

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
In [2]: df = pd.read_csv('laptop_data.csv')
```

```
In [10]: df.sample(5).T
```

```
Out[10]:
```

	1273	389	1259	15	162
Unnamed: 0	1273	389	1259	15	162
Company	HP	Lenovo	MSI	Apple	Dell
TypeName	Notebook	2 in 1 Convertible	Gaming	Ultrabook	Notebook
Inches	15.6	14.0	15.6	13.3	15.6
ScreenResolution	1366x768	Touchscreen 2560x1440	Full HD 1920x1080	IPS Panel Retina Display 2560x1600	Full HD 1920x1080
Cpu	Intel Core i7 6500U 2.5GHz	Intel Core i7 7500U 2.7GHz	Intel Core i7 6700HQ 2.6GHz	Intel Core i5 2.3GHz	Intel Core i5 8250U 1.6GHz
Ram	6GB	16GB	8GB	8GB	8GB
Memory	1TB HDD	512GB SSD	128GB SSD + 1TB HDD	256GB SSD	256GB SSD
Gpu	AMD Radeon R5 M330	Intel HD Graphics 620	Nvidia GeForce GTX 960M	Intel Iris Plus Graphics 640	AMD Radeon 520
OpSys	Windows 10	Windows 10	Windows 10	macOS	Windows 10
Weight	2.19kg	1.42kg	2.4kg	1.37kg	2.13kg
Price	40705.92	133679.52	65481.12	80908.344	38787.84

```
In [11]: df.shape
```

```
Out[11]: (1303, 12)
```

```
In [12]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0            1303 non-null   int64
1   Company               1303 non-null   object
2   TypeName              1303 non-null   object
3   Inches                1303 non-null   float64
4   ScreenResolution      1303 non-null   object
5   Cpu                   1303 non-null   object
6   Ram                   1303 non-null   object
7   Memory                1303 non-null   object
8   Gpu                   1303 non-null   object
9   OpSys                 1303 non-null   object
10  Weight                 1303 non-null   object
11  Price                  1303 non-null   float64
dtypes: float64(2), int64(1), object(9)
memory usage: 122.3+ KB
```

```
In [13]: df.duplicated().sum()
```

Out[13]: 0

```
In [14]: df.isnull().sum()
```

Out[14]:

Unnamed: 0	0
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 [15]: df['Unnamed: 0'].value_counts()
```

Out[15]:

Unnamed: 0	
0	1
856	1
874	1
873	1
872	1
..	
433	1
432	1
431	1
430	1
1302	1

Name: count, Length: 1303, dtype: int64

```
In [16]: df.drop(columns=['Unnamed: 0'],inplace=True)
```

```
In [20]: #help(df['Ram'].str.replace)
```

```
In [17]: df.head()
```

Out[17]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	P
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	71378.6
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5
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	30636.0
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	135195.3
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	96095.8

```
In [21]: df['Ram'] = df['Ram'].str.replace('GB', '')
df['Weight'] = df['Weight'].str.replace('kg', '')

In [22]: df.head()

Out[22]:
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Pr
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	71378.66
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.52
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	30636.00
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	135195.33
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	96095.80

```
In [24]: df['Ram'] = df['Ram'].astype('int32')
df['Weight'] = df['Weight'].astype('float32')
```

```
In [25]: df.info()

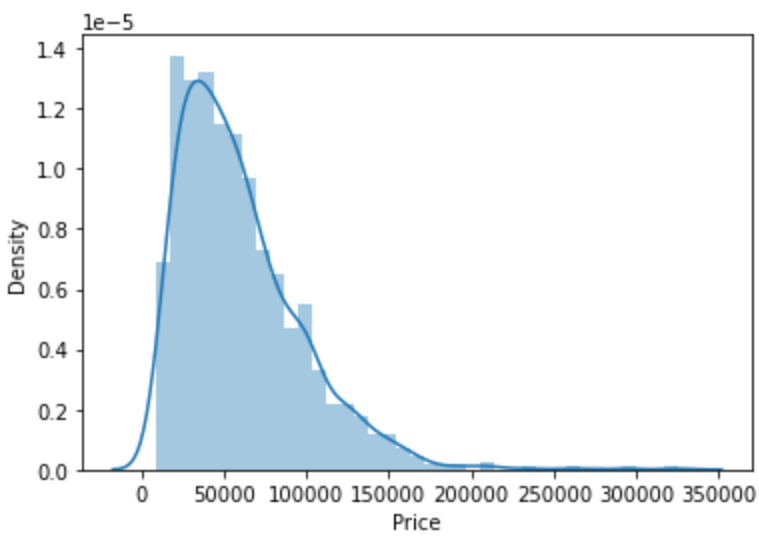
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Company               1303 non-null   object
1   TypeName              1303 non-null   object
2   Inches                1303 non-null   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
```

```
In [26]: import seaborn as sns
```

```
In [18]: sns.distplot(df['Price'])

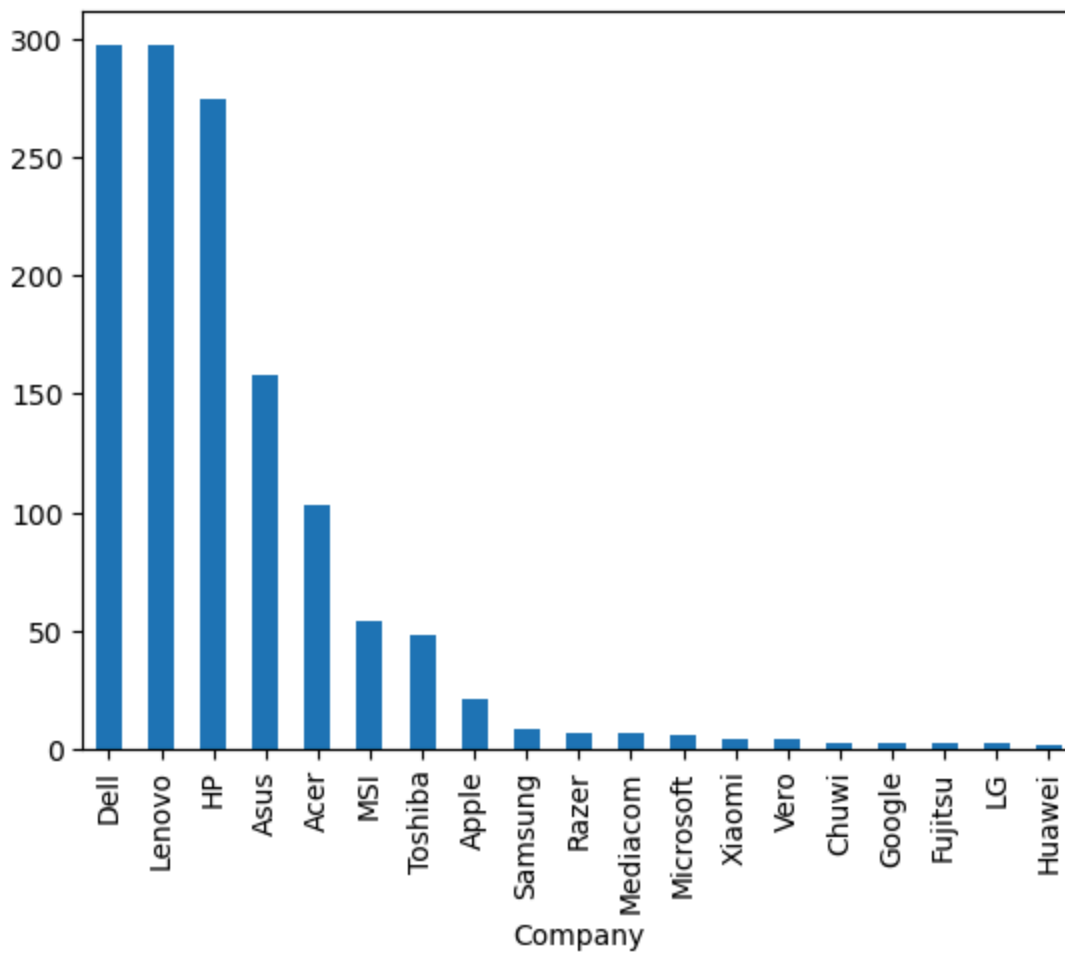
C:\Users\91842\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

