Project 4 - Mobile Price Analysis

Modules needed: Numpy,pandas,matplotlib,seaborn.

#import the needed libraries: import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt

#Loading the mobile dataset file:
df= pd.read_csv("/content/train.csv")

print(df)

0 1 2 3 4 	battery_power 842 1021 563 615 1821	2 0 1 1 3 1 5 1 1 1	clock	2.2 0.5 0.5 2.5 1.2	dual_sim 0 1 1 0 0	1 0 2 0 0 13 	0 1 1 0 1		7 53 41 10 44 	
1996 1997 1998 1999	1965 1911 1512 510	0 0		2.6 0.9 0.9 2.0	1 1 0 1	1 4	0 1 1 1		39 36 46 45	
0 1 2 3 4	m_dep mobile 0.6 0.7 0.9 0.8 0.6	e_wt n_c 188 136 145 131 141	2 3 5 6 2	p	x_height 20 905 1263 1216 1208	px_width 756 1988 1716 1786 1212	ram 2549 2631 2603 2769 1411	sc_h 9 17 11 16 8	sc_w 7 3 2 8 2	\
1995 1996 1997 1998 1999	0.8 0.2 0.7 0.1 0.9	106 187 108 145 168	6 4 8 5 6		1222 915 868 336 483	1890 1965 1632 670 754	668 2032 3057 869 3919	13 11 9 18 19	10 10 10 4	
0 1 2 3 4 1995 1996 1997 1998	talk_time the 19	nree_g 1 0 1 1 1 1 1 1 1 1 1	touch_	screen 0 1 1 0 1 1 1 1	wifi pr 1 0 0 0 0 0 1 0 1 1	ice_range 1 2 2 2 1 0 2 3 0				

[2000 rows x 21 columns]

#First 5 rows of the data: df.head()

$\overline{\Rightarrow}$	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

#Last 5 rows of the data:

5 rows × 21 columns

df.tail()

→		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_rang	ξ
•	1995	794	1	0.5	1	0	1	2	0.8	106	6		1222	1890	668	13	4	19	1	1	0		0
•	1996	1965	1	2.6	1	0	0	39	0.2	187	4		915	1965	2032	11	10	16	1	1	1		2
	1997	1911	0	0.9	1	1	1	36	0.7	108	8		868	1632	3057	9	1	5	1	1	0	;	3
	1998	1512	0	0.9	0	4	1	46	0.1	145	5		336	670	869	18	10	19	1	1	1	1	ζ
	1999	510	1	2.0	1	5	1	45	0.9	168	6		483	754	3919	19	4	2	1	1	1		3

5 rows × 21 columns

#Shape of the dataset: df.shape

→ (2000, 21)

#Description of the dataset:
df.describe()

\Rightarrow		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	ta
	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	2000.000000	2000.000000	200
	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	12.306500	5.767000	1
	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	4.213245	4.356398	
	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	5.000000	0.000000	
	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	9.000000	2.000000	
	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	12.000000	5.000000	1
	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	16.000000	9.000000	1
	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	19.000000	18.000000	2

