TRAIN 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\OneDrive\Documents\Mobile_Price_Classification_train.c
df
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_
0	842	0	2.2	0	1	0	7	0.6	1
1	1021	1	0.5	1	0	1	53	0.7	1
2	563	1	0.5	1	2	1	41	0.9	1
3	615	1	2.5	0	0	0	10	0.8	1
4	1821	1	1.2	0	13	1	44	0.6	1
...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	1
1996	1965	1	2.6	1	0	0	39	0.2	1
1997	1911	0	0.9	1	1	1	36	0.7	1
1998	1512	0	0.9	0	4	1	46	0.1	1
1999	510	1	2.0	1	5	1	45	0.9	1

2000 rows × 21 columns

In [3]:

```
1 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 [4]:

```
1 df.isnull().sum()
```

Out[4]:

```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [5]:

```
1 X=df.drop('dual_sim',axis=1)
2 y=df['dual_sim']
```

In [6]:

```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.4,random_state=40
3 X_train.shape,X_test.shape
```

Out[6]:

```
((800, 20), (1200, 20))
```

In [7]:

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

Out[7]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [8]:

```
rf=RandomForestClassifier()
```

In [9]:

```
params = {'max_depth':[1,2,3,4,5],'min_samples_leaf':[5,10,25,50,100,200],'n_estima
```

In [10]:

```
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[10]:

```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 25, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [21]:

```
grid_search.best_score_
```

Out[21]:

```
0.5487500000000001
```

In [12]:

```
rf_best = grid_search.best_estimator_
print(rf_best)
```

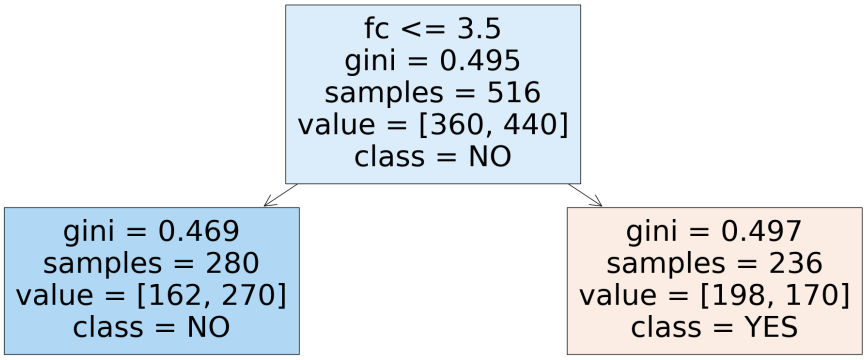
```
RandomForestClassifier(max_depth=1, min_samples_leaf=50)
```

In [25]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(30,10))
plot_tree(rf_best.estimators_[5],feature_names=X.columns,class_names=["YES","NO"],f
```

Out[25]:

```
[Text(0.5, 0.75, 'fc <= 3.5\ngini = 0.495\nsamples = 516\nvalue = [360, 4
40]\nclass = NO'),
 Text(0.25, 0.25, 'gini = 0.469\nsamples = 280\nvalue = [162, 270]\nclass
= NO'),
 Text(0.75, 0.25, 'gini = 0.497\nsamples = 236\nvalue = [198, 170]\nclass
= YES')]
```

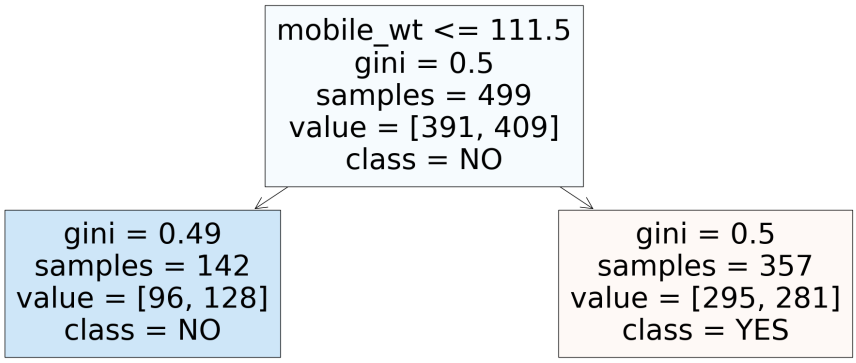


In [26]:

```
1 plt.figure(figsize=(30,10))
2 plot_tree(rf_best.estimators_[7],feature_names=X.columns,class_names=["YES","NO"],f
```

Out[26]:

```
[Text(0.5, 0.75, 'mobile_wt <= 111.5\ngini = 0.5\nsamples = 499\nvalue =
[391, 409]\nclass = NO'),
 Text(0.25, 0.25, 'gini = 0.49\nsamples = 142\nvalue = [96, 128]\nclass =
NO'),
 Text(0.75, 0.25, 'gini = 0.5\nsamples = 357\nvalue = [295, 281]\nclass =
YES')]
```



In [27]:

```
1 rf_best.feature_importances_
```

Out[27]:

```
array([0.06, 0.01, 0.05, 0.09, 0.  , 0.14, 0.08, 0.12, 0.05, 0.05, 0.04,
       0.08, 0.  , 0.12, 0.02, 0.07, 0.  , 0.  , 0.02, 0.  ])
```

In [28]:

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

In [29]:

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

Out[29]:

	Varname	Imp
5	int_memory	0.14
7	mobile_wt	0.12
13	sc_h	0.12
3	fc	0.09
11	px_width	0.08
6	m_dep	0.08
15	talk_time	0.07
0	battery_power	0.06
2	clock_speed	0.05
8	n_cores	0.05
9	pc	0.05
10	px_height	0.04
14	sc_w	0.02
18	wifi	0.02
1	blue	0.01
12	ram	0.00
4	four_g	0.00
16	three_g	0.00
17	touch_screen	0.00
19	price_range	0.00

TEST DATA

In [30]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [31]:

```
1 test_df = pd.read_csv(r"C:\Users\HP\OneDrive\Documents\Mobile_Price_Classification_
2 test_df
```

Out[31]:

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

1000 rows × 21 columns

In [32]:

```
1 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 [33]:

```
1 df.isnull().sum()
```

Out[33]:

```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [34]:

```
1 X=df.drop('dual_sim',axis=1)
2 y=df['dual_sim']
```

In [35]:

```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=6)
3 X_train.shape,X_test.shape
```

Out[35]:

```
((1500, 20), (500, 20))
```

In [37]:

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

Out[37]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [40]:

```
rf=RandomForestClassifier()
```

In [43]:

```
params = {'max_depth':[2,4,6,8,10],'min_samples_leaf':[5,10,20,30,50,60],'n_estimat
```

In [44]:

```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=rf,param_grid = params,cv=2,scoring = 'accurac
grid_search.fit(X_train,y_train)
```

Out[44]:

```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 4, 6, 8, 10],
                         'min_samples_leaf': [5, 10, 20, 30, 50, 60],
                         'n_estimators': [10, 50, 60, 100, 200, 300]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [45]:

```
grid_search.best_score_
```

Out[45]:

```
0.52
```

In [48]:

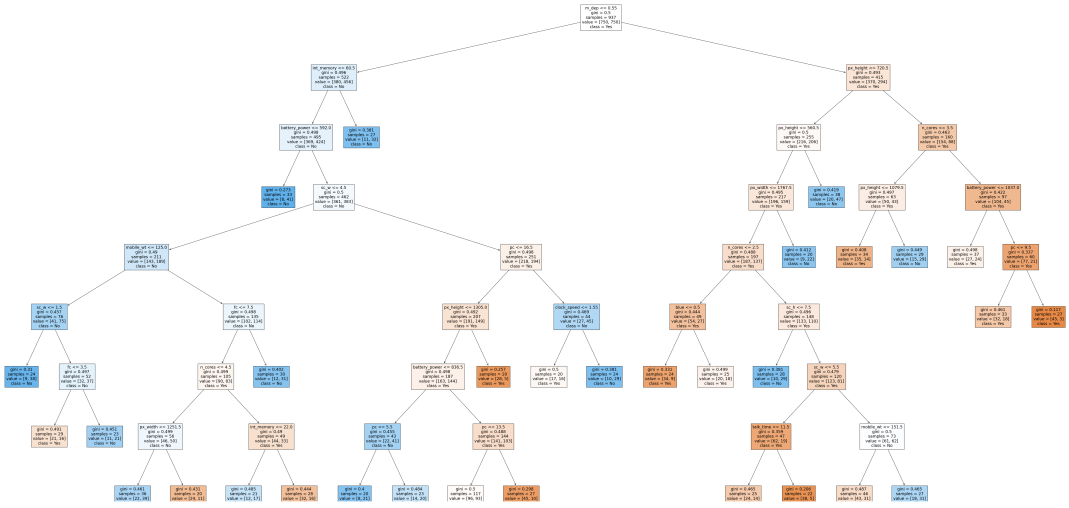
```
rf_best = grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=8, min_samples_leaf=20)
```

In [51]:

```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=X.columns,class_names=['Yes','No'],f
```

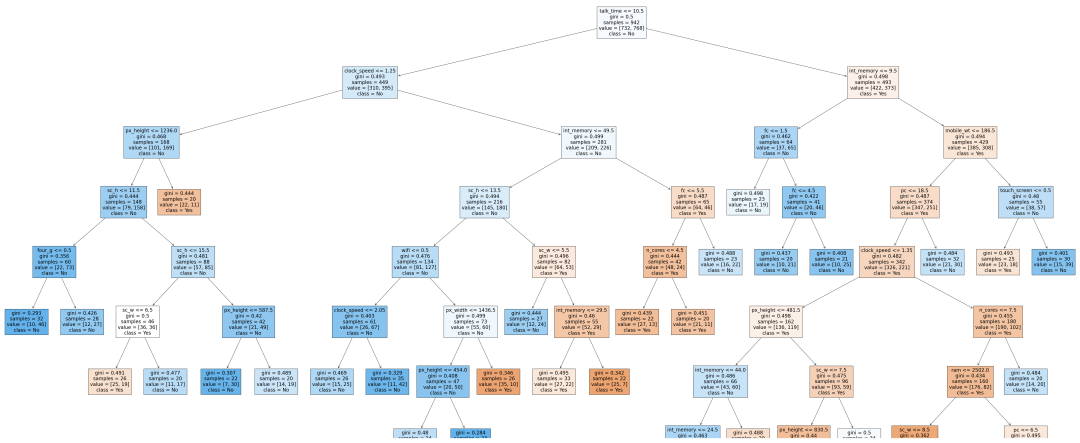
```
33\nvalue = [32, 18]\nclass = Yes'),
 Text(0.9743589743589743, 0.3888888888888889, 'gini = 0.117\nsamples =
27\nvalue = [45, 3]\nclass = Yes')]
```



In [53]:

```
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=X.columns,class_names=['Yes','No'],f
```

```
ni = 0.48\nsamples = 55\nvalue = [38, 57]\nclass = No'),
 Text(0.9230769230769231, 0.5, 'gini = 0.493\nsamples = 25\nvalue = [2
3, 18]\nclass = Yes'),
 Text(0.9743589743589743, 0.5, 'gini = 0.401\nsamples = 30\nvalue = [1
5, 39]\nclass = No')]
```



In [54]:

```
1 rf_best.feature_importances_
```

Out[54]:

```
array([0.10392769, 0.01476017, 0.05275833, 0.03242578, 0.01083849,
       0.05951906, 0.05022448, 0.09871062, 0.02637821, 0.04426562,
       0.08860137, 0.09012254, 0.11746061, 0.05019014, 0.05154751,
       0.06270661, 0.00375218, 0.00980958, 0.01330517, 0.01869583])
```

In [57]:

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

In [56]:

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

Out[56]:

	Varname	Imp
12	ram	0.117461
0	battery_power	0.103928
7	mobile_wt	0.098711
11	px_width	0.090123
10	px_height	0.088601
15	talk_time	0.062707
5	int_memory	0.059519
2	clock_speed	0.052758
14	sc_w	0.051548
6	m_dep	0.050224
13	sc_h	0.050190
9	pc	0.044266
3	fc	0.032426
8	n_cores	0.026378
19	price_range	0.018696
1	blue	0.014760
18	wifi	0.013305
4	four_g	0.010838
17	touch_screen	0.009810
16	three_g	0.003752

In [ ]:

```
1
```