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 [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	C
0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	m
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	m
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	N
3	3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	m
4	4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	m

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

Out[4]:
```
(1303, 12)
```

In [5]:
```
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 [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

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

Out[7]:
```
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 [8]:
```
df.drop(columns=['Unnamed: 0'],inplace=True)
```

In [9]:
```
df.head()
```

Out[9]:

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

In [10]:
```
df['Ram'] = df['Ram'].str.replace('GB','')
df['Weight'] = df['Weight'].str.replace('kg','')
```

In [11]:
```
df.head()
```

Out[11]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weigl
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.3
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.3
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.8
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.8
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.3

In [12]:

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

In [13]:

```
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 [14]:

```
import seaborn as sns
```

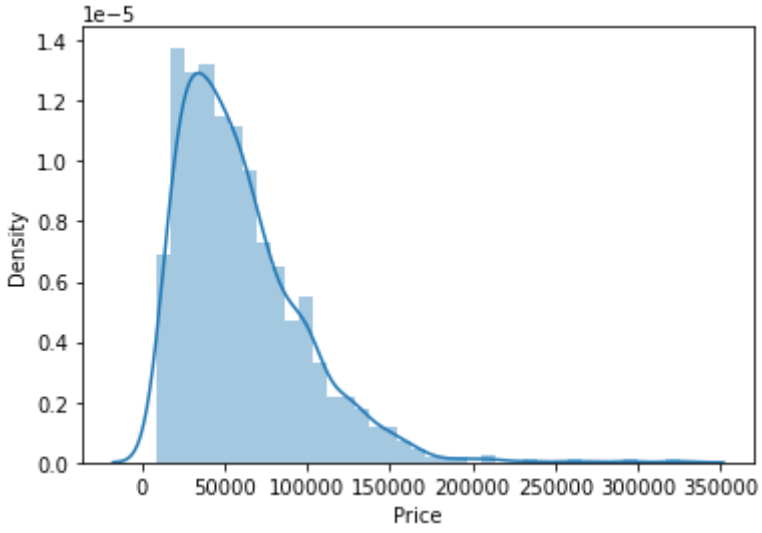
In [15]:

```
sns.distplot(df['Price'])
```

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

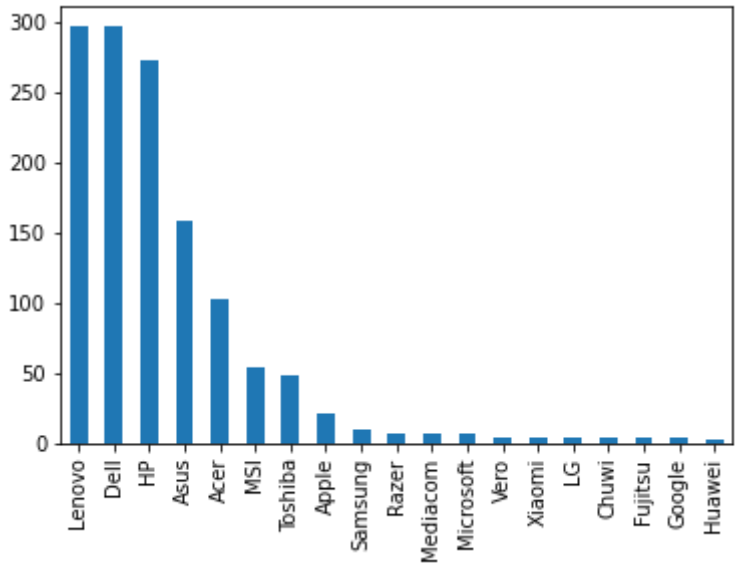
Out[15]:

