

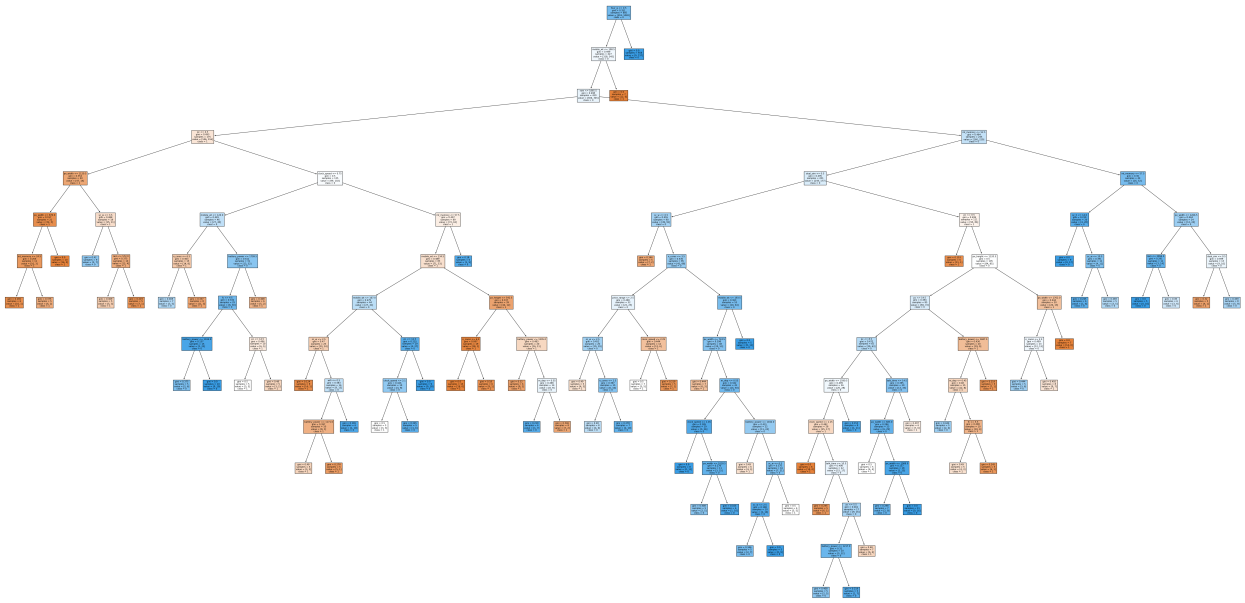
In [40]:
```
grid_search.best_score_
```

Out[40]:
```
0.7807142857142857
```

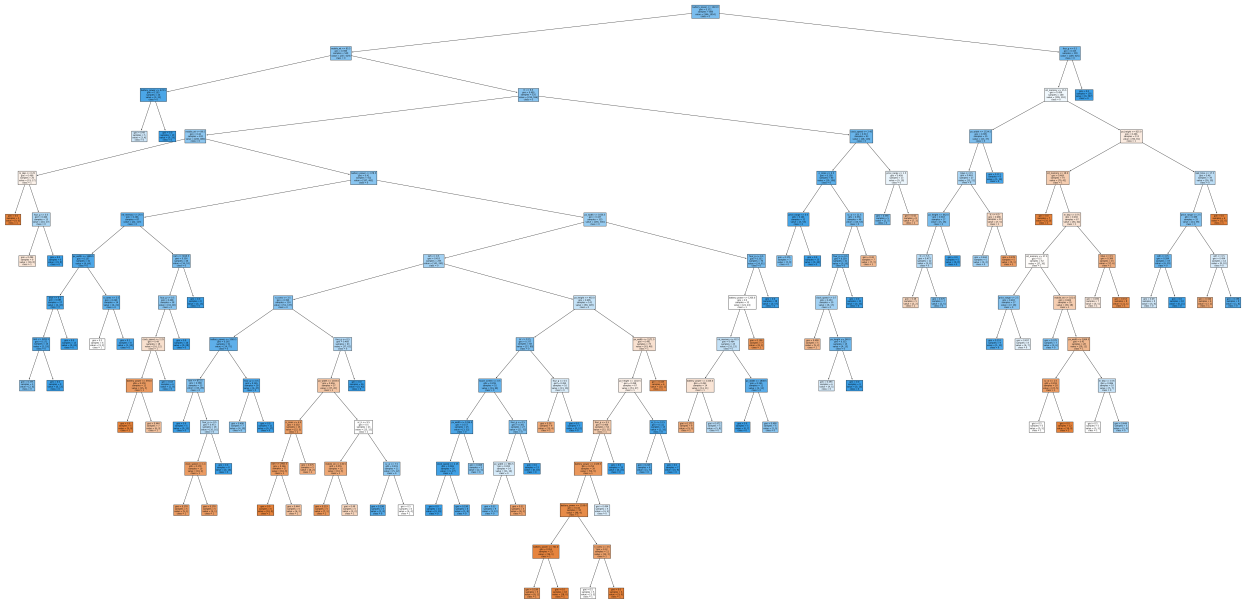
In [41]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=30)
```

In [42]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=["1","0"],filled=True)
```



In [43]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=["1","0"],filled=True)
```



In [44]:
```
rf_best.feature_importances_
```

Out[44]:
```
array([0.05435418, 0.00819718, 0.03898139, 0.00839901, 0.03432097,
       0.41497394, 0.03732129, 0.02513686, 0.04400458, 0.02272979,
       0.02688523, 0.05361046, 0.05244247, 0.05379899, 0.02347797,
       0.03565623, 0.03802804, 0.00886691, 0.00896868, 0.00984582])
```

In [45]:
```
imp_df=pd.DataFrame({"Varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[45]:

	Varname	Imp
5	four_g	0.414974
0	battery_power	0.054354
13	ram	0.053799
11	px_height	0.053610
12	px_width	0.052442
8	mobile_wt	0.044005
2	clock_speed	0.038981
16	talk_time	0.038028
6	int_memory	0.037321
15	sc_w	0.035656
4	fc	0.034321
10	pc	0.026885
7	m_dep	0.025137
14	sc_h	0.023478
9	n_cores	0.022730
19	price_range	0.009846
18	wifi	0.008969
17	touch_screen	0.008867
3	dual_sim	0.008399
1	blue	0.008197

In [ ]:
```

```