8 rows × 21 columns

#Cleaning the data for missing values and null values:
#Checking for null values:
df.isnull().sum()

```
battery_power 0
          blue
                  0
       clock_speed 0
        dual_sim
                  0
                  0
          fc
                  0
         four_g
                  0
       int_memory
         m_dep
                  0
                  0
        mobile_wt
                  0
        n_cores
                  0
          рс
        px_height
                  0
                  0
        px_width
                  0
          ram
                  0
          sc_h
                  0
         sc_w
                  0
        talk_time
                  0
         three_g
      touch_screen 0
          wifi
                  0
       price_range 0
     dtype: int64
#Checking for missing and null values:
null_df = df[df.isna().any(axis=1)]
null_df
 \overline{\mathbf{x}}
        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 rows × 21 columns
#Checking for missing and null values:
df.isnull().any().any()
 → False
#Info of the dataset:
df.info()
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 2000 entries, 0 to 1999
     Data columns (total 21 columns):
                         Non-Null Count Dtype
      # Column
          battery_power 2000 non-null int64
      0
                         2000 non-null int64
          blue
          clock_speed
                         2000 non-null
                                        float64
          dual_sim
                         2000 non-null
      3
                                         int64
                         2000 non-null
                                         int64
      4
          fc
                         2000 non-null
      5
          four_g
                                         int64
          int_memory
                         2000 non-null
                                        int64
      6
                         2000 non-null
      7
                                         float64
          m_dep
                         2000 non-null
      8
          mobile_wt
                                         int64
                         2000 non-null
                                         int64
      9
          n_cores
                         2000 non-null
                                         int64
      10 pc
                         2000 non-null
      11 px_height
                                         int64
                         2000 non-null int64
      12 px_width
                         2000 non-null
                                         int64
      13 ram
      14 sc_h
                         2000 non-null
                                         int64
                         2000 non-null
      15 sc_w
                                         int64
                         2000 non-null
      16 talk_time
                                         int64
                         2000 non-null
      17 three_g
                                         int64
      18 touch_screen 2000 non-null
                                        int64
                         2000 non-null
      19 wifi
                                         int64
      20 price_range
                         2000 non-null
                                        int64
     dtypes: float64(2), int64(19)
     memory usage: 328.2 KB
#Printing the column names:
df.columns
 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')
#Changing the column names:
df=df.rename(columns={"blue":"bluetooth","fc":"front_camera","pc": "Primary Camera mega pixels"})
print("Column Names has Updated Successfully")
 Column Names has Updated Successfully
#First 3 rows of the data:
df.head(3)
 \overline{\mathbf{x}}
         battery_power bluetooth clock_speed dual_sim front_camera 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
                                           2.2
      0
                   842
                                                                                             0.6
                                                                                                        188
                                                                                                                                 20
                                                                                                                                         756 2549
                                                                                                                                                                      19
                  1021
                                           0.5
                                                      1
                                                                    0
                                                                                       53
                                                                                             0.7
                                                                                                       136
                                                                                                                  3
                                                                                                                                905
                                                                                                                                         1988 2631
                                                                                                                                                      17
                                                                                                                                                            3
                                                                                                                                                                       7
                                                                    2
                                           0.5
                                                                                       41
                                                                                             0.9
                                                                                                        145
                                                                                                                  5
                                                                                                                               1263
                                                                                                                                         1716 2603
                                                                                                                                                      11
                                                                                                                                                                       9
     3 rows × 21 columns
#Printing the column names:
df.columns
```

#Changing the data:
mobile_data = pd.DataFrame()
mobile_data["bluetooth"] = np.where(df["bluetooth"]<1,"No","yes")
mobile_data["dual_sim"] = np.where(df["dual_sim"]<1,"No","yes")</pre>

dtype='object')

'Primary Camera mega pixels', 'px_height', 'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g', 'touch_screen', 'wifi', 'price_range'],

 \rightarrow

0

mobile_data["four_g"] = np.where(df["four_g"]<1,"No","yes")
mobile_data["three_g"] = np.where(df["three_g"]<1,"No","yes")
mobile_data["touch_screen"] = np.where(df["touch_screen"]<1,"No","yes")
mobile_data["wifi"] = np.where(df["wifi"]<1,"No","yes")
print("Column Data has updated Successfully")</pre>

Column Data has updated Successfully

#First 3 rows of the data:
mobile_data.head(3)

5		bluetooth	dual_sim	four_g	three_g	touch_screen	wifi
	0	No	No	No	No	No	yes
	1	yes	yes	yes	yes	yes	No
	2	yes	yes	yes	yes	yes	No

#First 3 rows of the data:
df.head(3)

u																					
€	}	battery_power	bluetooth	clock_speed	dual_sim	front_camera	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 w	ifi
	0	842	0	2.2	0	1	0	7	0.6	188	2		20	756	2549	9	7	19	0	0	1
	1	1021	1	0.5	1	0	1	53	0.7	136	3		905	1988	2631	17	3	7	1	1	0
	2	563	1	0.5	1	2	1	41	0.9	145	5		1263	1716	2603	11	2	9	1	1	0

3 rows × 21 columns

VISUALIZATION:

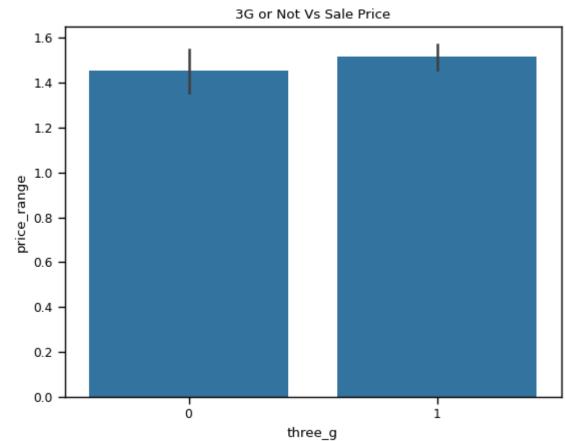
 $\overline{\pm}$

#Highest Correlated columns to PRICE RANGE:
plt.figure(figsize=(15, 12))
sns.set_context('paper')
mobile_corr=df.corr(numeric_only=True)
sns.heatmap(mobile_corr.corr(),cmap='Reds',annot=True)
plt.title(" Highest correlated columns to PRICE RANGE ",fontsize=20)
plt.show()

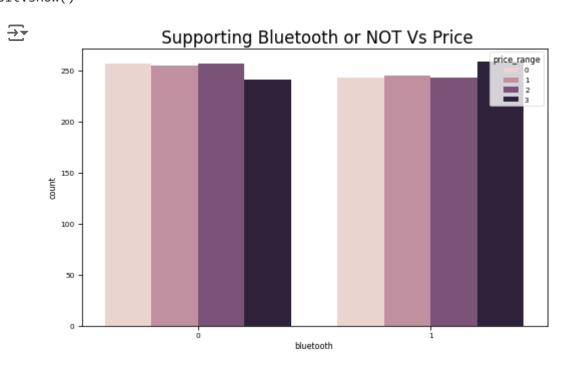
Highest correlated columns to PRICE RANGE -0.044-0.032 -0.14-0.0037-0.051-0.063 0.011 -0.053 -0.12-0.0072-0.049 -0.08 0.043 -0.15 -0.15 0.056 -0.052-0.087 -0.088 0.19 battery_power bluetooth -0.044 0.000680.024 -0.079 -0.068 <mark>0.031 -0.05 -0.071 0.021 -0.097 -0.12 -0.17 -0.032 -0.085 -0.086 -0.022 -0.12 -0.035 -0.11 -0.064 -0.086 -0.022 -0.12 -0.035 -0.11 -0.064</mark> clock_speed --0.0320.0006 -0.037-0.061 -0.16 <mark>-0.031-0.071-0.011-0.049-0.071 -0.1 -0.098-0.069 -0.12 -0.097-0.057 -0.16-<mark>0.0019</mark>-0.099 -0.099</mark> -0.12 -0.056-0.069-0.092-0.057-0.087 -0.11 -0.098-0.0510.0093-0.087-0.098 -0.12 -0.077-0.077<mark>0.00042</mark>0.037 dual sim - -0.14 0.024 -0.037 - 0.8 front camera -0.0037-0.079 -0.061 -0.12 1 -0.054 -0.08 -0.093 -0.12 -0.12 -0.14 -0.1 -0.082 -0.027 -0.013 -0.17 0.86 -0.03 -0.1 -0.1 four g =-0.051 -0.068 -0.16 -0.056 -0.12 - 0.6 m_dep - 0.011 -0.05 -0.071-0.092-0.057 -0.08 -0.04 1 0.0017-0.055-0.026-0.023-0.027-0.098 -0.13 -0.13-0.0067-0.091-0.055 -0.12 -0.099 mobile wt --0.053-0.071-0.011-0.057-0.0069-0.093 -0.12 0.0017 -0.081<mark>-0.016</mark>-0.075-0.077 -0.1 -0.14 -0.13 <mark>-0.027-0.066-0.071-0.053</mark> -0.14 1 -0.078-0.073-0.033-0.066-0.055-0.022-0.014 -0.1 0.0056-0.073-0.088 Primary Camera mega pixels -0.0072-0.097 -0.071 -0.11 0.9 -0.12 -0.15 -0.026 -0.016 -0.078 1 -0.14 -0.12 -0.051 -0.11 -0.16 <mark>-0.041</mark>-0.098-0.096-0.061-0.072 - 0.4 px height --0.049 -0.12 -0.1 -0.098 -0.13 -0.14 -0.054-0.023-0.075-0.073 -0.14 1 0.79 -0.055 0.024 0.0022-0.087 -0.16 -0.04 0.031 0.11 px_width = -0.08 -0.17 -0.098-0.051 -0.12 -0.1 -0.086-0.027-0.077-0.033 -0.12 0.79 1 -0.02 -0.031-0.027-0.068 -0.11 -0.075-0.0019 0.14 ram - 0.043 -0.032 -0.0690.0093 -0.066 -0.0820.0036 -0.098 -0.1 -0.066 -0.051 -0.055 -0.02 -0.061 -0.04 -0.046 -0.066 -0.15 -0.039 0.97 - 0.2 sc h = -0.15 -0.085 -0.12 -0.087 -0.12 -0.0270.0099 -0.13 -0.14 -0.055 -0.11 0.024 -0.031 -0.061 1 0.78 -0.11 -0.045 -0.1 -0.0064-0.077 sc w = -0.15 -0.086-0.097-0.098 -0.14 -0.013-0.032 -0.13 -0.13 -0.022 -0.16 0.0022-0.027 -0.04 0.78 1 -0.13 -0.024 -0.0580.000350.059 talk time - 0.056 -0.022 -0.057 -0.12 -0.065 -0.17 -0.053-0.0067-0.027 -0.014 -0.041 -0.087 -0.068 -0.046 -0.11 -0.13 -0.16 -0.012 -0.12 -0.048 three_g =-0.052 -0.12 -0.16 -0.077-0.095 0.86 -0.078-0.091-0.066 -0.1 -0.098 -0.16 -0.11 -0.066-0.045-0.024 -0.16 - 0.0 touch screen -0.087-0.035-0.0019-0.077-0.099-0.03-0.11-0.055-0.0710.0056-0.096-0.04-0.075-0.15-0.1 -0.058-0.012-0.032 wifi =-0.088 -0.11 -0.0990.000420.039 -0.1 -0.041 -0.12 -0.053 -0.073 -0.061 0.031-0.0019-0.039-0.00640.00035-0.12 -0.071 -0.032 price_range - 0.19 -0.064-0.099-0.037-0.085 -0.1 -0.0064-0.099 -0.14 -0.088-0.072 0.11 0.14 0.97 -0.077-0.059-0.048-0.086 -0.17 -0.056 bluetooth four_g dual_sim m_dep mobile_wt n_cores px_height px_width W mega pixels

#3G or Not 3G Mobile VS Sale Price:
sns.barplot(x='three_g', y='price_range', data=df)
plt.title('3G or Not Vs Sale Price')
plt.show()



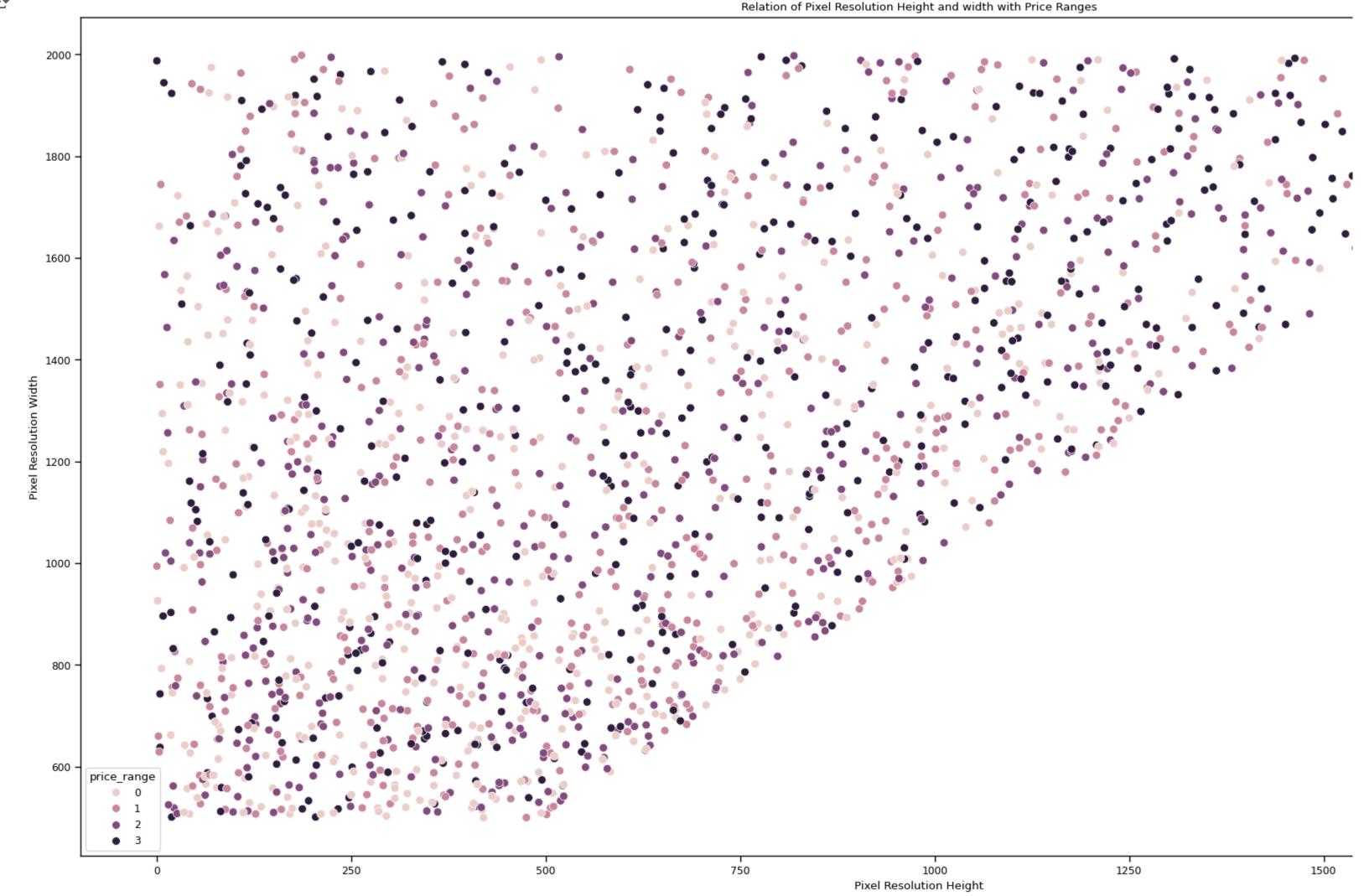


#Count Plot For Supporting Bluetooth or NOT Vs Price:
plt.figure(figsize = (10,6),dpi=60)
sns.countplot(data=df, x="bluetooth", hue="price_range")
plt.title(" Supporting Bluetooth or NOT Vs Price",fontsize=20)
plt.show()



Relation of Pixel Resolution Height and Pixel Resolution Width with Price Ranges
fig=plt.figure(figsize=(20,10))
ax=fig.add_axes([0,0,1,1])
sns.scatterplot(x = "px_height", y = "px_width", data = df, hue = "price_range",s=50)
ax.set(xlabel = "Pixel Resolution Height" , ylabel = "Pixel Resolution Width")
ax.set(title = "Relation of Pixel Resolution Height and width with Price Ranges")
plt.show()

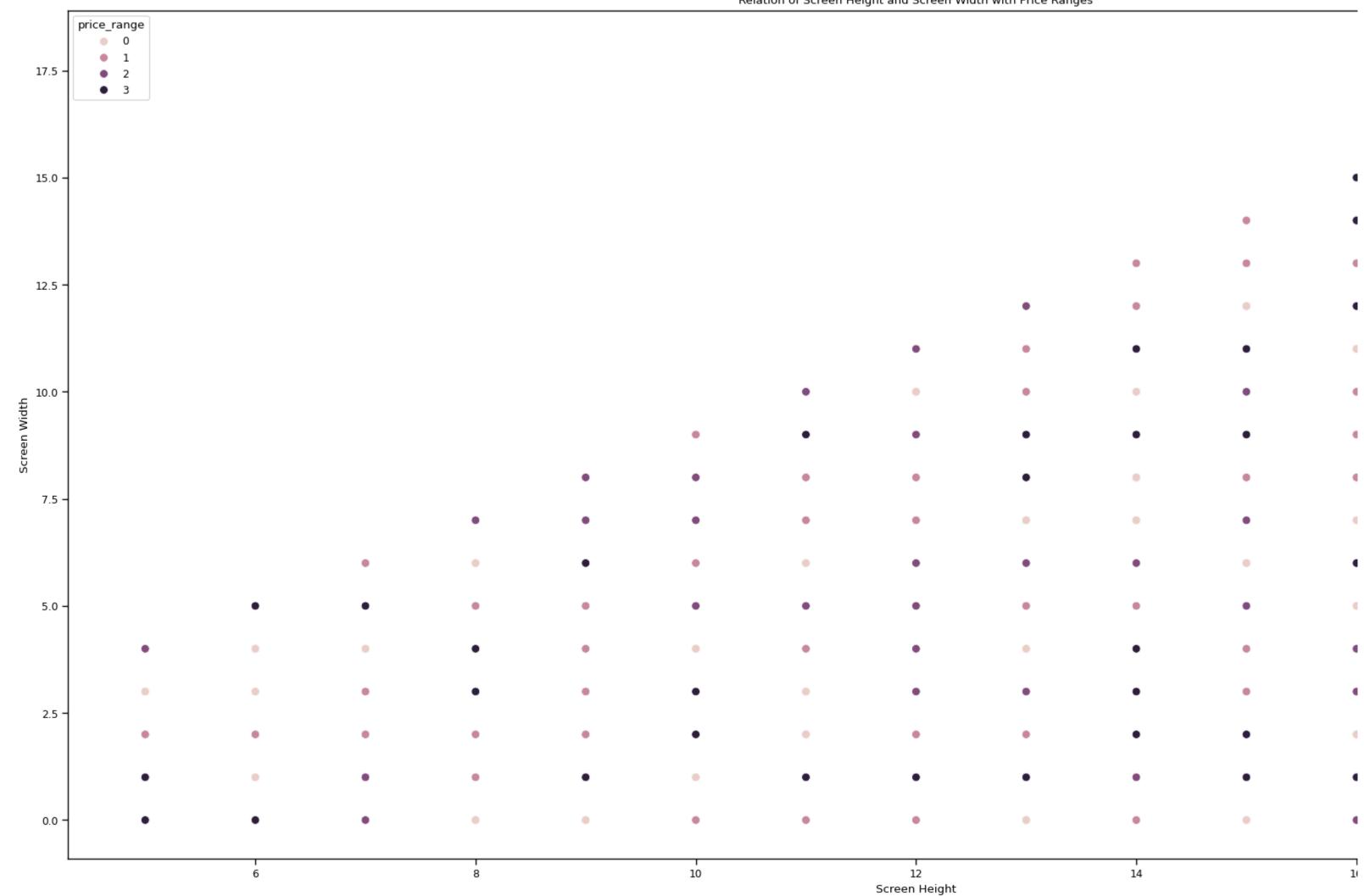




Relation of Screen Height and Screen Width with Price Ranges
fig=plt.figure(figsize=(20,10))
ax=fig.add_axes([0,0,1,1])
sns.scatterplot(x = "sc_h", y = "sc_w", data = df, hue = "price_range",s=50)
ax.set(xlabel = "Screen Height",ylabel = "Screen Width")
ax.set(title = "Relation of Screen Height and Screen Width with Price Ranges")
plt.show()



Relation of Screen Height and Screen Width with Price Ranges



#Changing of data:
df["price_range"].replace({1:"Low Cost", 2:"Medium Cost", 3:"High Cost", 4:"Very High Cost"}, inplace=True)
print("Column Data has updated Successfully")

Column Data has updated Successfully

#First five rows of the data:
df.head()

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

5 rows × 21 columns