```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```

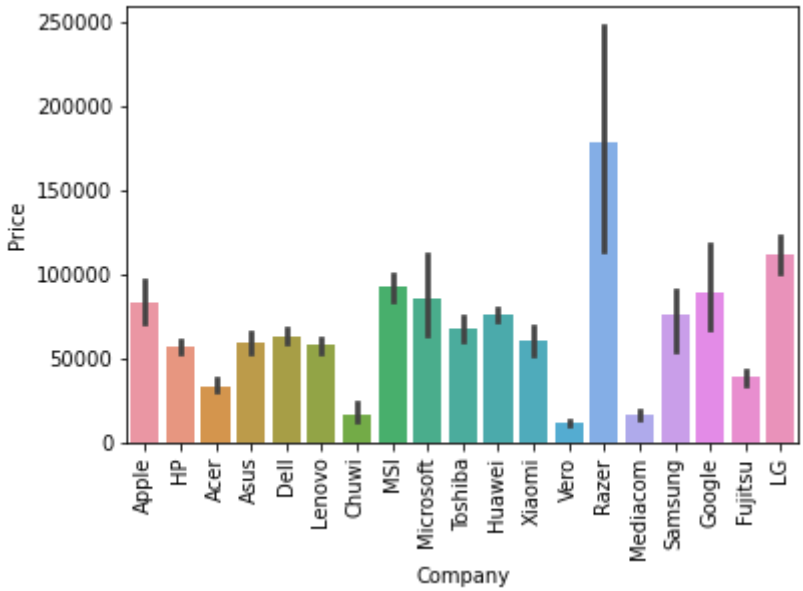


In [16]:
```
df['Company'].value_counts().plot(kind='bar')
```

Out[16]: <AxesSubplot:>

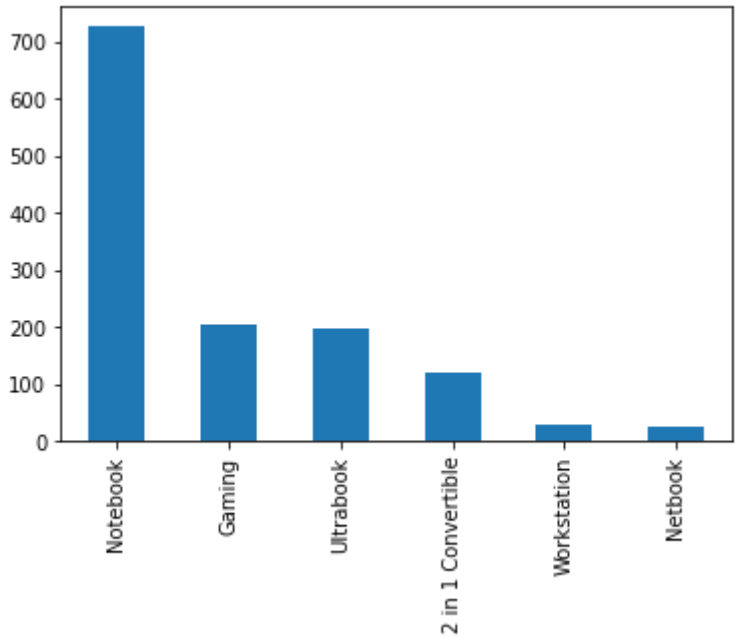


In [17]:
```
sns.barplot(x=df['Company'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

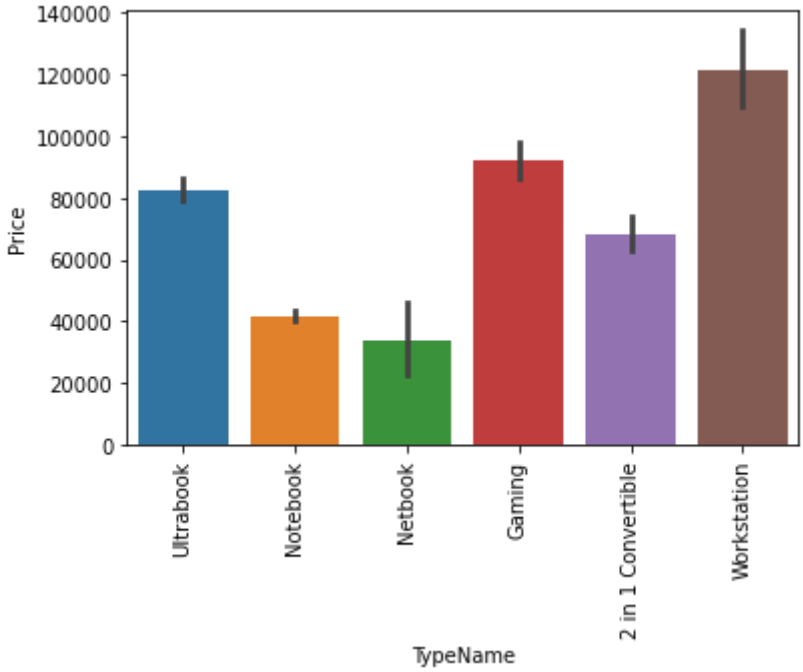


In [18]:
```
df['TypeName'].value_counts().plot(kind='bar')
```

Out[18]: <AxesSubplot:>



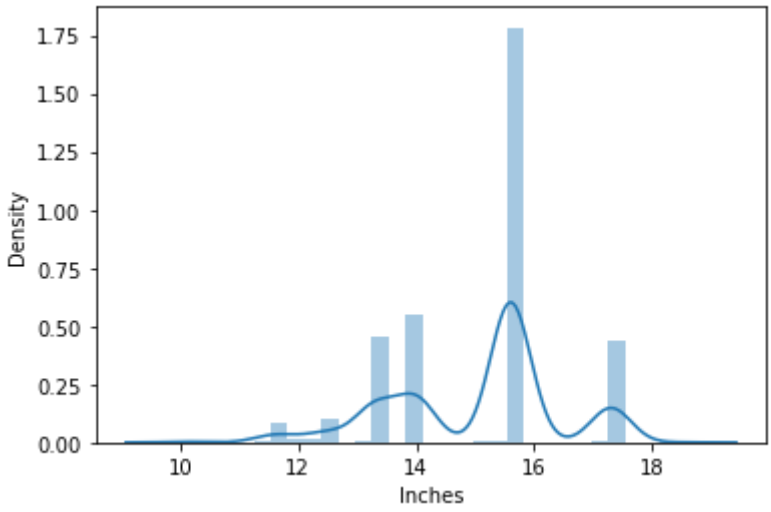
In [19]:
```
sns.barplot(x=df['TypeName'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



