A MACHINE LEARNING PROJECT ON MOBILE PRICE CLASSIFICATION

BY TEAM ORION

CONTENTS

- INTRODUCTION
- IMPORTED MODULES
- DATASET
- DATA PROCESSING
- GRAPHS
- CORRELATION
- DATA PREPROCESSING
- TRAIN AND TEST SPLIT
- RANDOM FOREST CLASSIFIER
- CLASSIFICATION

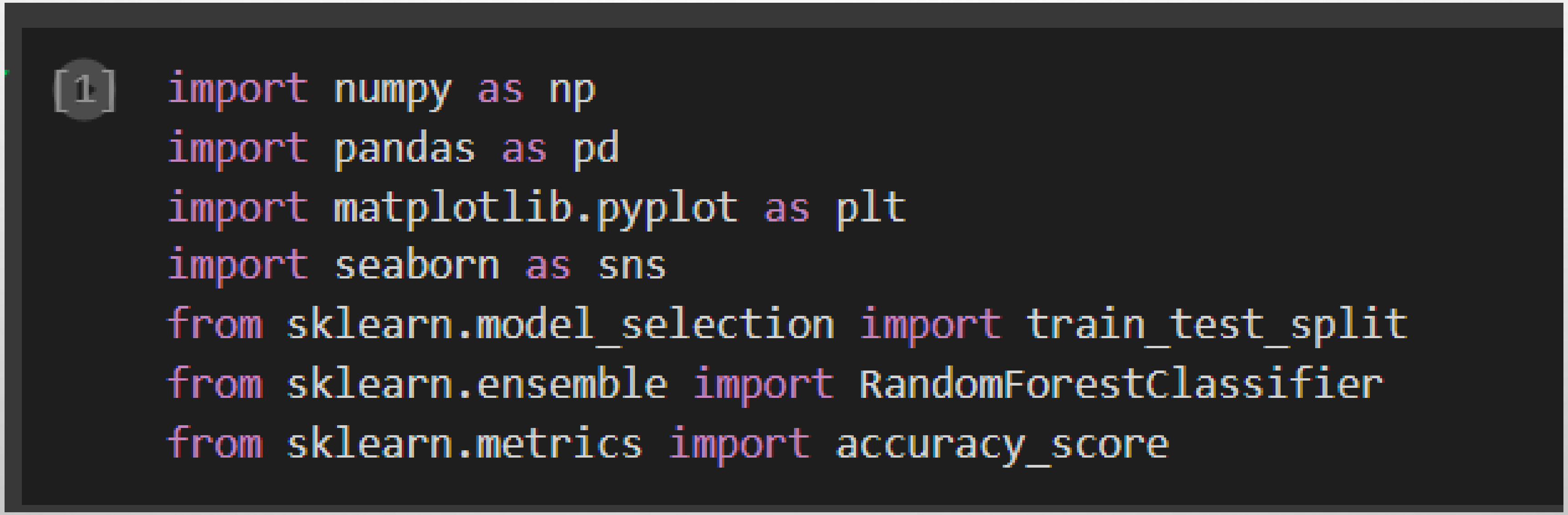
DATASET

WE GOT THE DATASET FOR THIS PROJECT FROM KAGGLE, FROM THIS LINK:

HTTPS://WWW.KAGGLE.COM/DATASETS/IABHISHEKOFFICIAL/MOBILE-PRICE-CLASSIFICATION

IMPORTED MODULES

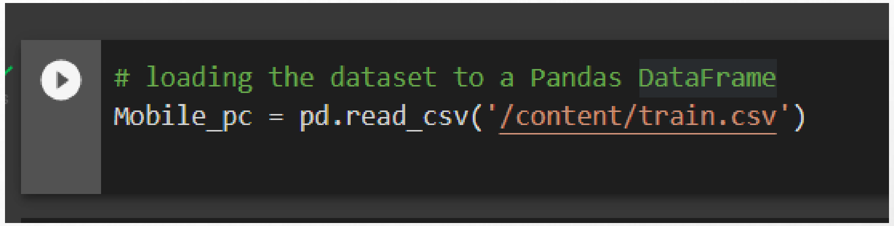
WE IMPORTED THE FOLLOWING MODULES FOR THIS PROJECT:



```
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.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

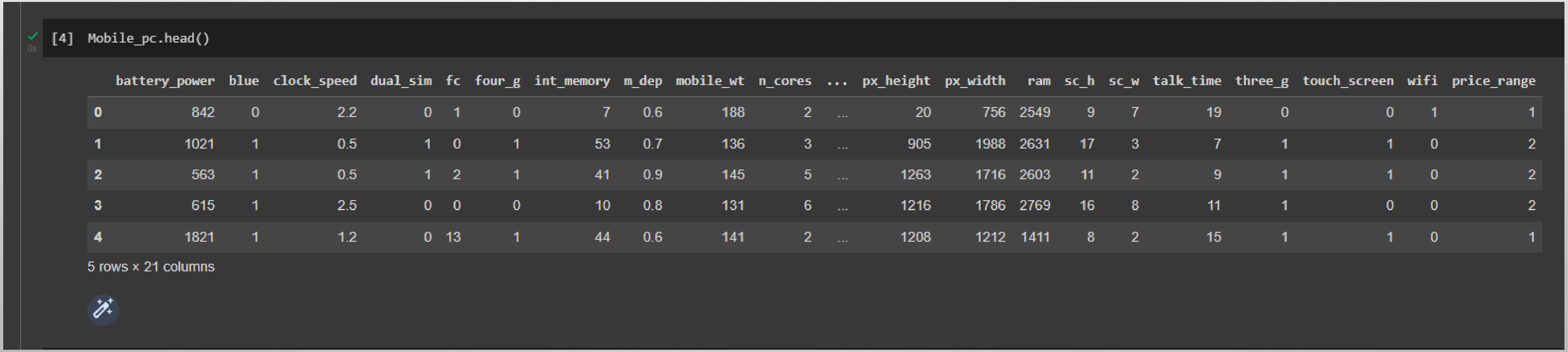
DATA PROCESSING

- FIRST WE LOADED OUR DATASET TO A DATAFRAME



```
# loading the dataset to a Pandas DataFrame
Mobile_pc = pd.read_csv('/content/train.csv')
```

- THEN FOUND THE NUMBER OF ROWS AND COLUMNS

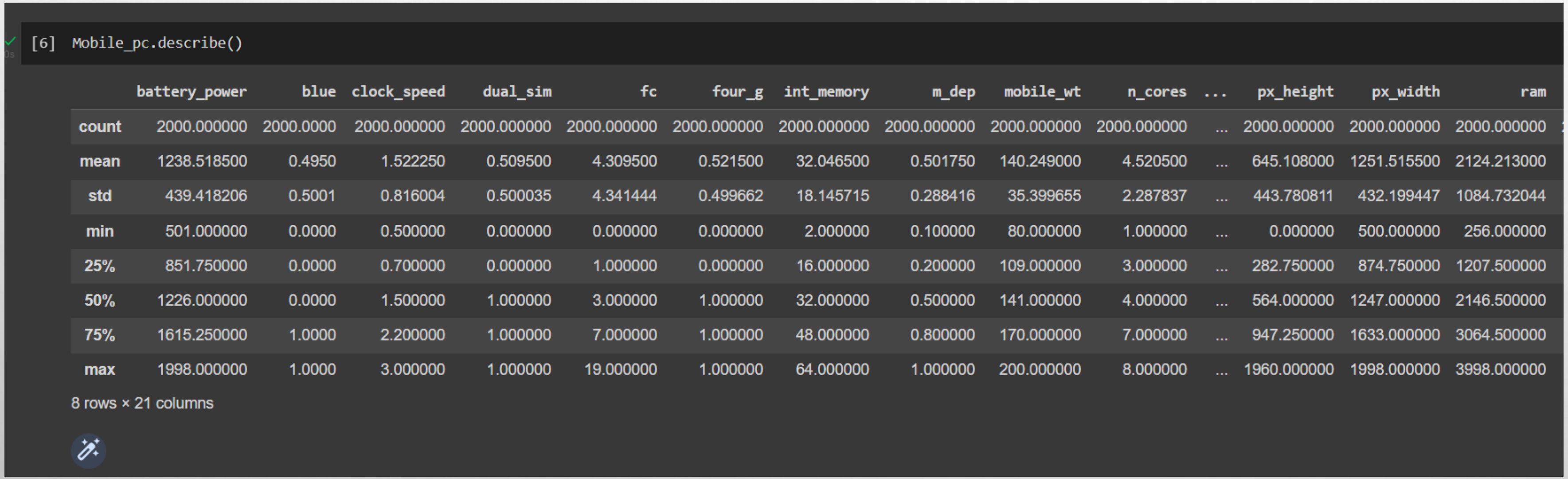


```
[4]  Mobile_pc.head()
```

```
   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
```

DATA PROCESSING

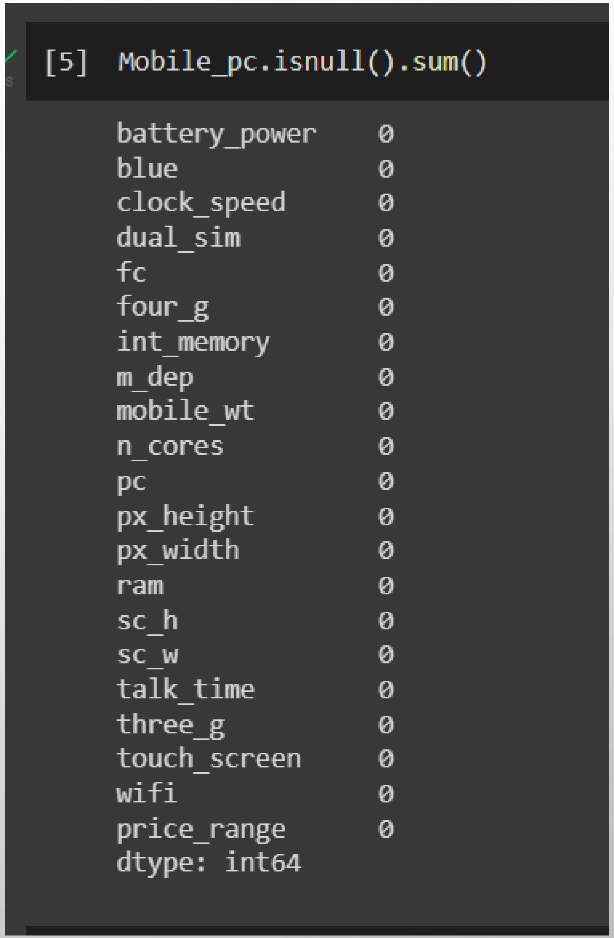
- THEN SEARCHED FOR NULL VALUES.
- AFTER THAT, GOT THE DESCRIPTION OF THE DATAFRAME.



```
[6] Mobile_pc.describe()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	...	645.108000	1251.515500	2124.213000
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	...	443.780811	432.199447	1084.732044
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	...	0.000000	500.000000	256.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	...	282.750000	874.750000	1207.500000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	...	564.000000	1247.000000	2146.500000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	...	947.250000	1633.000000	3064.500000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	...	1960.000000	1998.000000	3998.000000

8 rows × 21 columns



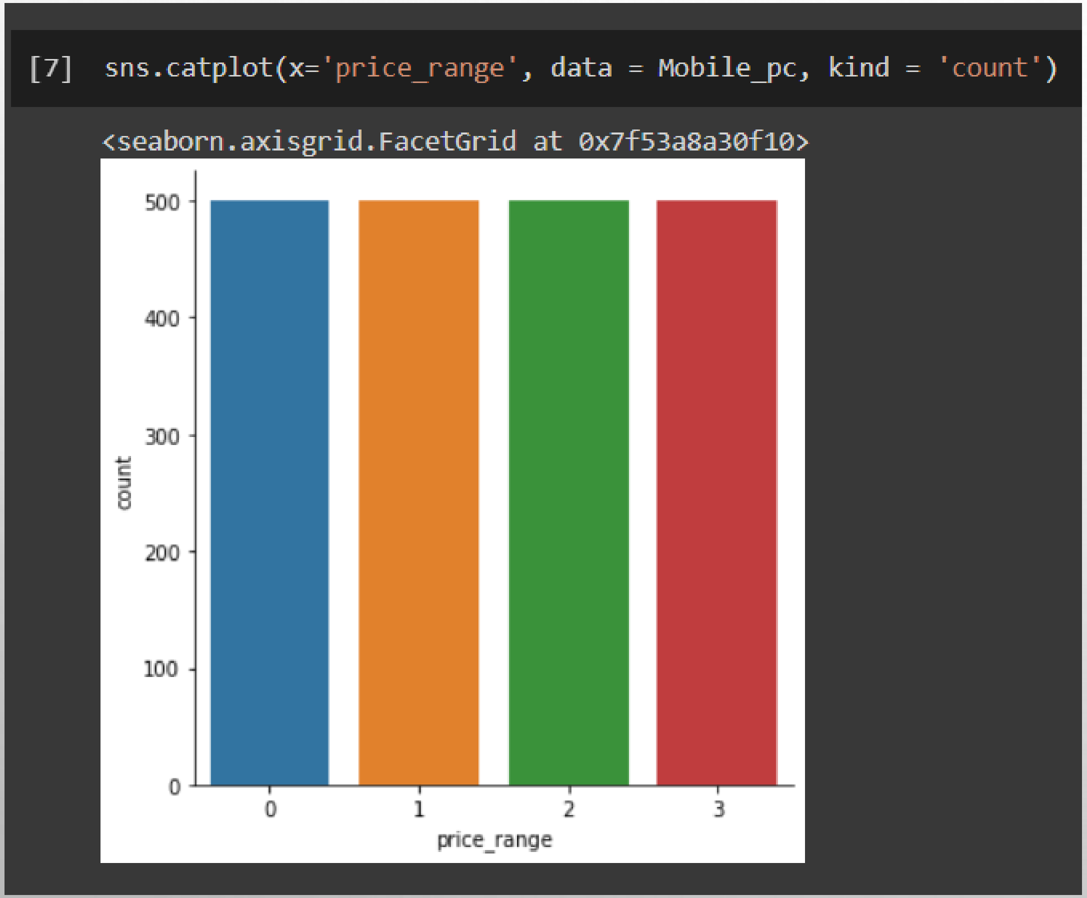
```
[5] Mobile_pc.isnull().sum()

    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
```

GRAPHS

PLOTTED THE FOLLOWING GRAPHS:

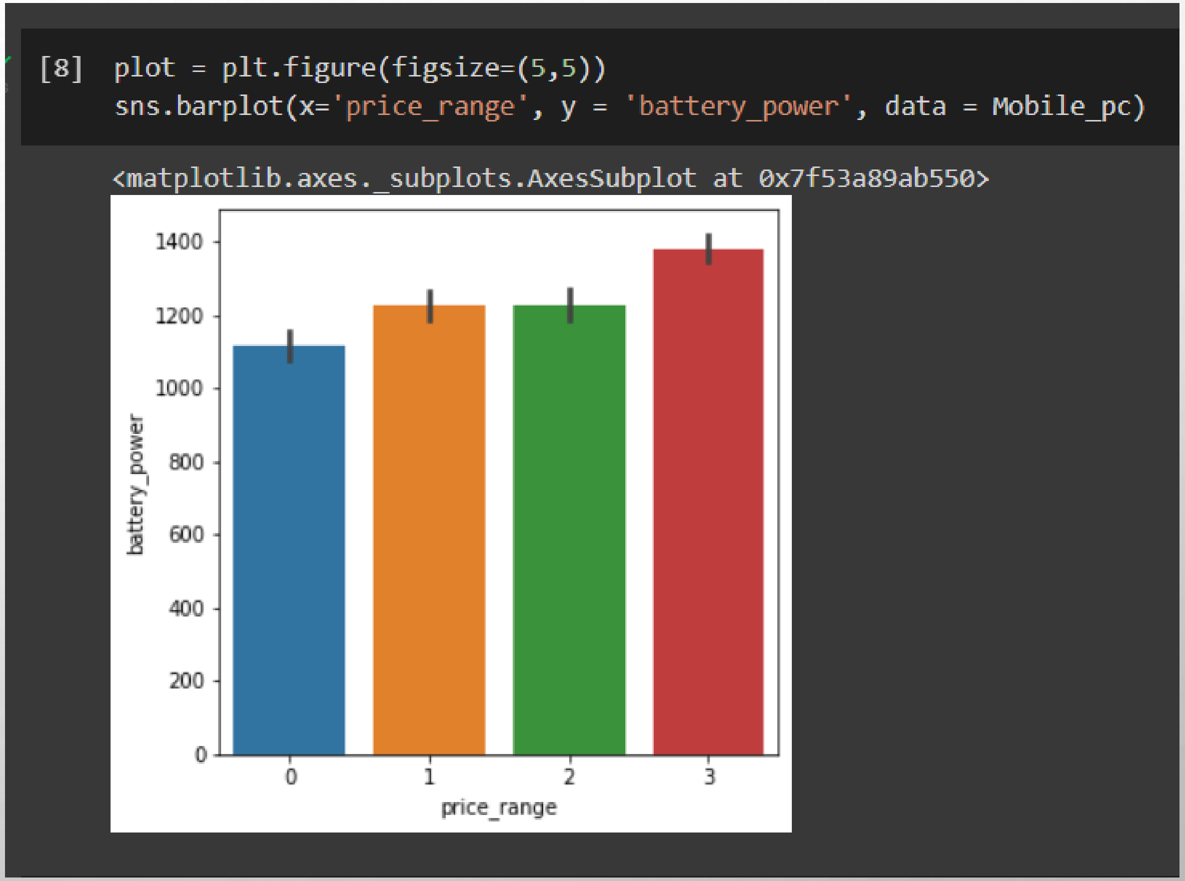
1.



```
[7]  sns.catplot(x='price_range', data = Mobile_pc, kind = 'count')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f53a8a30f10>
```

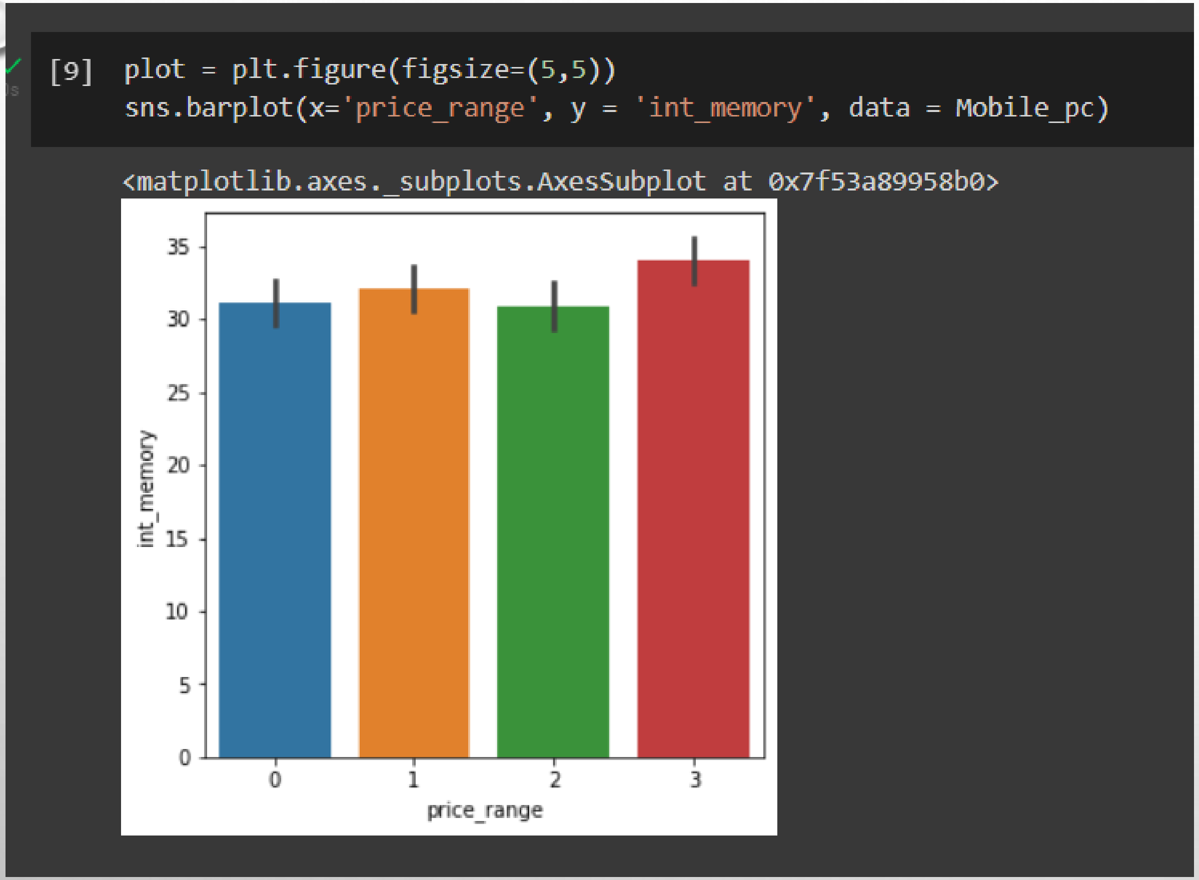
2.



```
[8]  plot = plt.figure(figsize=(5,5))
     sns.barplot(x='price_range', y = 'battery_power', data = Mobile_pc)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f53a89ab550>
```

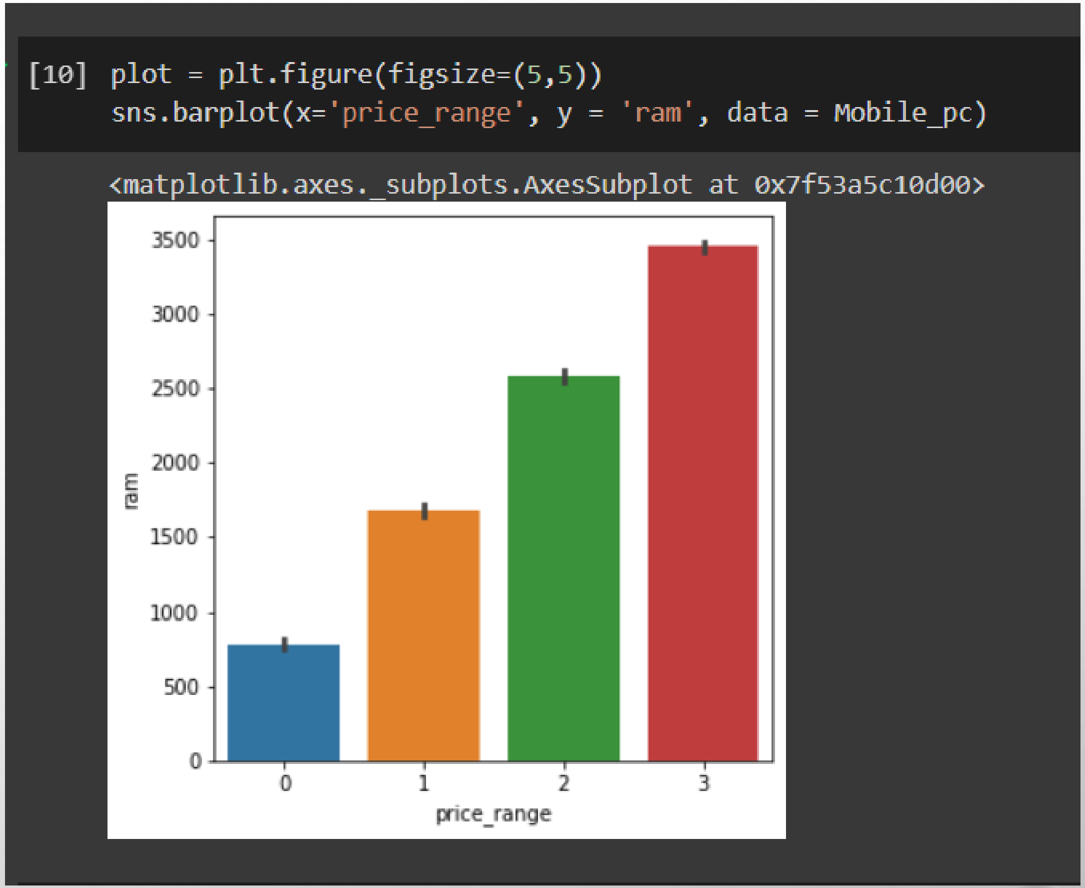
3.



```
[9] plot = plt.figure(figsize=(5,5))
    sns.barplot(x='price_range', y = 'int_memory', data = Mobile_pc)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f53a89958b0>
```

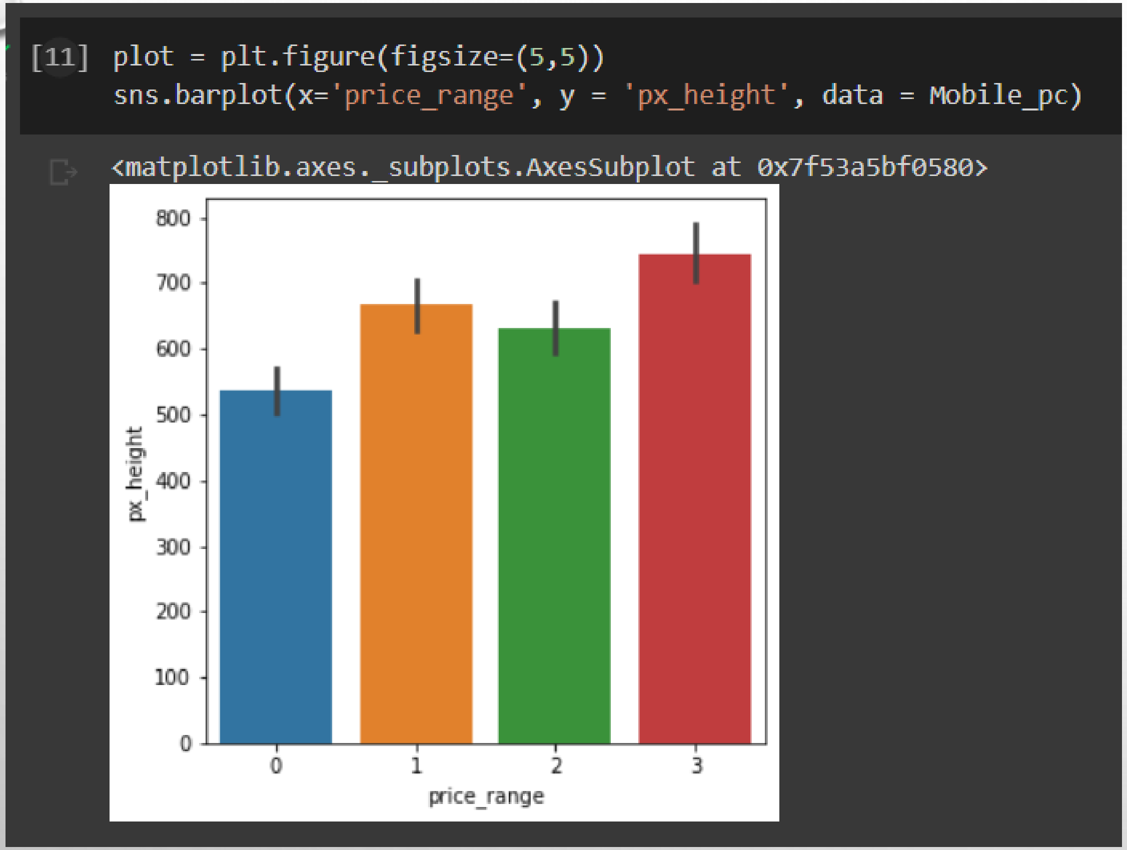
4.



```
[10] plot = plt.figure(figsize=(5,5))
     sns.barplot(x='price_range', y = 'ram', data = Mobile_pc)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f53a5c10d00>
```

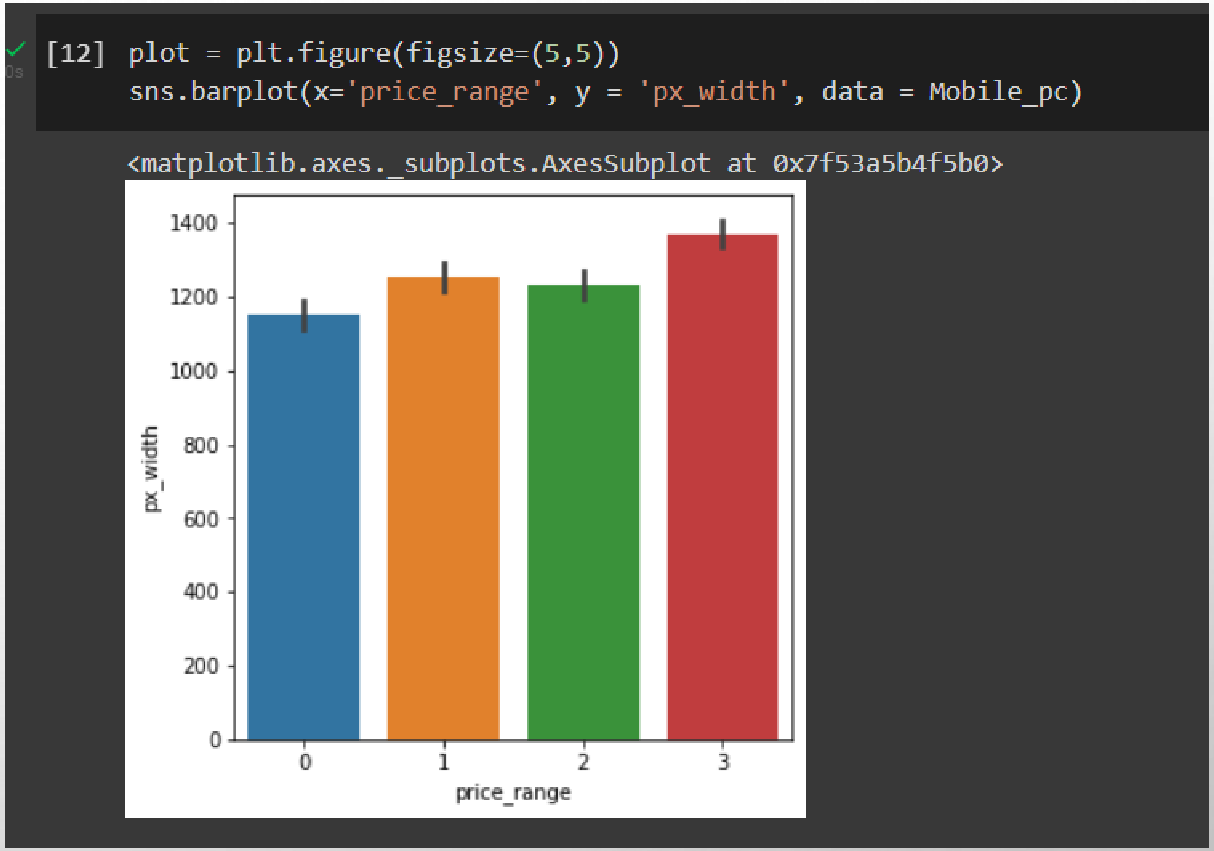
5.



```
[11] plot = plt.figure(figsize=(5,5))
     sns.barplot(x='price_range', y = 'px_height', data = Mobile_pc)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f53a5bf0580>
```

6.

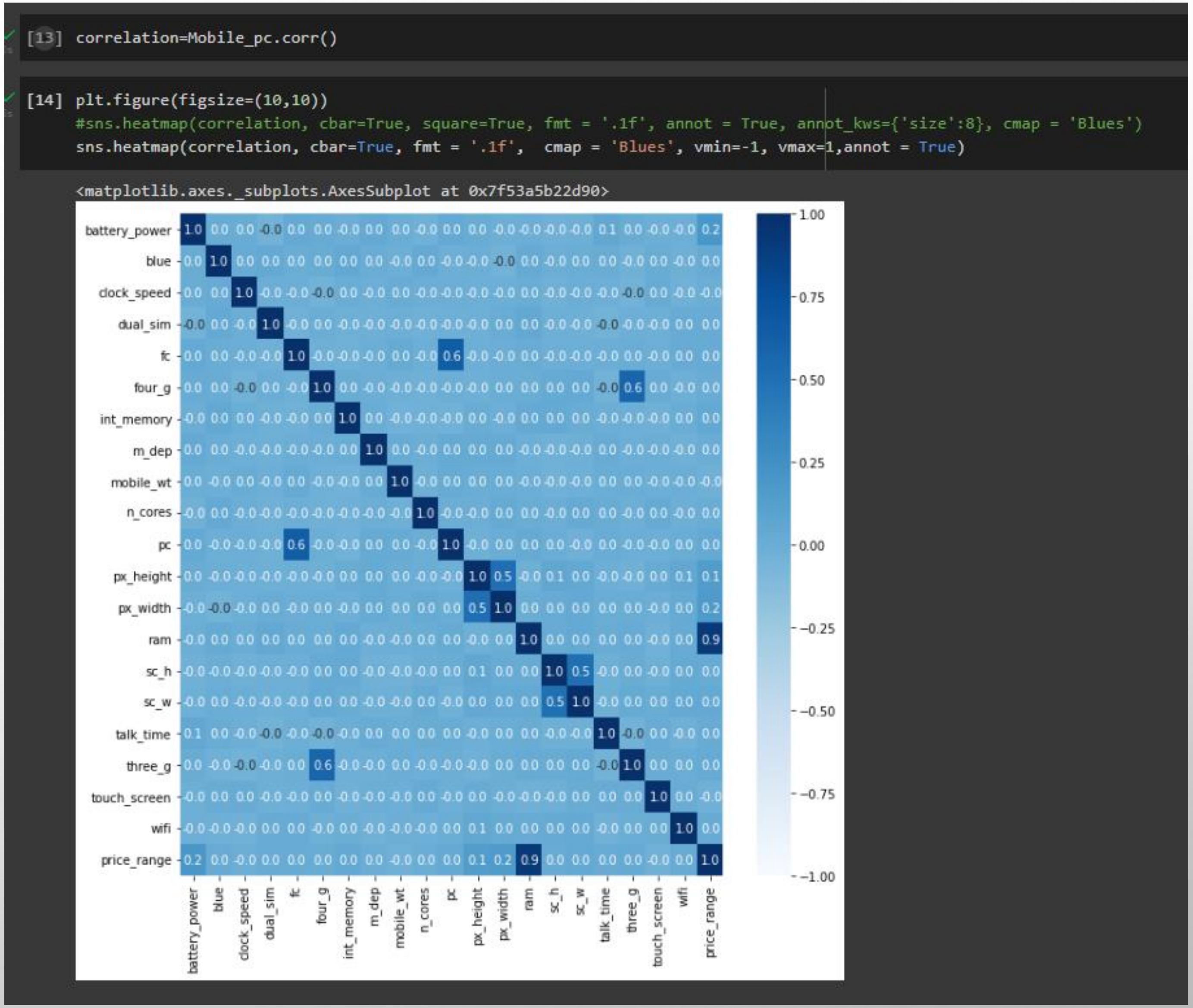


```
[12] plot = plt.figure(figsize=(5,5))
     sns.barplot(x='price_range', y = 'px_width', data = Mobile_pc)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f53a5b4f5b0>
```

CORRELATION

THE GRAPH TO DEPICT THE CORRELATION AMONG THE SPECIFICATIONS IS FORMED AS FOLLOWS:



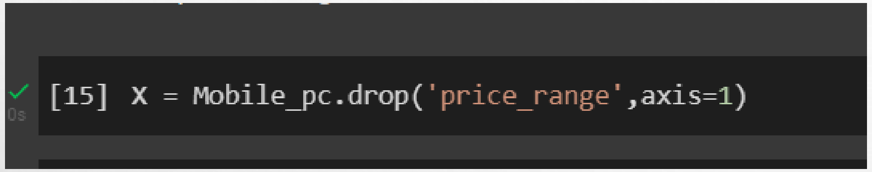
```
[13] correlation=Mobile_pc.corr()
```

```
[14] plt.figure(figsize=(10,10))
     #sns.heatmap(correlation, cbar=True, square=True, fmt = '.1f', annot = True, annot_kws={'size':8}, cmap = 'Blues')
     sns.heatmap(correlation, cbar=True, fmt = '.1f',  cmap = 'Blues', vmin=-1, vmax=1,annot = True)
```

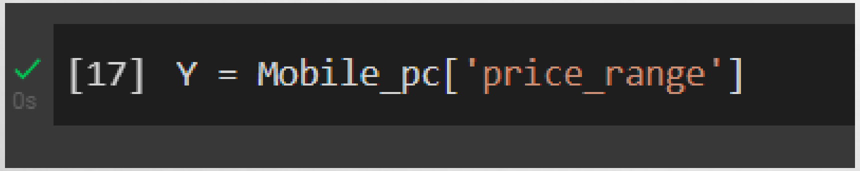
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f53a5b22d90>
```

DATA PREPROCESSING

MADE TWO SEPARATE DATAFRAMES WITH AND WITHOUT THE RESULT COLUMN.



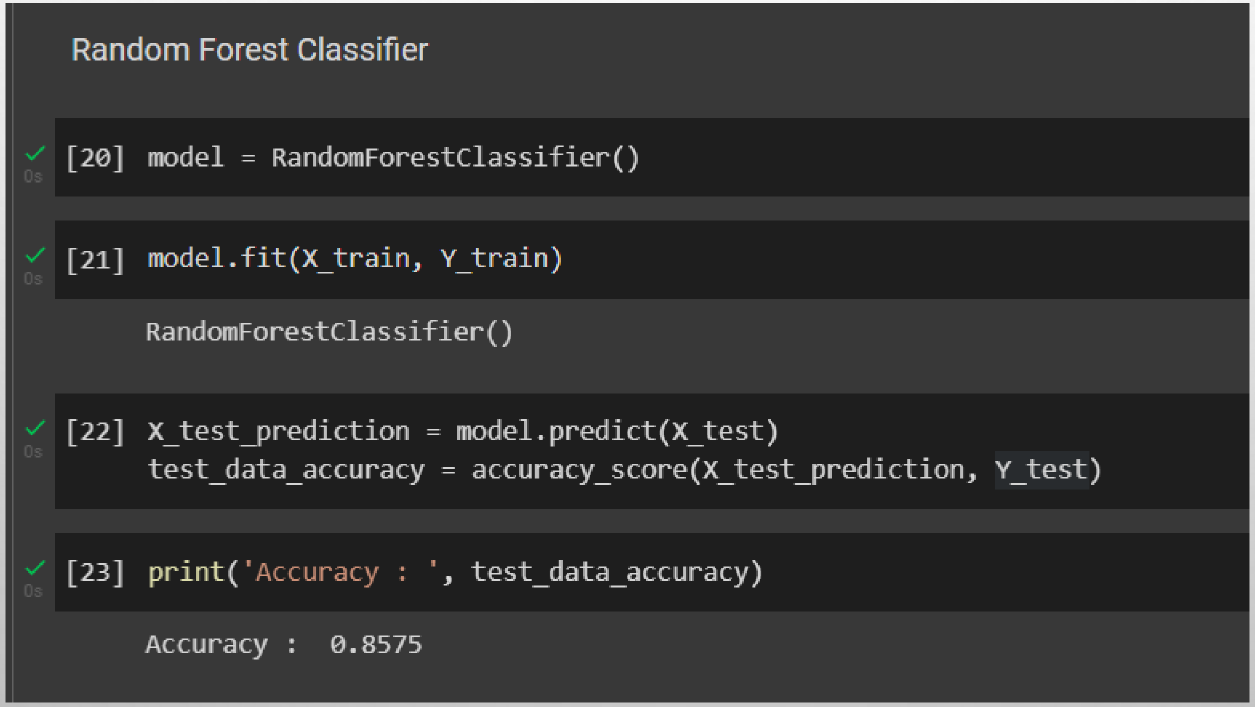
```
[15]  X = Mobile_pc.drop('price_range',axis=1)
```



```
[17]  Y = Mobile_pc['price_range']
```

RANDOM FOREST CLASSIFIER

WE USED RANDOM FOREST CLASSIFIER TO CLASSIFY THE DATA INTO THE PRICE RANGE AND CALCULATED ITS ACCURACY.



Random Forest Classifier

```
[20] model = RandomForestClassifier()
```

```
[21] model.fit(X_train, Y_train)

     RandomForestClassifier()
```

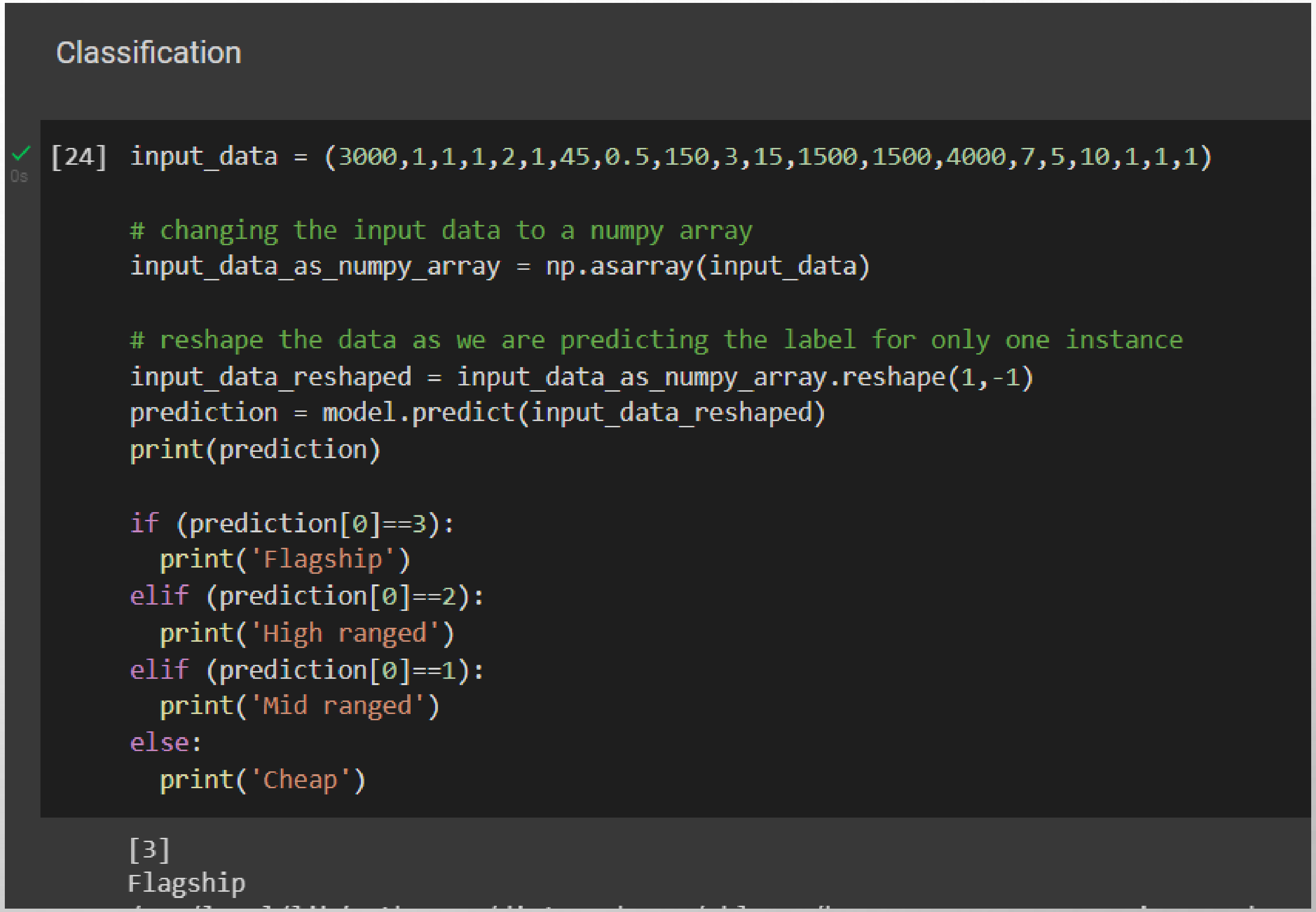
```
[22] X_test_prediction = model.predict(X_test)
     test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
[23] print('Accuracy : ', test_data_accuracy)

     Accuracy :  0.8575
```

CLASSIFICATION

THEN WE INPUT DATA WHICH WAS SUITABLE FOR A FLAGSHIP DEVICE, AND RECEIVED THE EXPECTED RESULT.



Classification

```
[24] input_data = (3000,1,1,1,2,1,45,0.5,150,3,15,1500,1500,4000,7,5,10,1,1,1)

     # changing the input data to a numpy array
     input_data_as_numpy_array = np.asarray(input_data)

     # reshape the data as we are predicting the label for only one instance
     input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
     prediction = model.predict(input_data_reshaped)
     print(prediction)

     if (prediction[0]==3):
       print('Flagship')
     elif (prediction[0]==2):
       print('High ranged')
     elif (prediction[0]==1):
       print('Mid ranged')
     else:
       print('Cheap')
```

```
[3]
Flagship
```

THANK YOU