Laptop Price Prediction

Importing Basic Dependencies

In [120]:
```
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
%matplotlib inline
```

In [121]:
```
df = pd.read_csv('laptop_data.csv')
df.head()
```

Out[121]:

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0000
3	3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.3360
4	4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.8080

In [122]:
```
df.columns
```

Out[122]:
```
Index(['Unnamed: 0', 'Company', 'TypeName', 'Inches', 'ScreenResolution',
       'Cpu', 'Ram', 'Memory', 'Gpu', 'OpSys', 'Weight', 'Price'],
      dtype='object')
```

In [123]:
```
# removing the unnamed: 0 col

df = df[['Company', 'TypeName', 'Inches', 'ScreenResolution',
       'Cpu', 'Ram', 'Memory', 'Gpu', 'OpSys', 'Weight', 'Price']]
df.head()
```

Out[123]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0000
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.3360
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.8080

In [124]:
```
df.isnull().sum()
```

Out[124]:
```
Company             0
TypeName            0
Inches              0
ScreenResolution    0
Cpu                 0
Ram                 0
Memory              0
Gpu                 0
OpSys               0
Weight              0
Price               0
dtype: int64
```

In [125]:
```
# checking for duplicated rows

df.duplicated().sum()
```

Out[125]:
```
29
```

In [126]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Company           1303 non-null   object 
 1   TypeName          1303 non-null   object 
 2   Inches            1303 non-null   float64
 3   ScreenResolution  1303 non-null   object 
 4   Cpu               1303 non-null   object 
 5   Ram               1303 non-null   object 
 6   Memory            1303 non-null   object 
 7   Gpu               1303 non-null   object 
 8   OpSys             1303 non-null   object 
 9   Weight            1303 non-null   object 
 10  Price             1303 non-null   float64
dtypes: float64(2), object(9)
memory usage: 112.1+ KB
```

In [127]:
```
catvars = df.select_dtypes(include=['object']).columns
numvars = df.select_dtypes(include = ['int32','int64','float32','float64']).columns

catvars,numvars
```

Out[127]:
```
(Index(['Company', 'TypeName', 'ScreenResolution', 'Cpu', 'Ram', 'Memory',
        'Gpu', 'OpSys', 'Weight'],
       dtype='object'),
 Index(['Inches', 'Price'], dtype='object'))
```

In [128]:
```
def uniquevals(col):
    print(f'Details of the particular col {col} is : {df[col].unique()}')

def valuecounts(col):
    print(f'Valuecounts of the particular col {col} is : {df[col].value_counts()}')



for col in df.columns:
    uniquevals(col)
    print("-"*75)
```

```
Details of the particular col Company is : ['Apple' 'HP' 'Acer' 'Asus' 'Dell' 'Lenovo' 'Chuwi' 'MSI' 'Microsoft'
 'Toshiba' 'Huawei' 'Xiaomi' 'Vero' 'Razer' 'Mediacom' 'Samsung' 'Google'
 'Fujitsu' 'LG']
---------------------------------------------------------------------------
Details of the particular col TypeName is : ['Ultrabook' 'Notebook' 'Netbook' 'Gaming' '2 in 1 Convertible'
 'Workstation']
---------------------------------------------------------------------------
Details of the particular col Inches is : [13.3 15.6 15.4 14.  12.  11.6 17.3 10.1 13.5 12.5 13.  18.4 13.9 12.3
 17.  15.  14.1 11.3]
---------------------------------------------------------------------------
Details of the particular col ScreenResolution is : ['IPS Panel Retina Display 2560x1600' '1440x900' 'Full HD 1920x
1080'
 'IPS Panel Retina Display 2880x1800' '1366x768'
 'IPS Panel Full HD 1920x1080' 'IPS Panel Retina Display 2304x1440'
 'IPS Panel Full HD / Touchscreen 1920x1080'
 'Full HD / Touchscreen 1920x1080' 'Touchscreen / Quad HD+ 3200x1800'
 'IPS Panel Touchscreen 1920x1200' 'Touchscreen 2256x1504'
 'Quad HD+ / Touchscreen 3200x1800' 'IPS Panel 1366x768'
 'IPS Panel 4K Ultra HD / Touchscreen 3840x2160'
```

In [129]:
```
'''
so on observation we can see that if we remove "GB" from RAM,i can
make it as an integer value then after,now same goes with Memory as
well as Weight,for Weight i can classify it as floating variable
using the str.replace() as shown ↓
'''

df['Ram'] = df['Ram'].str.replace('GB','')
df['Weight'] = df['Weight'].str.replace('kg','')

# converting from string->integer for ram column

df['Ram'] = df['Ram'].astype('int32')

# converting from string-> float for the weight column

df['Weight'] = df['Weight'].astype('float32')

df.head()
```

Out[129]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080

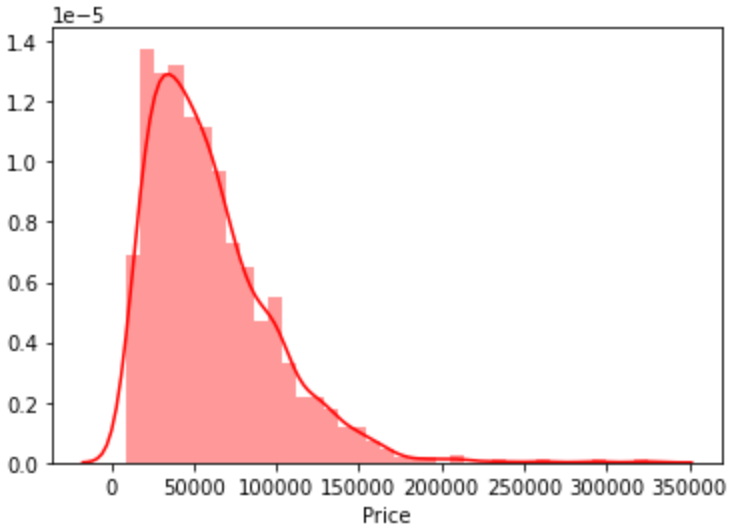
In [130]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1303 non-null   object
 1   TypeName          1303 non-null   object
 2   Inches            1303 non-null   float64
 3   ScreenResolution  1303 non-null   object
 4   Cpu               1303 non-null   object
 5   Ram               1303 non-null   int32
 6   Memory            1303 non-null   object
 7   Gpu               1303 non-null   object
 8   OpSys             1303 non-null   object
 9   Weight            1303 non-null   float32
 10  Price             1303 non-null   float64
dtypes: float32(1), float64(2), int32(1), object(7)
memory usage: 101.9+ KB
```

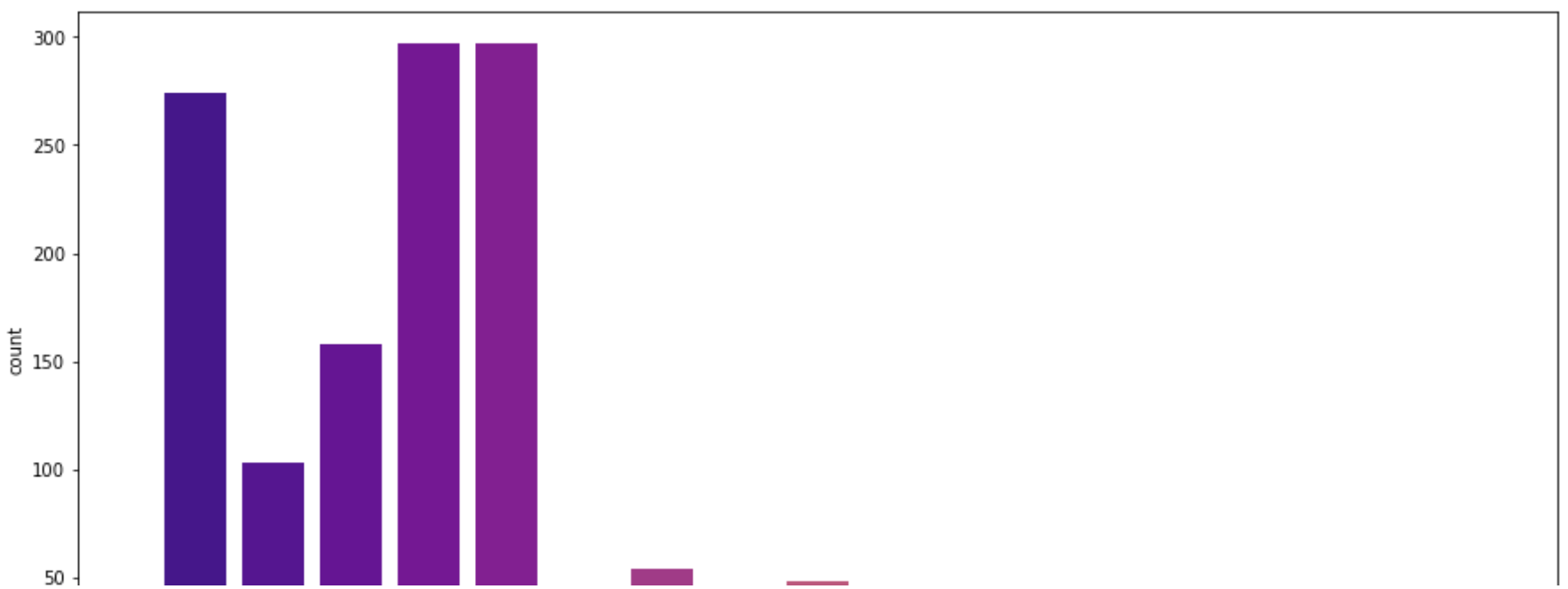
Exploratory Data Analysis

In [131]:
```
# viewing the distribution of the price column

sn.distplot(df['Price'],color='red')
```

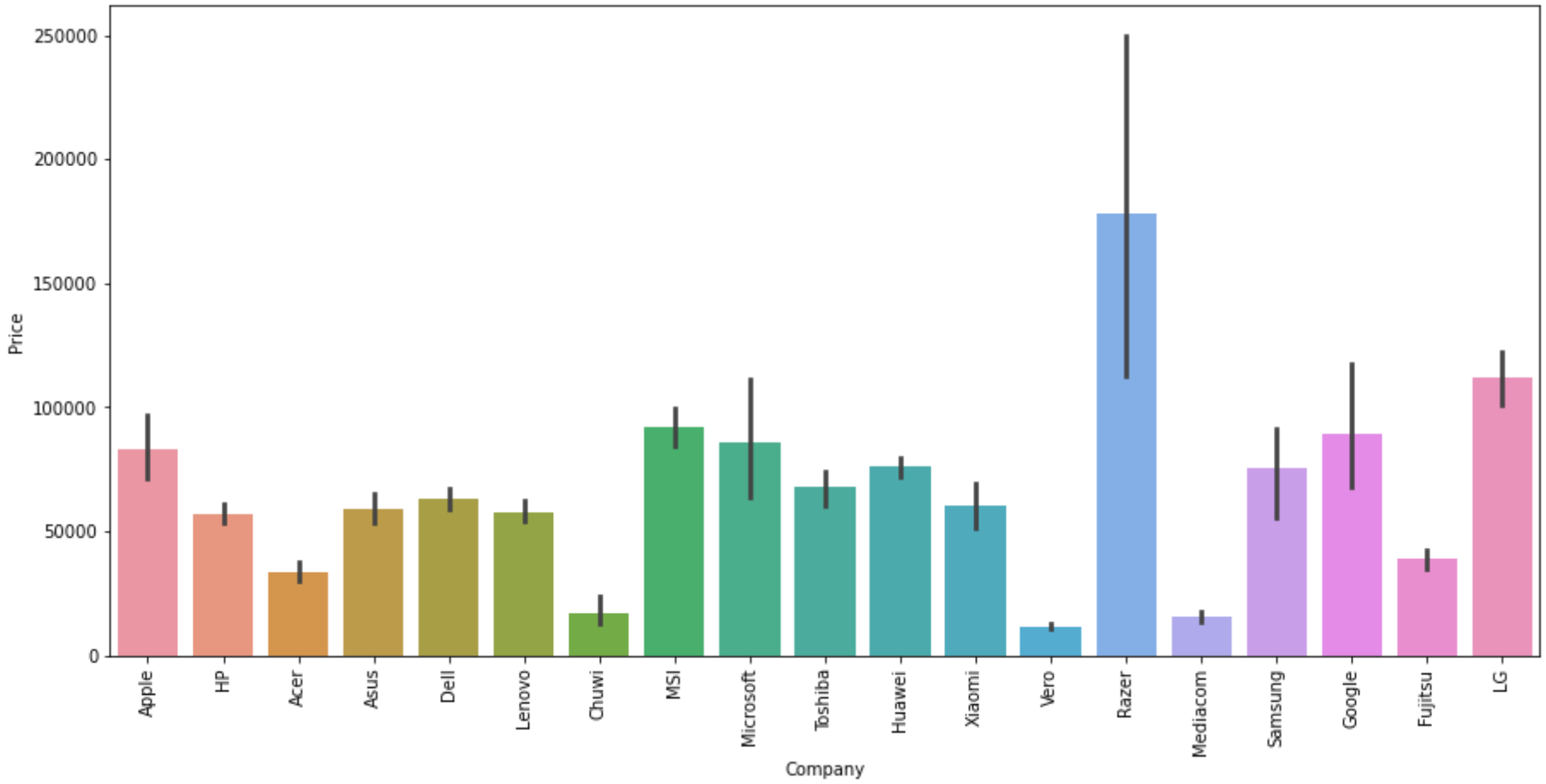
Out[131]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120b5f3f8c8>
```



In [132]:
```
## plotting countplots for the categorical variables

def drawplot(col):
    plt.figure(figsize=(15,7))
    sn.countplot(df[col],palette='plasma')
    plt.xticks(rotation='vertical')

toview = ['Company', 'TypeName','Ram','OpSys']
for col in toview:
    drawplot(col)
```

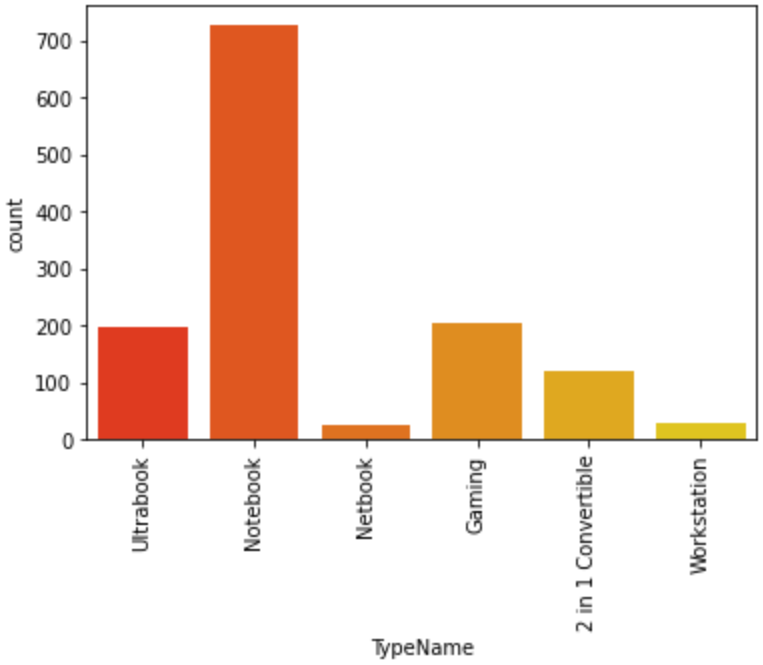


In [133]:
```
# average price for each of the laptop brands
# this will say us the insight that as per company the price of the laptop vary


plt.figure(figsize=(15,7))
sn.barplot(x = df['Company'],y = df['Price'])
plt.xticks(rotation = 'vertical')
plt.show()
```



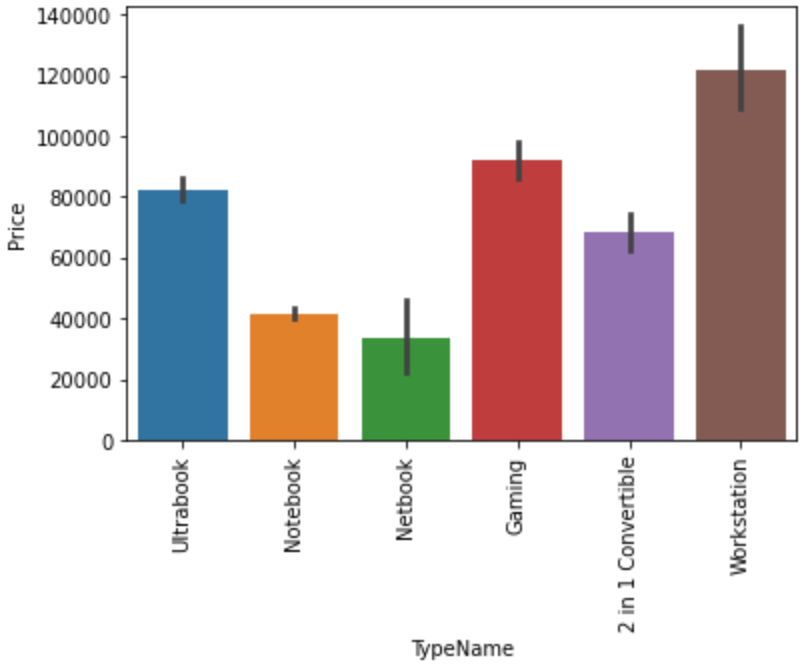
In [134]:
```
## various types of laptops

sn.countplot(df['TypeName'],palette='autumn')
plt.xticks(rotation = 'vertical')
```

Out[134]: (array([0, 1, 2, 3, 4, 5]), <a list of 6 Text major ticklabel objects>)



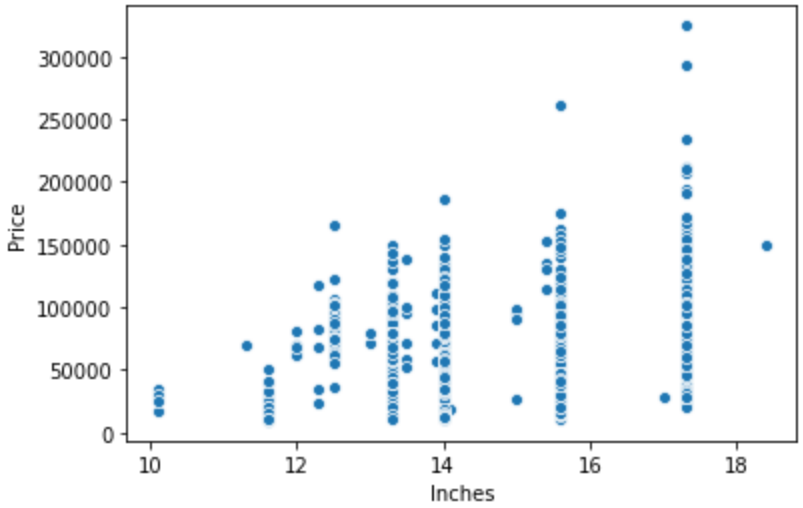
In [135]:
```
# laptop type and variation about the price

sn.barplot(x = df['TypeName'],y = df['Price'])
plt.xticks(rotation = 'vertical')
```

Out[135]: (array([0, 1, 2, 3, 4, 5]), <a list of 6 Text major ticklabel objects>)



In [136]:
```
# variation of inches towards the price

sn.scatterplot(x = df['Inches'],y = df['Price'])
```

Out[136]: <matplotlib.axes._subplots.AxesSubplot at 0x120bc808508>



For the `Screen Resolution` column we have many types of Screen Resolutions out there as shown `Touch Screen` and `Normal` and `IPS Panel` are the 3 parts on basis of which we can segregate the things

In [137]:
```
df['ScreenResolution'].value_counts()
```

Out[137]:
```
Full HD 1920x1080                                507
1366x768                                         281
IPS Panel Full HD 1920x1080                      230
IPS Panel Full HD / Touchscreen 1920x1080         53
Full HD / Touchscreen 1920x1080                   47
1600x900                                          23
Touchscreen 1366x768                              16
Quad HD+ / Touchscreen 3200x1800                  15
IPS Panel 4K Ultra HD 3840x2160                   12
IPS Panel 4K Ultra HD / Touchscreen 3840x2160     11
4K Ultra HD / Touchscreen 3840x2160               10
Touchscreen 2560x1440                              7
4K Ultra HD 3840x2160                              7
IPS Panel 1366x768                                 7
IPS Panel Quad HD+ / Touchscreen 3200x1800         6
Touchscreen 2256x1504                              6
IPS Panel Retina Display 2560x1600                 6
IPS Panel Retina Display 2304x1440                 6
IPS Panel Touchscreen 2560x1440                    5
IPS Panel 2560x1440                                4
IPS Panel Retina Display 2880x1800                 4
1440x900                                           4
IPS Panel Touchscreen 1920x1200                    4
2560x1440                                          3
1920x1080                                          3
IPS Panel Quad HD+ 2560x1440                       3
IPS Panel Touchscreen 1366x768                     3
Touchscreen 2400x1600                              3
Quad HD+ 3200x1800                                 3
IPS Panel Full HD 2160x1440                        2
IPS Panel Quad HD+ 3200x1800                       2
IPS Panel Touchscreen / 4K Ultra HD 3840x2160      2
Touchscreen / Full HD 1920x1080                    1
Touchscreen / Quad HD+ 3200x1800                   1
Touchscreen / 4K Ultra HD 3840x2160                1
IPS Panel Full HD 1920x1200                        1
IPS Panel Full HD 2560x1440                        1
IPS Panel Retina Display 2736x1824                 1
IPS Panel Touchscreen 2400x1600                    1
IPS Panel Full HD 1366x768                         1
Name: ScreenResolution, dtype: int64
```

In [138]:
```
# creating a new col,touchscreen if the value is 1 that laptop is touch screen

df['TouchScreen'] = df['ScreenResolution'].apply(lambda element:1
                                                 if 'Touchscreen' in element else 0)

df.head()
```

Out[138]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0

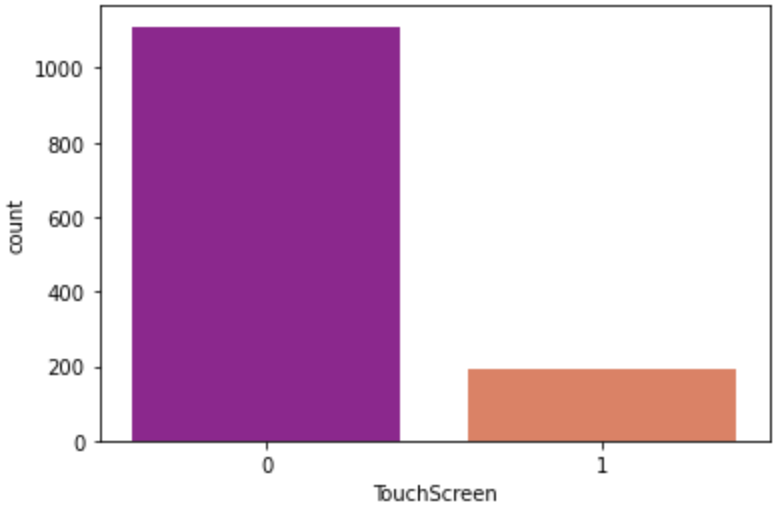
In [139]:
```
df.sample(5)
```

Out[139]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen
56	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i3 6006U 2GHz	4	128GB SSD	Intel HD Graphics 520	Windows 10	1.91	23389.9200	0
607	Lenovo	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	6	1TB HDD	Intel HD Graphics 620	Windows 10	2.40	30049.9200	0
66	HP	Notebook	15.6	1366x768	Intel Core i3 6006U 2GHz	4	500GB HDD	AMD Radeon 520	Windows 10	1.86	23373.4032	0
291	Asus	Gaming	17.3	Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	8	1TB HDD	Nvidia GeForce GTX 1050	Windows 10	3.00	63243.3600	0
987	Lenovo	Gaming	15.6	IPS Panel Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	8	128GB SSD + 1TB HDD	Nvidia GeForce GTX 1060	Windows 10	2.50	63349.9200	0

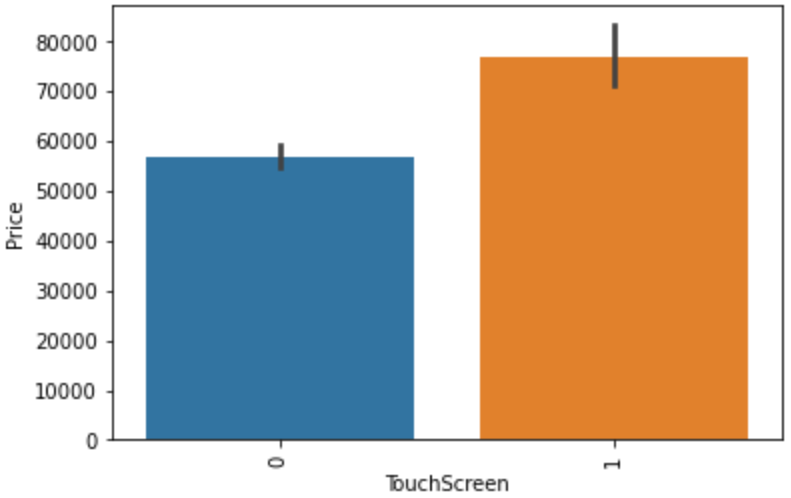
In [140]:
```
sn.countplot(df['TouchScreen'],palette='plasma')
```

Out[140]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120bc92aa88>
```



In [141]:
```
# touch screen on comparision with price of laptop

sn.barplot(x = df['TouchScreen'],y = df['Price'])
plt.xticks(rotation = 'vertical')
```

Out[141]:
```
(array([0, 1]), <a list of 2 Text major ticklabel objects>)
```



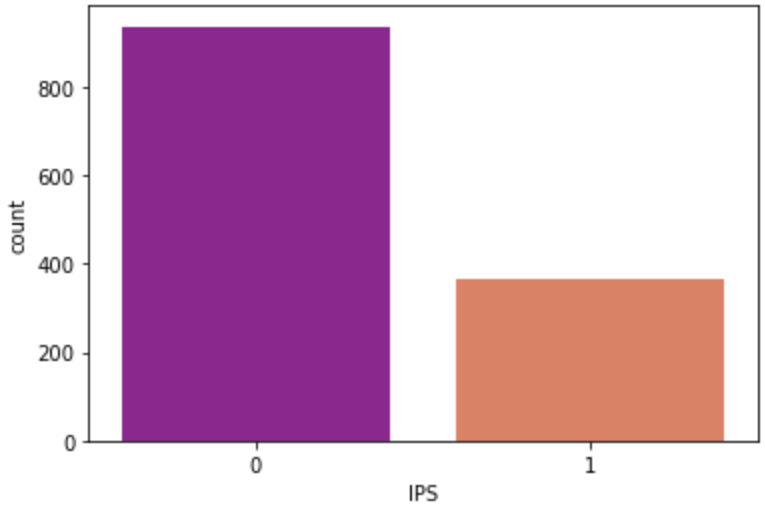
In [142]:
```
# creating a new col named IPS,does the laptop have IPS facility or not

df['IPS'] = df['ScreenResolution'].apply(
    lambda element:1 if "IPS" in element else 0
)
df.sample(5)
```

Out[142]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS
695	Acer	Netbook	11.6	1366x768	Intel Celeron Dual Core N3050 1.6GHz	4	32GB Flash Storage	Intel HD Graphics	Windows 10	1.4	14332.32	0	0
178	Lenovo	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	128GB SSD + 1TB HDD	AMD R17M-M1-70	Windows 10	1.9	43316.64	0	0
884	Dell	Notebook	15.6	1366x768	Intel Pentium Quad Core N3710 1.6GHz	4	500GB HDD	Intel HD Graphics	Windows 10	2.2	19660.32	0	0
1203	Dell	Ultrabook	13.3	Quad HD+ / Touchscreen 3200x1800	Intel Core i7 7500U 2.7GHz	16	512GB SSD	Intel HD Graphics 620	Windows 10	1.2	142790.40	1	0
588	Lenovo	Notebook	15.6	Touchscreen 1366x768	Intel Core i7 8550U 1.8GHz	12	1TB HDD	Intel HD Graphics 620	Windows 10	2.2	32447.52	1	0

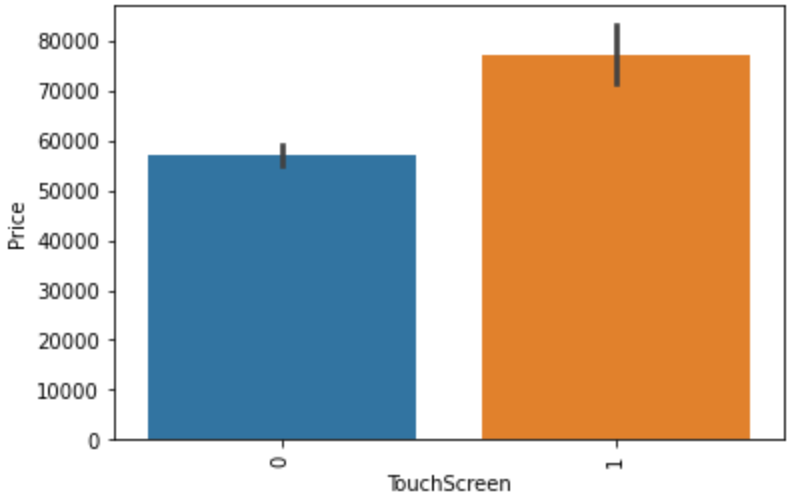
In [143]:
```
sn.countplot(df['IPS'],palette='plasma')
```

Out[143]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120bc9c8508>
```



In [144]:
```
# price variation with respect to the IPS col

sn.barplot(x = df['TouchScreen'],y = df['Price'])
plt.xticks(rotation = 'vertical')
```

Out[144]:
```
(array([0, 1]), <a list of 2 Text major ticklabel objects>)
```



Extracting the X Resolution and the Y Resolution

In [145]:
```
# we will split the text at the "x" letter and seperate the 2 parts
# from this we can observe that one of the col is Y res we need to do
# some feature engineering on the X res col

splitdf = df['ScreenResolution'].str.split('x',n = 1,expand=True)
splitdf.head()
```

Out[145]:

	0	1
0	IPS Panel Retina Display 2560	1600
1	1440	900
2	Full HD 1920	1080
3	IPS Panel Retina Display 2880	1800
4	IPS Panel Retina Display 2560	1600

In [146]:
```
splitdf = df['ScreenResolution'].str.split('x',n = 1,expand=True)

df['X_res'] = splitdf[0]
df['Y_res'] = splitdf[1]
df.head()
```

Out[146]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	X_res	Y_re
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	IPS Panel Retina Display 2560	160
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	1440	90
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	Full HD 1920	108
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	IPS Panel Retina Display 2880	180
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	IPS Panel Retina Display 2560	160

In [147]:
```
'''
So basically from that whole text of the X_res col,we need to
extract the digits from it,but the problem is the numbers are scattered
in some cases,that is the reason why i am using regex,if we use this
we will exactly get the numbers which we are looking for!,
so firstly replace all the "," with "" and then find all numbers
from that string as "\d+\.?\d+",\d means that integer number and \.?
all the numbers which come after an number and \d+ the string must end with number


'''


df['X_res'] = df['X_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)').apply(lambda x:x[0])

df.head()
```

Out[147]:

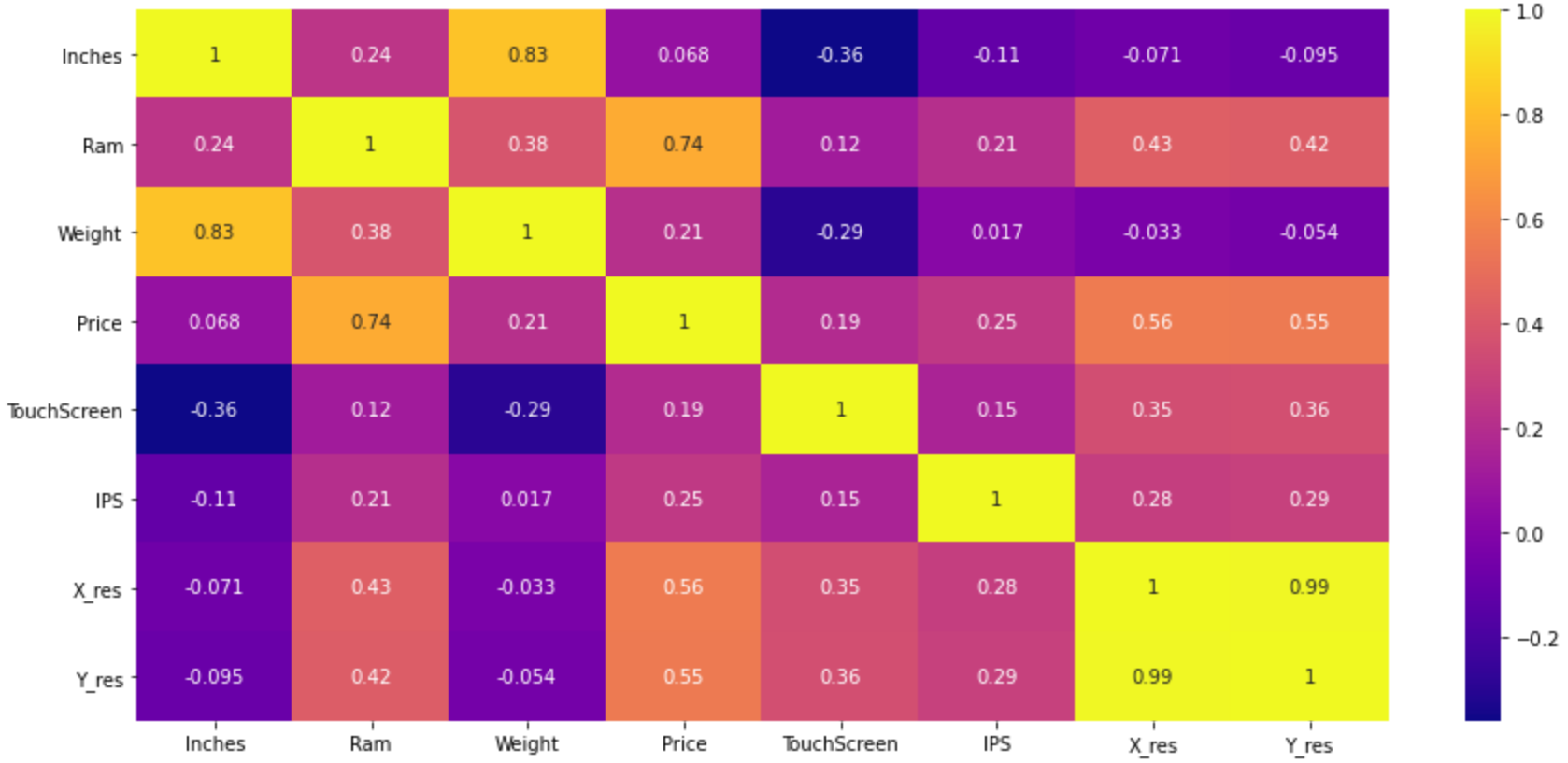
	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	X_res	Y_res
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	2560	160
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	1440	90
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	1920	108
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	2880	180
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	2560	160

In [148]:
```
df['X_res'] = df['X_res'].astype('int')
df['Y_res'] = df['Y_res'].astype('int')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Company           1303 non-null   object 
 1   TypeName          1303 non-null   object 
 2   Inches            1303 non-null   float64
 3   ScreenResolution  1303 non-null   object 
 4   Cpu               1303 non-null   object 
 5   Ram               1303 non-null   int32  
 6   Memory            1303 non-null   object 
 7   Gpu               1303 non-null   object 
 8   OpSys             1303 non-null   object 
 9   Weight            1303 non-null   float32
 10  Price             1303 non-null   float64
 11  TouchScreen       1303 non-null   int64  
 12  IPS               1303 non-null   int64  
 13  X_res             1303 non-null   int32  
 14  Y_res             1303 non-null   int32  
dtypes: float32(1), float64(2), int32(3), int64(2), object(7)
memory usage: 132.5+ KB
```

In [149]:
```
plt.figure(figsize=(15,7))
sn.heatmap(df.corr(),annot=True,cmap='plasma')
```

Out[149]: <matplotlib.axes._subplots.AxesSubplot at 0x120bc85f288>



In [150]:
```
df.corr()['Price']
```

Out[150]:
```
Inches         0.068197
Ram            0.743007
Weight         0.210370
Price          1.000000
TouchScreen    0.191226
IPS            0.252208
X_res          0.556529
Y_res          0.552809
Name: Price, dtype: float64
```

From the correlation plot we observed that as the X_res and Y_res is increasing,the price of the laptop is also increasing,so `X_res` and `Y_res` are positively correlated and they are giving much information,so that is the reason why i had splitted `Resolution` column into

So to make things good,we can create a new column named `PPI{pixels per inch}` ,now as we saw from the correlation plot that the `X_res and Y_res` are having much collinearity,so why not combine them with `Inches` which is having less collinearity,so we will combine them as follows ↓,so here is the formula of how to calculate `PPI` {pixels per inch}

$$PPI(pixelsperinch) = \frac{\sqrt{X_resolution^2 + Y_resolution^2}}{inches}$$

In [151]:
```
df['PPI'] = (((df['X_res']**2+df['Y_res']**2))**0.5/df['Inches']).astype('float')
df.head()
```

Out[151]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	X_res	Y_res
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	2560	160(
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	1440	90(
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	1920	108(
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	2880	180(
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	2560	160(

In [152]:
```
df.corr()['Price']
```

Out[152]:
```
Inches         0.068197
Ram            0.743007
Weight         0.210370
Price          1.000000
TouchScreen    0.191226
IPS            0.252208
X_res          0.556529
Y_res          0.552809
PPI            0.473487
Name: Price, dtype: float64
```

So as we observe from the correlation data that the `PPI` is having good correlation,so we will be using that,as that is a combination of 3 features and that gives collective results of 3 columns,so we will drop `Inches,X_res,Y_res` as well

In [153]:
```
df.drop(columns=['ScreenResolution','Inches','X_res','Y_res'],inplace=True)
df.head()
```

Out[153]:

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005

Now we will work on `CPU` column,as that also has much text data and we need to process it efficiently as we may get good insights from them

In [154]:
```
df['Cpu'].value_counts()
```

Out[154]:
```
Intel Core i5 7200U 2.5GHz     190
Intel Core i7 7700HQ 2.8GHz    146
Intel Core i7 7500U 2.7GHz     134
Intel Core i7 8550U 1.8GHz      73
Intel Core i5 8250U 1.6GHz      72
                              ...
Intel Core i7 6920HQ 2.9GHz      1
Intel Core i7 2.9GHz             1
AMD E-Series E2-9000 2.2GHz      1
Intel Core M 7Y30 1.0GHz         1
Intel Core i7 6560U 2.2GHz       1
Name: Cpu, Length: 118, dtype: int64
```

Most common processors are made by intel right,so we will be clustering their `processors` into different categories like `i5,i7,other` ,now other means the processors of intel which do not have i3,i5 or i7 attached to it,they're completely different so that's the reason i will clutter them into `other` and other category is `AMD` which is a different category in whole

So if we observe we need to extract the first 3 words of the CPU column,as the first 3 words of every row under the CPU col is the type of the CPU,so we will be using them as shown ↓

In [155]:
```
df['CPU_name'] = df['Cpu'].apply(lambda text:" ".join(text.split()[:3]))
df.head()
```

Out[155]:

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

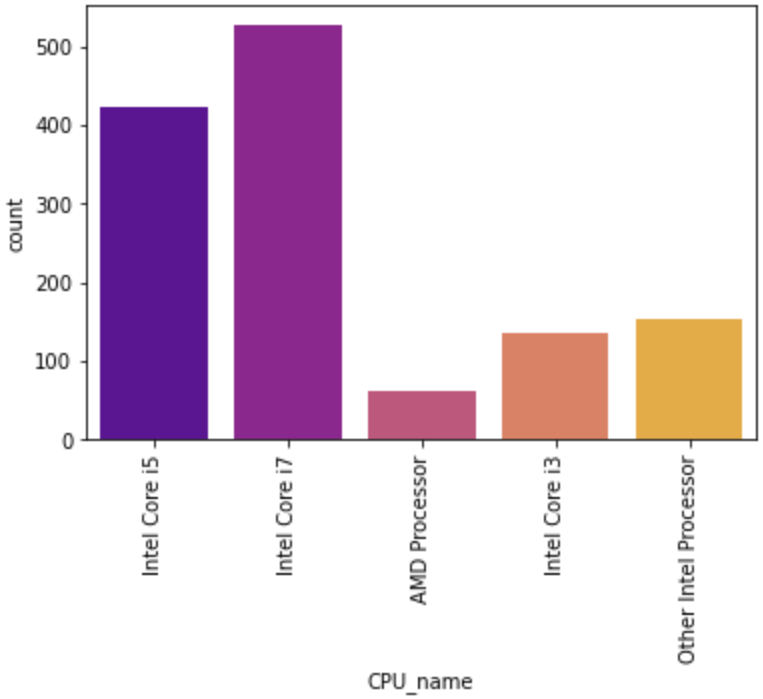
In [156]:
```
'''
As mentioned earlier,if we get any of the intel `i3,i5 or i7` versions
we will return them as it is,but if we get any other processor
we will first check whether is that a variant of the intel? or not
if yes,then we will tag it as "Other Intel Processor" else we will
say it as `AMD Processor`

'''

def processortype(text):

    if text=='Intel Core i7' or text=='Intel Core i5' or text=='Intel Core i3':
        return text

    else:
        if text.split()[0]=='Intel':
            return 'Other Intel Processor'

        else:
            return 'AMD Processor'



df['CPU_name'] = df['CPU_name'].apply(lambda text:processortype(text))
df.head()
```

Out[156]:

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

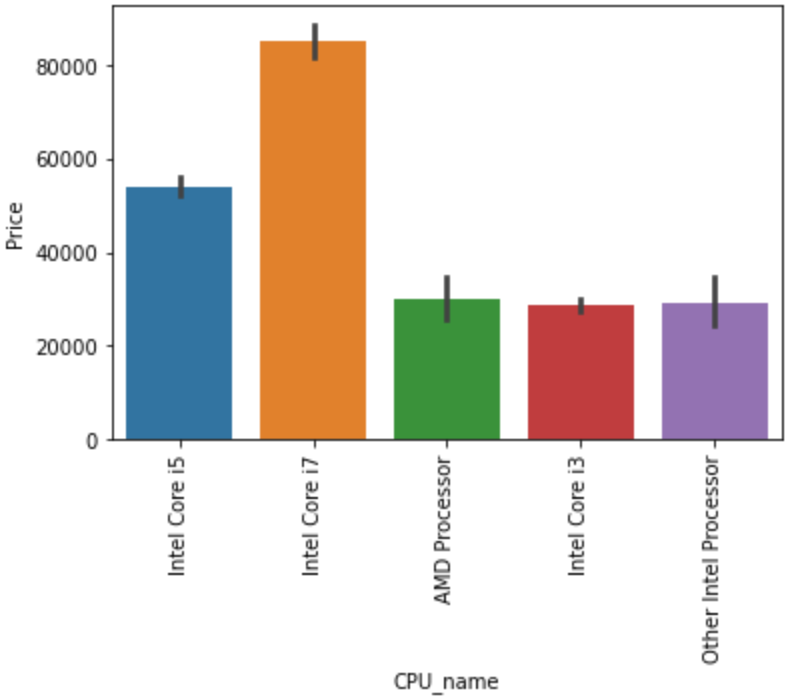
In [157]:
```
sn.countplot(df['CPU_name'],palette='plasma')
plt.xticks(rotation = 'vertical')
```

Out[157]: (array([0, 1, 2, 3, 4]), <a list of 5 Text major ticklabel objects>)



In [158]:
```
# price vs processor variation

sn.barplot(df['CPU_name'],df['Price'])
plt.xticks(rotation = 'vertical')
```

Out[158]: (array([0, 1, 2, 3, 4]), <a list of 5 Text major ticklabel objects>)



In [159]:
```
## dropping the cpu column

df.drop(columns=['Cpu'],inplace=True)
df.head()
```

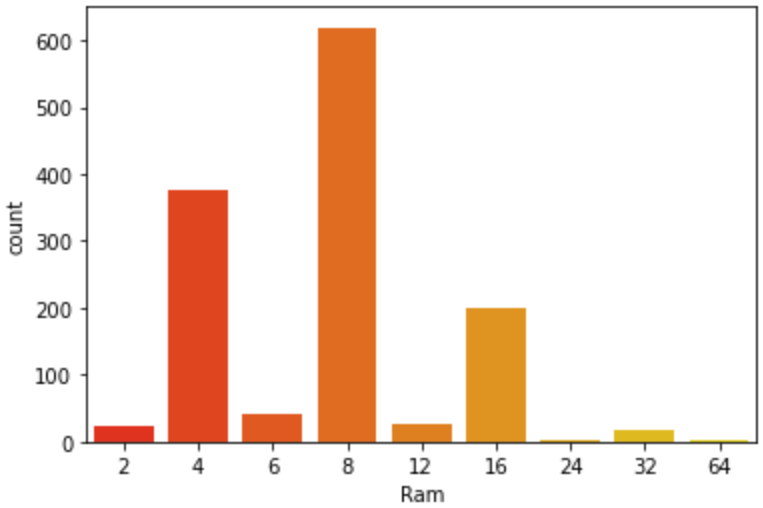
Out[159]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

Analysis on the RAM column

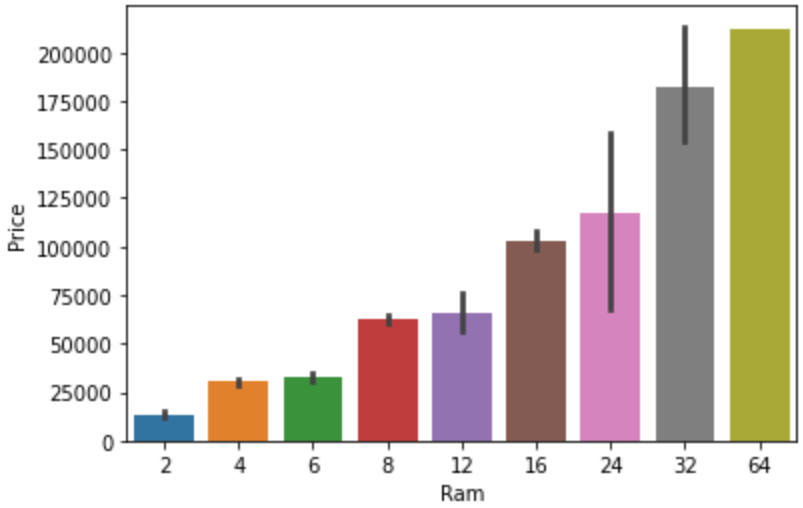
In [160]:
```
sn.countplot(df['Ram'],palette='autumn')
```

Out[160]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120c4b5be08>
```



In [161]:
```
## ram is having good relation with price

sn.barplot(df['Ram'],df['Price'])
```

Out[161]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120c5ba34c8>
```



About the memory column

We will seperate the `Type` of memory and the value of it,just similar to the one which is done in the previous part

This part involves things which are needed to be done in steps,so here we do not have the memory as a complete we have it in different dimension as `128GB SSD + 1TB HDD` ,so inorder to for it come in a same dimension we need to do some modifications which are done below as shown

In [162]:
```
df['Memory'].iloc[:1][0]
```

Out[162]:
```
'128GB SSD'
```

In [163]:
```
# we have different categories and also different kinds of variations

df['Memory'].value_counts()
```

Out[163]:
```
256GB SSD                        412
1TB HDD                          223
500GB HDD                        132
512GB SSD                        118
128GB SSD +  1TB HDD              94
128GB SSD                         76
256GB SSD +  1TB HDD              73
32GB Flash Storage                38
2TB HDD                           16
64GB Flash Storage                15
1TB SSD                           14
512GB SSD +  1TB HDD              14
256GB SSD +  2TB HDD              10
1.0TB Hybrid                       9
256GB Flash Storage                8
16GB Flash Storage                 7
32GB SSD                           6
180GB SSD                          5
128GB Flash Storage                4
512GB SSD +  2TB HDD               3
16GB SSD                           3
1TB SSD +  1TB HDD                 2
256GB SSD +  500GB HDD             2
512GB Flash Storage                2
256GB SSD +  256GB SSD             2
128GB SSD +  2TB HDD               2
64GB Flash Storage +  1TB HDD      1
1.0TB HDD                          1
508GB Hybrid                       1
8GB SSD                            1
512GB SSD +  256GB SSD             1
512GB SSD +  1.0TB Hybrid          1
1TB HDD +  1TB HDD                 1
128GB HDD                          1
256GB SSD +  1.0TB Hybrid          1
32GB HDD                           1
64GB SSD                           1
240GB SSD                          1
512GB SSD +  512GB SSD             1
Name: Memory, dtype: int64
```

In [164]:
```
## 4 most common variants observed : HHD,SSD,Flash,Hybrid

# this expression will remove the decimal space for example 1.0 TB will be 1TB

df['Memory'] = df['Memory'].astype(str).replace('\.0','',regex = True)

# replace the GB word with " "

df['Memory'] = df['Memory'].str.replace('GB','')

# replace the TB word with "000"

df['Memory'] = df['Memory'].str.replace('TB','000')

# split the word accross the "+" character

newdf = df['Memory'].str.split("+",n = 1,expand = True)

newdf
```

Out[164]:

	0	1
0	128 SSD	None
1	128 Flash Storage	None
2	256 SSD	None
3	512 SSD	None
4	256 SSD	None
...	...	...
1298	128 SSD	None
1299	512 SSD	None
1300	64 Flash Storage	None
1301	1000 HDD	None
1302	500 HDD	None

1303 rows × 2 columns

In [165]:
```
# we will strip up all the white spaces,basically eliminating white space

df['first'] = newdf[0]
df['first'] = df['first'].str.strip()
df.head()
```

Out[165]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	first
0	Apple	Ultrabook	8	128 SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	128 SSD
1	Apple	Ultrabook	8	128 Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	128 Flash Storage
2	HP	Notebook	8	256 SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	256 SSD
3	Apple	Ultrabook	16	512 SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	512 SSD
4	Apple	Ultrabook	8	256 SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	256 SSD

In [166]:
```
def applychanges(value):

    df['Layer1'+value] = df['first'].apply(lambda x:1 if value in x else 0)


listtoapply = ['HDD','SSD','Hybrid','FlashStorage']
for value in listtoapply:
    applychanges(value)


df.head()
```

Out[166]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	first	Layer1HDD	La
0	Apple	Ultrabook	8	128 SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	128 SSD	0	
1	Apple	Ultrabook	8	128 Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	128 Flash Storage	0	
2	HP	Notebook	8	256 SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	256 SSD	0	
3	Apple	Ultrabook	16	512 SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	512 SSD	0	
4	Apple	Ultrabook	8	256 SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	256 SSD	0	

In [167]:
```
# remove all the characters just keep the numbers

df['first'] = df['first'].str.replace(r'\D','')
df['first'].value_counts()
```

Out[167]:
```
256     508
1000    250
128     177
512     140
500     132
32       45
64       17
2000     16
16       10
180       5
240       1
8         1
508       1
Name: first, dtype: int64
```

In [168]:
```
df['Second'] = newdf[1]
df.head()
```

Out[168]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	first	Layer1HDD	Layer
0	Apple	Ultrabook	8	128 SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	128	0	
1	Apple	Ultrabook	8	128 Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	128	0	
2	HP	Notebook	8	256 SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	256	0	
3	Apple	Ultrabook	16	512 SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	512	0	
4	Apple	Ultrabook	8	256 SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	256	0	

In [169]:
```
def applychanges1(value):

    df['Layer2'+value] = df['Second'].apply(lambda x:1 if value in x else 0)


listtoapply1 = ['HDD','SSD','Hybrid','FlashStorage']
df['Second'] = df['Second'].fillna("0")
for value in listtoapply1:
    applychanges1(value)


# remove all the characters just keep the numbers

df['Second'] = df['Second'].str.replace(r'\D','')
df['Second'].value_counts()
```

Out[169]:
```
0       1095
1000     187
2000      15
256        3
500        2
512        1
Name: Second, dtype: int64
```

In [170]:
```
df['first'] = df['first'].astype('int')
df['Second'] = df['Second'].astype('int')
df.head()
```

Out[170]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	...	first	Layer1HDD	Layer1SSD	Layer1Hybrid
0	Apple	Ultrabook	8	128 SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	...	128	0	1	0
1	Apple	Ultrabook	8	128 Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	...	128	0	0	0
2	HP	Notebook	8	256 SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	...	256	0	1	0
3	Apple	Ultrabook	16	512 SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	...	512	0	1	0
4	Apple	Ultrabook	8	256 SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	...	256	0	1	0

5 rows × 22 columns

In [171]:
```
# multiplying the elements and storing the result in subsequent columns


df["HDD"]=(df["first"]*df["Layer1HDD"]+df["Second"]*df["Layer2HDD"])
df["SSD"]=(df["first"]*df["Layer1SSD"]+df["Second"]*df["Layer2SSD"])
df["Hybrid"]=(df["first"]*df["Layer1Hybrid"]+df["Second"]*df["Layer2Hybrid"])
df["Flash_Storage"]=(df["first"]*df["Layer1FlashStorage"]+df["Second"]*df["Layer2FlashStorage"])


## dropping of uncessary columns

df.drop(columns=['first', 'Second', 'Layer1HDD', 'Layer1SSD', 'Layer1Hybrid',
       'Layer1FlashStorage', 'Layer2HDD', 'Layer2SSD', 'Layer2Hybrid',
       'Layer2FlashStorage'],inplace=True)
```

In [172]:
```
df.sample(5)
```

Out[172]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	HDD	SSD	Hybrid
308	Lenovo	Notebook	8	128 SSD + 1000 HDD	Nvidia GeForce 940MX	Windows 10	2.30	43636.320	0	0	141.211998	Intel Core i5	1000	128	0
1109	Asus	Gaming	16	128 SSD + 1000 HDD	Nvidia GeForce GTX 960M	Windows 10	2.59	71341.920	0	1	141.211998	Intel Core i7	1000	128	0
700	Dell	Gaming	8	1000 HDD	Nvidia GeForce GTX 1050	Windows 10	2.56	43636.320	0	0	141.211998	Intel Core i5	1000	0	0
349	Dell	Ultrabook	8	1000 HDD	AMD Radeon 530	Windows 10	1.90	35324.640	0	0	141.211998	Intel Core i5	1000	0	0
1049	Asus	Netbook	4	16 Flash Storage	Intel HD Graphics 400	Chrome OS	1.20	15339.312	0	0	135.094211	Other Intel Processor	0	0	0

In [173]:
```
df.drop(columns=['Memory'],inplace=True)
df.sample(5)
```

Out[173]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	HDD	SSD	Hybrid	Flash_Stora
609	Acer	Notebook	4	Intel HD Graphics 405	Windows 10	1.60	18594.72	0	0	111.935204	Other Intel Processor	0	0	0	
463	Lenovo	Notebook	8	AMD Radeon R7 M460	No OS	1.50	42570.72	0	1	157.350512	Intel Core i7	0	512	0	
1172	Asus	Notebook	4	Intel HD Graphics	Windows 10	2.20	19660.32	0	0	100.454670	Other Intel Processor	500	0	0	
94	Asus	Ultrabook	8	Intel HD Graphics 620	Windows 10	1.25	55890.72	0	0	157.350512	Intel Core i7	0	256	0	
231	HP	Notebook	4	AMD Radeon R2	Windows 10	2.10	17582.40	0	0	100.454670	AMD Processor	500	0	0	

In [174]:
```
df.corr()['Price']
```

Out[174]:
```
Ram              0.743007
Weight           0.210370
Price            1.000000
TouchScreen      0.191226
IPS              0.252208
PPI              0.473487
HDD             -0.096441
SSD              0.670799
Hybrid           0.007989
Flash_Storage         NaN
Name: Price, dtype: float64
```

Based on the correlation we observe that `Hybrid` and `Flash Storage` are almost negligible,so we can simply drop them off,where as HDD and SDD are having good correlation,we find that HDD has -ve relation with Price,and that's true,if the price of laptop is increasing there is more probability that the laptop is gonna use SDD instead of HDD and vice versa as well

In [175]:
```
df.columns
```

Out[175]:
```
Index(['Company', 'TypeName', 'Ram', 'Gpu', 'OpSys', 'Weight', 'Price',
       'TouchScreen', 'IPS', 'PPI', 'CPU_name', 'HDD', 'SSD', 'Hybrid',
       'Flash_Storage'],
      dtype='object')
```

In [176]:
```
df.drop(columns = ['Hybrid','Flash_Storage'],inplace=True)
df.head()
```

Out[176]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	HDD	SSD
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256

Analysis on GPU

In [177]:
```
df['Gpu'].value_counts()
```

Out[177]:
```
Intel HD Graphics 620      281
Intel HD Graphics 520      185
Intel UHD Graphics 620      68
Nvidia GeForce GTX 1050     66
Nvidia GeForce GTX 1060     48
                          ...
Nvidia GeForce 960M          1
AMD Radeon R7 M360           1
AMD Radeon Pro 455           1
AMD Radeon R9 M385           1
AMD R17M-M1-70               1
Name: Gpu, Length: 110, dtype: int64
```

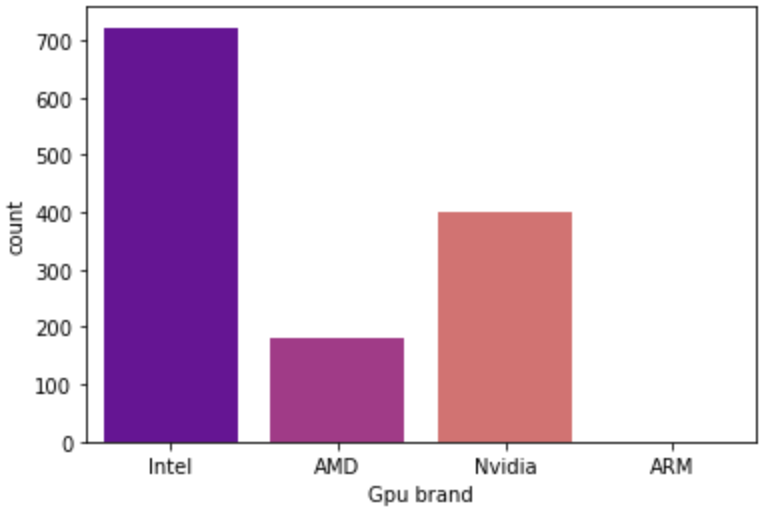
Here as we are having less data regarding the laptops,its better that we focus on `GPU brands` instead focusing on the values which are present there beside them,we will focus on the `brands`

In [178]:
```
# this is what we will be doing,extracting the brands
a = df['Gpu'].iloc[1]
print(a.split()[0])
```

```
Intel
```

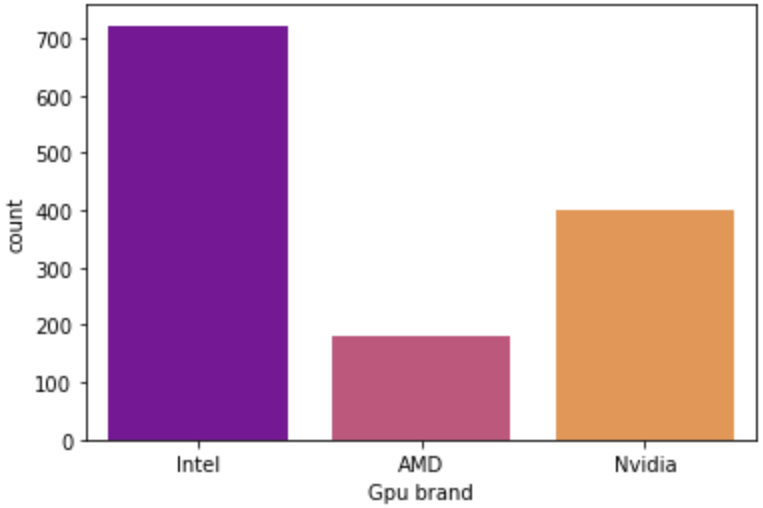
In [179]:
```
df['Gpu brand'] = df['Gpu'].apply(lambda x:x.split()[0])
sn.countplot(df['Gpu brand'],palette='plasma')
```

Out[179]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120c5c8f908>
```



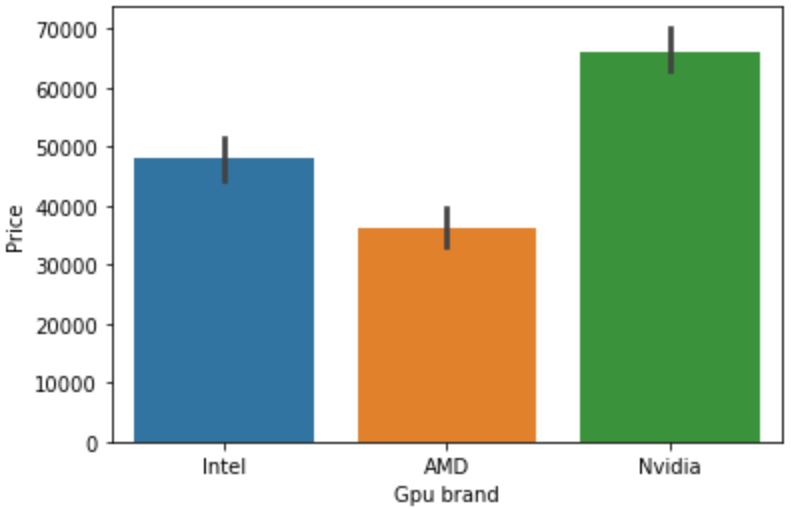
In [180]:
```
# removing the "ARM" tuple

df = df[df['Gpu brand']!='ARM']
sn.countplot(df['Gpu brand'],palette='plasma')
```

Out[180]: <matplotlib.axes._subplots.AxesSubplot at 0x120c5d0d388>



In [181]:
```
# price-GPU analysis,i used np.median inorder to check if there is any
# inpact of outlier or not

sn.barplot(df['Gpu brand'],df['Price'],estimator=np.median)
```

Out[181]: <matplotlib.axes._subplots.AxesSubplot at 0x120c5d5fc08>



In [182]:
```
df = df.drop(columns=['Gpu'])
df.head()
```

Out[182]:

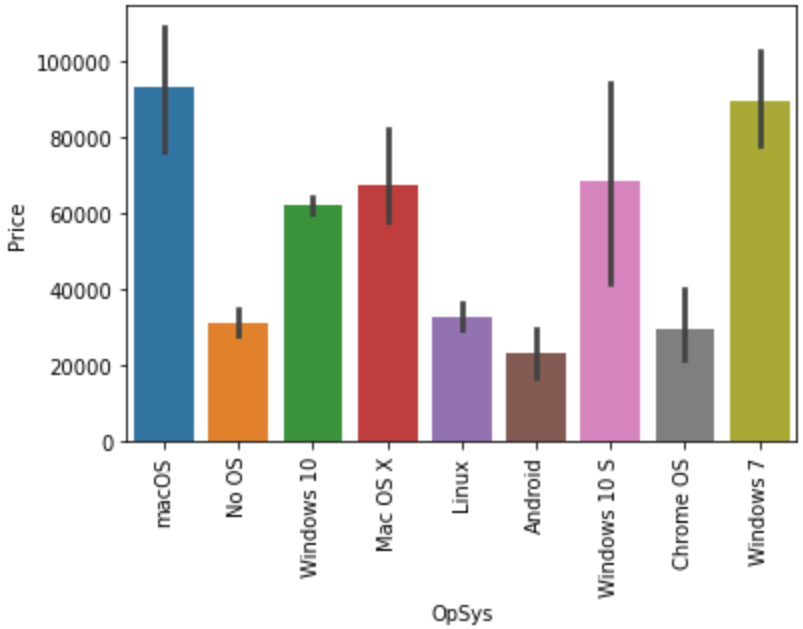
	Company	TypeName	Ram	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	HDD	SSD	Gpu brand
0	Apple	Ultrabook	8	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	Intel
1	Apple	Ultrabook	8	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	Intel
2	HP	Notebook	8	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256	Intel
3	Apple	Ultrabook	16	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512	AMD
4	Apple	Ultrabook	8	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256	Intel

Operating System analysis

In [183]:
```
df['OpSys'].value_counts()
```

Out[183]:
```
Windows 10      1072
No OS             66
Linux             62
Windows 7         45
Chrome OS         26
macOS             13
Windows 10 S       8
Mac OS X           8
Android            2
Name: OpSys, dtype: int64
```

In [184]:
```
sn.barplot(df['OpSys'],df['Price'])
plt.xticks(rotation = 'vertical')
plt.show()
```



In [185]:
```
df['OpSys'].unique()
```

Out[185]:
```
array(['macOS', 'No OS', 'Windows 10', 'Mac OS X', 'Linux', 'Android',
       'Windows 10 S', 'Chrome OS', 'Windows 7'], dtype=object)
```

In [186]:
```
# club {Windows 10,Windows 7,Windows 7 S}-->Windows
# club {macOS,mac OS X}--> mac
# else return Others

def setcategory(text):

    if text=='Windows 10' or text=='Windows 7' or text=='Windows 10 S':
        return 'Windows'

    elif text=='Mac OS X' or text=='macOS':
        return 'Mac'

    else:
        return 'Other'


df['OpSys'] = df['OpSys'].apply(lambda x:setcategory(x))

df.head()
```

Out[186]:

	Company	TypeName	Ram	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	HDD	SSD	Gpu brand
0	Apple	Ultrabook	8	Mac	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	Intel
1	Apple	Ultrabook	8	Mac	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	Intel
2	HP	Notebook	8	Other	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256	Intel
3	Apple	Ultrabook	16	Mac	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512	AMD
4	Apple	Ultrabook	8	Mac	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256	Intel

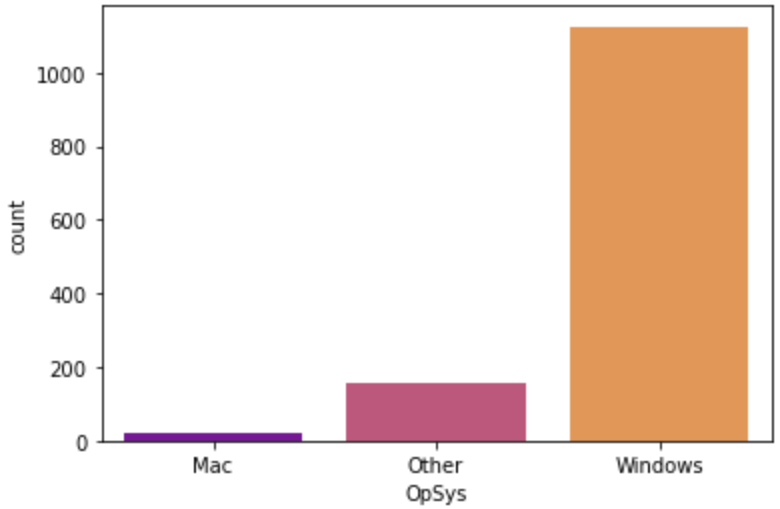
In [187]:
```
df.sample(5)
```

Out[187]:

	Company	TypeName	Ram	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	HDD	SSD	Gpu brand
1094	HP	Netbook	4	Windows	2.40	85194.72	0	0	125.367428	Intel Core i5	0	128	Intel
538	HP	Gaming	12	Windows	2.62	95850.72	0	0	127.335675	Intel Core i7	1000	0	Nvidia
995	Asus	Notebook	8	Windows	1.40	61272.00	0	0	276.053530	Intel Core i5	0	256	Intel
879	HP	Notebook	4	Windows	2.04	44701.92	0	0	141.211998	Intel Core i5	0	256	Intel
515	Asus	Netbook	2	Windows	1.10	13053.60	0	0	135.094211	Other Intel Processor	0	0	Intel

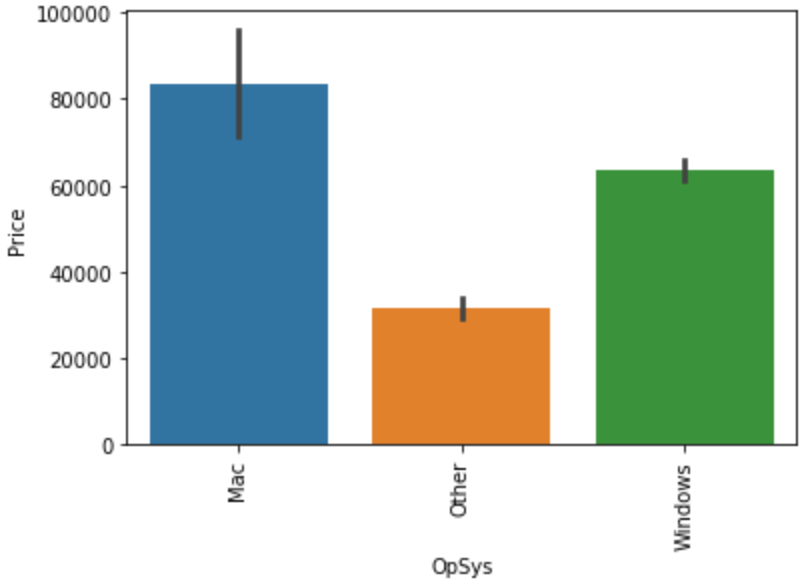
In [188]:
```
sn.countplot(df['OpSys'],palette='plasma')
```

Out[188]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120c5e7cb88>
```



In [189]:
```
sn.barplot(x = df['OpSys'],y = df['Price'])
plt.xticks(rotation = 'vertical')
```

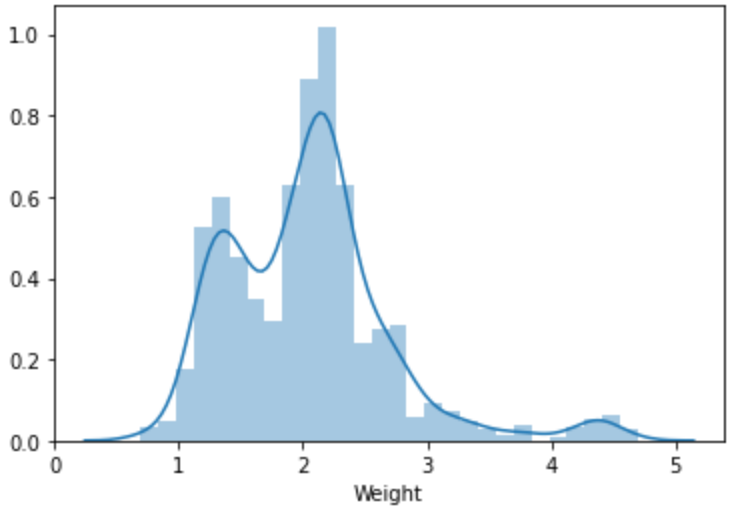
Out[189]:
```
(array([0, 1, 2]), <a list of 3 Text major ticklabel objects>)
```



Weight analysis

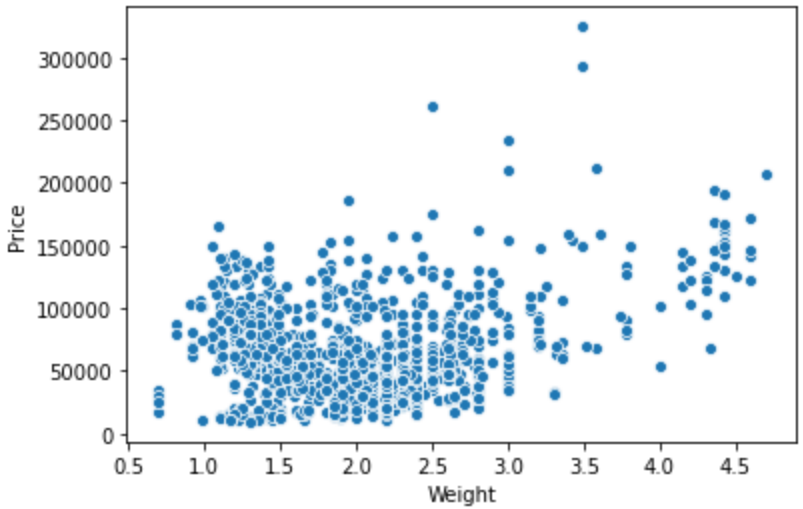
In [190]:
```
sn.distplot(df['Weight'])
```

Out[190]: <matplotlib.axes._subplots.AxesSubplot at 0x120c5f23cc8>



In [191]:
```
sn.scatterplot(df['Weight'],df['Price'])
```

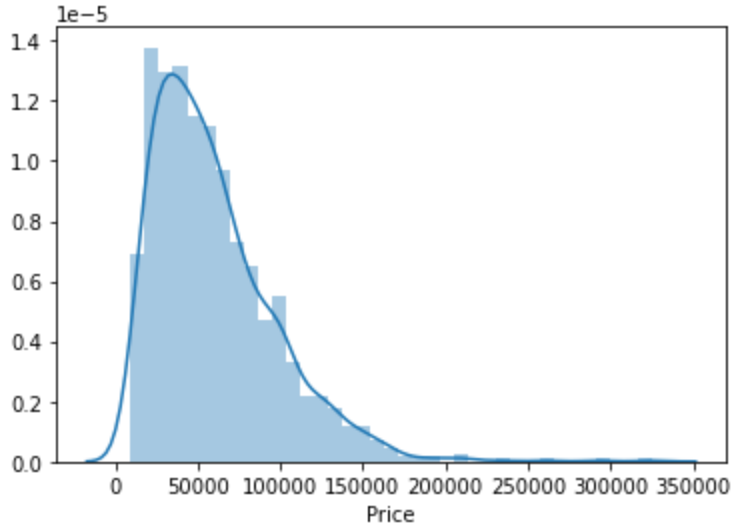
Out[191]: <matplotlib.axes._subplots.AxesSubplot at 0x120c5fd5508>



Price Analysis

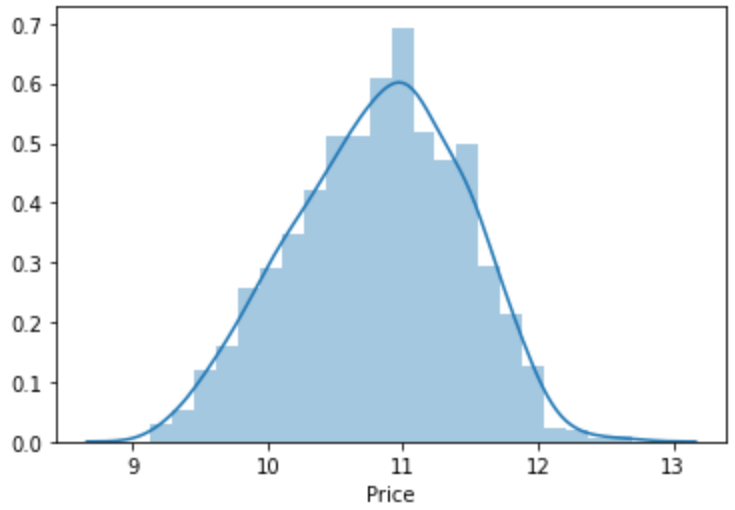
In [192]:
```
sn.distplot(df['Price'])
```

Out[192]: <matplotlib.axes._subplots.AxesSubplot at 0x120c5e07608>



In [193]:
```
# so if we apply np.log to the Price col we get a gaussian distibution

sn.distplot(np.log(df['Price']))
```

Out[193]: <matplotlib.axes._subplots.AxesSubplot at 0x120bc51c4c8>

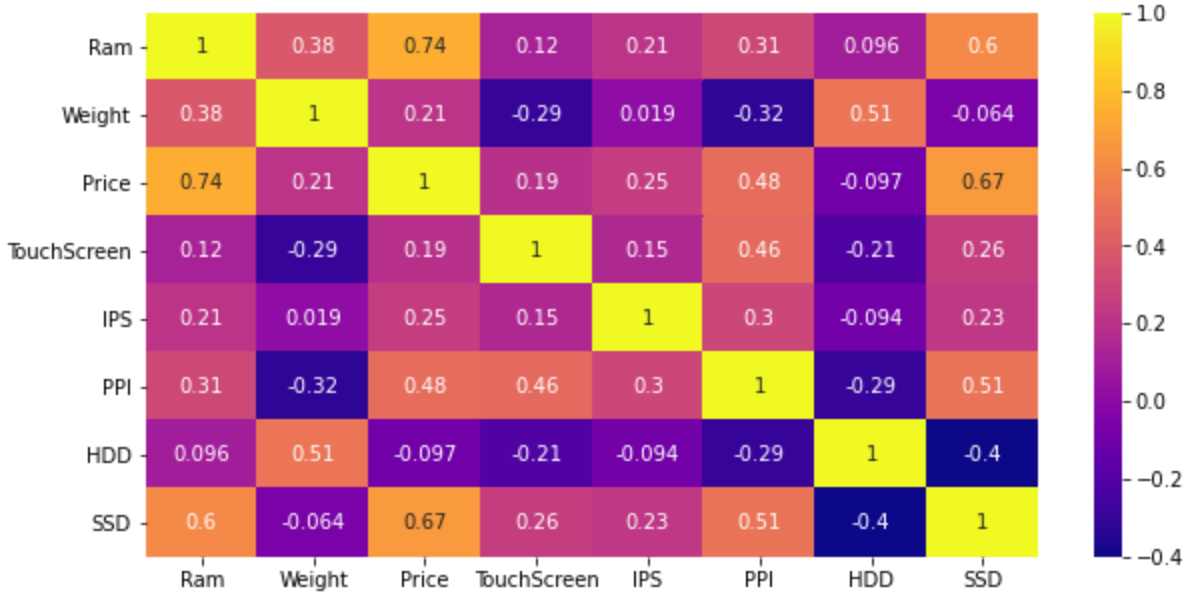


In [194]:
```
## correlation with price

df.corr()['Price']
```

Out[194]:
```
Ram            0.742905
Weight         0.209867
Price          1.000000
TouchScreen    0.192917
IPS            0.253320
PPI            0.475368
HDD           -0.096891
SSD            0.670660
Name: Price, dtype: float64
```

In [195]:
```
plt.figure(figsize=(10,5))
sn.heatmap(df.corr(),annot=True,cmap='plasma')
```

Out[195]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120c48d8908>
```



Model Building

In [196]:
```
test = np.log(df['Price'])
train = df.drop(['Price'],axis = 1)
```

In [197]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from sklearn import metrics
from sklearn.model_selection import RandomizedSearchCV
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn import tree
```

In [198]:
```
X_train, X_test, y_train, y_test = train_test_split(train,test,
                                                    test_size=0.15,random_state=2)

X_train.shape,X_test.shape
```

Out[198]:
```
((1106, 12), (196, 12))
```

There's a Class which we imported named as `Column Trasnformer` we use this widely while building our models using `Pipelines` ,so for this we have to get the index numbers of the columns which are having categorical variables

In [199]:
```
mapper = {i:value for i,value in enumerate(X_train.columns)}
mapper
```

Out[199]:
```
{0: 'Company',
 1: 'TypeName',
 2: 'Ram',
 3: 'OpSys',
 4: 'Weight',
 5: 'TouchScreen',
 6: 'IPS',
 7: 'PPI',
 8: 'CPU_name',
 9: 'HDD',
 10: 'SSD',
 11: 'Gpu brand'}
```

Linear Regression

In [200]:
```
# we will apply one hot encoding on the columns with this indices-->[0,1,3,8,11]
# the remainder we keep as passthrough i.e no other col must get effected
# except the ones undergoing the transformation!

step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],remainder='passthrough')

step2 = LinearRegression()

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8073277448418599
MAE 0.21017827976428746
```

In [201]:
```
## now mae is 0.21 so if you want to check how much difference is there do this

## we see there is a difference of 1.23 only as per the orignal value
## that is our model predicts +-0.21 more/less than the original price!

np.exp(0.21)
```

Out[201]:
```
1.2336780599567432
```

Ridge Regression

In [202]:
```
# we will apply one hot encoding on the columns with this indices-->[0,1,3,8,11]
# the remainder we keep as passthrough i.e no other col must get effected
# except the ones undergoing the transformation!

step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],remainder='passthrough')

step2 = Ridge(alpha=10)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.812733103131181
MAE 0.20926802242582962
```

LassoRegression

In [203]:
```
# we will apply one hot encoding on the columns with this indices-->[0,1,3,8,11]
# the remainder we keep as passthrough i.e no other col must get effected
# except the ones undergoing the transformation!

step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],remainder='passthrough')

step2 = Lasso(alpha=0.001)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8071857196899418
MAE 0.21114350716913166
```

Decision Tree

In [204]:
```
# we will apply one hot encoding on the columns with this indices-->[0,1,3,8,11]
# the remainder we keep as passthrough i.e no other col must get effected
# except the ones undergoing the transformation!

step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],remainder='passthrough')

step2 = DecisionTreeRegressor(max_depth=8)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8437664803528917
MAE 0.1808159839491631
```

Random Forest

In [207]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],remainder='passthrough')

step2 = RandomForestRegressor(n_estimators=100,
                              random_state=3,
                              max_samples=0.5,
                              max_features=0.75,
                              max_depth=15)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8840242410385177
MAE 0.15974965172059183
```

In [208]:
```
import pickle

pickle.dump(df,open('df.pkl','wb'))
pickle.dump(pipe,open('pipe.pkl','wb'))
```

In [88]:
```
train.head()
```

Out[88]:

	Company	TypeName	Ram	OpSys	Weight	TouchScreen	IPS	PPI	CPU_name	HDD	SSD	Gpu brand
0	Apple	Ultrabook	8	Mac	1.37	0	1	226.983005	Intel Core i5	0	128	Intel
1	Apple	Ultrabook	8	Mac	1.34	0	0	127.677940	Intel Core i5	0	0	Intel
2	HP	Notebook	8	Other	1.86	0	0	141.211998	Intel Core i5	0	256	Intel
3	Apple	Ultrabook	16	Mac	1.83	0	1	220.534624	Intel Core i7	0	512	AMD
4	Apple	Ultrabook	8	Mac	1.37	0	1	226.983005	Intel Core i5	0	256	Intel

In [89]:
```
train.to_csv('traineddata.csv',index=None)
```

Hyperparameter Tuning for Random Forest

In [90]:
```
indexlist = [0,1,3,8,11]
transformlist = []
for key,value in mapper.items():
    if key in indexlist:
        transformlist.append(value)

transformlist
```

Out[90]:
```
['Company', 'TypeName', 'OpSys', 'CPU_name', 'Gpu brand']
```

In [91]:
```
train = pd.get_dummies(train,columns=transformlist,drop_first=True)
train.head()
```

Out[91]:

	Ram	Weight	TouchScreen	IPS	PPI	HDD	SSD	Company_Apple	Company_Asus	Company_Chuwi	...	TypeName_Ultrabook	TypeName
0	8	1.37	0	1	226.983005	0	128	1	0	0	...	1	
1	8	1.34	0	0	127.677940	0	0	1	0	0	...	1	
2	8	1.86	0	0	141.211998	0	256	0	0	0	...	0	
3	16	1.83	0	1	220.534624	0	512	1	0	0	...	1	
4	8	1.37	0	1	226.983005	0	256	1	0	0	...	1	

5 rows × 38 columns

In [92]:
```
X_train, X_test, y_train, y_test = train_test_split(train,test,
                                                    test_size=0.15,random_state=2)

X_train.shape,X_test.shape
```

Out[92]:
```
((1106, 38), (196, 38))
```

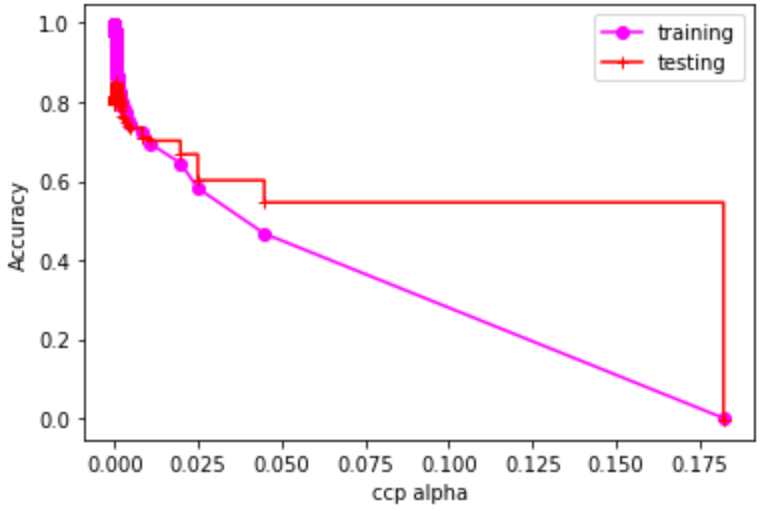
In [93]:
```
reg = DecisionTreeRegressor(random_state=0)
reg.fit(X_train,y_train)
plt.figure(figsize=(16,9))
tree.plot_tree(reg,filled=True,feature_names=train.columns)
```

Out[93]:
```
[Text(434.7535615670393, 478.60434782608695, 'Ram <= 7.0\nmse = 0.39\nsamples = 1106\nvalue = 10.821'),
 Text(197.29260701854207, 457.3330434782609, 'CPU_name_Other Intel Processor <= 0.5\nmse = 0.192\nsamples = 362\nva
lue = 10.209'),
 Text(130.012151081063, 436.0617391304348, 'CPU_name_Intel Core i5 <= 0.5\nmse = 0.13\nsamples = 256\nvalue = 10.38
7'),
 Text(87.3165974964857, 414.7904347826087, 'CPU_name_Intel Core i7 <= 0.5\nmse = 0.08\nsamples = 154\nvalue = 10.23
3'),
 Text(37.29773746569382, 393.5191304347826, 'Weight <= 1.645\nmse = 0.07\nsamples = 135\nvalue = 10.183'),
 Text(6.693460070955218, 372.24782608695654, 'TypeName_Ultrabook <= 0.5\nmse = 0.064\nsamples = 10\nvalue = 10.58
7'),
 Text(4.781042907825156, 350.9765217391304, 'Weight <= 1.395\nmse = 0.02\nsamples = 8\nvalue = 10.478'),
 Text(3.8248343262601243, 329.70521739130436, 'mse = 0.0\nsamples = 1\nvalue = 10.208'),
 Text(5.737251489390187, 329.70521739130436, 'Company_Lenovo <= 0.5\nmse = 0.011\nsamples = 7\nvalue = 10.516'),
 Text(4.781042907825156, 308.43391304347824, 'Weight <= 1.615\nmse = 0.003\nsamples = 6\nvalue = 10.555'),
 Text(1.9124171631300622, 287.1626086956522, 'PPI <= 137.589\nmse = 0.002\nsamples = 3\nvalue = 10.597'),
 Text(0.9562085815650311, 265.89130434782606, 'mse = 0.0\nsamples = 1\nvalue = 10.66'),
 Text(2.8686257446950933, 265.89130434782606, 'PPI <= 161.491\nmse = 0.0\nsamples = 2\nvalue = 10.565'),
 Text(1.9124171631300622, 244.62, 'mse = 0.0\nsamples = 1\nvalue = 10.577'),
 Text(3.8248343262601243, 244.62, 'mse = 0.0\nsamples = 1\nvalue = 10.553'),
```

In [94]:
```
path = reg.cost_complexity_pruning_path(X_train,y_train)
ccp_alphas = path.ccp_alphas
```

In [95]:
```
alphalist = []
for alpha in ccp_alphas:
    reg = DecisionTreeRegressor(random_state=0,ccp_alpha=alpha)
    reg.fit(X_train,y_train)
    alphalist.append(reg)
```

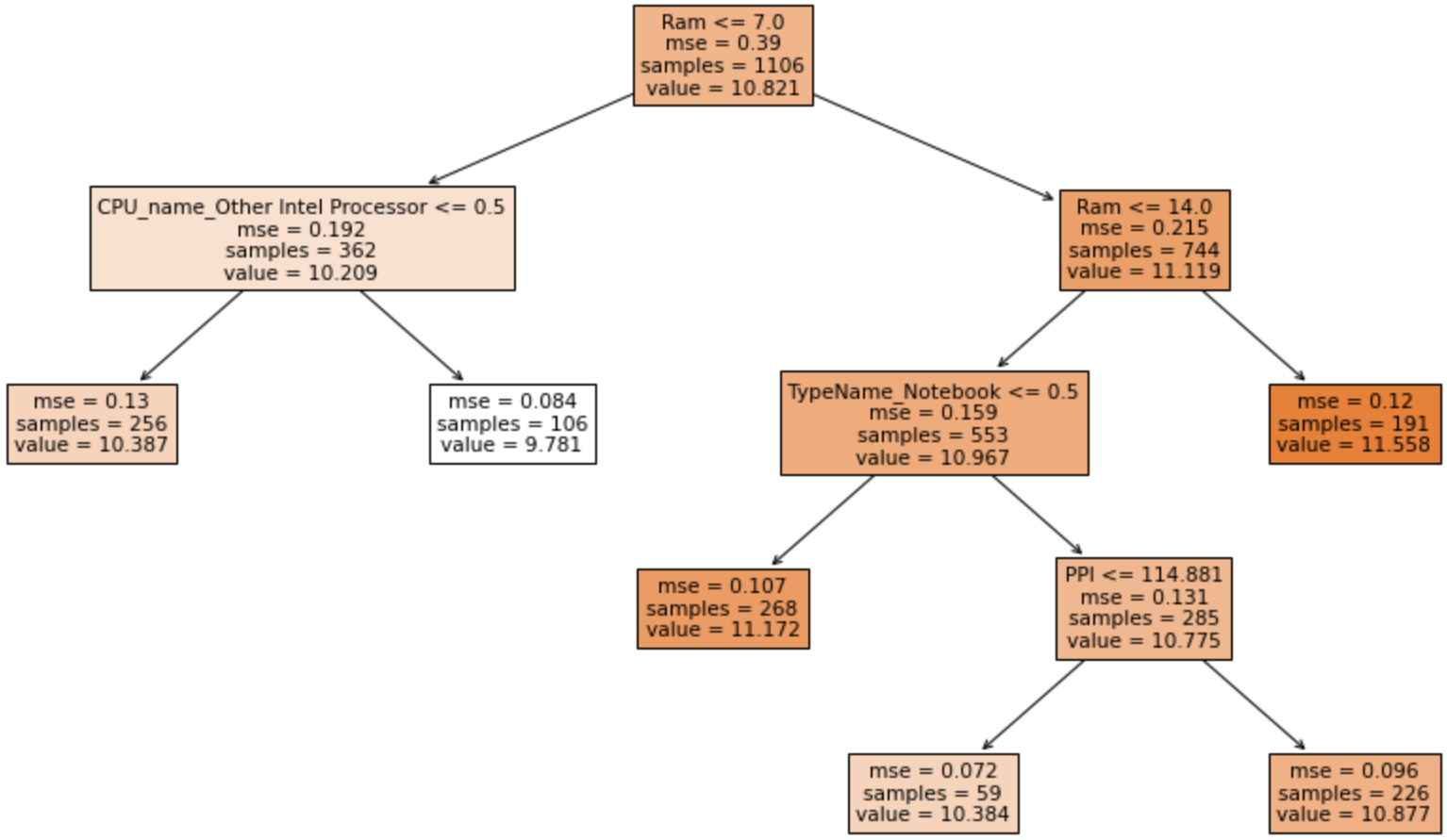
In [96]:
```
train_score = [reg.score(X_train,y_train) for reg in alphalist]
test_score = [reg.score(X_test,y_test) for reg in alphalist]

plt.xlabel('ccp alpha')
plt.ylabel('Accuracy')

plt.plot(ccp_alphas,train_score,marker = 'o',
        label = 'training',color = 'magenta')
plt.plot(ccp_alphas,test_score,marker = '+',
         label = 'testing',color = 'red',drawstyle = 'steps-post')
plt.legend()
plt.show()
```



possible values of alpha can lie between `[0.0025-->0.0075]`

In [97]:
```
reg = DecisionTreeRegressor(random_state=0,ccp_alpha=0.0085)
reg.fit(X_train,y_train)
plt.figure(figsize=(16,9))
tree.plot_tree(reg,filled=True,feature_names=train.columns)
```

Out[97]:
```
[Text(446.4, 440.31600000000003, 'Ram <= 7.0\nmse = 0.39\nsamples = 1106\nvalue = 10.821'),
 Text(223.2, 342.468, 'CPU_name_Other Intel Processor <= 0.5\nmse = 0.192\nsamples = 362\nvalue = 10.209'),
 Text(111.6, 244.62, 'mse = 0.13\nsamples = 256\nvalue = 10.387'),
 Text(334.79999999999995, 244.62, 'mse = 0.084\nsamples = 106\nvalue = 9.781'),
 Text(669.5999999999999, 342.468, 'Ram <= 14.0\nmse = 0.215\nsamples = 744\nvalue = 11.119'),
 Text(558.0, 244.62, 'TypeName_Notebook <= 0.5\nmse = 0.159\nsamples = 553\nvalue = 10.967'),
 Text(446.4, 146.772, 'mse = 0.107\nsamples = 268\nvalue = 11.172'),
 Text(669.5999999999999, 146.772, 'PPI <= 114.881\nmse = 0.131\nsamples = 285\nvalue = 10.775'),
 Text(558.0, 48.924000000000035, 'mse = 0.072\nsamples = 59\nvalue = 10.384'),
 Text(781.1999999999999, 48.924000000000035, 'mse = 0.096\nsamples = 226\nvalue = 10.877'),
 Text(781.1999999999999, 244.62, 'mse = 0.12\nsamples = 191\nvalue = 11.558')]
```



In [98]:
```
params=  {

    'RandomForest':{
        'model' : RandomForestRegressor(),
        'params':{
            'n_estimators':[int(x) for x in np.linspace(100,1200,10)],
            'criterion':["mse", "mae"],
            'max_depth':[int(x) for x in np.linspace(1,30,5)],
            'max_features':['auto','sqrt','log2'],
            'ccp_alpha':[x for x in np.linspace(0.0025,0.0125,5)],
            'min_samples_split':[2,5,10,14],
            'min_samples_leaf':[2,5,10,14],
        }
    },
    'Decision Tree':{
        'model':DecisionTreeRegressor(),
        'params':{
            'criterion':["mse", "mae"],
            'max_depth':[int(x) for x in np.linspace(1,30,5)],
            'max_features':['auto','sqrt','log2'],
            'ccp_alpha':[x for x in np.linspace(0.0025,0.0125,5)],
            'min_samples_split':[2,5,10,14],
            'min_samples_leaf':[2,5,10,14],
        }
    }
}
```

In [99]:
```
scores = []
for modelname,mp in params.items():
    clf = RandomizedSearchCV(mp['model'],
                             param_distributions=mp['params'],cv = 5,
                             n_iter=10,scoring='neg_mean_squared_error',verbose=2)
    clf.fit(X_train,y_train)
    scores.append({
        'model_name':modelname,
        'best_score':clf.best_score_,
        'best_estimator':clf.best_estimator_,
    })
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END ccp_alpha=0.0075, criterion=mae, max_depth=8, max_features=sqrt, min_samples_leaf=14, min_samples_split=5,
n_estimators=100; total time=   0.3s
[CV] END ccp_alpha=0.0075, criterion=mae, max_depth=8, max_features=sqrt, min_samples_leaf=14, min_samples_split=5,
n_estimators=100; total time=   0.3s
[CV] END ccp_alpha=0.0075, criterion=mae, max_depth=8, max_features=sqrt, min_samples_leaf=14, min_samples_split=5,
n_estimators=100; total time=   0.4s
[CV] END ccp_alpha=0.0075, criterion=mae, max_depth=8, max_features=sqrt, min_samples_leaf=14, min_samples_split=5,
n_estimators=100; total time=   0.3s
[CV] END ccp_alpha=0.0075, criterion=mae, max_depth=8, max_features=sqrt, min_samples_leaf=14, min_samples_split=5,
n_estimators=100; total time=   0.3s
[CV] END ccp_alpha=0.005, criterion=mse, max_depth=8, max_features=log2, min_samples_leaf=10, min_samples_split=5,
n_estimators=955; total time=   1.0s
[CV] END ccp_alpha=0.005, criterion=mse, max_depth=8, max_features=log2, min_samples_leaf=10, min_samples_split=5,
n_estimators=955; total time=   1.0s
[CV] END ccp_alpha=0.005, criterion=mse, max_depth=8, max_features=log2, min_samples_leaf=10, min_samples_split=5,
n_estimators=955; total time=   1.0s
[CV] END ccp_alpha=0.005, criterion=mse, max_depth=8, max_features=log2, min_samples_leaf=10, min_samples_split=5,
n_estimators=955; total time=   1.2s
```

In [100]:
```
scores_df = pd.DataFrame(scores,columns=['model_name','best_score','best_estimator'])
scores_df
```

Out[100]:

	model_name	best_score	best_estimator
0	RandomForest	-0.097905	(DecisionTreeRegressor(ccp_alpha=0.005, max_de...
1	Decision Tree	-0.094662	DecisionTreeRegressor(ccp_alpha=0.005, criteri...

In [101]:
```
scores
```

Out[101]:
```
[{'model_name': 'RandomForest',
  'best_score': -0.09790477698157705,
  'best_estimator': RandomForestRegressor(ccp_alpha=0.005, max_depth=8, max_features='log2',
                        min_samples_leaf=10, min_samples_split=5,
                        n_estimators=955)},
 {'model_name': 'Decision Tree',
  'best_score': -0.09466171354689881,
  'best_estimator': DecisionTreeRegressor(ccp_alpha=0.005, criterion='mae', max_depth=15,
                        max_features='auto', min_samples_leaf=5,
                        min_samples_split=10)}]
```

In [102]:
```
rf = RandomForestRegressor(ccp_alpha=0.0025, max_depth=22, min_samples_leaf=14,
                        min_samples_split=5, n_estimators=1200)

rf.fit(X_train,y_train)
ypred = rf.predict(X_test)
print(metrics.r2_score(y_test,y_pred))
```

```
0.8840242410385177
```

Prediction on the whole Dataset

In [103]:
```
predicted = []
testtrain = np.array(train)
for i in range(len(testtrain)):
    predicted.append(rf.predict([testtrain[i]]))

predicted
```

...

In [105]:
```
# as we transformed our price variable to np.log
# we have to retranform it from np.log-->np.exp inorder to get the result

ans = [np.exp(predicted[i][0]) for i in range(len(predicted))]
```

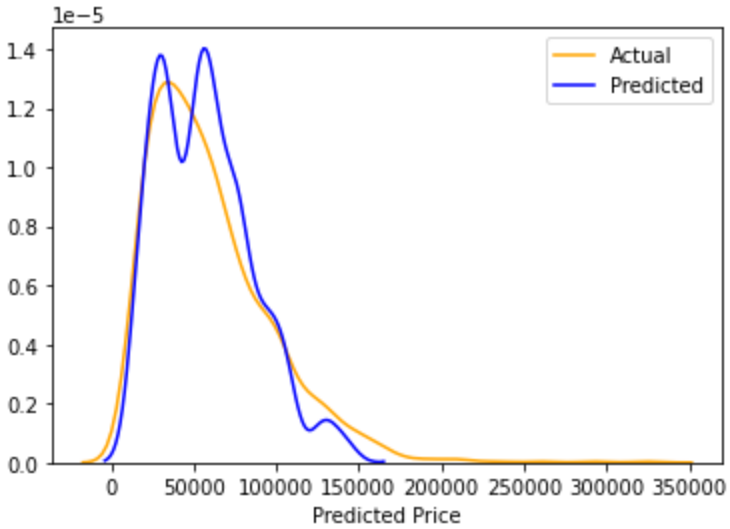
In [106]:
```
df['Predicted Price'] = np.array(ans)
df
```

Out[106]:

	Company	TypeName	Ram	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	HDD	SSD	Gpu brand	Predicted Price
0	Apple	Ultrabook	8	Mac	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	Intel	76157.830254
1	Apple	Ultrabook	8	Mac	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	Intel	71583.255543
2	HP	Notebook	8	Other	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256	Intel	48400.360553
3	Apple	Ultrabook	16	Mac	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512	AMD	105866.345413
4	Apple	Ultrabook	8	Mac	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256	Intel	77610.078256
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	4	Windows	1.80	33992.6400	1	1	157.350512	Intel Core i7	0	128	Intel	31429.732242
1299	Lenovo	2 in 1 Convertible	16	Windows	1.30	79866.7200	1	1	276.053530	Intel Core i7	0	512	Intel	105789.258479
1300	Lenovo	Notebook	2	Windows	1.50	12201.1200	0	0	111.935204	Other Intel Processor	0	0	Intel	17650.740005
1301	HP	Notebook	6	Windows	2.19	40705.9200	0	0	100.454670	Intel Core i7	1000	0	AMD	28714.685145
1302	Asus	Notebook	4	Windows	2.20	19660.3200	0	0	100.454670	Other Intel Processor	500	0	Intel	17723.721745

1302 rows × 14 columns

In [107]:
```
sn.distplot(df['Price'],hist=False,color='orange',label='Actual')
sn.distplot(df['Predicted Price'],hist=False,color='blue',label='Predicted')
plt.legend()
plt.show()
```



Random Forest Regressor version_2

In [108]:
```
rf1 = RandomForestRegressor(n_estimators=100,
                            random_state=3,
                            max_samples=0.5,
                            max_features=0.75,
                            max_depth=15)

rf1.fit(X_train,y_train)
print(f'R2 score : {metrics.r2_score(y_test,rf1.predict(X_test))}')
```

```
R2 score : 0.8876880244703835
```

In [109]:
```
predicted = []
testtrain = np.array(train)
for i in range(len(testtrain)):
    predicted.append(rf1.predict([testtrain[i]]))

predicted
```

...

In [110]:
```
# as we transformed our price variable to np.log
# we have to retranform it from np.log-->np.exp inorder to get the result

ans = [np.exp(predicted[i][0]) for i in range(len(predicted))]
```

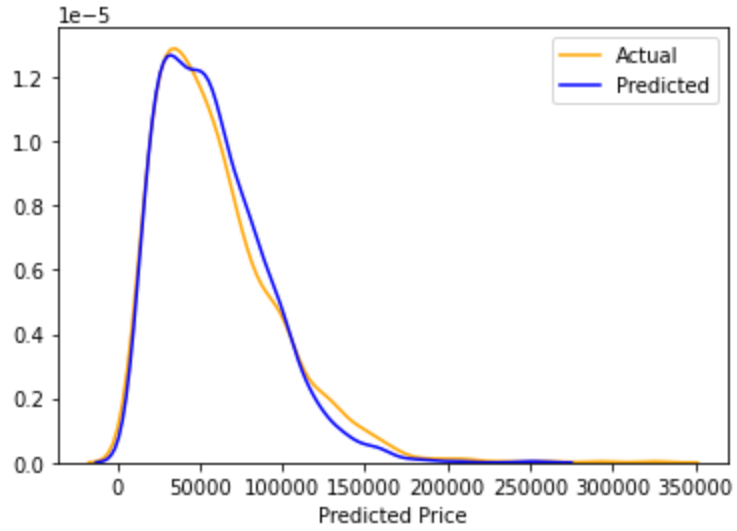
In [111]:
```
data = df.copy()
data['Predicted Price'] = np.array(ans)
data
```

Out[111]:

	Company	TypeName	Ram	OpSys	Weight	Price	TouchScreen	IPS	PPI	CPU_name	HDD	SSD	Gpu brand	Predicted Price
0	Apple	Ultrabook	8	Mac	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	Intel	72954.959187
1	Apple	Ultrabook	8	Mac	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	Intel	53469.111472
2	HP	Notebook	8	Other	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256	Intel	38363.798401
3	Apple	Ultrabook	16	Mac	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512	AMD	135714.258702
4	Apple	Ultrabook	8	Mac	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256	Intel	82764.044180
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	4	Windows	1.80	33992.6400	1	1	157.350512	Intel Core i7	0	128	Intel	36457.409553
1299	Lenovo	2 in 1 Convertible	16	Windows	1.30	79866.7200	1	1	276.053530	Intel Core i7	0	512	Intel	87623.187293
1300	Lenovo	Notebook	2	Windows	1.50	12201.1200	0	0	111.935204	Other Intel Processor	0	0	Intel	12778.590180
1301	HP	Notebook	6	Windows	2.19	40705.9200	0	0	100.454670	Intel Core i7	1000	0	AMD	38074.548690
1302	Asus	Notebook	4	Windows	2.20	19660.3200	0	0	100.454670	Other Intel Processor	500	0	Intel	19569.661649

1302 rows × 14 columns

In [112]:
```
sn.distplot(data['Price'],hist=False,color='orange',label='Actual')
sn.distplot(data['Predicted Price'],hist=False,color='blue',label='Predicted')
plt.legend()
plt.show()
```



In [113]:
```
import pickle
file = open('laptoppricepredictor.pkl','wb')
pickle.dump(rf1,file)
file.close()
```

In [ ]:
```
X_train.iloc[0]
```

In [ ]:
```

```

In [ ]:
```

```