Business Problem :

a mobile's factory owner needs to estimate the price ranges of mobile based on deferent mobile Specification

data dictionary :

- id:ID
- battery_power:Total energy a battery can store in one time measured in mAh
- blue:Has bluetooth or not
- clock_speed:speed at which microprocessor executes instructions
- dual_sim:Has dual sim support or not
- fc:Front Camera mega pixels
- four_g:Has 4G or not
- int_memory:Internal Memory in Gigabytes
- m_dep:Mobile Depth in cm
- mobile_wt:Weight of mobile phone
- n_cores:Number of cores of processor
- pc:Primary Camera mega pixels
- px_height:Pixel Resolution Height
- px_width:Pixel Resolution Width
- ram:Random Access Memory in Megabytes
- sc_h:Screen Height of mobile in cm
- sc_w:Screen Width of mobile in cm
- talk_time:longest time that a single battery charge will last when you are
- three_g:Has 3G or not
- touch_screen:Has touch screen or not
- wifi:Has wifi or not

In [1]:

```
#pip install cufflinks
```

In [2]:

```
#pip install chart_studio
```

In [3]:

```
#pip install plotly
```

In [4]:

```
#pip install lightgbm
```

In [5]:

```
#pip install pandas_profiling
```

In [6]:

```
import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style("darkgrid")
```

In [7]:

```
from collections import Counter
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import label_binarize
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve
from sklearn.multiclass import OneVsRestClassifier
from lightgbm import LGBMClassifier
```

In [8]:

```
import warnings
warnings.filterwarnings("ignore")
import chart_studio.plotly as py
import cufflinks as cf
import plotly.express as px
%matplotlib inline
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected = True)
cf.go_offline();
import plotly.graph_objs as go
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Load and Check Data

In [9]:

```
df = pd.read_csv('train.csv')
df.head(10)
```

Out[9]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	
5	1859	0	0.5	1	3	0	22	0.7	164	
6	1821	0	1.7	0	4	1	10	0.8	139	
7	1954	0	0.5	1	0	0	24	0.8	187	
8	1445	1	0.5	0	0	0	53	0.7	174	
9	509	1	0.6	1	2	1	9	0.1	93	

10 rows × 21 columns

In [10]:

```
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 [11]:

```
1 df.profile_report()
```

Summarize dataset: 34/34 [00:45<00:00, 1.33s/it, 100% Completed]

Generate report structure: 1/1 [00:27<00:00, 27.49s/it] 100%

Render HTML: 100% 1/1 [00:17<00:00, 17.97s/it]

Overview

Dataset statistics

Number of variables	21
Number of observations	2000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	328.2 KiB
Average record size in memory	168.1 B

Variable types

Numeric	14
Categorical	7

Warnings

`fc` is highly correlated with `pc`	**High correlation**
`four_g` is highly correlated with `three_g`	**High correlation**
`pc` is highly correlated with `fc`	**High correlation**
`px_height` is highly correlated with `px_width`	**High correlation**

Out[11]:

In [ ]:

```
1
```

In [ ]:

```
1
```

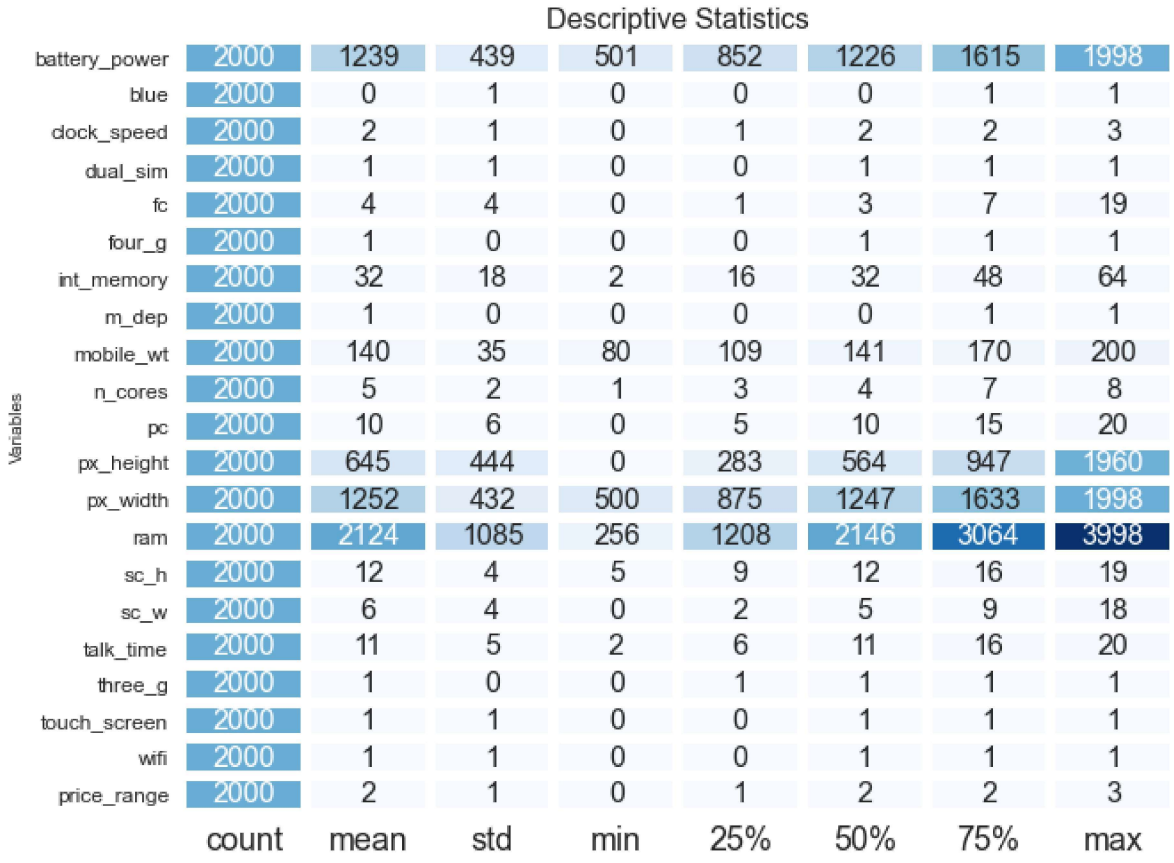
In [ ]:

```
1
```

Descriptive Statistics

In [12]:

```
1  desc = df.describe().T
2
3  df1 = pd.DataFrame(index=['battery_power','blue','clock_speed','dual_sim','fc','four_g'
4  'px_width','ram','sc_h','sc_w','talk_time','three_g','touch_screen','wifi','price_range
5                    columns= ["count","mean","std","min",
6                              "25%","50%","75%","max"], data= desc )
7
8  f,ax = plt.subplots(figsize=(10,8))
9
10 sns.heatmap(df1, annot=True,cmap = "Blues", fmt= '.0f',
11             ax=ax,linewidths = 5, cbar = False,
12             annot_kws={"size": 16})
13
14 plt.xticks(size = 18)
15 plt.yticks(size = 12, rotation = 0)
16 plt.ylabel("Variables")
17 plt.title("Descriptive Statistics", size = 16)
18 plt.show()
```



In [ ]:

```
1
```