Out[18]: <AxesSubplot:xlabel='Price', ylabel='Density'>
```

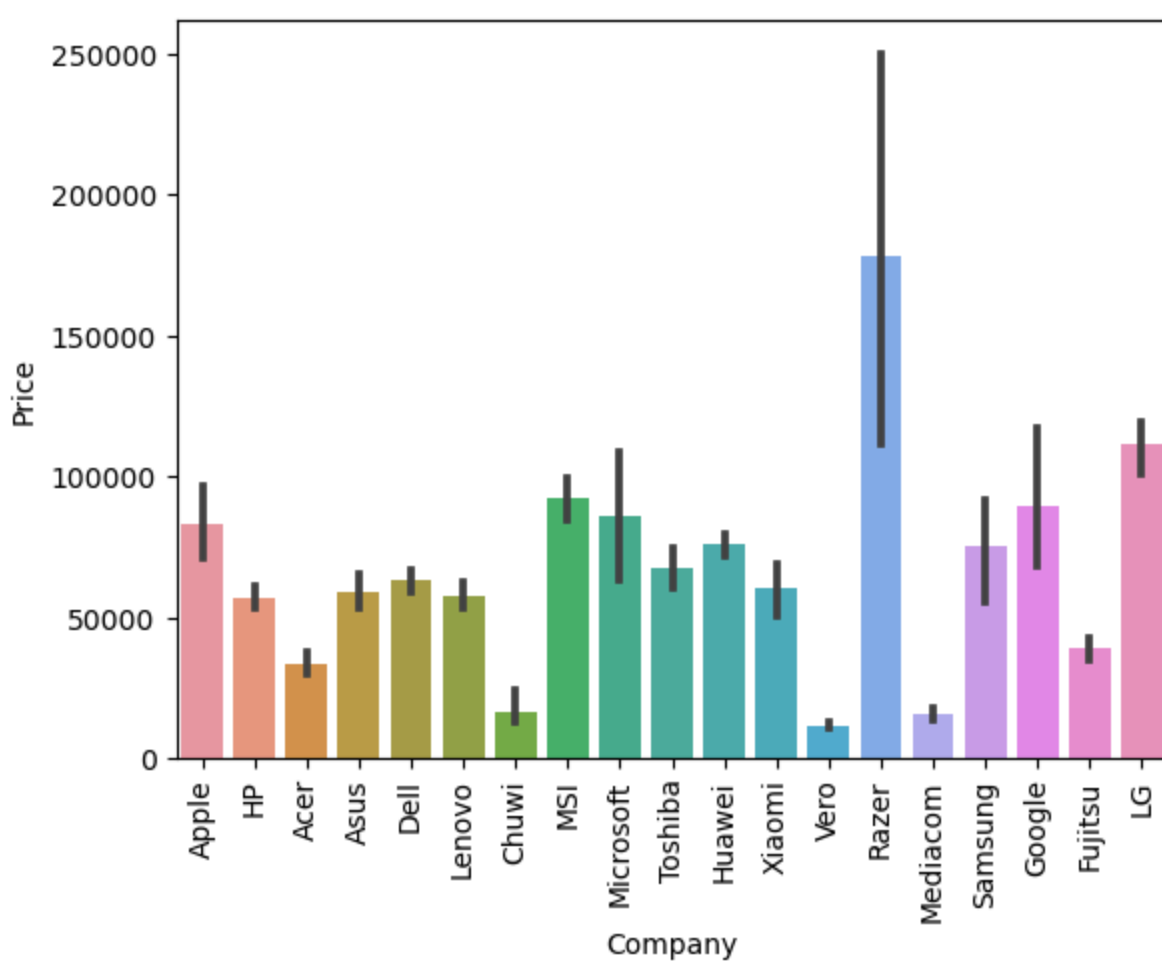


```
In [27]: df['Company'].value_counts().plot(kind='bar')
```

```
Out[27]: <Axes: xlabel='Company'>
```

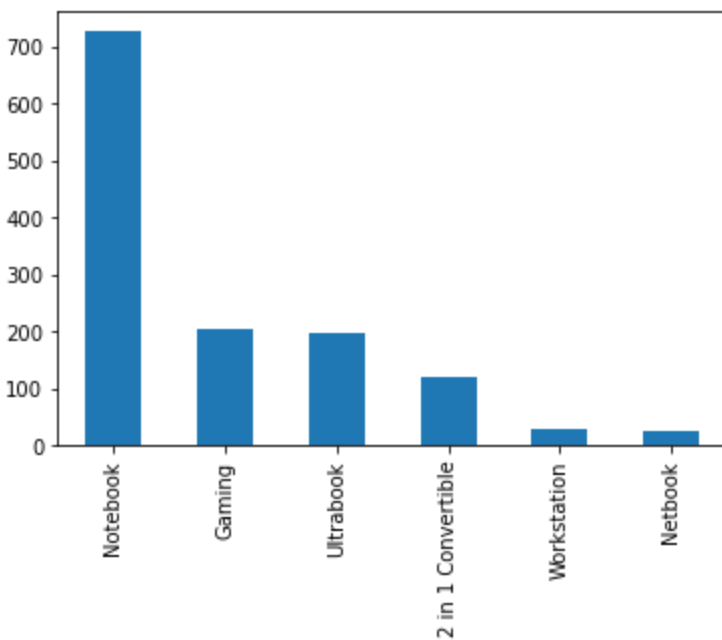


```
In [28]: sns.barplot(x=df['Company'], y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

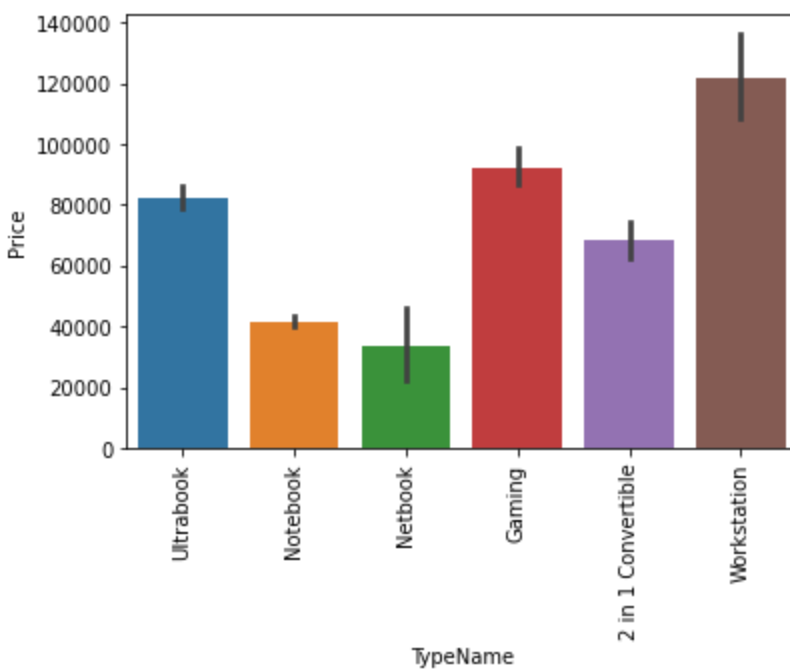


```
In [28]: df['TypeName'].value_counts().plot(kind='bar')
```

```
Out[28]: <AxesSubplot:>
```



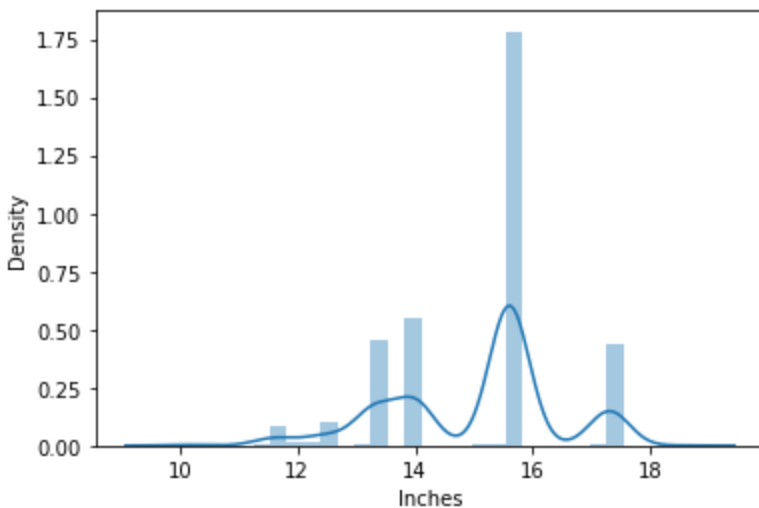
```
In [29]: sns.barplot(x=df['TypeName'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



```
In [30]: sns.distplot(df['Inches'])
```

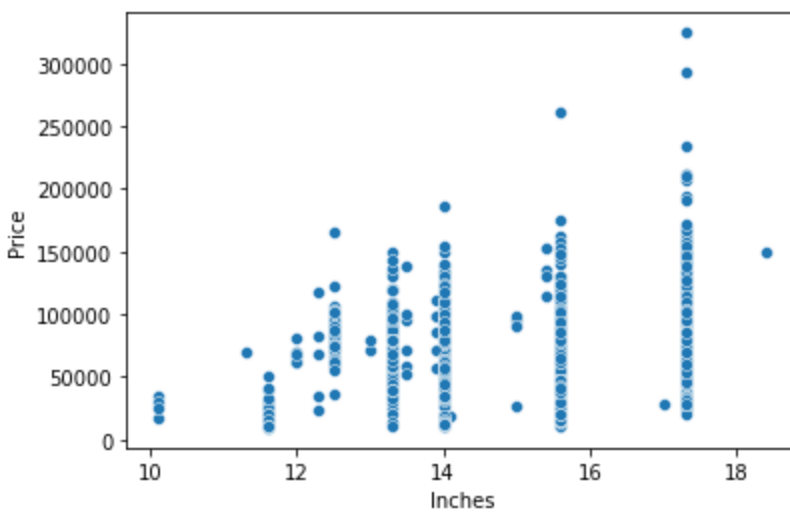
C:\Users\91842\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

```
Out[30]: <AxesSubplot:xlabel='Inches', ylabel='Density'>
```



```
In [31]: sns.scatterplot(x=df['Inches'], y=df['Price'])
```

```
Out[31]: <AxesSubplot:xlabel='Inches', ylabel='Price'>
```



```
In [32]: df['ScreenResolution'].value_counts()
```

```
Out[32]: 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
IPS Panel 1366x768                         7
4K Ultra HD 3840x2160                     7
IPS Panel Quad HD+ / Touchscreen 3200x1800   6
Touchscreen 2256x1504                      6
IPS Panel Retina Display 2304x1440           6
IPS Panel Retina Display 2560x1600           6
IPS Panel Touchscreen 2560x1440             5
IPS Panel 2560x1440                        4
IPS Panel Retina Display 2880x1800           4
IPS Panel Touchscreen 1920x1200             4
1440x900                                    4
Quad HD+ 3200x1800                         3
IPS Panel Quad HD+ 2560x1440                3
1920x1080                                   3
Touchscreen 2400x1600                      3
IPS Panel Touchscreen 1366x768              3
2560x1440                                  3
IPS Panel Full HD 2160x1440                 2
IPS Panel Touchscreen / 4K Ultra HD 3840x2160 2
IPS Panel Quad HD+ 3200x1800               2
Touchscreen / Full HD 1920x1080            1
IPS Panel Retina Display 2736x1824         1
IPS Panel Full HD 1920x1200                1
IPS Panel Full HD 1366x768                 1
Touchscreen / 4K Ultra HD 3840x2160        1
IPS Panel Touchscreen 2400x1600            1
IPS Panel Full HD 2560x1440                1
Touchscreen / Quad HD+ 3200x1800           1
Name: ScreenResolution, dtype: int64
```

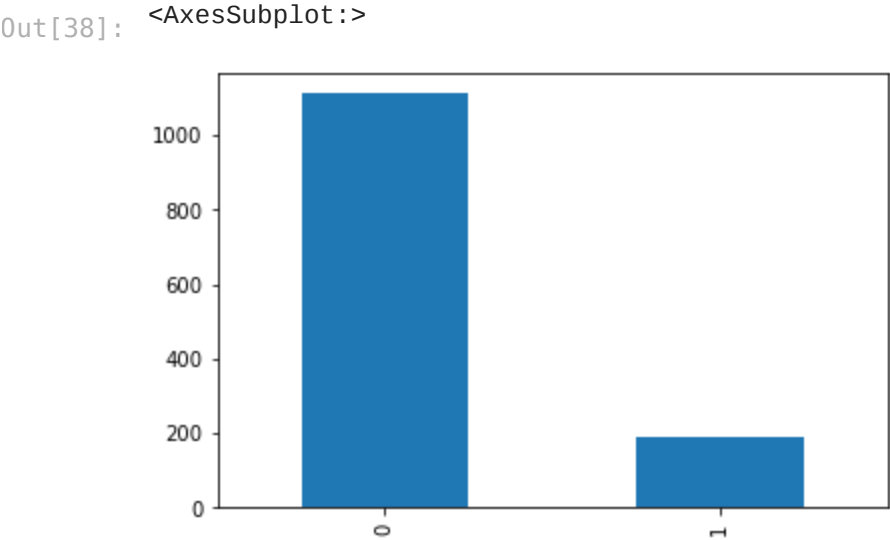
```
In [34]: df['Touchscreen'] = df['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in x else 0)
```

Out[37]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	
1154	Dell	Notebook	15.6	IPS Panel Touchscreen / 4K Ultra HD 3840x2160	Intel Core i5 6300HQ 2.3GHz	8	256GB SSD	Nvidia GeForce 960M	Windows 10	2.04	119lb
750	Lenovo	Netbook	11.6	Touchscreen 1366x768	Intel Celeron Dual Core N3060 1.6GHz	4	128GB SSD	Intel HD Graphics 400	Windows 10	1.40	25lb
1246	Dell	Notebook	14.0	1366x768	Intel Core i5 7200U 2.5GHz	4	500GB HDD	Intel HD Graphics 620	Windows 10	1.60	46lb
879	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	4	256GB SSD	Intel HD Graphics 620	Windows 10	2.04	44lb
1021	Toshiba	Ultrabook	13.3	Full HD 1920x1080	Intel Core i5 6200U 2.3GHz	8	256GB SSD	Intel HD Graphics 520	Windows 10	1.20	84lb

In [38]:

```
df['Touchscreen'].value_counts().plot(kind='bar')
```

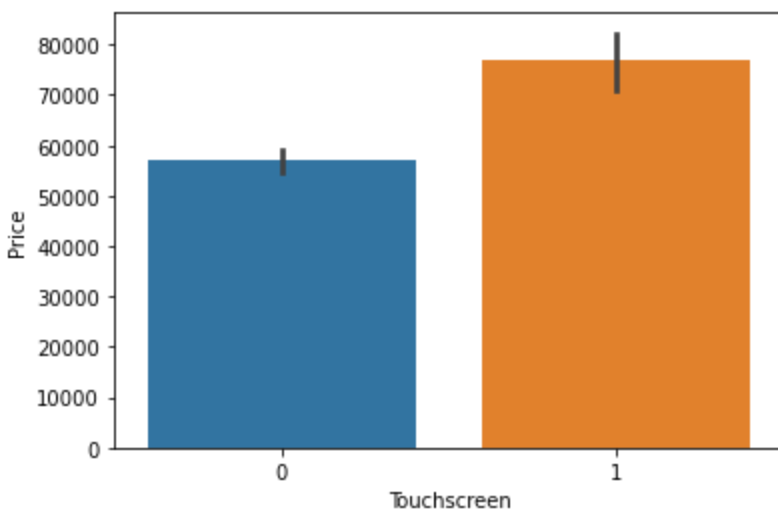


In [39]:

```
sns.barplot(x=df['Touchscreen'],y=df['Price'])
```

Out[39]:

<AxesSubplot:xlabel='Touchscreen', ylabel='Price'>



```
In [40]: df['Ips'] = df['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)
```

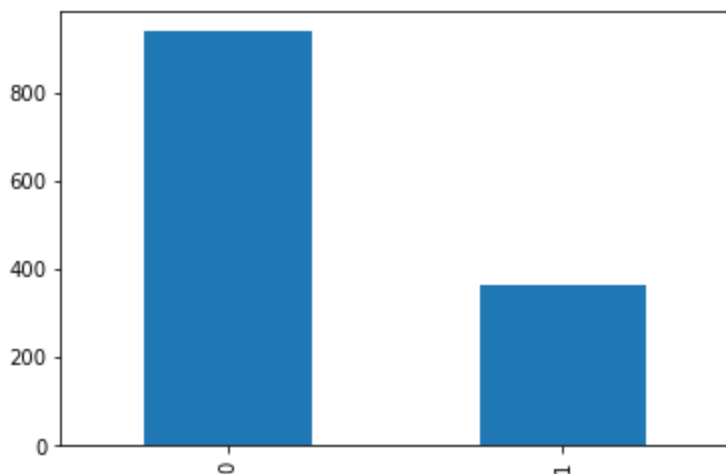
```
In [41]: df.head()
```

```
Out[41]:
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Pr
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	71378.66
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.52
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	30636.00
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	135195.33
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	96095.80

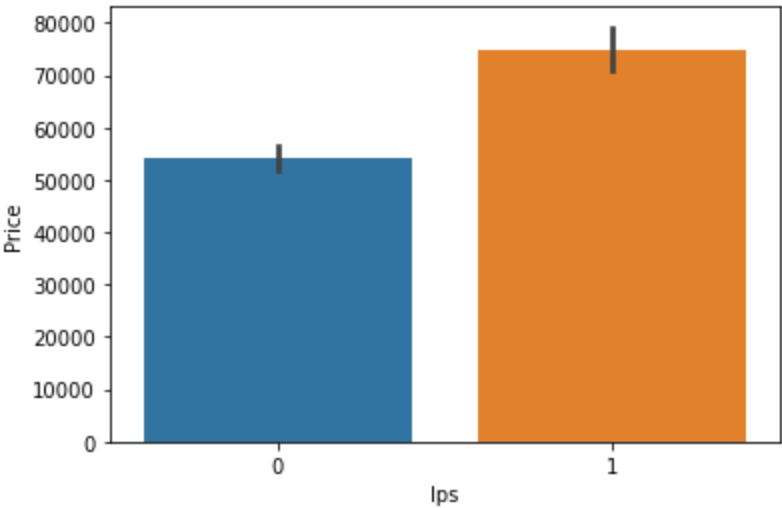
```
In [42]: df['Ips'].value_counts().plot(kind='bar')
```

```
Out[42]: <AxesSubplot:>
```



```
In [43]: sns.barplot(x=df['Ips'],y=df['Price'])
```

```
Out[43]: <AxesSubplot:xlabel='Ips', ylabel='Price'>
```



```
In [47]: new = df['ScreenResolution'].str.split('x',n=1,expand=True)
```

```
In [48]: df['X_res'] = new[0]
df['Y_res'] = new[1]
```

```
In [50]: df.sample(5)
```

```
Out[50]:
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	
141	Lenovo	Notebook	14.0	IPS Panel Full HD 1920x1080	Intel Core i5 8250U 1.6GHz	8	256GB SSD	AMD Radeon RX 550	Windows 10	1.75	5940
1055	HP	Notebook	15.6	1366x768	Intel Core i3 6100U 2.3GHz	4	500GB HDD	Intel HD Graphics 520	Windows 10	2.31	3750
75	Asus	Gaming	15.6	Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	8	1TB HDD	Nvidia GeForce GTX 1050	Windows 10	2.20	5050
984	Toshiba	Notebook	14.0	1366x768	Intel Core i5 6200U 2.3GHz	4	500GB HDD	Intel HD Graphics 520	Windows 10	1.75	4870
337	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	1.84	6090

```
In [59]: df['X_res'] = df['X_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)').apply(lambda x:
```

```
In [60]: df.head()
```

Out[60]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Pr
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	71378.66
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.52
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	30636.00
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	135195.33
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	96095.80

In [62]:

```
df['X_res'] = df['X_res'].astype('int')
df['Y_res'] = df['Y_res'].astype('int')
```

In [63]:

```
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 [65]:

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

Out[65]:

```
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
```

In [68]:

```
df['ppi'] = (((df['X_res']**2) + (df['Y_res']**2))**0.5/df['Inches']).astype('float')
```

```
In [69]: df.corr()['Price']
```

```
Out[69]: 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
```

```
In [70]: df.drop(columns=['ScreenResolution'], inplace=True)
```

```
In [71]: df.head()
```

Out[71]:	Company	TypeName	Inches	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen
0	Apple	Ultrabook	13.3	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0
1	Apple	Ultrabook	13.3	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0
2	HP	Notebook	15.6	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	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0
4	Apple	Ultrabook	13.3	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0

```
In [72]: df.drop(columns=['Inches', 'X_res', 'Y_res'], inplace=True)
```

```
In [73]: df.head()
```

Out[73]:

	Company	Type	Name	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips
0	Apple	Ultrabook		Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1 226
1	Apple	Ultrabook		Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0 127
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
3	Apple	Ultrabook		Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1 220
4	Apple	Ultrabook		Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1 226

In [31]:

df['Cpu'].value_counts()

Out[31]:

```
Cpu
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 M M3-6Y30 0.9GHz      1
AMD A9-Series 9420 2.9GHz        1
Intel Core i3 6006U 2.2GHz        1
AMD A6-Series 7310 2GHz           1
Intel Xeon E3-1535M v6 3.1GHz      1
Name: count, Length: 118, dtype: int64
```

In [29]:

df['Cpu Name'] = df['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))

In [32]:

df['Cpu Name'].value_counts()

```
Out[32]:
```

Cpu Name	
Intel Core i7	527
Intel Core i5	423
Intel Core i3	136
Intel Celeron Dual	80
Intel Pentium Quad	27
Intel Core M	19
AMD A9-Series 9420	12
Intel Celeron Quad	8
AMD A6-Series 9220	8
AMD A12-Series 9720P	7
Intel Atom x5-Z8350	5
AMD A8-Series 7410	4
Intel Atom x5-Z8550	4
Intel Pentium Dual	3
AMD A9-Series 9410	3
AMD Ryzen 1700	3
AMD A9-Series A9-9420	2
AMD A10-Series 9620P	2
Intel Atom X5-Z8350	2
AMD E-Series E2-9000e	2
Intel Xeon E3-1535M	2
Intel Xeon E3-1505M	2
AMD E-Series 7110	2
AMD A10-Series 9600P	2
AMD A6-Series A6-9220	2
AMD A10-Series A10-9620P	2
AMD Ryzen 1600	1
Intel Atom x5-Z8300	1
AMD E-Series E2-6110	1
AMD FX 9830P	1
AMD E-Series E2-9000	1
AMD A6-Series 7310	1
Intel Atom Z8350	1
AMD A12-Series 9700P	1
AMD A4-Series 7210	1
AMD FX 8800P	1
AMD E-Series 9000e	1
Samsung Cortex A72&A53	1
AMD E-Series 9000	1
AMD E-Series 6110	1

Name: count, dtype: int64

```
In [30]: df.head()
```

Out[30]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Pr
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	71378.66
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.52
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	30636.00
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	135195.33
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	96095.80

In [81]:

```
def fetch_processor(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'
```

In [82]:

```
df['Cpu brand'] = df['Cpu Name'].apply(fetch_processor)
```

In [83]:

```
df.head()
```

Out[83]:

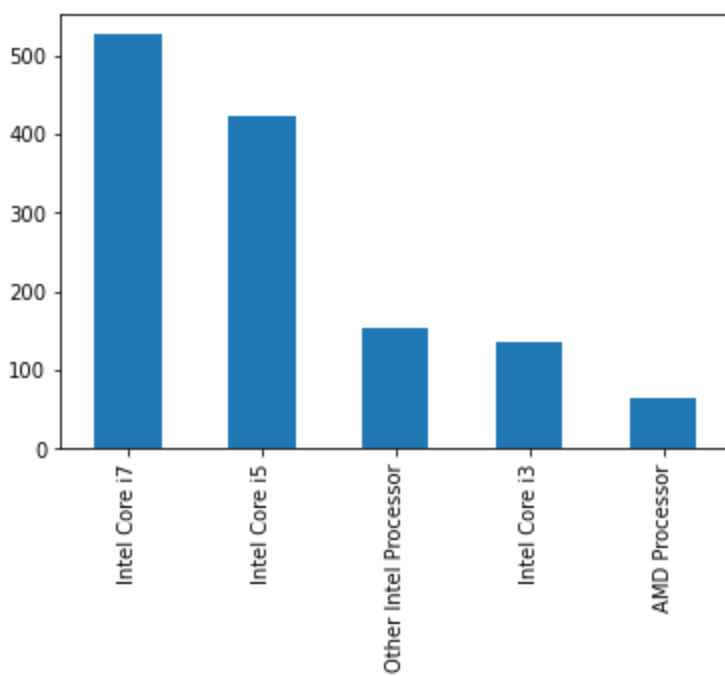
	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1 226
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0 127
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0 141
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1 220
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1 226

In [84]:

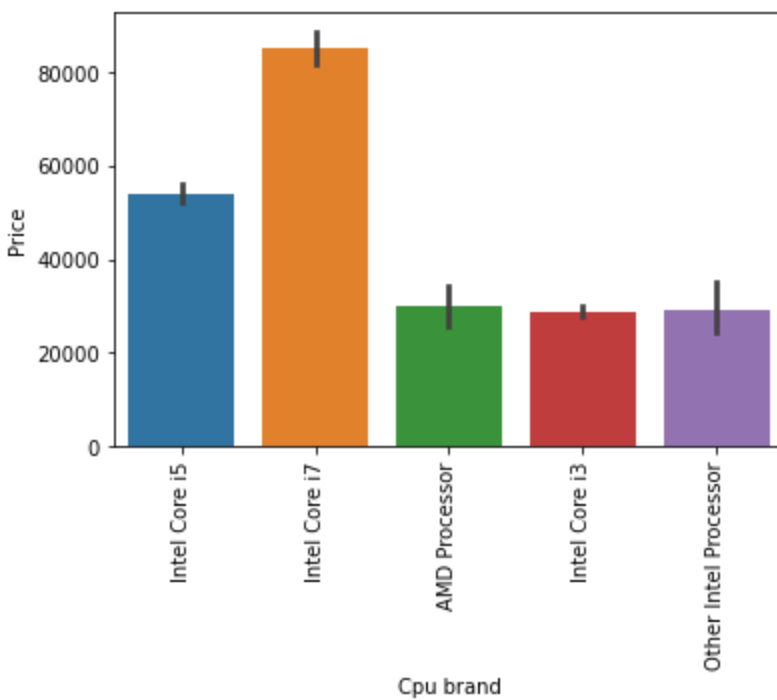
```
df['Cpu brand'].value_counts().plot(kind='bar')
```

Out[84]:

```
<AxesSubplot:>
```



```
In [87]: sns.barplot(x=df['Cpu brand'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



```
In [88]: df.drop(columns=['Cpu', 'Cpu Name'], inplace=True)
```

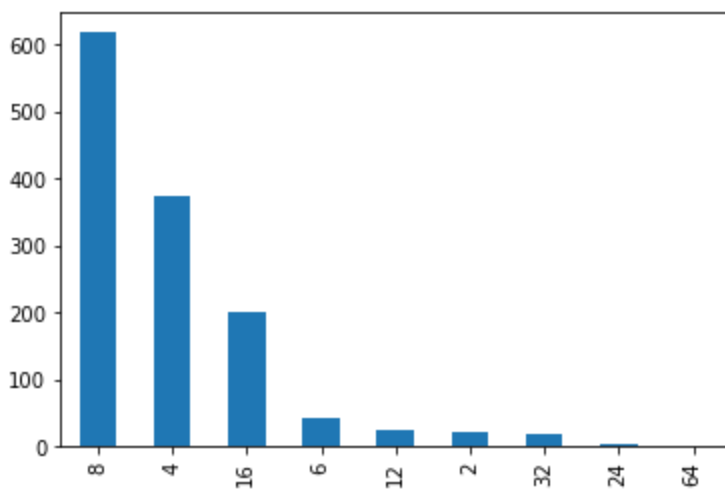
```
In [89]: df.head()
```


Out[89]:

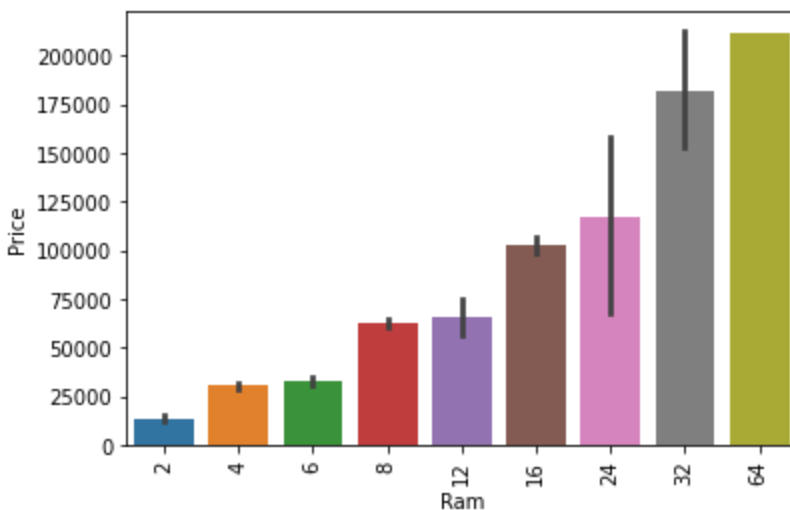
	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005
1	Apple	Ultrabook	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940
2	HP	Notebook	8	256GB SSD	Intel HD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

```
In [90]: df['Ram'].value_counts().plot(kind='bar')
```

Out[90]: <AxesSubplot:>



```
In [91]: sns.barplot(x=df['Ram'], y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



```
In [92]: df['Memory'].value_counts()
```

```
Out[92]: 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
512GB SSD + 1TB HDD 14
1TB SSD 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
16GB SSD 3
512GB SSD + 2TB HDD 3
256GB SSD + 256GB SSD 2
128GB SSD + 2TB HDD 2
256GB SSD + 500GB HDD 2
512GB Flash Storage 2
1TB SSD + 1TB HDD 2
32GB HDD 1
64GB SSD 1
1.0TB HDD 1
512GB SSD + 256GB SSD 1
512GB SSD + 1.0TB Hybrid 1
8GB SSD 1
240GB SSD 1
128GB HDD 1
1TB HDD + 1TB HDD 1
512GB SSD + 512GB SSD 1
256GB SSD + 1.0TB Hybrid 1
508GB Hybrid 1
64GB Flash Storage + 1TB HDD 1
Name: Memory, dtype: int64
```

```
In [93]: df['Memory'] = df['Memory'].astype(str).replace('\.0', '', regex=True)
df["Memory"] = df["Memory"].str.replace('GB', '')
df["Memory"] = df["Memory"].str.replace('TB', '000')
new = df["Memory"].str.split("+", n = 1, expand = True)

df["first"] = new[0]
df["first"] = df["first"].str.strip()

df["second"] = new[1]

df["Layer1HDD"] = df["first"].apply(lambda x: 1 if "HDD" in x else 0)
df["Layer1SSD"] = df["first"].apply(lambda x: 1 if "SSD" in x else 0)
df["Layer1Hybrid"] = df["first"].apply(lambda x: 1 if "Hybrid" in x else 0)
df["Layer1Flash_Storage"] = df["first"].apply(lambda x: 1 if "Flash Storage" in x else 0)

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

df["second"].fillna("0", inplace = True)

df["Layer2HDD"] = df["second"].apply(lambda x: 1 if "HDD" in x else 0)
df["Layer2SSD"] = df["second"].apply(lambda x: 1 if "SSD" in x else 0)
```

```

df["Layer2Hybrid"] = df["second"].apply(lambda x: 1 if "Hybrid" in x else 0)
df["Layer2Flash_Storage"] = df["second"].apply(lambda x: 1 if "Flash Storage" in x else 0)

df['second'] = df['second'].str.replace(r'\D', '')

df["first"] = df["first"].astype(int)
df["second"] = df["second"].astype(int)

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["Layer1Flash_Storage"]+df["second"]*df["Layer2Flash_Storage"])

df.drop(columns=['first', 'second', 'Layer1HDD', 'Layer1SSD', 'Layer1Hybrid', 'Layer1Flash_Storage', 'Layer2HDD', 'Layer2SSD', 'Layer2Hybrid', 'Layer2Flash_Storage'],inplace=True)

```

```

<ipython-input-93-10829db803de>:16: FutureWarning: The default value of regex will change from True to False in a future version.
  df['first'] = df['first'].str.replace(r'\D', '')
<ipython-input-93-10829db803de>:25: FutureWarning: The default value of regex will change from True to False in a future version.
  df['second'] = df['second'].str.replace(r'\D', '')

```

In [98]: `df.sample(5)`

Out[98]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	p
1247	Asus	Gaming	16	256 SSD + 1000 HDD	Nvidia GeForce GTX 1070	Windows 10	2.34	123876.000	0	1	141.2119
505	Lenovo	Notebook	8	256 SSD	Intel HD Graphics 620	Windows 10	1.44	50562.720	0	0	165.6321
820	Lenovo	Notebook	4	500 HDD	Intel HD Graphics 520	Windows 10	2.10	26101.872	0	0	100.4546
21	Lenovo	Gaming	8	128 SSD + 1000 HDD	Nvidia GeForce GTX 1050	Windows 10	2.50	53226.720	0	1	141.2119
301	Asus	Gaming	16	256 SSD + 1000 HDD	Nvidia GeForce GTX 1070	Windows 10	2.90	113060.160	0	0	127.3356

In [99]: `df.drop(columns=['Memory'],inplace=True)`

In [100... `df.head()`

Out[100]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

In [101...

df.corr()['Price']

Out[101]:

Ram0.743007
Weight0.210370
Price1.000000
Touchscreen0.191226
Ips0.252208
ppi0.473487
HDD-0.096441
SSD0.670799
Hybrid0.007989
Flash_Storage-0.040511
Name: Price, dtype: float64

In [102...

df.drop(columns=['Hybrid', 'Flash_Storage'], inplace=True)

In [103...

df.head()

Out[103]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

In [104...

df['Gpu'].value_counts()

```
Out[104]: 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
...
Intel HD Graphics 540      1
AMD FirePro W6150M         1
AMD Radeon R5 M315         1
AMD Radeon R7 M360         1
AMD FirePro W5130M         1
Name: Gpu, Length: 110, dtype: int64
```

```
In [106... df['Gpu brand'] = df['Gpu'].apply(lambda x:x.split()[0])
```

```
In [107... df.head()
```

```
Out[107]:
```

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	t
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	

```
In [108... df['Gpu brand'].value_counts()
```

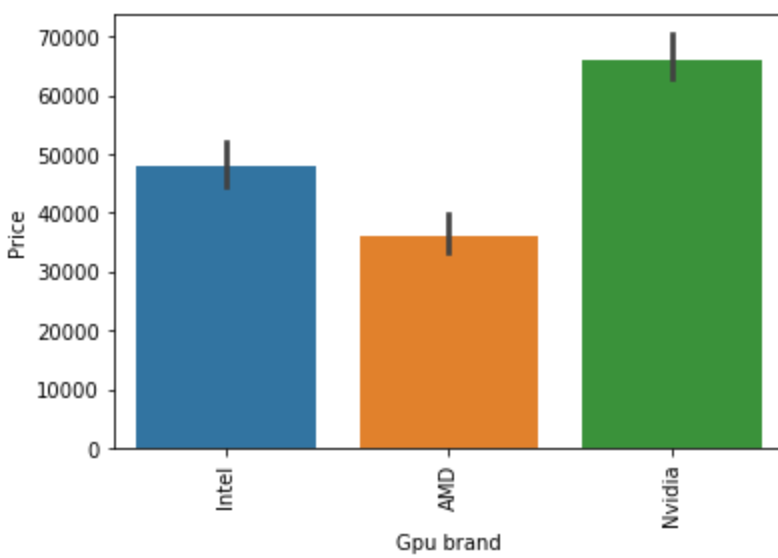
```
Out[108]: Intel      722
Nvidia    400
AMD       180
ARM        1
Name: Gpu brand, dtype: int64
```

```
In [111... df = df[df['Gpu brand'] != 'ARM']
```

```
In [112... df['Gpu brand'].value_counts()
```

```
Out[112]: Intel      722
Nvidia    400
AMD       180
Name: Gpu brand, dtype: int64
```

```
In [115... sns.barplot(x=df['Gpu brand'],y=df['Price'],estimator=np.median)
plt.xticks(rotation='vertical')
plt.show()
```



```
In [116... df.drop(columns=['Gpu'], inplace=True)
```

C:\Users\91842\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
return super().drop()

```
In [117... df.head()
```

```
Out[117]:
```

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

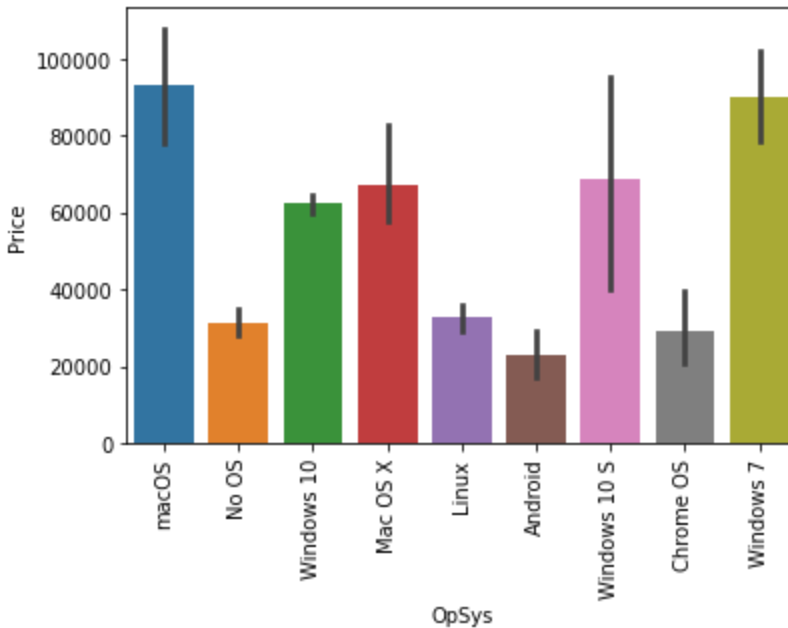
```
In [118... df['OpSys'].value_counts()
```

```
Out[118]:
```

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 [120... sns.barpplot(x=df['OpSys'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



```
In [121... def cat_os(inp):
    if inp == 'Windows 10' or inp == 'Windows 7' or inp == 'Windows 10 S':
        return 'Windows'
    elif inp == 'macOS' or inp == 'Mac OS X':
        return 'Mac'
    else:
        return 'Others/No OS/Linux'
```

```
In [122... df['os'] = df['OpSys'].apply(cat_os)
```

<ipython-input-122-38671a3c07bd>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['os'] = df['OpSys'].apply(cat_os)

```
In [123... df.head()
```

Out[123]:

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

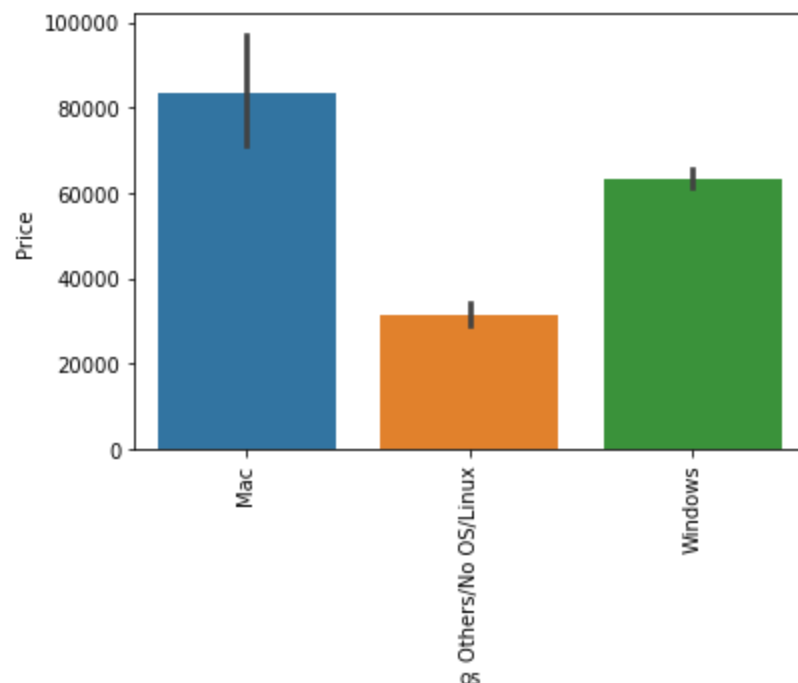
In [124... `df.drop(columns=['OpSys'], inplace=True)`

C:\Users\91842\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
return super().drop()
```

In [125... `sns.barplot(x=df['os'], y=df['Price'])`
`plt.xticks(rotation='vertical')`
`plt.show()`

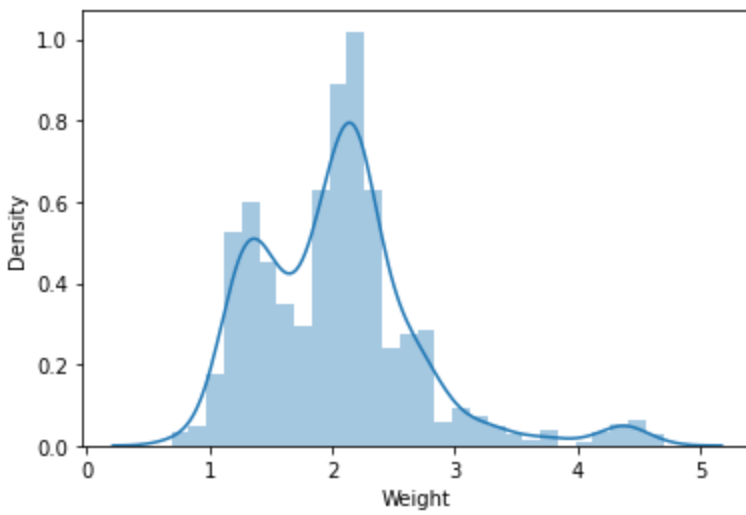


In [126... `sns.distplot(df['Weight'])`

C:\Users\91842\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
'distplot' is a deprecated function and will be removed in a future version. Please adapt
your code to use either 'displot' (a figure-level function with similar flexibility) or
'histplot' (an axes-level function for histograms).

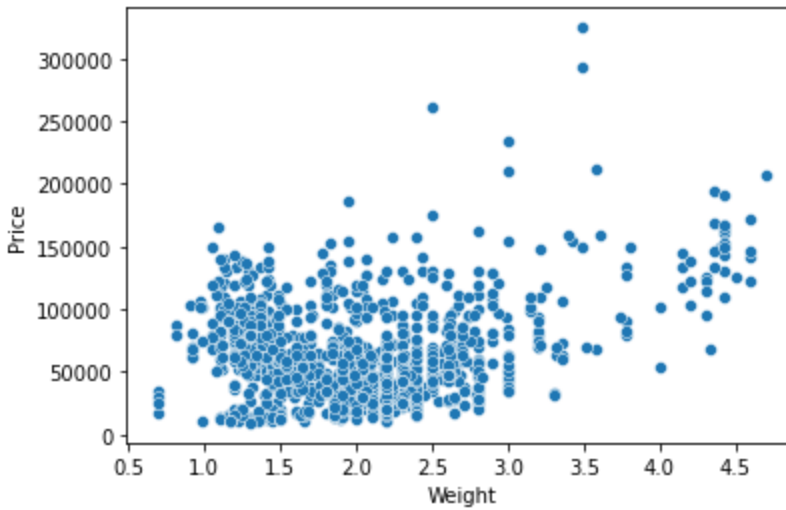
```
return(msg, FutureWarning)
```


Out[126]: <AxesSubplot:xlabel='Weight', ylabel='Density'>



In [127]: `sns.scatterplot(x=df['Weight'],y=df['Price'])`

Out[127]: <AxesSubplot:xlabel='Weight', ylabel='Price'>



In [128]: `df.corr()['Price']`

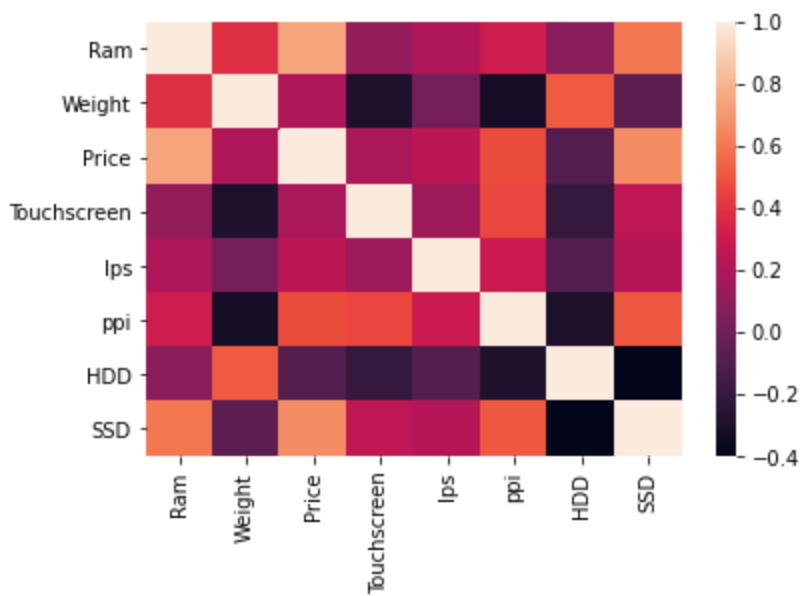
Out[128]:

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 [130]: `sns.heatmap(df.corr())`

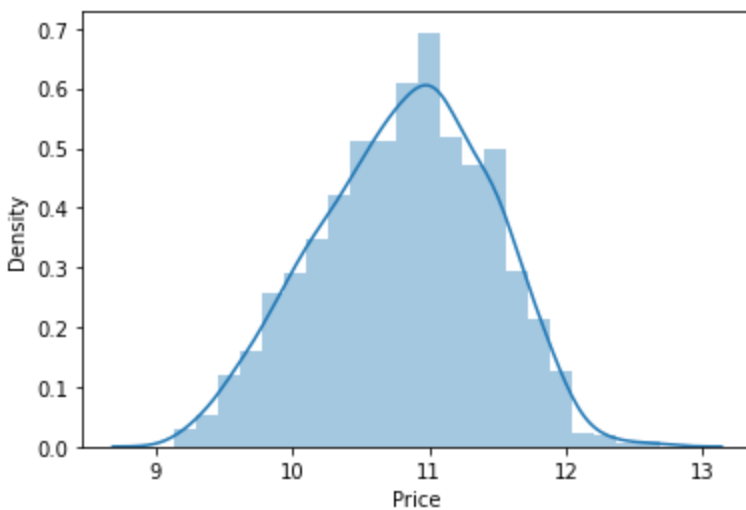
Out[130]: <AxesSubplot:>



```
In [133... sns.distplot(np.log(df['Price']))
```

C:\Users\91842\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

```
Out[133]: <AxesSubplot:xlabel='Price', ylabel='Density'>
```



```
In [134... X = df.drop(columns=['Price'])
y = np.log(df['Price'])
```

```
In [135... X
```

Out[135]:

	Company	Type	Name	Ram	Weight	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Gpu brand	
0	Apple	Ultrabook		8	1.37	0	1	226.983005	Intel Core i5	0	128	Intel	M
1	Apple	Ultrabook		8	1.34	0	0	127.677940	Intel Core i5	0	0	Intel	M
2	HP	Notebook		8	1.86	0	0	141.211998	Intel Core i5	0	256	Intel	Others OS/Li
3	Apple	Ultrabook		16	1.83	0	1	220.534624	Intel Core i7	0	512	AMD	M
4	Apple	Ultrabook		8	1.37	0	1	226.983005	Intel Core i5	0	256	Intel	M
...
1298	Lenovo	2 in 1 Convertible		4	1.80	1	1	157.350512	Intel Core i7	0	128	Intel	Windc
1299	Lenovo	2 in 1 Convertible		16	1.30	1	1	276.053530	Intel Core i7	0	512	Intel	Windc
1300	Lenovo	Notebook		2	1.50	0	0	111.935204	Other Intel Processor	0	0	Intel	Windc
1301	HP	Notebook		6	2.19	0	0	100.454670	Intel Core i7	1000	0	AMD	Windc
1302	Asus	Notebook		4	2.20	0	0	100.454670	Other Intel Processor	500	0	Intel	Windc

1302 rows × 12 columns

In [136...

y

Out[136]:

011.175755

110.776777

210.329931

311.814476

411.473101

...

129810.433899

129911.288115

13009.409283

130110.614129

13029.886358

Name: Price, Length: 1302, dtype: float64

In [137...

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.15,random_state=2)
```

In [138...

X_train

Out[138]:

	Company	Type	Name	Ram	Weight	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Gpu brand	
183	Toshiba	Notebook		8	2.00	0	0	100.454670	Intel Core i5	0	128	Intel	Wind
1141	MSI	Gaming		8	2.40	0	0	141.211998	Intel Core i7	1000	128	Nvidia	Wind
1049	Asus	Netbook		4	1.20	0	0	135.094211	Other Intel Processor	0	0	Intel	Others OS/Li
1020	Dell	2 in 1 Convertible		4	2.08	1	1	141.211998	Intel Core i3	1000	0	Intel	Wind
878	Dell	Notebook		4	2.18	0	0	141.211998	Intel Core i5	1000	128	Nvidia	Wind
...	
466	Acer	Notebook		4	2.20	0	0	100.454670	Intel Core i3	500	0	Nvidia	Wind
299	Asus	Ultrabook		16	1.63	0	0	141.211998	Intel Core i7	0	512	Nvidia	Wind
493	Acer	Notebook		8	2.20	0	0	100.454670	AMD Processor	1000	0	AMD	Wind
527	Lenovo	Notebook		8	2.20	0	0	100.454670	Intel Core i3	2000	0	Nvidia	Others OS/Li
1193	Apple	Ultrabook		8	0.92	0	1	226.415547	Other Intel Processor	0	0	Intel	M

1106 rows × 12 columns

```
In [144... from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score, mean_absolute_error
```

```
In [141... from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
```

Linear regression

```
In [145... step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 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', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))
```

R2 score 0.8073277448418521
MAE 0.21017827976429174

Ridge Regression

In [158...

```
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 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', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))
```

R2 score 0.8127331031311811
MAE 0.20926802242582954

Lasso Regression

In [171...

```
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 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', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))
```

R2 score 0.8071853945317105
MAE 0.21114361613472565

KNN

In [180...

```
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 11])
], remainder='passthrough')

step2 = KNeighborsRegressor(n_neighbors=3)
```

```
pipe = Pipeline([
```

```

        ('step1', step1),
        ('step2', step2)
    ])

    pipe.fit(X_train, y_train)

    y_pred = pipe.predict(X_test)

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

```

R2 score 0.8021984604448553
MAE 0.19319716721521116

Decision Tree

In [191]

```

step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 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', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))

```

R2 score 0.8466456692979233
MAE 0.1806340977609143

SVM

In [213]

```

step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 11])
], remainder='passthrough')

step2 = SVR(kernel='rbf', C=10000, epsilon=0.1)

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

```

R2 score 0.8083180902257614
MAE 0.20239059427481307

Random Forest

In [306...

```
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 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', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))
```

R2 score 0.8873402378382488
MAE 0.15860130110457718

ExtraTrees

In [228...

```
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 11])
], remainder='passthrough')

step2 = ExtraTreesRegressor(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', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))
```

R2 score 0.8753793123440623
MAE 0.15979519126758127

AdaBoost

In [244...

```
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 11])
], remainder='passthrough')

step2 = AdaBoostRegressor(n_estimators=15, learning_rate=1.0)

pipe = Pipeline([
    ('step1', step1),
```

```

    ('step2', step2)
])

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

```

R2 score 0.7929652659237908
MAE 0.23296532406396742

Gradient Boost

In [260...

```

step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 11])
], remainder='passthrough')

step2 = GradientBoostingRegressor(n_estimators=500)

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

```

R2 score 0.8823244736036472
MAE 0.15929506744611283

XgBoost

In [287...

```

step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 11])
], remainder='passthrough')

step2 = XGBRegressor(n_estimators=45, max_depth=5, learning_rate=0.5)

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

```

R2 score 0.8811773435850243
MAE 0.16496203512600974

Voting Regressor

In [305...

```
from sklearn.ensemble import VotingRegressor, StackingRegressor

step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 11])
], remainder='passthrough')

rf = RandomForestRegressor(n_estimators=350, random_state=3, max_samples=0.5, max_features=
gbdt = GradientBoostingRegressor(n_estimators=100, max_features=0.5)
xgb = XGBRegressor(n_estimators=25, learning_rate=0.3, max_depth=5)
et = ExtraTreesRegressor(n_estimators=100, random_state=3, max_samples=0.5, max_features=0.

step2 = VotingRegressor([('rf', rf), ('gbdt', gbdt), ('xgb', xgb), ('et', et)], weights=[5,

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

R2 score 0.8901036732986811
MAE 0.15847265699907628

Stacking

In [296...

```
from sklearn.ensemble import VotingRegressor, StackingRegressor

step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0, 1, 7, 10, 11])
], remainder='passthrough')

estimators = [
    ('rf', RandomForestRegressor(n_estimators=350, random_state=3, max_samples=0.5, max_fea
    ('gbdt', GradientBoostingRegressor(n_estimators=100, max_features=0.5)),
    ('xgb', XGBRegressor(n_estimators=25, learning_rate=0.3, max_depth=5))
]

step2 = StackingRegressor(estimators=estimators, final_estimator=Ridge(alpha=100))

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

R2 score 0.8816958647512341
MAE 0.1663048975120589

Exporting the Model

```
In [308... import pickle

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

```
In [307... df
```

Out[307]:

	Company	TypeName	Ram	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD
0	Apple	Ultrabook	8	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128
1	Apple	Ultrabook	8	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0
2	HP	Notebook	8	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256
3	Apple	Ultrabook	16	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512
4	Apple	Ultrabook	8	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256
...
1298	Lenovo	2 in 1 Convertible	4	1.80	33992.6400	1	1	157.350512	Intel Core i7	0	128
1299	Lenovo	2 in 1 Convertible	16	1.30	79866.7200	1	1	276.053530	Intel Core i7	0	512
1300	Lenovo	Notebook	2	1.50	12201.1200	0	0	111.935204	Other Intel Processor	0	0
1301	HP	Notebook	6	2.19	40705.9200	0	0	100.454670	Intel Core i7	1000	0
1302	Asus	Notebook	4	2.20	19660.3200	0	0	100.454670	Other Intel Processor	500	0

1302 rows × 13 columns

```
In [309... X_train
```

Out[309]:

	Company	TypeName	Ram	Weight	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Gpu brand	
183	Toshiba	Notebook	8	2.00	0	0	100.454670	Intel Core i5	0	128	Intel	Windows
1141	MSI	Gaming	8	2.40	0	0	141.211998	Intel Core i7	1000	128	Nvidia	Windows
1049	Asus	Netbook	4	1.20	0	0	135.094211	Other Intel Processor	0	0	Intel	Others OS/Linux
1020	Dell	2 in 1 Convertible	4	2.08	1	1	141.211998	Intel Core i3	1000	0	Intel	Windows
878	Dell	Notebook	4	2.18	0	0	141.211998	Intel Core i5	1000	128	Nvidia	Windows
...
466	Acer	Notebook	4	2.20	0	0	100.454670	Intel Core i3	500	0	Nvidia	Windows
299	Asus	Ultrabook	16	1.63	0	0	141.211998	Intel Core i7	0	512	Nvidia	Windows
493	Acer	Notebook	8	2.20	0	0	100.454670	AMD Processor	1000	0	AMD	Windows
527	Lenovo	Notebook	8	2.20	0	0	100.454670	Intel Core i3	2000	0	Nvidia	Others OS/Linux
1193	Apple	Ultrabook	8	0.92	0	1	226.415547	Other Intel Processor	0	0	Intel	Mac OS

1106 rows × 12 columns

In []: