y-of-mobile-price-range-prediction

August 4, 2024

[1]:
```
from google.colab import drive
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
df=pd.read_csv('/content/data_mobile_price_range.csv')
df.head()
```

[3]:
```
   battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  m_dep  \
0            842     0          2.2         0   1       0           7    0.6
1           1021     1          0.5         1   0       1          53    0.7
2            563     1          0.5         1   2       1          41    0.9
3            615     1          2.5         0   0       0          10    0.8
4           1821     1          1.2         0  13       1          44    0.6

   mobile_wt  n_cores  ...  px_height  px_width   ram  sc_h  sc_w  talk_time  \
0        188        2  ...         20       756  2549     9     7         19
1        136        3  ...        905      1988  2631    17     3          7
2        145        5  ...       1263      1716  2603    11     2          9
3        131        6  ...       1216      1786  2769    16     8         11
4        141        2  ...       1208      1212  1411     8     2         15

   three_g  touch_screen  wifi  price_range
0        0             0     1            1
1        1             1     0            2
2        1             1     0            2
3        1             0     0            2
4        1             1     0            1

[5 rows x 21 columns]
```

[4]:
```
df.shape
```

[4]:
```
(2000, 21)
```

[5]:
```
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
```

checking for duplicate values

```
[6]: duplicated_values_count=len(df[df.duplicated()])
     duplicated_values_count
```

```
[6]: 0
```

checking for null values

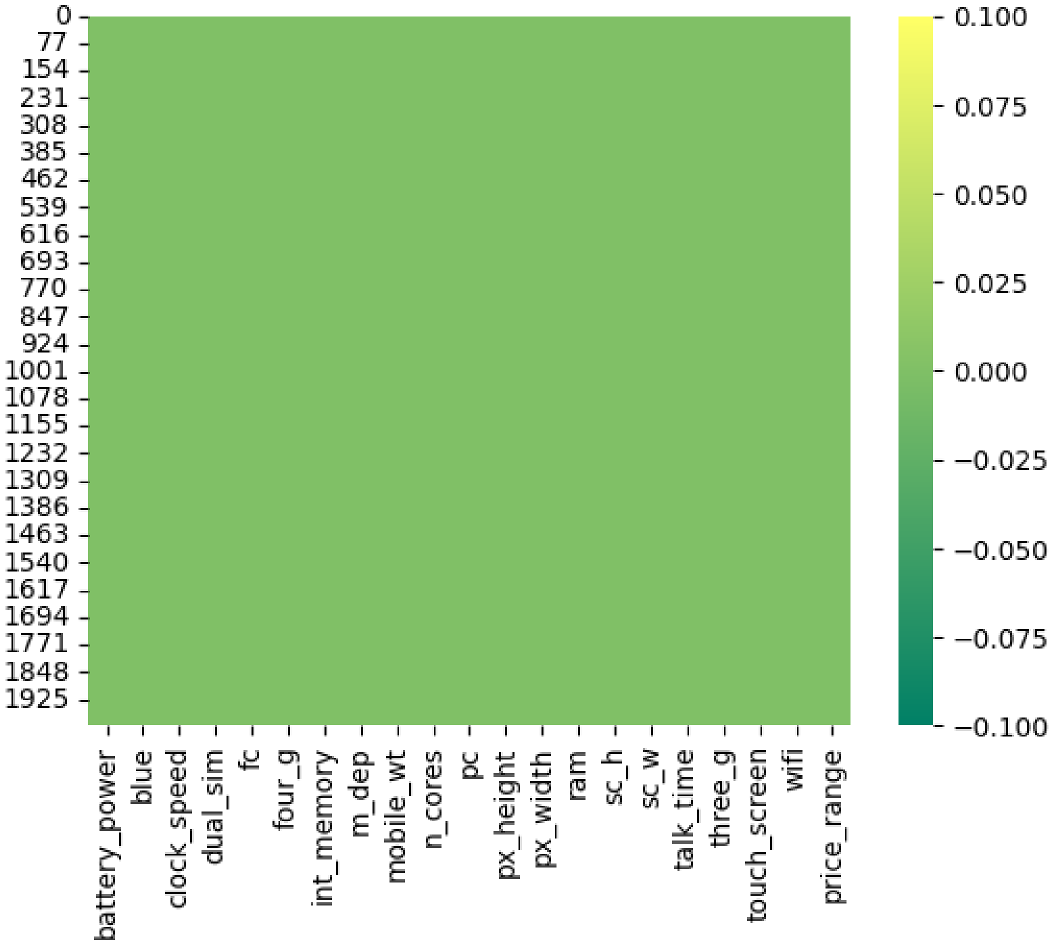
```
[7]: df.isnull().sum()
```

```
[7]: 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
```

[8]:

```
import seaborn as sns
x=sns.heatmap(df.isnull(),cmap='summer',cbar=True)
```



[9]:
```
df.columns
```

[9]:
```
Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g',
       'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
       'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
       'touch_screen', 'wifi', 'price_range'],
      dtype='object')
```

[10]:
```
len(df.columns)
```

[10]:
```
21
```

[11]:
```
df.describe()
```

[11]:
```
       battery_power       blue  clock_speed     dual_sim           fc  \
count    2000.000000  2000.0000  2000.000000  2000.000000  2000.000000
mean     1238.518500     0.4950     1.522250     0.509500     4.309500
std       439.418206     0.5001     0.816004     0.500035     4.341444
min       501.000000     0.0000     0.500000     0.000000     0.000000
25%       851.750000     0.0000     0.700000     0.000000     1.000000
50%      1226.000000     0.0000     1.500000     1.000000     3.000000
75%      1615.250000     1.0000     2.200000     1.000000     7.000000
max      1998.000000     1.0000     3.000000     1.000000    19.000000

            four_g   int_memory        m_dep    mobile_wt      n_cores  …  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  …
mean      0.521500    32.046500     0.501750   140.249000     4.520500  …
std       0.499662    18.145715     0.288416    35.399655     2.287837  …
min       0.000000     2.000000     0.100000    80.000000     1.000000  …
25%       0.000000    16.000000     0.200000   109.000000     3.000000  …
50%       1.000000    32.000000     0.500000   141.000000     4.000000  …
75%       1.000000    48.000000     0.800000   170.000000     7.000000  …
max       1.000000    64.000000     1.000000   200.000000     8.000000  …

         px_height     px_width          ram         sc_h         sc_w  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean    645.108000  1251.515500  2124.213000    12.306500     5.767000
std     443.780811   432.199447  1084.732044     4.213245     4.356398
min       0.000000   500.000000   256.000000     5.000000     0.000000
25%     282.750000   874.750000  1207.500000     9.000000     2.000000
50%     564.000000  1247.000000  2146.500000    12.000000     5.000000
75%     947.250000  1633.000000  3064.500000    16.000000     9.000000
max    1960.000000  1998.000000  3998.000000    19.000000    18.000000

         talk_time      three_g  touch_screen         wifi  price_range
count  2000.000000  2000.000000   2000.000000  2000.000000  2000.000000
mean     11.011000     0.761500      0.503000     0.507000     1.500000
```

```
std       5.463955    0.426273     0.500116    0.500076    1.118314
min       2.000000    0.000000     0.000000    0.000000    0.000000
25%       6.000000    1.000000     0.000000    0.000000    0.750000
50%      11.000000    1.000000     1.000000    1.000000    1.500000
75%      16.000000    1.000000     1.000000    1.000000    2.250000
max      20.000000    1.000000     1.000000    1.000000    3.000000

[8 rows x 21 columns]
```

```
[12]: df.nunique()
```

```
[12]: battery_power    1094
      blue                2
      clock_speed        26
      dual_sim            2
      fc                 20
      four_g              2
      int_memory         63
      m_dep              10
      mobile_wt         121
      n_cores             8
      pc                 21
      px_height        1137
      px_width         1109
      ram              1562
      sc_h               15
      sc_w               19
      talk_time          19
      three_g             2
      touch_screen        2
      wifi                2
      price_range         4
      dtype: int64
```

```
[13]: sc_w_zero_count = sum(df.sc_w==0)
      print(f"number of phones with screen width=0: {sc_w_zero_count}")
```

```
number of phones with screen width=0: 180
```

```
[14]: px_height_zero_count=sum(df.px_height==0)
      print(f"number of phones with pixel height=0 :{px_height_zero_count}")
```

```
number of phones with pixel height=0 :2
```

replacing zero with mean values

```
[15]: sc_w_mean=df.sc_w.mean()
      np.where(df.sc_w==0, sc_w_mean,df.sc_w)
```

```python
print(df)
```

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

      m_dep  mobile_wt  n_cores  …  px_height  px_width   ram  sc_h  sc_w  \
0       0.6        188        2  …         20       756  2549     9     7
1       0.7        136        3  …        905      1988  2631    17     3
2       0.9        145        5  …       1263      1716  2603    11     2
3       0.8        131        6  …       1216      1786  2769    16     8
4       0.6        141        2  …       1208      1212  1411     8     2
…     …        …      …  …        …       …   …   …   …
1995    0.8        106        6  …       1222      1890   668    13     4
1996    0.2        187        4  …        915      1965  2032    11    10
1997    0.7        108        8  …        868      1632  3057     9     1
1998    0.1        145        5  …        336       670   869    18    10
1999    0.9        168        6  …        483       754  3919    19     4

      talk_time  three_g  touch_screen  wifi  price_range
0            19        0             0     1            1
1             7        1             1     0            2
2             9        1             1     0            2
3            11        1             0     0            2
4            15        1             1     0            1
…           …      …           …   …          …
1995         19        1             1     0            0
1996         16        1             1     1            2
1997          5        1             1     0            3
1998         19        1             1     1            0
1999          2        1             1     1            3

[2000 rows x 21 columns]
```

[16]:

```python
px_height_mean=df.px_height.mean()
np.where(df.px_height==0,px_height_mean,df.px_height)
print(df)
```

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

      m_dep  mobile_wt  n_cores  …  px_height  px_width   ram  sc_h  sc_w  \
0       0.6        188        2  …         20       756  2549     9     7
1       0.7        136        3  …        905      1988  2631    17     3
2       0.9        145        5  …       1263      1716  2603    11     2
3       0.8        131        6  …       1216      1786  2769    16     8
4       0.6        141        2  …       1208      1212  1411     8     2
…     …        …      …  …        …       …   …   …   …
1995    0.8        106        6  …       1222      1890   668    13     4
1996    0.2        187        4  …        915      1965  2032    11    10
1997    0.7        108        8  …        868      1632  3057     9     1
1998    0.1        145        5  …        336       670   869    18    10
1999    0.9        168        6  …        483       754  3919    19     4

      talk_time  three_g  touch_screen  wifi  price_range
0            19        0             0     1            1
1             7        1             1     0            2
2             9        1             1     0            2
3            11        1             0     0            2
4            15        1             1     0            1
…         …      …           …   …          …
1995         19        1             1     0            0
1996         16        1             1     1            2
1997          5        1             1     0            3
1998         19        1             1     1            0
1999          2        1             1     1            3

[2000 rows x 21 columns]
```

[17]:
```
len(df[df.duplicated()])
```

[17]:
```
0
```

[18]:
```
df.drop_duplicates(subset='battery_power',keep='last')
```

[18]:

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
5              1859     0          0.5         1   3       0          22
9               509     1          0.6         1   2       1           9
12             1815     0          2.8         0   2       0          33
16              838     0          0.5         0   1       1          13
18             1131     1          0.5         1  11       0          49
…             …   …          …       …  ..     …         …
1995            794     1          0.5         1   0       1           2
1996           1965     1          2.6         1   0       0          39
1997           1911     0          0.9         1   1       1          36
1998           1512     0          0.9         0   4       1          46
1999            510     1          2.0         1   5       1          45

      m_dep  mobile_wt  n_cores  …  px_height  px_width   ram  sc_h  sc_w  \
5       0.7        164        1  …       1004      1654  1067    17     1
9       0.1         93        5  …       1137      1224   513    19    10
12      0.6        159        4  …        607       748  1482    18     0
16      0.1        196        8  …        984      1850  3554    10     9
18      0.6        101        5  …        658       878  1835    19    13
…     …        …      …  …        …       …   …   …   …
1995    0.8        106        6  …       1222      1890   668    13     4
1996    0.2        187        4  …        915      1965  2032    11    10
1997    0.7        108        8  …        868      1632  3057     9     1
1998    0.1        145        5  …        336       670   869    18    10
1999    0.9        168        6  …        483       754  3919    19     4

      talk_time  three_g  touch_screen  wifi  price_range
5            10        1             0     0            1
9            12        1             0     0            0
12            2        1             0     0            1
16           19        1             0     1            3
18           16        1             1     0            1
…         …      …           …   …          …
1995         19        1             1     0            0
1996         16        1             1     1            2
1997          5        1             1     0            3
1998         19        1             1     1            0
1999          2        1             1     1            3

[1094 rows x 21 columns]
```

[ ]:

```
df.isnull()
```

[ ]:

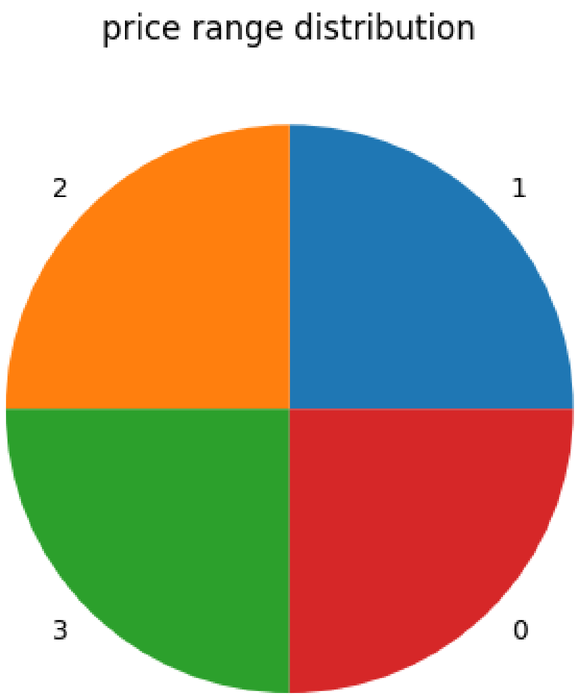
```
      battery_power   blue  clock_speed  dual_sim     fc  four_g  int_memory  \
0             False  False        False     False  False   False       False
1             False  False        False     False  False   False       False
2             False  False        False     False  False   False       False
```

```
3             False  False        False     False  False   False       False
4             False  False        False     False  False   False       False
…             …    …          …       …    …     …         …
1995          False  False        False     False  False   False       False
1996          False  False        False     False  False   False       False
1997          False  False        False     False  False   False       False
1998          False  False        False     False  False   False       False
1999          False  False        False     False  False   False       False

      m_dep  mobile_wt  n_cores  …  px_height  px_width    ram   sc_h  \
0     False      False    False  …      False     False  False  False
1     False      False    False  …      False     False  False  False
2     False      False    False  …      False     False  False  False
3     False      False    False  …      False     False  False  False
4     False      False    False  …      False     False  False  False
…     …        …      …  …        …       …    …    …
1995  False      False    False  …      False     False  False  False
1996  False      False    False  …      False     False  False  False
1997  False      False    False  …      False     False  False  False
1998  False      False    False  …      False     False  False  False
1999  False      False    False  …      False     False  False  False

       sc_w  talk_time  three_g  touch_screen   wifi  price_range
0     False      False    False         False  False        False
1     False      False    False         False  False        False
2     False      False    False         False  False        False
3     False      False    False         False  False        False
4     False      False    False         False  False        False
…     …        …      …           …    …          …
1995  False      False    False         False  False        False
1996  False      False    False         False  False        False
1997  False      False    False         False  False        False
1998  False      False    False         False  False        False
1999  False      False    False         False  False        False

[2000 rows x 21 columns]
```

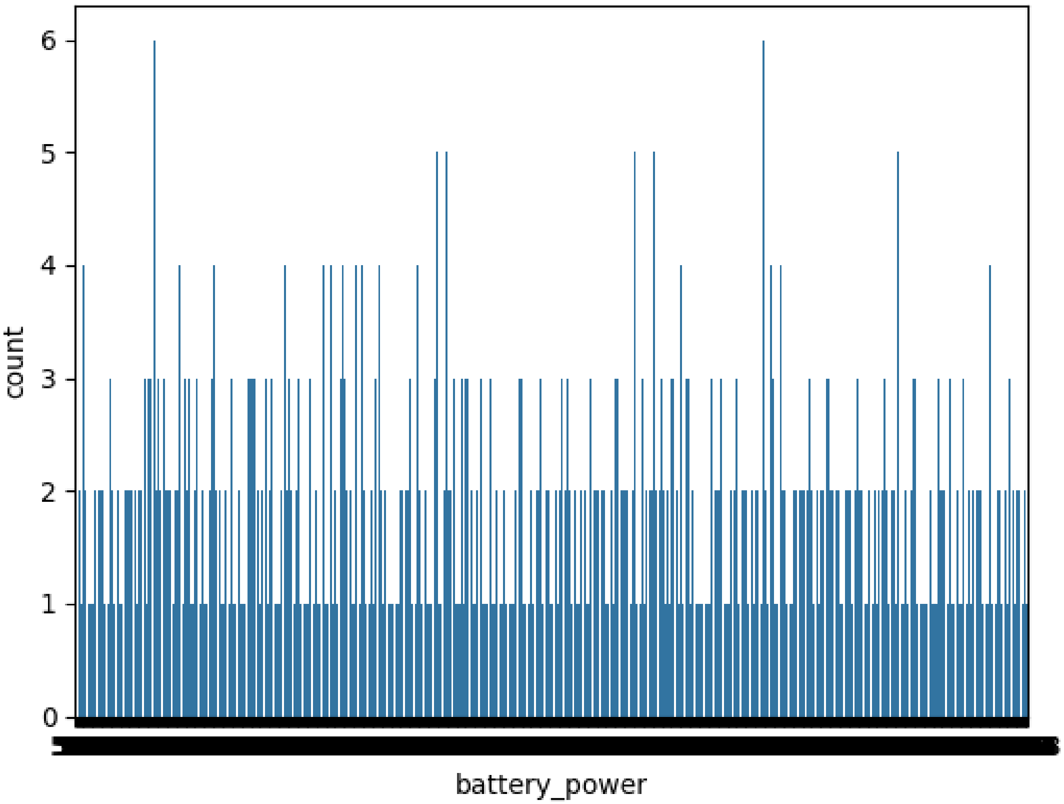
[ ]:
```
df.sc_w.unique()
```

[ ]:
```
array([ 7,  3,  2,  8,  1, 10,  9,  0, 15, 13,  5, 11,  4, 12,  6, 17, 14,
       16, 18])
```

[19]:
```
x=df['price_range'].value_counts()
plt.pie(x,labels=x.index)
plt.title('price range distribution')
plt.show()
```



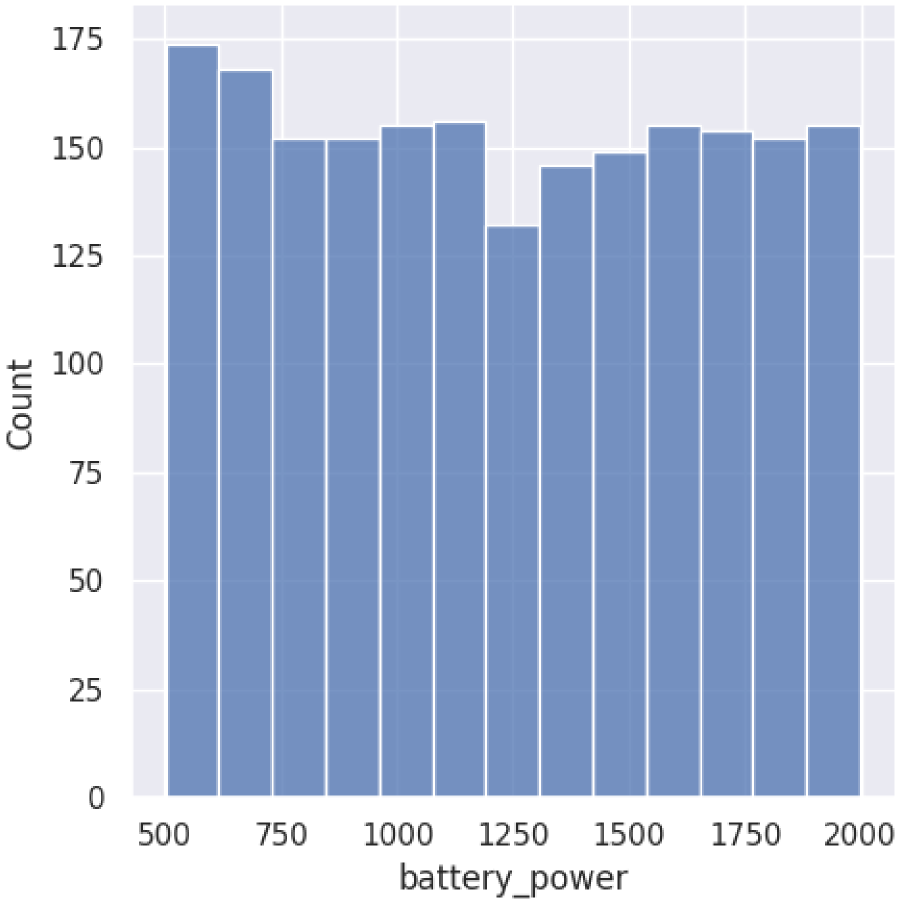
```
[20]: sns.countplot(x='battery_power',data=df)
      plt.figure(figsize=(20,20))
      plt.show()
```



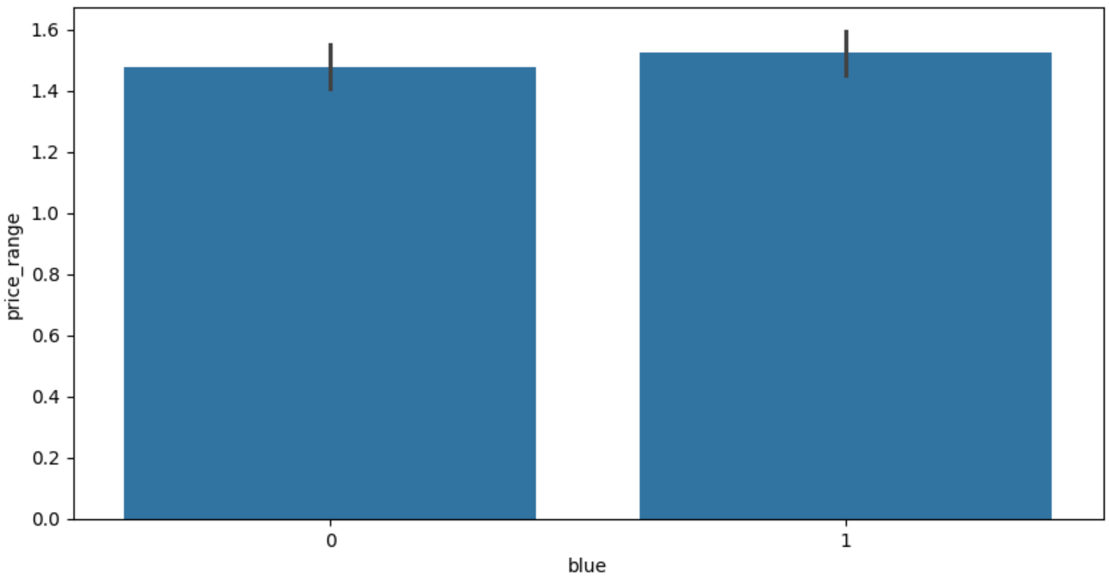
```
<Figure size 2000x2000 with 0 Axes>
```

as the axis level is not adjusting so we have to set the axis level

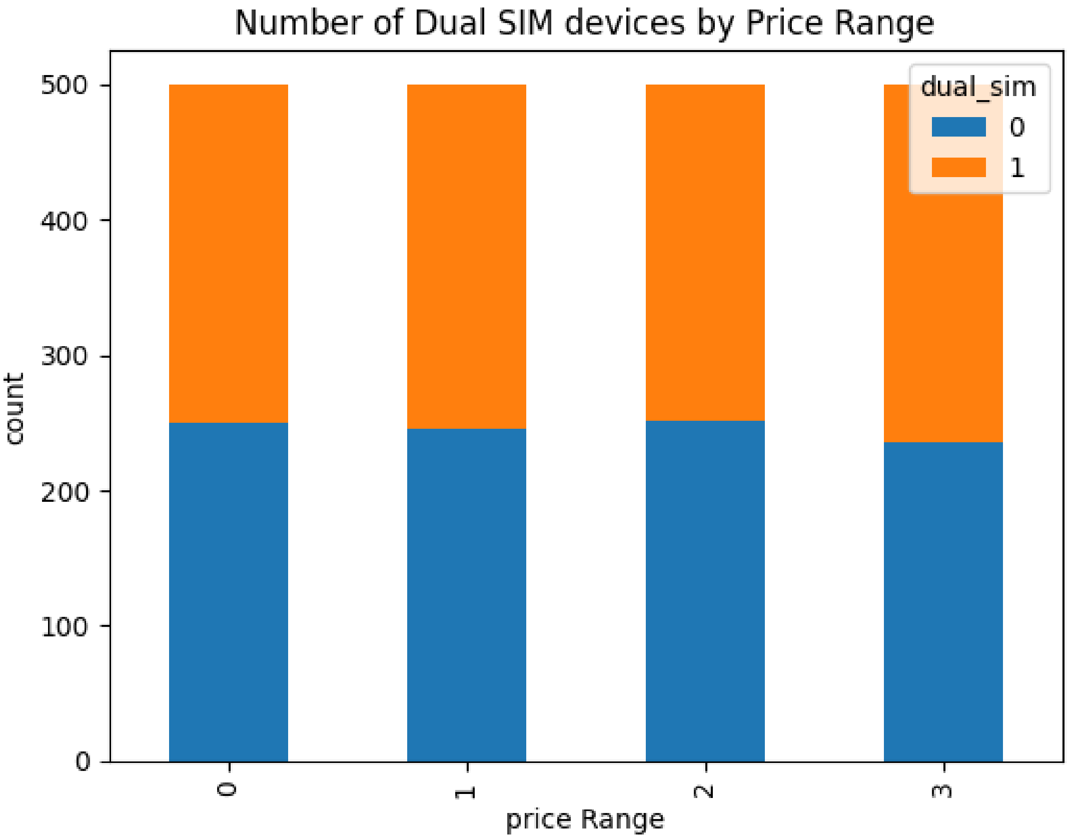
```
[ ]: sns.set(rc={'figure.figsize':(3,4)})
     sns.displot(df['battery_power'])
     plt.show()
```



[23]:
```
fig, ax=plt.subplots(figsize=(10,5))
sns.barplot(data=df,x='blue',y='price_range', ax=ax)
plt.show()
```



[25]:
```
sim_count=df.groupby(['price_range','dual_sim'])['dual_sim'].count()
sim_count=sim_count.unstack()
sim_count.plot(kind='bar',stacked=True)
plt.xlabel('price Range')
plt.ylabel('count')
plt.title('Number of Dual SIM devices by Price Range')
plt.show()
```



model selection

```
[27]: from sklearn.model_selection import train_test_split
      x=df.drop(['price_range'],axis=1)
      y=df['price_range']
      x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=5)
```

```
[ ]: print(x_train)
```

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
836             902     1          0.6         1   0       0          63
755            1018     1          0.7         1   7       0          63
138             536     0          2.4         1  12       1           3
61              799     1          2.3         0   1       1          63
384             625     1          1.9         0  12       1          33
…             …   …          …       …   ..    …          …
1142           1193     1          3.0         0  10       0          56
998            1373     1          1.9         1   1       1          29
1725           1117     1          0.5         1   2       0          21
206            1642     0          0.5         1  16       1           8
867            1498     1          0.7         0   3       1           8
```

```
      m_dep  mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  \
836     0.7        122        5  14        364      1360  3654    18     8
755     0.1        155        5  18        856       883  3048    10     3
138     0.3        182        7  14       1386      1539   284    12     9
61      0.8        144        8   6        361       975   431    15     6
384     0.2        191        1  20        431       550  3801    10     6
…     …        …      …  ..        …       …   …   …   …
1142    0.4        196        3  17        674       864  2394    19    11
998     0.9        141        6  12       1220      1348  2752    15     2
1725    0.1        177        2  19        495      1035  1999    15     9
206     0.3        171        6  17        129       873  2984    13     4
867     0.1        170        7   4        347      1076  3358     7     3

      talk_time  three_g  touch_screen  wifi
836          15        0             1     1
755           2        0             0     1
138           4        1             1     0
61            6        1             1     1
384           2        1             0     0
…         …      …           …   …
1142         14        1             1     0
998           7        1             1     1
1725          2        1             0     1
206          17        1             0     1
867          19        1             0     0

[1400 rows x 20 columns]
```

```
[ ]: print(y_train)
```

```
836     3
755     2
138     0
61      0
384     2
       ..
1142    2
998     3
1725    1
206     2
867     3
Name: price_range, Length: 1400, dtype: int64
```

```
[ ]: len(x_train)
```

```
[ ]: 1400
```

[ ]:
```
len(x_test)
```

[ ]:
```
600
```

[28]:
```
x.shape
```

[28]:
```
(2000, 20)
```

[29]:
```
y.shape
```

[29]:
```
(2000,)
```

[ ]:
```

```

[ ]:
```
from sklearn.linear_model import LinearRegression
mymodel=LinearRegression()
```

[ ]:
```
mymodel.fit(x_train,y_train) #train the model
```

[ ]:
```
LinearRegression()
```

[ ]:
```
pred=mymodel.predict(x_test)
pred
```

[ ]:
```
array([ 1.96240214,  2.60074578,  2.40573903,  0.19204357,  1.19772059,
        0.80854306,  1.77986553,  3.33576508,  1.69238379,  0.72888059,
        2.4171309 , -0.05496171,  2.5868986 ,  3.2618825 ,  2.34389635,
        0.02772633,  1.32983697,  1.96785926,  0.93654089,  3.27450501,
       -0.45533469,  0.8391667 ,  2.97399029,  1.38449747,  2.05315665,
        1.61542494,  1.95230803,  3.27051741,  0.15051929,  1.43100422,
        0.80134992,  1.72323133,  0.22658088,  0.25272216,  3.6074405 ,
        2.10296606,  0.6430281 ,  3.70113276,  2.18401473, -0.30025608,
        2.80030464,  0.71163586,  2.4803529 ,  1.77739622,  3.18459357,
        2.96582174, -0.10294317,  0.12563189,  1.64716439,  1.61069819,
        2.28734963,  2.76196938,  2.194089  ,  1.46073206,  1.43181693,
        0.21420486,  0.85664362, -0.29793003,  1.77671055,  0.97874936,
        0.05065619,  0.10200921,  2.3480865 ,  2.72749569,  0.68597355,
        1.42793596,  1.86477221,  2.33901408,  1.86103147,  1.14164384,
        1.87479806,  2.34989719, -0.06511619, -0.43916558,  0.06839285,
        2.11443871,  1.59752415,  0.66475027,  0.51056126,  2.86489559,
        1.32130925,  3.50100427,  1.152135  ,  0.90520464,  1.21712201,
        1.37241326,  0.71871659,  1.02591802,  2.43390904,  1.17605274,
        3.23358056,  0.00686964,  2.75522249,  3.17592344,  2.25050164,
        1.88741619,  1.0122828 ,  1.84377731,  3.66538144,  2.89652308,
        2.96059196,  1.20052236,  0.92103534,  0.17478779,  2.79480086,
        2.48769142,  0.48624651,  1.85716201,  0.06963057,  3.12774183,
       -0.06127111,  1.99520549, -0.17615302,  0.33496113,  1.34725936,
 1.1738657 ,  1.48872871,  2.48772222,  2.46120017,  0.00436287,
 1.23645861,  1.84976831,  1.08409044,  0.20152922,  3.73352728,
 2.33051246,  0.14403288,  0.25809646,  0.9238448 ,  2.61005407,
 1.89758471,  2.31329123,  1.97844955, -0.02972142,  1.01305441,
-0.05461355,  1.88779611,  0.51350184,  0.06814574,  1.74577417,
 1.38829296,  2.92277928,  0.24390816,  1.32312465,  2.69685353,
 1.39190487, -0.33322304, -0.12967106,  0.19688851,  1.49549226,
 1.28967389,  0.86944002,  1.4767795 ,  0.23562871,  1.91621725,
 1.7933708 ,  1.96703165,  0.88811906, -0.00506223,  3.1510289 ,
 3.02402766,  3.12744791,  1.85897184,  1.43772388,  1.12289347,
 0.44020657,  1.10293123, -0.16143861, -0.11884886,  1.60079432,
 2.32875176,  1.60986186,  1.75298446, -0.2006015 ,  1.69621312,
 2.38180836,  0.17545245,  1.80274696,  2.68580333,  0.25409576,
-0.07633703, -0.0392554 ,  2.24277037,  1.8177244 ,  2.48816776,
 1.90063621,  3.08451283,  1.21167642,  2.59116616,  2.11127835,
 2.70057218,  1.09815055,  0.91658283,  1.38944749,  3.19998077,
 1.80795925,  2.14899943,  1.7323144 ,  0.03099085,  1.47839328,
 2.24607951,  0.39955288,  3.43960768,  0.27074933,  1.75111024,
 3.15650583,  3.25298754,  2.61830883,  1.21398832,  0.22396931,
 0.95457512,  3.03759253,  1.31365828,  2.67925658,  1.20444878,
 1.19330976,  2.33889721,  0.77906702,  1.23575963,  2.37791197,
 3.13393321,  0.07494136,  2.70604083,  1.88676506, -0.40400064,
 1.71379073,  1.19652241,  1.46079793,  0.42542039,  2.92482007,
 2.6709246 ,  1.525536  ,  1.51148093,  2.94162625,  1.81462503,
 0.7136394 ,  2.12542763,  1.2372093 ,  0.61032848,  2.42386267,
-0.15636565,  0.99387707,  1.54431242,  0.51458293,  0.57315032,
 0.50543212,  2.10480207,  1.7554677 , -0.56690115, -0.27117854,
 2.46598655,  2.99760754, -0.20597562,  2.16515189,  2.35673744,
 0.6358538 ,  0.20922404,  3.36118811,  2.92787035,  1.88113962,
 3.15203483,  0.42015666,  1.94431031,  1.02988142,  0.18925985,
 2.14276666,  3.31590267,  2.67496199,  2.48977921,  2.51579146,
 0.06333655,  1.12935891,  0.8750562 ,  1.63070024,  2.15694639,
 0.42300687,  3.02236597,  0.15550044,  2.76563874,  2.87902856,
 1.73048125,  0.13332702,  0.79179787,  1.03913034,  0.97665161,
 3.48112295,  2.65505758,  2.74870834, -0.28049619,  0.27921579,
 1.11242513,  1.53739158,  1.30006749,  0.42638552,  1.14671559,
 0.82111047,  0.26206283,  0.07444301,  2.47660414,  3.21665954,
 2.64954965,  3.24970757,  2.2750735 ,  0.18530455,  2.41654514,
 3.62864152,  0.2575017 ,  3.38442977,  0.91983888,  1.39254816,
 2.95397197,  2.41956924, -0.09689264,  0.74767631,  0.0847655 ,
 0.34174419,  2.20784283,  0.0186711 ,  2.09721587,  2.99933458,
-0.08482084,  0.28052069,  0.33974123,  1.44364003,  1.47132876,
 2.07888843,  2.35051134,  1.39718132,  1.4341576 ,  1.30074889,
 1.3558278 ,  2.70251232,  1.75395907,  1.84181395, -0.10957711,
 0.17415347,  0.20643558,  0.50513386,  0.2950989 ,  1.36330578,
 3.33360119,  0.51391821,  2.25862356,  1.39229374,  1.89183797,
 0.06265464,  0.97658955,  1.68830526,  0.22100906,  1.55271435,
 0.25678731,  2.3990872 ,  1.84881576,  2.92876057,  1.39503557,
 0.1497577 ,  0.72765784,  0.06285433,  2.97322368,  2.75465459,
 1.5257041 ,  2.50338403,  3.29655225, -0.34512406,  2.03013966,
 2.55016147, -0.05186488,  1.4003869 ,  1.8883466 ,  1.46010961,
 0.58677277, -0.03534526,  3.20020218,  2.50347785,  1.68592987,
-0.09918669,  0.22789138,  3.3358754 ,  1.9586939 ,  2.0654349 ,
 2.34333747,  0.93057163,  3.59279096,  3.46154707,  2.89713136,
 1.44082216,  3.28815155, -0.12913616,  0.56653495,  0.23594097,
 3.35520994,  1.84511137,  1.2513283 ,  1.60564881,  0.53562565,
 2.47536017,  2.07669331,  2.71351737,  1.05486243, -0.66323774,
 2.37711088,  0.38884336,  2.81626331,  0.5577853 ,  0.64835005,
 1.52003786, -0.09614317,  0.08081814,  0.07783023,  1.35077002,
 0.97069596,  0.78064501,  0.62853055,  1.71060631,  0.22264679,
 1.57498972,  2.90987713, -0.01063422,  3.3524179 ,  1.78014504,
 1.35737621, -0.39024145,  0.62191351,  1.20631688,  1.87509925,
 1.23629328,  2.62733128,  2.23359607,  2.14207592,  0.79622843,
 0.29314663,  1.61997698,  1.99192821,  2.62189347,  0.68478284,
 3.13584219,  0.62945746,  3.09729729,  0.53719772,  3.42984721,
 0.53438088,  3.53169663,  0.81795947,  0.91185123,  2.56625724,
 1.70128873,  2.0556729 ,  2.26231581,  2.3633309 ,  0.16289753,
 1.86063712,  1.1655461 , -0.16779175,  1.83880662,  0.66344892,
 0.29503011,  1.84636403,  2.88222147,  1.50409019,  3.15814245,
 1.96603966,  2.31841055, -0.01997589,  0.43636662,  1.41291136,
 0.00895026,  2.25735425,  1.03993205,  2.46722117,  1.90954188,
 2.76217224,  2.05972323,  0.48350629,  1.75514225,  0.64428077,
 3.8163736 ,  1.23132385,  3.75865719,  1.52315104,  1.75563036,
-0.27047398,  0.54976509,  2.51805287,  0.85037407,  0.71285446,
 0.41330814,  0.10344961,  1.46341666,  1.85790143,  3.59159702,
 1.43716126,  1.61298436,  1.13413072,  0.17069539,  0.83648365,
 0.41116714,  2.43988087,  0.94112223, -0.59272411,  2.20672975,
 1.10557322,  3.46074715,  3.71815454,  2.21945973,  2.68548591,
 2.93513798,  0.19734363,  1.73288013,  1.17666726, -0.26480213,
 0.08841615,  1.35423958,  1.57505857,  3.36257729,  1.99128216,
 3.39278593,  1.78118172,  2.54417849,  0.55079145,  0.25612392,
 0.23132966,  2.57304528,  2.93923935,  1.99364907,  0.23852986,
 1.12341173,  0.13511232,  1.42503802,  0.31870563,  1.22569441,
 2.60393388,  0.28194056,  2.57105315,  3.07228563,  2.45848086,
 0.76342541,  1.87094744,  0.96081344,  0.87174811,  1.66209852,
-0.55871462, -0.04158233,  0.47488916,  0.6092443 ,  1.25272379,
 2.00221118,  0.71443655,  0.65933665,  1.95338515,  3.11158333,
 0.23936365,  2.3466925 ,  0.47637854,  2.67392422,  1.2269335 ,
 0.98206884,  2.54039487,  1.20788734,  2.2548539 ,  0.89787576,
 1.32273829,  0.66394799,  2.38007525,  2.49845745,  0.74528839,
 2.87080749,  0.98774044,  1.80843311,  2.1137098 ,  1.48963575,
 1.35504687, -0.13293906, -0.50999473,  2.93571858,  0.98515882,
 1.96088995,  3.45408853,  1.65568998,  2.2059907 ,  1.28329948,
 2.23526301,  2.14771962,  1.02574437,  0.24494396,  3.45151146,
-0.04236978,  1.20534572,  0.3435824 ,  2.58798185,  1.93311682,
 1.36354525,  1.23825585,  0.18815686,  2.99904797,  0.4270519 ,
 2.44926863,  0.943152  ,  2.99786033,  1.14556284, -0.66171321])
```

[ ]:
```
y_test
```

[ ]:
```
51      2
1327    3
1488    3
1432    0
417     1
       ..
1907    3
1075    1
1596    3
1280    1
1378    0
Name: price_range, Length: 600, dtype: int64
```

[ ]:
```
mymodel.score(x_test,y_test)
```

[ ]:
```
0.9188050447010625
```

[32]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.metrics import accuracy_score,classification_report
```

[ ]:
```
print('R2:',metrics.r2_score(y_test,pred))
```

```
R2: 0.9188050447010625
```

[30]:
```
from sklearn.ensemble import RandomForestClassifier
clsr=RandomForestClassifier(n_estimators=300)
clsr.fit(x_train,y_train)
```

[30]:
```
RandomForestClassifier(n_estimators=300)
```

[35]:
```
y_pred=clsr.predict(x_test)
test_score= accuracy_score(y_test,y_pred)
test_score
```

[35]:
```
0.8833333333333333
```

[37]:
```
y_pred_train=clsr.predict(x_train)
train_score=accuracy_score(y_train,y_pred_train)
train_score
```

```
[37]: 1.0
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```