In [20]:
```
sns.distplot(df['Inches'])
```

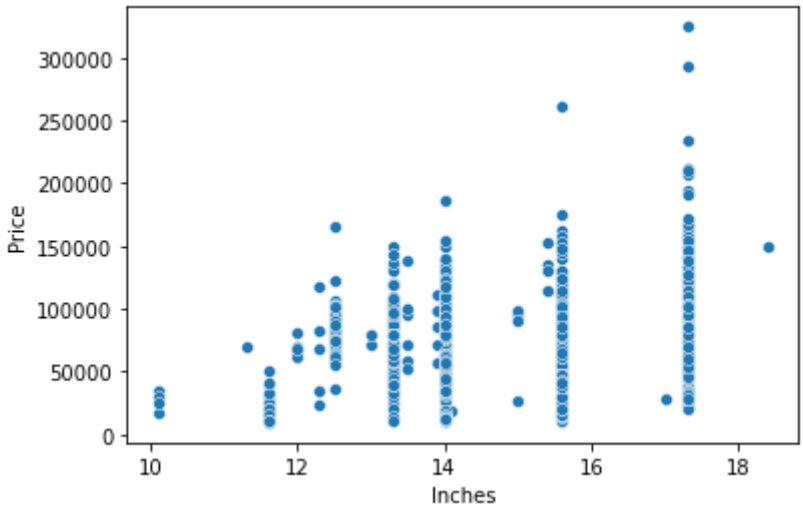
```
C:\Users\Varun kumar\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function with sim
ilar flexibility) or `histplot` (an axes-level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

Out[20]: <AxesSubplot:xlabel='Inches', ylabel='Density'>



In [21]:
```
sns.scatterplot(x=df['Inches'],y=df['Price'])
```

Out[21]: <AxesSubplot:xlabel='Inches', ylabel='Price'>



In [22]:
```
df['ScreenResolution'].value_counts()
```

Out[22]:
```
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
IPS Panel 1366x768                                 7
4K Ultra HD 3840x2160                              7
Touchscreen 2560x1440                              7
IPS Panel Retina Display 2304x1440                 6
Touchscreen 2256x1504                              6
IPS Panel Quad HD+ / Touchscreen 3200x1800         6
IPS Panel Retina Display 2560x1600                 6
IPS Panel Touchscreen 2560x1440                    5
IPS Panel Touchscreen 1920x1200                    4
IPS Panel 2560x1440                                4
IPS Panel Retina Display 2880x1800                 4
1440x900                                           4
IPS Panel Quad HD+ 2560x1440                       3
1920x1080                                          3
IPS Panel Touchscreen 1366x768                     3
Touchscreen 2400x1600                              3
2560x1440                                          3
Quad HD+ 3200x1800                                 3
IPS Panel Full HD 2160x1440                        2
IPS Panel Touchscreen / 4K Ultra HD 3840x2160      2
IPS Panel Quad HD+ 3200x1800                       2
Touchscreen / Quad HD+ 3200x1800                   1
IPS Panel Retina Display 2736x1824                 1
IPS Panel Full HD 1920x1200                        1
Touchscreen / 4K Ultra HD 3840x2160                1
Touchscreen / Full HD 1920x1080                    1
IPS Panel Touchscreen 2400x1600                    1
IPS Panel Full HD 1366x768                         1
IPS Panel Full HD 2560x1440                        1
Name: ScreenResolution, dtype: int64
```

In [23]:
```
df['Touchscreen'] = df['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in x el
```

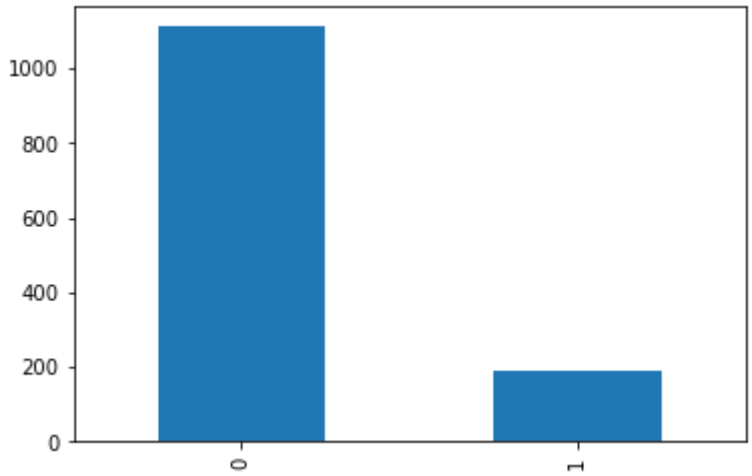
In [24]:
```
df.sample(5)
```

Out[24]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
549	Lenovo	Notebook	15.6	IPS Panel Full HD 1920x1080	Intel Core i7 7500U 2.7GHz	8	512GB SSD	Intel HD Graphics 620	Windows 10
1137	Dell	Notebook	15.6	1366x768	Intel Core i5 7200U 2.5GHz	8	1TB HDD	AMD Radeon R7 M445	Windows 10
1040	Lenovo	2 in 1 Convertible	15.6	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4	256GB SSD	Intel HD Graphics 520	Windows 10
1146	HP	Ultrabook	12.5	Full HD 1920x1080	Intel Core i7 6500U 2.50GHz	8	256GB SSD	Intel HD Graphics 520	Windows 7
455	Dell	2 in 1 Convertible	11.6	Touchscreen 1366x768	Intel Pentium Quad Core N3710 1.6GHz	4	500GB HDD	Intel HD Graphics 405	Windows 10

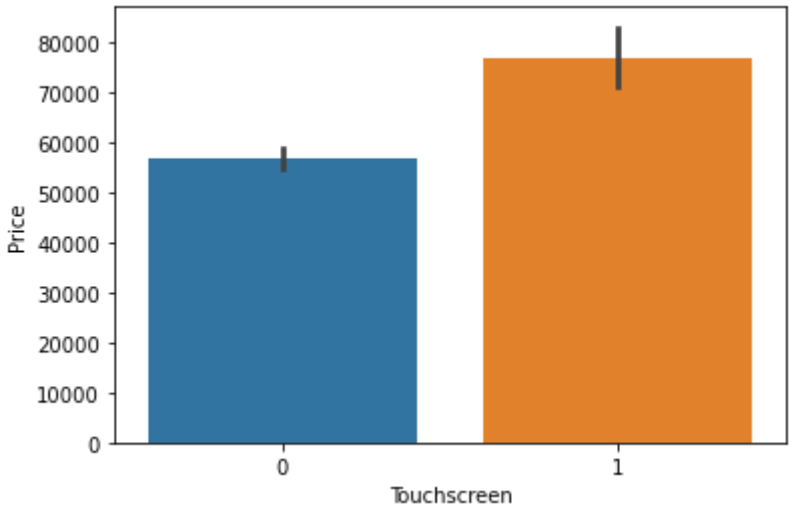
In [25]:
```
df['Touchscreen'].value_counts().plot(kind='bar')
```

Out[25]:
```
<AxesSubplot:>
```



In [26]:
```
sns.barplot(x=df['Touchscreen'],y=df['Price'])
```

Out[26]:
```
<AxesSubplot:xlabel='Touchscreen', ylabel='Price'>
```



In [27]:
```
df['Ips'] = df['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)
```

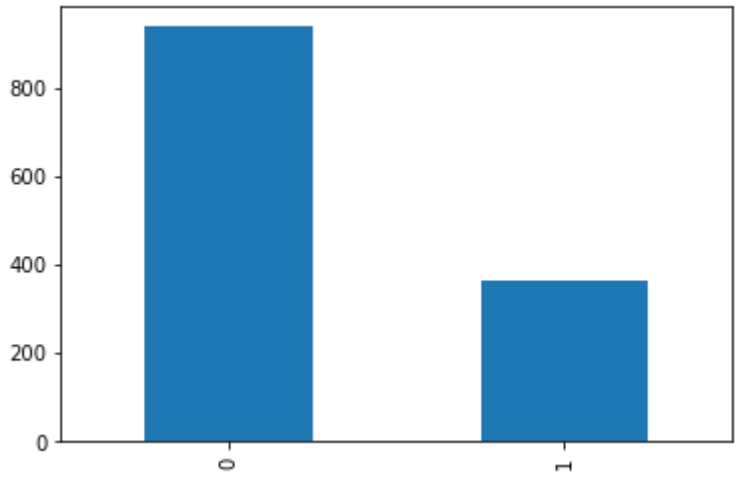
In [28]:
```
df.head()
```

Out[28]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weigl
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.3
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.3
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.8
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.8
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.3

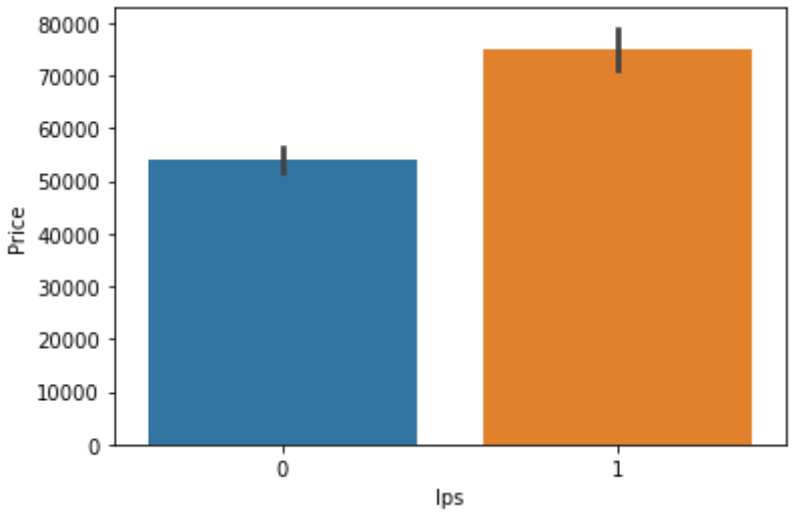
In [29]:
```
df['Ips'].value_counts().plot(kind='bar')
```

Out[29]:
```
<AxesSubplot:>
```



In [30]:
```
sns.barplot(x=df['Ips'],y=df['Price'])
```

Out[30]: <AxesSubplot:xlabel='Ips', ylabel='Price'>



In [31]:
```
new = df['ScreenResolution'].str.split('x',n=1,expand=True)
```

In [32]:
```
df['X_res'] = new[0]
df['Y_res'] = new[1]
```

In [33]:
```
df.sample(5)
```

Out[33]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
432	Lenovo	Notebook	14.0	IPS Panel Full HD 1920x1080	Intel Core i3 7130U 2.7GHz	4	128GB SSD	Intel HD Graphics 620	Windows 10
1230	Lenovo	Netbook	12.5	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.50GHz	16	512GB SSD	Intel HD Graphics 520	Windows 10
787	Lenovo	Gaming	15.6	IPS Panel Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	4	1TB HDD	Nvidia GeForce GTX 1050	Windows 10

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
50	Lenovo	2 in 1 Convertible	10.1	IPS Panel Touchscreen 1920x1200	Intel Atom x5-Z8550 1.44GHz	4	64GB Flash Storage	Intel HD Graphics 400	Android
1263	Acer	Notebook	15.6	1366x768	Intel Celeron Dual Core N3060 1.6GHz	4	500GB HDD	Intel HD Graphics 400	Linux

In [34]:
```
df['X_res'] = df['X_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)').apply(lamb
```

In [35]:
```
df.head()
```

Out[35]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weigh
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.3
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.3
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.8
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.8
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.3

In [36]:
```
df['X_res'] = df['X_res'].astype('int')
df['Y_res'] = df['Y_res'].astype('int')
```

In [37]:
```
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 [38]:
```
df.corr()['Price']
```

