Random Forest/ Bootstrap Aggregation

Mobile classification Case study

In [1]:
```
import pandas as pd
```

In [3]:
```
mobile = pd.read_csv(r'c:\Users\Admin\Desktop\Mobile Classification.csv')
mobile.head()
```

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1

5 rows × 21 columns

In [4]:
```
mobile.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 [5]:
```
mobile.isnull().sum()
```

Out[5]:
```
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 [7]:
```
mobile['price_range'].nunique()
```

Out[7]:
```
4
```

In [6]:
```
# Finding out the number of categories in price range

mobile['price_range'].unique()
```

Out[6]:
```
array([1, 2, 3, 0], dtype=int64)
```

Assumption:- If the price range 0, the mobiles are cheaper and if the price range is 3, mobiles are costlier.

To validate the above assumption, we need to group the price ranges based on average ram.

In [16]:
```
mobile.groupby('price_range')[['ram']].mean().reset_index()
```

Out[16]:

	price_range	ram
0	0	785.314
1	1	1679.490
2	2	2582.816
3	3	3449.232

In [17]:
```
# Indentifying the output and input variables

y = mobile[['price_range']]
x = mobile.drop(columns=['price_range'])
```

In [19]:
```
# Splitting the data into train and test

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

In [20]:
```
len(x_train),len(x_test),len(y_train),len(y_test)
```

Out[20]:
```
(1600, 400, 1600, 400)
```

Building Random Forest

In [21]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 500)
model = rf.fit(x_train,y_train)
print('The model has been built succesfuly !! Yeah!')
```

```
C:\Users\Admin\AppData\Local\Temp/ipykernel_13596/413783706.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel
().
  model = rf.fit(x_train,y_train)
The model has been built succesfuly !! Yeah!
```

In [22]:
```
# Predicting on test data

y_test['Prediction'] = model.predict(x_test)
```

```
C:\Users\Admin\AppData\Local\Temp/ipykernel_13596/2239895179.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  y_test['Prediction'] = model.predict(x_test)
```

In [23]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [24]:
```
print(confusion_matrix(y_test['price_range'],y_test['Prediction']))
```

```
[[ 93   2   0   0]
 [  5  74  13   0]
 [  0  15  75   9]
 [  0   0   7 107]]
```

In [25]:
```
print(accuracy_score(y_test['price_range'],y_test['Prediction']))
```

```
0.8725
```

So by building Random forest model we are getting accuracy of 87%.

In [ ]:
```

```