Out[38]:
```
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 [39]:
```
df['ppi'] = (((df['X_res']**2) + (df['Y_res']**2))**0.5/df['Inches']).astype('float'
```

In [40]:
```
df.corr()['Price']
```

Out[40]:
```
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 [41]:
```
df.drop(columns=['ScreenResolution'],inplace=True)
```

In [42]:
```
df.head()
```

Out[42]:

	Company	TypeName	Inches	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	To
0	Apple	Ultrabook	13.3	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	
1	Apple	Ultrabook	13.3	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	
2	HP	Notebook	15.6	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	

	Company	TypeName	Inches	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	To
3	Apple	Ultrabook	15.4	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	
4	Apple	Ultrabook	13.3	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	

In [43]:
```
df.drop(columns=['Inches','X_res','Y_res'],inplace=True)
```

In [44]:
```
df.head()
```

Out[44]:

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

In [45]:
```
df['Cpu'].value_counts()
```

Out[45]:
```
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 i5 2.0GHz             1
AMD Ryzen 1600 3.2GHz            1
AMD A6-Series 7310 2GHz          1
Intel Core i5 6440HQ 2.6GHz      1
AMD A4-Series 7210 2.2GHz        1
Name: Cpu, Length: 118, dtype: int64
```

In [46]:
```
df['Cpu Name'] = df['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))
```

In [47]:
```
df.head()
```

Out[47]:

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

In [48]:
```
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 [49]:
```
df['Cpu brand'] = df['Cpu Name'].apply(fetch_processor)
```

In [50]:
```
df.head()
```

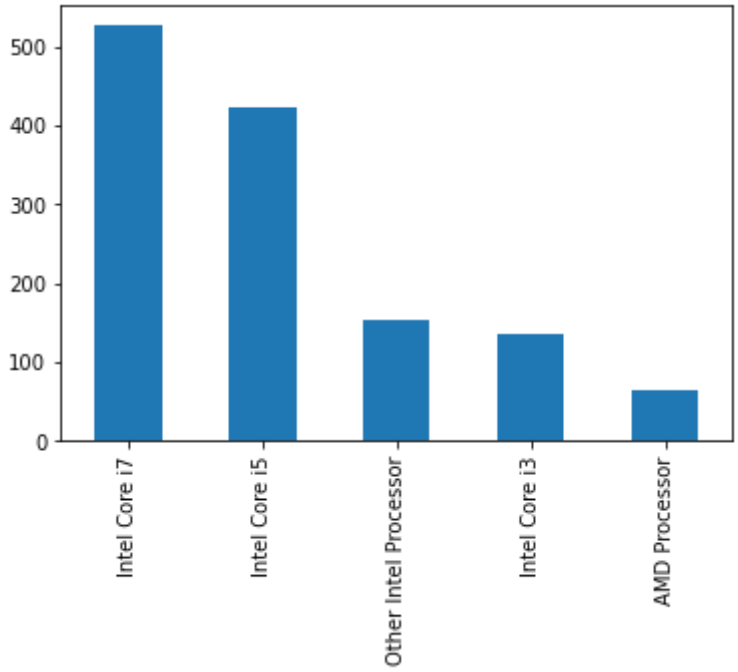
Out[50]:

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscre
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	

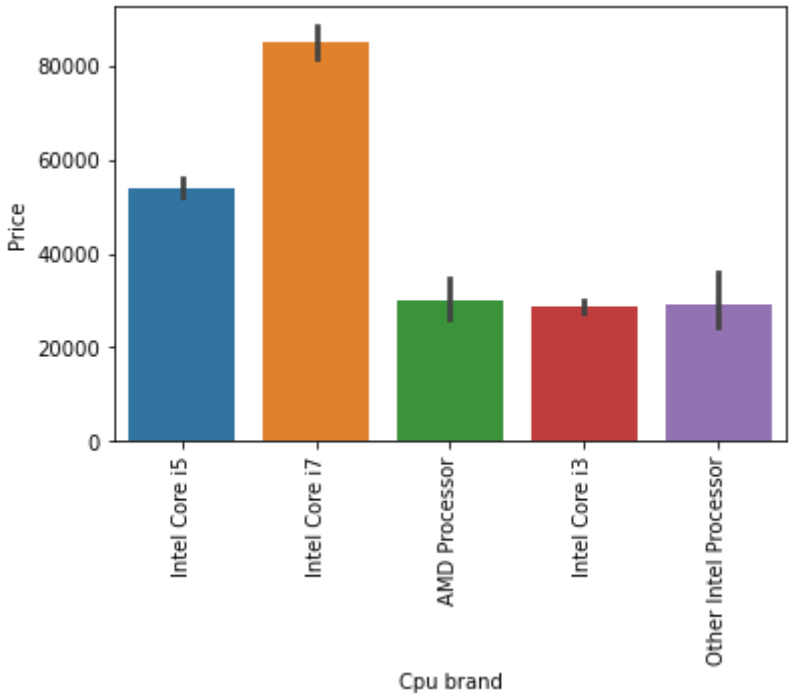
	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscre
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	

In [51]:
```
df['Cpu brand'].value_counts().plot(kind='bar')
```

Out[51]: <AxesSubplot:>



In [52]:
```
sns.barplot(x=df['Cpu brand'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



In [53]:
```
df.drop(columns=['Cpu','Cpu Name'],inplace=True)
```

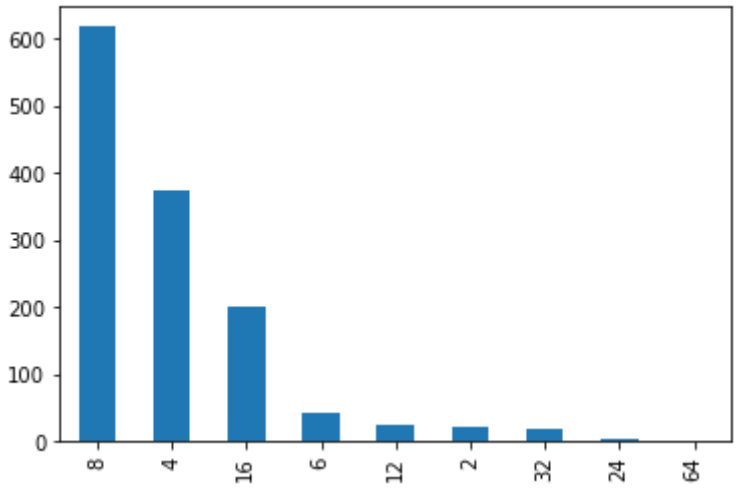
In [54]:
```
df.head()
```

Out[54]:

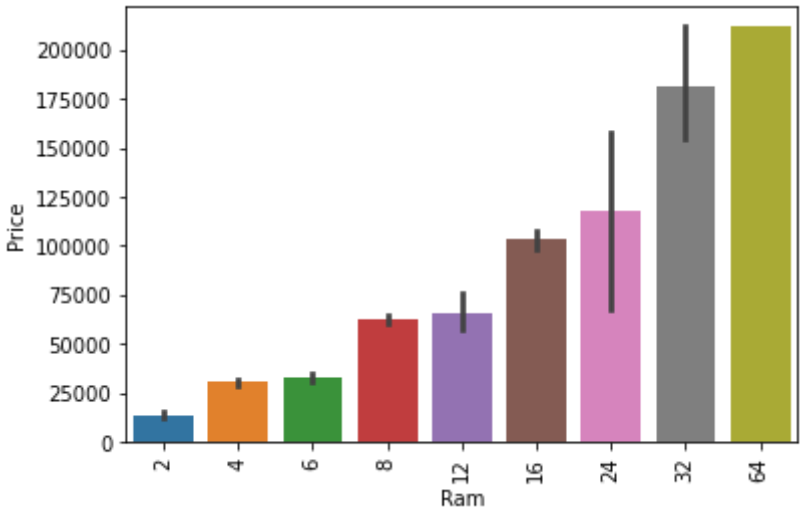
	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1
1	Apple	Ultrabook	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0
2	HP	Notebook	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000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

In [55]:
```
df['Ram'].value_counts().plot(kind='bar')
```

Out[55]:
```
<AxesSubplot:>
```



In [56]:
```
sns.barplot(x=df['Ram'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



In [57]:
```
df['Memory'].value_counts()
```

Out[57]:
```
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
512GB SSD +  2TB HDD               3
16GB SSD                           3
512GB Flash Storage                2
128GB SSD +  2TB HDD               2
256GB SSD +  256GB SSD             2
```

```
256GB SSD +  500GB HDD             2
1TB SSD +  1TB HDD                 2
512GB SSD +  512GB SSD             1
240GB SSD                          1
256GB SSD +  1.0TB Hybrid          1
64GB SSD                           1
64GB Flash Storage +  1TB HDD      1
1.0TB HDD                          1
128GB HDD                          1
512GB SSD +  1.0TB Hybrid          1
1TB HDD +  1TB HDD                 1
508GB Hybrid                       1
8GB SSD                            1
512GB SSD +  256GB SSD             1
32GB HDD                           1
Name: Memory, dtype: int64
```

In [58]:

```
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 el

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 e

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["Layer2Fl

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

```
<ipython-input-58-10829db803de>:16: FutureWarning: The default value of regex will c
hange from True to False in a future version.
  df['first'] = df['first'].str.replace(r'\D', '')
<ipython-input-58-10829db803de>:25: FutureWarning: The default value of regex will c
hange from True to False in a future version.
  df['second'] = df['second'].str.replace(r'\D', '')
```

In [59]:

```
df.sample(5)
```

Out[59]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen
379	Acer	Notebook	8	1000 HDD	Nvidia GeForce GTX 1050	Linux	2.4	45074.8800	0
907	Acer	Notebook	4	16 Flash Storage	Intel HD Graphics	Chrome OS	2.2	19127.5200	0
541	Dell	Notebook	4	128 SSD	Intel HD Graphics 520	Windows 10	1.6	39207.1536	0
1267	Dell	Notebook	8	1000 HDD	AMD Radeon R5 M430	Linux	2.3	42943.1472	0
1202	Dell	2 in 1 Convertible	16	512 SSD	Intel UHD Graphics 620	Windows 10	2.0	68464.8000	1

In [60]:
```
df.drop(columns=['Memory'],inplace=True)
```

In [61]:
```
df.head()
```

Out[61]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	pp
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.98300
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.67794
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.21199
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.53462
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.98300

In [62]:
```
df.corr()['Price']
```

Out[62]:
```
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   -0.040511
Name: Price, dtype: float64
```

In [63]:
```
df.drop(columns=['Hybrid','Flash_Storage'],inplace=True)
```

In [64]:
```
df.head()
```

Out[64]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	pp
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.98300
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.67794
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.21199
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.53462
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.98300

In [65]:
```
df['Gpu'].value_counts()
```

Out[65]:
```
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
                          ...
AMD Radeon Pro 555           1
Intel Iris Pro Graphics      1
AMD FirePro W5130M           1
AMD Radeon RX 560            1
Nvidia Quadro M500M          1
Name: Gpu, Length: 110, dtype: int64
```

In [66]:
```
df['Gpu brand'] = df['Gpu'].apply(lambda x:x.split()[0])
```

In [67]:
```
df.head()
```

Out[67]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	pp
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.98300
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.67794
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.21199
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.53462
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.98300

In [68]:
```
df['Gpu brand'].value_counts()
```

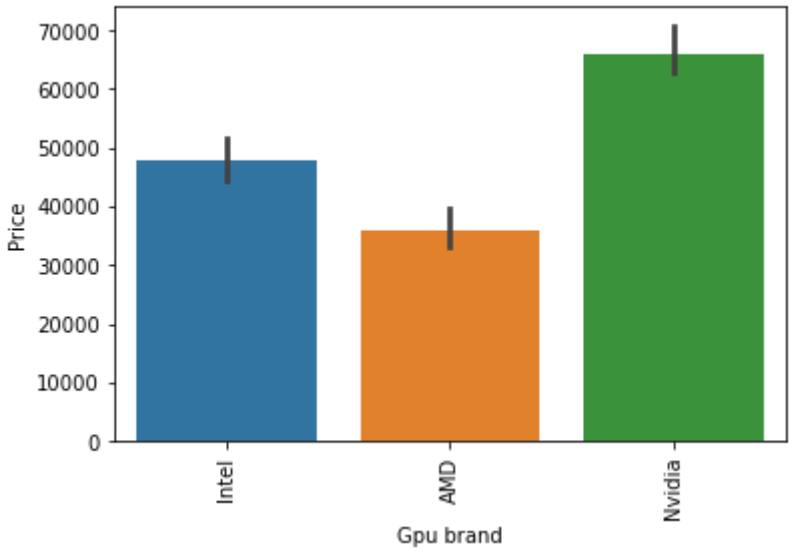
Out[68]:
```
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: Gpu brand, dtype: int64
```

In [69]:
```
df = df[df['Gpu brand'] != 'ARM']
```

In [70]:
```
df['Gpu brand'].value_counts()
```

Out[70]:
```
Intel     722
Nvidia    400
AMD       180
Name: Gpu brand, dtype: int64
```

In [71]:
```
sns.barplot(x=df['Gpu brand'],y=df['Price'],estimator=np.median)
plt.xticks(rotation='vertical')
plt.show()
```



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

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

Out[73]:

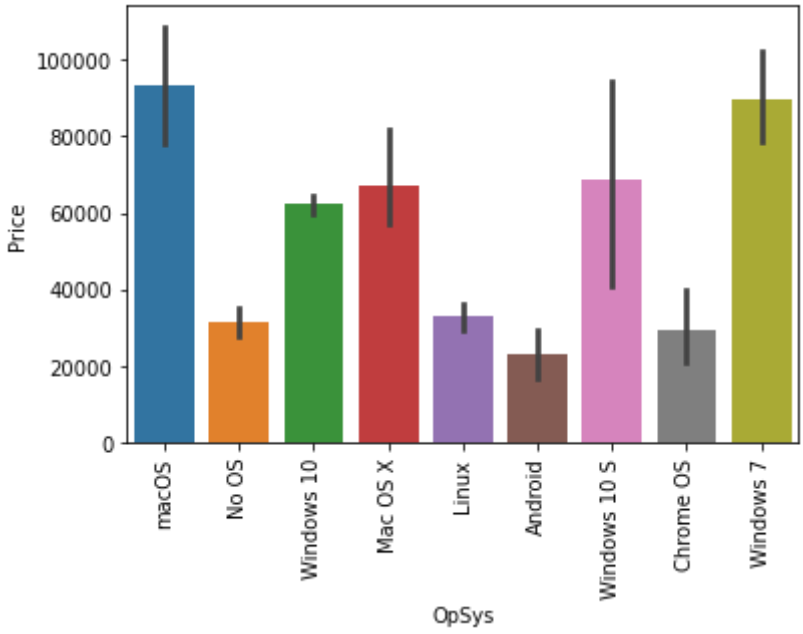
	Company	TypeName	Ram	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand
0	Apple	Ultrabook	8	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	8	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	8	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	16	macOS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

In [74]:
```
df['OpSys'].value_counts()
```

Out[74]:
```
Windows 10      1072
No OS             66
Linux             62
Windows 7         45
Chrome OS         26
macOS             13
Mac OS X           8
Windows 10 S       8
Android            2
Name: OpSys, dtype: int64
```

In [75]:
```
sns.barplot(x=df['OpSys'],y=df['Price'])
```

```
plt.xticks(rotation='vertical')
plt.show()
```



In [76]:

```
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 [77]:

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

In [78]:

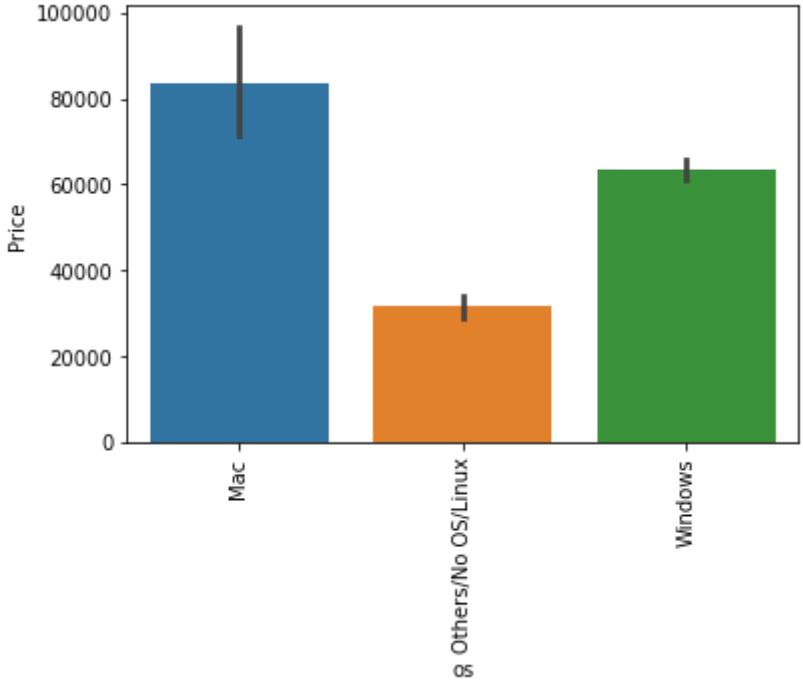
```
df.head()
```

Out[78]:

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

In [79]:
```
df.drop(columns=['OpSys'],inplace=True)
```

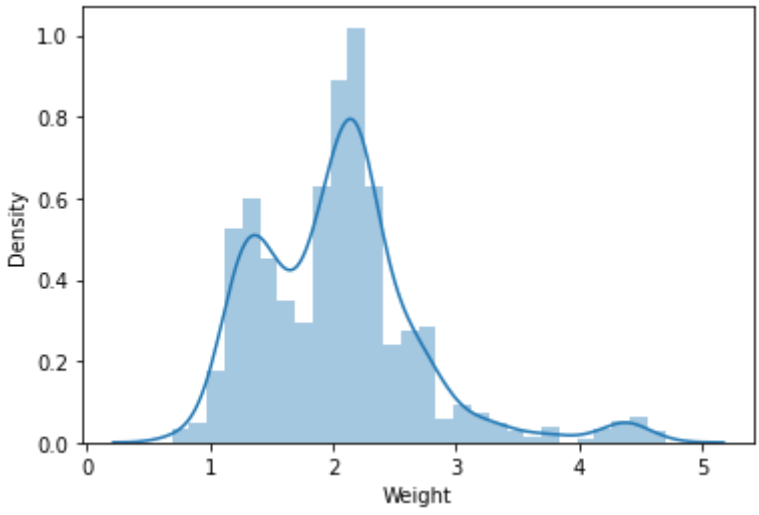
In [80]:
```
sns.barplot(x=df['os'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



In [81]:
```
sns.distplot(df['Weight'])
```

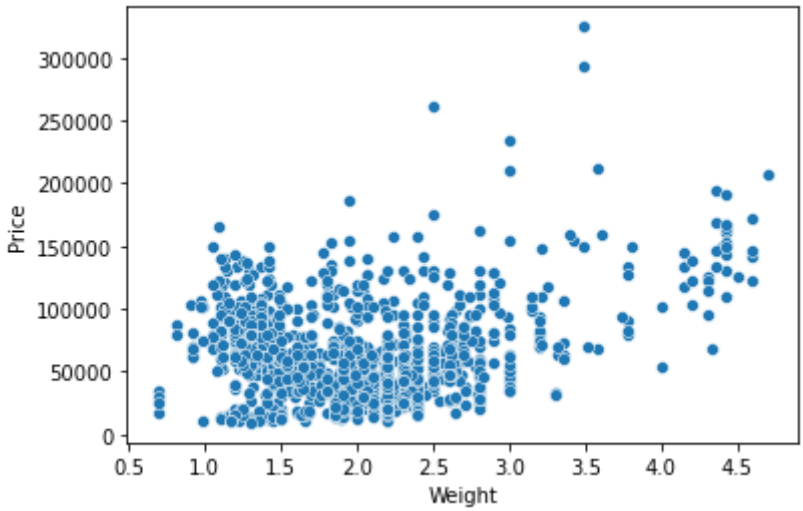
```
C:\Users\Varun kumar\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function with sim
ilar flexibility) or `histplot` (an axes-level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

Out[81]: <AxesSubplot:xlabel='Weight', ylabel='Density'>



In [82]:
```
sns.scatterplot(x=df['Weight'],y=df['Price'])
```

Out[82]: <AxesSubplot:xlabel='Weight', ylabel='Price'>

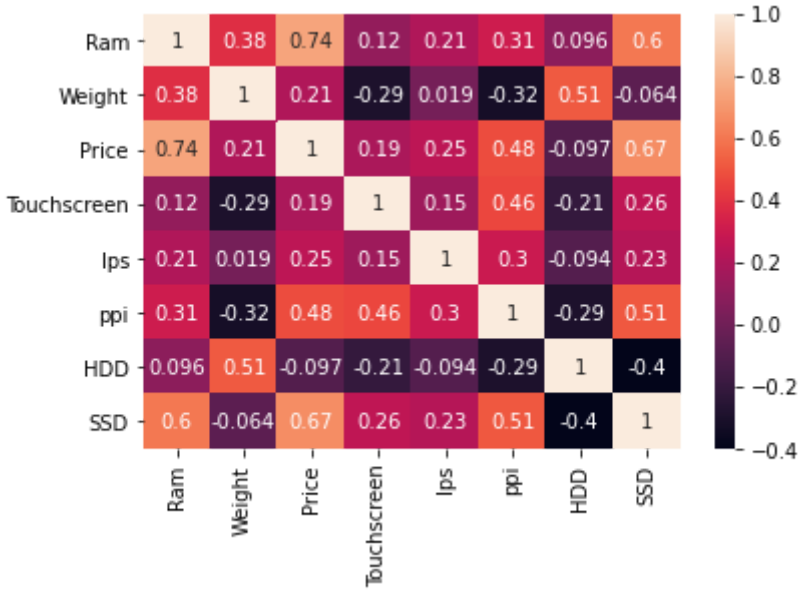


In [83]:
```
df.corr()['Price']
```

Out[83]:
```
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 [140…
```
sns.heatmap(df.corr(),annot=True)
```

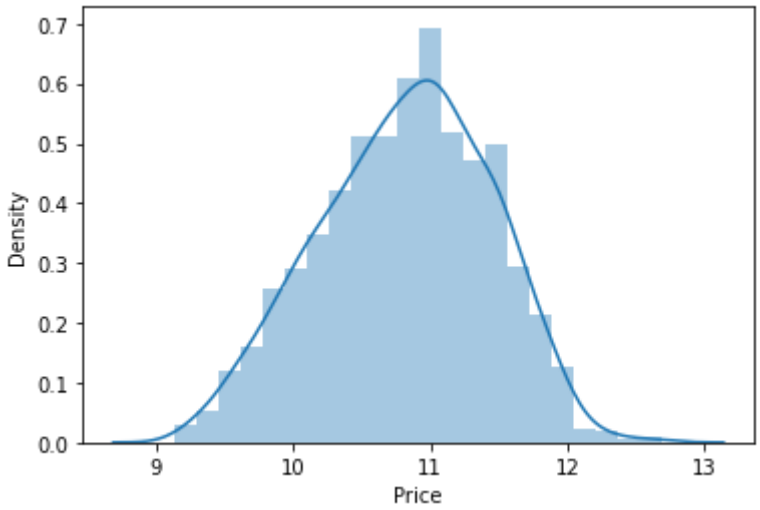
Out[140… <AxesSubplot:>



In [85]:
```
sns.distplot(np.log(df['Price']))
```

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

Out[85]: <AxesSubplot:xlabel='Price', ylabel='Density'>



In [86]:
```
X = df.drop(columns=['Price'])
y = np.log(df['Price'])
```

In [87]:
```
X
```

Out[87]:

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

1302 rows × 12 columns

In [88]:
```
y
```

Out[88]:
```
0       11.175755
1       10.776777
2       10.329931
3       11.814476
4       11.473101
          ...
1298    10.433899
1299    11.288115
1300     9.409283
1301    10.614129
1302     9.886358
Name: Price, Length: 1302, dtype: float64
```

In [89]:
```
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 [90]:
```
X_train
```

Out[90]:

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

1106 rows × 12 columns

In [91]:
```
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 [92]:

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

Linear regression

In [93]:

```
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 0.8073277448418522
MAE 0.21017827976429174
```

Ridge Regression

In [94]:

```
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 0.8127331031311809
MAE 0.20926802242582976
```

Lasso Regression

In [139…

```
step1 = ColumnTransformer(transformers=[
```

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

step2 = Lasso(alpha=0.002)

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 0.8048986293268664
MAE 0.21252926520036242
```

KNN

Decision Tree

In [97]:

```
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 0.831356290461679
MAE 0.18686616252396962
```

SVM

In [122…

```
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 [130…

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

step2 = RandomForestRegressor(n_estimators=800,
                              random_state=3,
                              max_samples=0.5,
                              max_features=0.85,
                              max_depth=19)

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 0.8842238734380571
MAE 0.16010485229215574
```

ExtraTrees

AdaBoost

In [114…

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

step2 = AdaBoostRegressor(n_estimators=80,learning_rate=1.1)

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 0.7933259540611934
MAE 0.22899914522846598
```

Gradient Boost

In [131…
```
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 0.8803238273292837
MAE 0.16019652415915453
```

XgBoost

In [137…
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = XGBRegressor(n_estimators=140,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 0.8746950689623854
MAE 0.16582715402265588
```

Voting Regressor

In [108…
```
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
    ('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 0.8809533645736113
MAE 0.16662383385951088
```

In [ ]:

In [ ]:

In [ ]: