Laptop Price Prediction for SmartTech Co

Name : Marri Madhava Narahari

student_id :S7802

project overview

SmartTech Co. has partnered with our data science team to develop a robust machine learning model that predicts laptop prices accurately. As the market for laptops continues to expand with a myriad of brands and specifications, having a precise pricing model becomes crucial for both consumers and manufacturers.

In [ ]:

```

```

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
```

In [2]:

```
df=pd.read_csv('laptop.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Unnamed: 0.1	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Me
0	0	0.0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	1
1	1	1.0	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	1 St
2	2	2.0	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	2
3	3	3.0	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	5
4	4	4.0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	2

Data cleaning

droping the unwanted columns

In [4]:
```
df.drop(columns=['Unnamed: 0.1','Unnamed: 0'],
    inplace=True)
```

In [5]:
```
df.head()
```

Out[5]:

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

Droping duplicates

In [6]:
```
df.drop_duplicates(inplace=True)
```

Handling missing values

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

Out[7]:
```
Company             1
TypeName            1
Inches              1
ScreenResolution    1
Cpu                 1
Ram                 1
Memory              1
Gpu                 1
OpSys               1
Weight              1
Price               1
dtype: int64
```

In [8]:
```
#Droping the duplicate values

df.dropna(inplace=True)
```

removing outliers

In [9]:
```
q1=df.Price.quantile(0.25)
q3=df.Price.quantile(0.75)
iqr=q3-q1
```

In [10]:
```
lower=q1-1.5*iqr
upper=q3+1.5*iqr
```

In [11]:
```
upper
```

Out[11]:
```
150550.4322
```

In [12]:
```
df[df['Price']<lower]
```

Out[12]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight

In [13]:
```
df[df['Price']>upper].index
```

Out[13]:
```
Int64Index([  17,  196,  204,  238,  247,  297,  517,  530,  563,  610,  6
59,
             723,  744,  749,  758,  778,  780,  830,  841,  911,  955,  9
68,
            1017, 1066, 1081, 1103, 1136, 1231],
           dtype='int64')
```

In [14]:
```
df.drop(index=df[df['Price']>upper].index,inplace=True)
```

Data information

In [15]:
```
df.nunique()
```

Out[15]:
```
Company              19
TypeName              6
Inches               25
ScreenResolution     40
Cpu                 115
Ram                  10
Memory               37
Gpu                 102
OpSys                 9
Weight              185
Price               752
dtype: int64
```

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1216 entries, 0 to 1273
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Company           1216 non-null   object 
 1   TypeName          1216 non-null   object 
 2   Inches            1216 non-null   object 
 3   ScreenResolution  1216 non-null   object 
 4   Cpu               1216 non-null   object 
 5   Ram               1216 non-null   object 
 6   Memory            1216 non-null   object 
 7   Gpu               1216 non-null   object 
 8   OpSys             1216 non-null   object 
 9   Weight            1216 non-null   object 
 10  Price             1216 non-null   float64
dtypes: float64(1), object(10)
memory usage: 114.0+ KB
```

In [17]:
```
df.dtypes
```

Out[17]:
```
Company              object
TypeName             object
Inches               object
ScreenResolution     object
Cpu                  object
Ram                  object
Memory               object
Gpu                  object
OpSys                object
Weight               object
Price               float64
dtype: object
```

In [18]:
```
df.select_dtypes(include='object')
```

Out[18]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	C
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	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	m
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	N
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	m
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	m
...	...	...	...	...	...	...	...	...	
1269	Asus	Notebook	15.6	1366x768	Intel Core i7 6500U 2.5GHz	4GB	500GB HDD	Nvidia GeForce 920M	Wir
1270	Lenovo	2 in 1 Convertible	14	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4GB	128GB SSD	Intel HD Graphics 520	Wir
1271	Lenovo	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16GB	512GB SSD	Intel HD Graphics 520	Wir
1272	Lenovo	Notebook	14	1366x768	Intel Celeron Dual Core N3050 1.6GHz	2GB	64GB Flash Storage	Intel HD Graphics	Wir
1273	HP	Notebook	15.6	1366x768	Intel Core i7 6500U 2.5GHz	6GB	1TB HDD	AMD Radeon R5 M330	Wir

1216 rows × 10 columns

In [ ]:

In [ ]:

remove the GB in ram column and kg in weight column

In [19]:
```
df['Ram']=df['Ram'].str.replace("GB","")
```

In [20]:
```
df['Weight']=df['Weight'].str.replace("kg","")
```

In [21]:
```
df.head()
```

Out[21]:

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

In [22]:
```
# inches -476
# Memory- 770
# Weight - 208
df[(df['Price']=='?')]
```

Out[22]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight

In [23]:
```
df.drop(index=[208,476,770],inplace=True)
```

In [24]:
```
df.index=np.arange(1213)
```

changing the data type of columns

In [25]:
```
df['Ram']=df['Ram'].astype('int32')
df['Weight']=df['Weight'].astype('float32')
df['Inches']=df['Inches'].astype('float32')
```

In [ ]:

cleaning and Analysis on screen resolution column

In [26]:
```
df['ScreenResolution'].value_counts()
```

Out[26]:
```
Full HD 1920x1080                                   484
1366x768                                            255
IPS Panel Full HD 1920x1080                         212
IPS Panel Full HD / Touchscreen 1920x1080            50
Full HD / Touchscreen 1920x1080                      45
1600x900                                             23
Touchscreen 1366x768                                 16
Quad HD+ / Touchscreen 3200x1800                     14
IPS Panel 4K Ultra HD / Touchscreen 3840x2160        11
IPS Panel 1366x768                                    7
IPS Panel Retina Display 2560x1600                    6
Touchscreen 2560x1440                                 6
4K Ultra HD / Touchscreen 3840x2160                   6
IPS Panel 4K Ultra HD 3840x2160                       6
IPS Panel Retina Display 2304x1440                    6
Touchscreen 2256x1504                                 6
4K Ultra HD 3840x2160                                 5
IPS Panel Touchscreen 2560x1440                       5
1440x900                                              4
```

In [27]:
```
# adding the touchscreen and IPS columns
```

In [28]:
```
df['Touchscreen']=df['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' :
df['IPS']=df['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)
```

In [29]:
```
# adding the pixel per inch column
for i in df.index:
    x=df.loc[i,'ScreenResolution'].split()[-1].split('x')[0]
    y=df.loc[i,'ScreenResolution'].split()[-1].split('x')[1]
    z=math.sqrt(int(x)**2+int(y)**2)
    df.loc[i,"PPI"]=z/float(df.loc[i,'Inches'])
```

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

Out[30]:

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

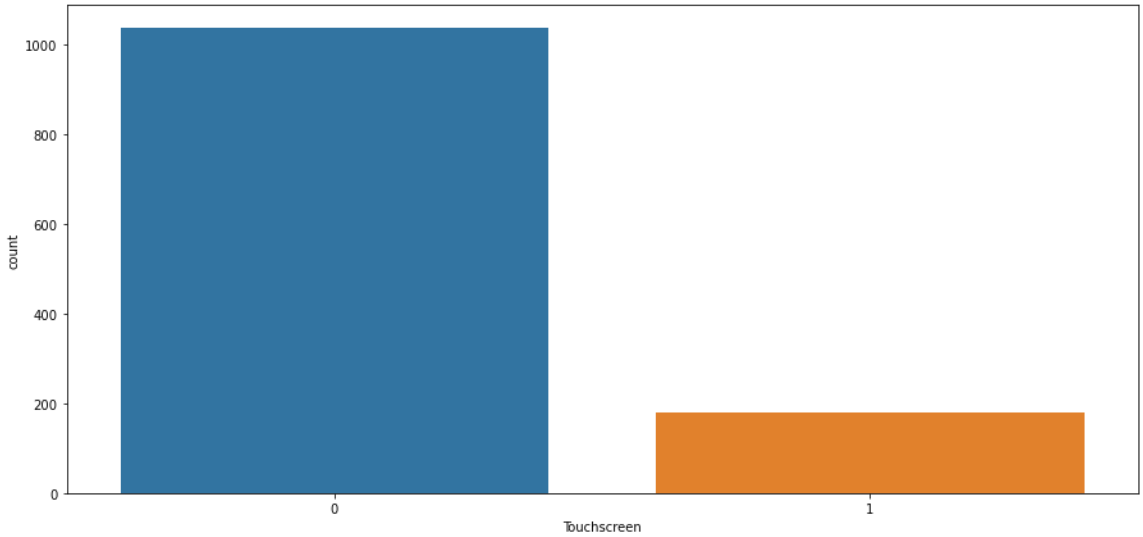
In [31]:
```
# groping the screen resolution column
df.drop(columns='ScreenResolution',inplace=True)
```

In [32]:
```
df.head()
```

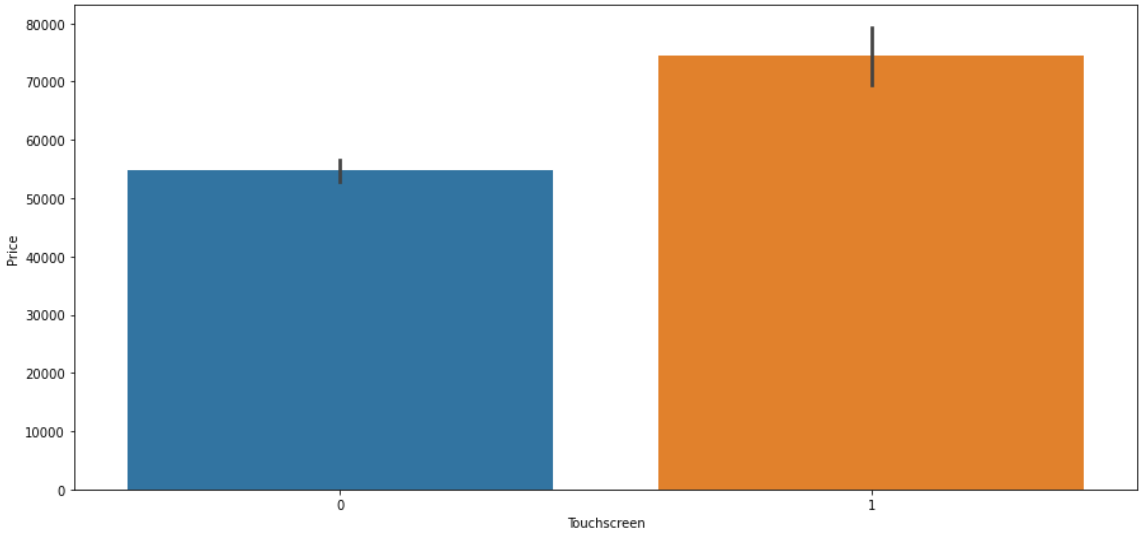
Out[32]:

	Company	TypeName	Inches	Cpu	Ram	Memory	Gpu	OpSys	Weight	Pr
0	Apple	Ultrabook	13.3	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.68
1	Apple	Ultrabook	13.3	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.52
2	HP	Notebook	15.6	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.00
3	Apple	Ultrabook	15.4	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.33
4	Apple	Ultrabook	13.3	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.80

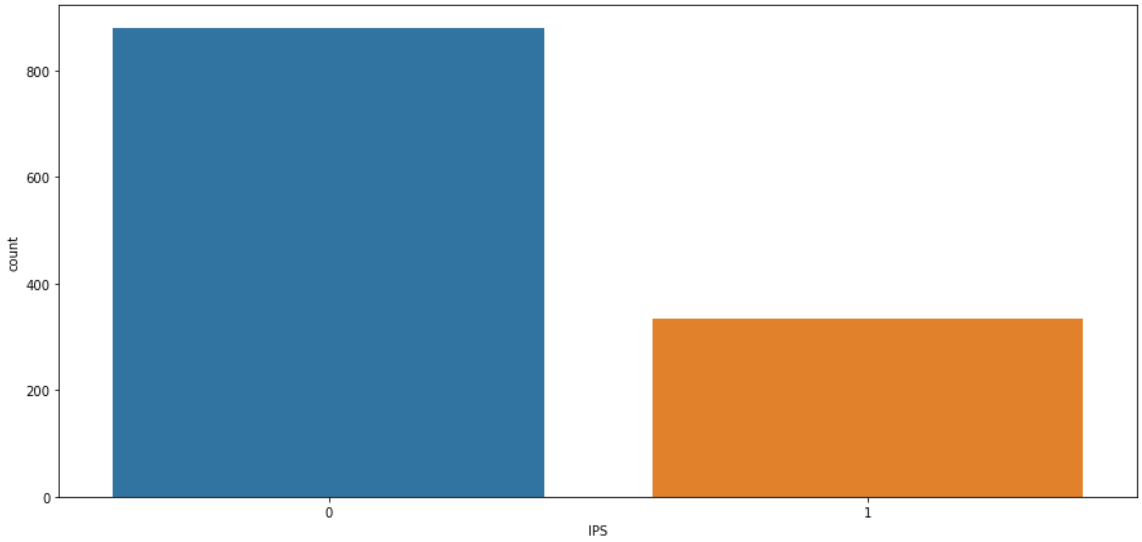
In [33]:
```
plt.figure(figsize=(15,7))
sns.countplot(data=df,x='Touchscreen')
plt.show()
```



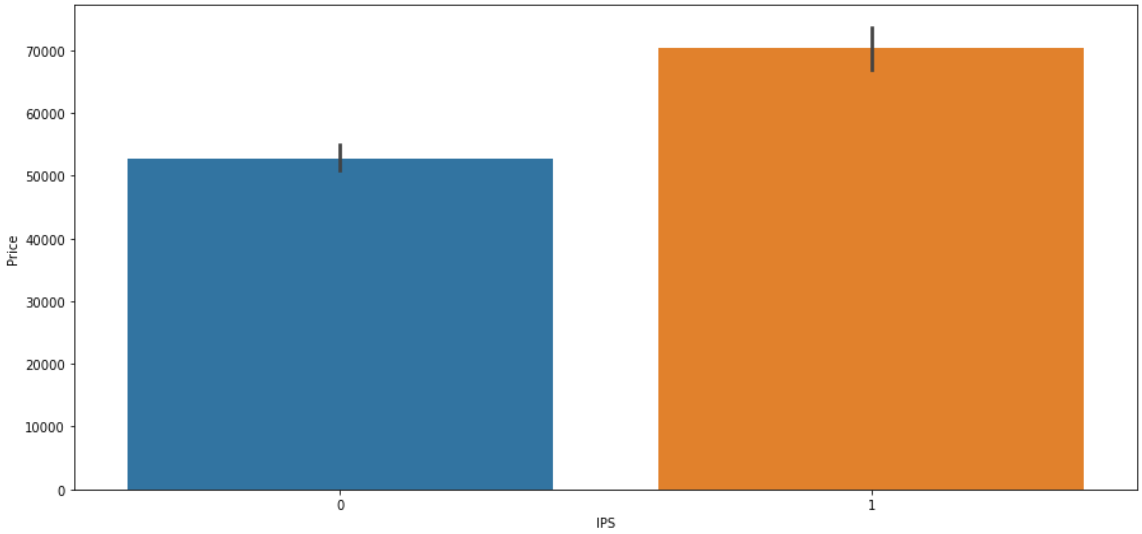
In [34]:
```
plt.figure(figsize=(15,7))
sns.barplot(data=df,x='Touchscreen',y='Price')
plt.show()
```



In [35]:
```
plt.figure(figsize=(15,7))
sns.countplot(data=df,x='IPS')
plt.show()
```



In [36]:
```
plt.figure(figsize=(15,7))
sns.barplot(data=df,x='IPS',y='Price')
plt.show()
```



In [ ]:

cleaning and Analysis on CPU column

In [37]:
```
df['Cpu'].unique()
```

Out[37]:
```
array(['Intel Core i5 2.3GHz', 'Intel Core i5 1.8GHz',
       'Intel Core i5 7200U 2.5GHz', 'Intel Core i7 2.7GHz',
       'Intel Core i5 3.1GHz', 'AMD A9-Series 9420 3GHz',
       'Intel Core i7 2.2GHz', 'Intel Core i7 8550U 1.8GHz',
       'Intel Core i5 8250U 1.6GHz', 'Intel Core i3 6006U 2GHz',
       'Intel Core i7 2.8GHz', 'Intel Core M m3 1.2GHz',
       'Intel Core i7 7500U 2.7GHz', 'Intel Core i3 7100U 2.4GHz',
       'Intel Core i5 7300HQ 2.5GHz', 'AMD E-Series E2-9000e 1.5GHz',
       'Intel Core i5 1.6GHz', 'Intel Core i7 8650U 1.9GHz',
       'Intel Atom x5-Z8300 1.44GHz', 'AMD E-Series E2-6110 1.5GHz',
       'AMD A6-Series 9220 2.5GHz',
       'Intel Celeron Dual Core N3350 1.1GHz',
       'Intel Core i3 7130U 2.7GHz', 'Intel Core i7 7700HQ 2.8GHz',
       'Intel Core i5 2.0GHz', 'AMD Ryzen 1700 3GHz',
       'Intel Pentium Quad Core N4200 1.1GHz',
       'Intel Celeron Dual Core N3060 1.6GHz', 'Intel Core i5 1.3GHz',
       'AMD FX 9830P 3GHz', 'Intel Core i7 7560U 2.4GHz',
       'AMD E-Series 6110 1.5GHz', 'Intel Core i5 6200U 2.3GHz',
       'Intel Core M 6Y75 1.2GHz', 'Intel Core i5 7500U 2.7GHz',
```

In [38]:
```
df['Cpu'].value_counts()
```

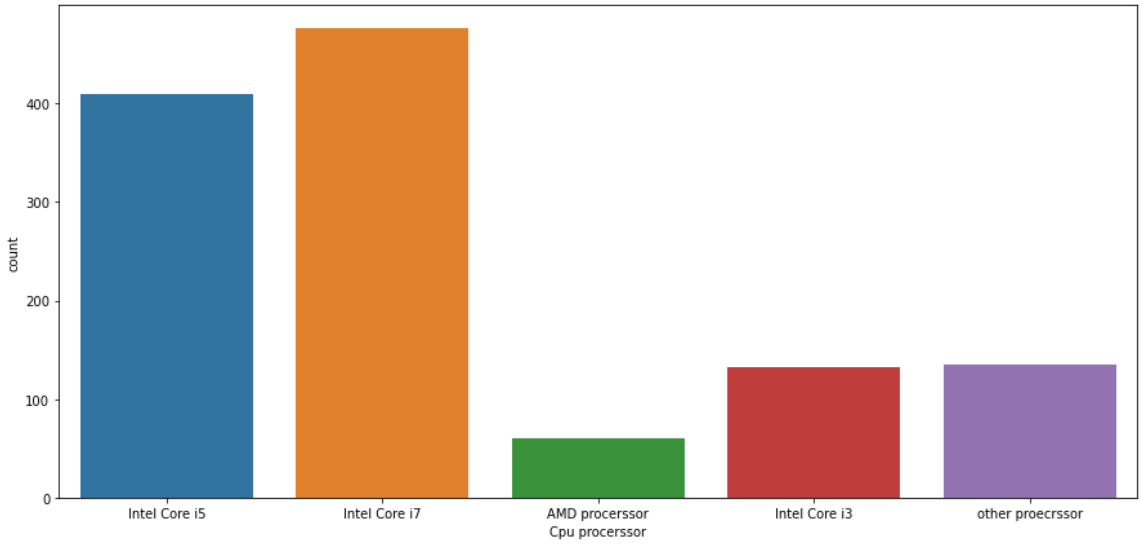
Out[38]:
```
Intel Core i5 7200U 2.5GHz              183
Intel Core i7 7700HQ 2.8GHz             129
Intel Core i7 7500U 2.7GHz              125
Intel Core i7 8550U 1.8GHz               71
Intel Core i5 8250U 1.6GHz               68
                                        ...
Intel Celeron Dual Core N3350 2.0GHz      1
Intel Core M 7Y30 1.0GHz                  1
Intel Core i3 6100U 2.1GHz                1
AMD E-Series E2-9000 2.2GHz               1
AMD A9-Series 9410 2.9GHz                 1
Name: Cpu, Length: 115, dtype: int64
```

In [39]:
```
# adding the cpu processor column
for i in df.index:
    if 'Intel Core i5' in  df.loc[i,'Cpu']:
        df.loc[i,'Cpu procerssor']='Intel Core i5'
    elif 'Intel Core i7'  in  df.loc[i,'Cpu']:
        df.loc[i,'Cpu procerssor']='Intel Core i7'
    elif 'Intel Core i3' in  df.loc[i,'Cpu']:
        df.loc[i,'Cpu procerssor']='Intel Core i3'
    elif 'AMD' in  df.loc[i,'Cpu']:
        df.loc[i,'Cpu procerssor']='AMD procerssor'
    else:
        df.loc[i,'Cpu procerssor']='other proecrssor'
```

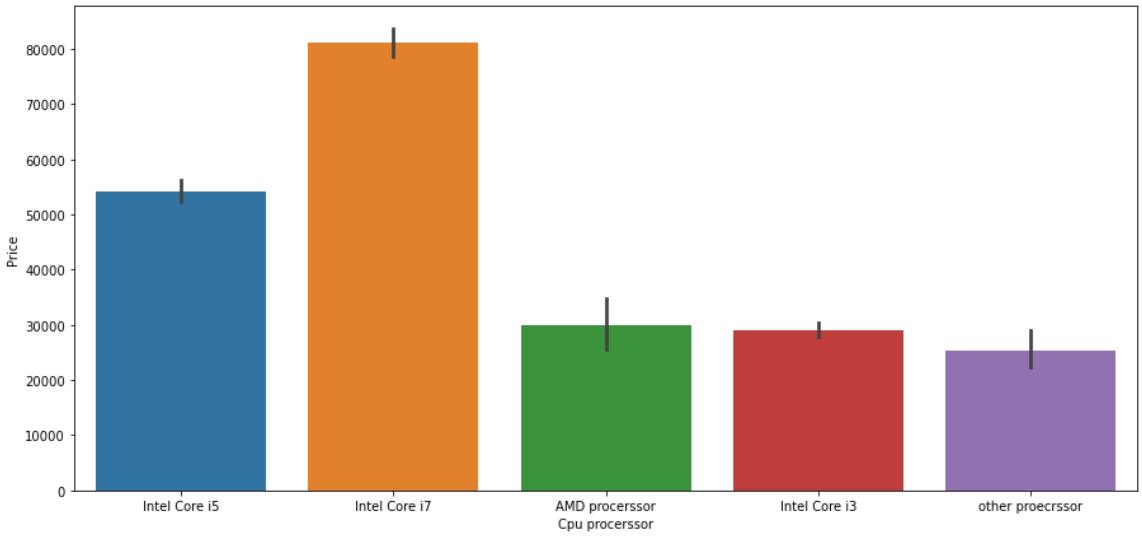
In [40]:
```
df['Cpu procerssor'].value_counts()
```

Out[40]:
```
Intel Core i7       476
Intel Core i5       410
other proecrssor    135
Intel Core i3       132
AMD procerssor       60
Name: Cpu procerssor, dtype: int64
```

In [41]:
```
#visualizing the count of cpu procesors
plt.figure(figsize=(15,7))
sns.countplot(data=df,x='Cpu procerssor')
plt.show()
```



In [42]:
```
plt.figure(figsize=(15,7))
sns.barplot(data=df,x='Cpu procerssor',y='Price')
plt.show()
```



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

In [ ]:

cleaning and Analysis on GPU column

In [44]:
```
df['Gpu'].unique()
```

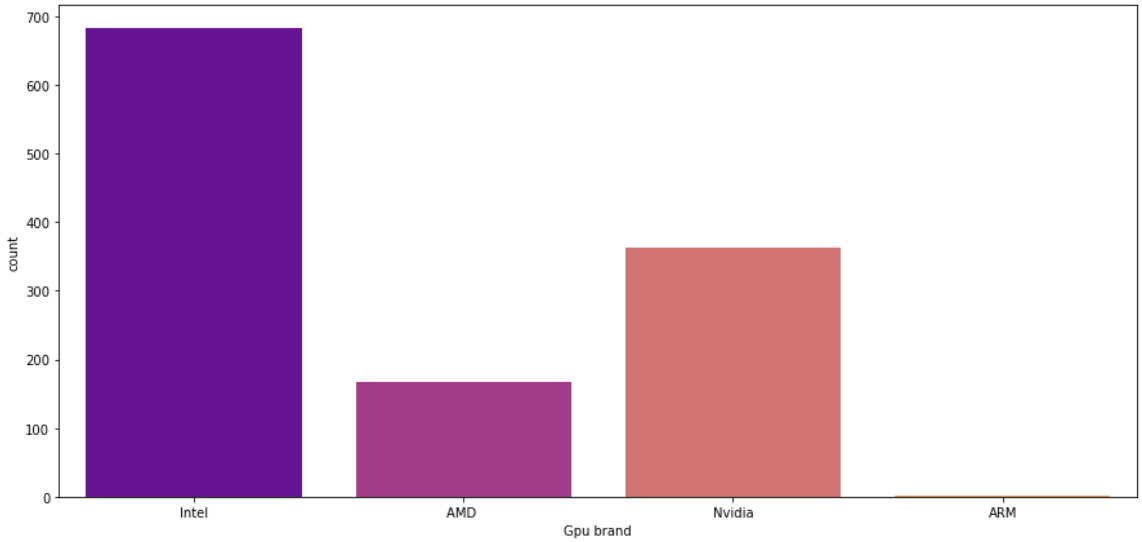
Out[44]:
```
array(['Intel Iris Plus Graphics 640', 'Intel HD Graphics 6000',
       'Intel HD Graphics 620', 'AMD Radeon Pro 455',
       'Intel Iris Plus Graphics 650', 'AMD Radeon R5',
       'Intel Iris Pro Graphics', 'Nvidia GeForce MX150',
       'Intel UHD Graphics 620', 'Intel HD Graphics 520',
       'AMD Radeon Pro 555', 'AMD Radeon R5 M430',
       'Intel HD Graphics 615', 'Nvidia GeForce 940MX',
       'Nvidia GeForce GTX 1050', 'AMD Radeon R2', 'AMD Radeon 530',
       'Nvidia GeForce 930MX', 'Intel HD Graphics',
       'Intel HD Graphics 500', 'Nvidia GeForce 930MX ',
       'Nvidia GeForce GTX 1060', 'Nvidia GeForce 150MX',
       'Intel Iris Graphics 540', 'AMD Radeon RX 580',
       'Nvidia GeForce 920MX', 'AMD Radeon R4 Graphics', 'AMD Radeon 52
0',
       'Nvidia GeForce GTX 1070', 'Nvidia GeForce GTX 1050 Ti',
       'Intel HD Graphics 400', 'Nvidia GeForce MX130', 'AMD R4 Graphic
s',
       'Nvidia GeForce GTX 940MX', 'AMD Radeon RX 560',
       'Nvidia GeForce 920M', 'AMD Radeon R7 M445', 'AMD Radeon RX 550',
```

In [45]:
```
#creating the GPU brand column that contain manufacturer of gpu
for i in df.index:
    if 'Intel' in  df.loc[i,'Gpu']:
        df.loc[i,'Gpu brand']='Intel'
    elif 'Nvidia'  in  df.loc[i,'Gpu']:
        df.loc[i,'Gpu brand']='Nvidia'
    elif 'ARM' in  df.loc[i,'Gpu']:
        df.loc[i,'Gpu brand']='ARM'
    elif 'AMD' in  df.loc[i,'Gpu']:
        df.loc[i,'Gpu brand']='AMD '
```

In [46]:
```
df['Gpu brand'].value_counts()
```

Out[46]:
```
Intel     682
Nvidia    363
AMD       167
ARM         1
Name: Gpu brand, dtype: int64
```

In [47]:
```
plt.figure(figsize=(15,7))
sns.countplot(data=df,x='Gpu brand',palette='plasma')
plt.show()
```



In [48]:
```
df.drop(columns=['Gpu'],inplace=True)
```

In [49]:
```
df.head()
```

Out[49]:

	Company	TypeName	Inches	Ram	Memory	OpSys	Weight	Price	Touchscreen
0	Apple	Ultrabook	13.3	8	128GB SSD	macOS	1.37	71378.6832	0
1	Apple	Ultrabook	13.3	8	128GB Flash Storage	macOS	1.34	47895.5232	0
2	HP	Notebook	15.6	8	256GB SSD	No OS	1.86	30636.0000	0
3	Apple	Ultrabook	15.4	16	512GB SSD	macOS	1.83	135195.3360	0
4	Apple	Ultrabook	13.3	8	256GB SSD	macOS	1.37	96095.8080	0

In [ ]:

cleaning and Analysis on Opsys column

In [50]:
```
df['OpSys'].value_counts()
```

Out[50]:
```
Windows 10      996
No OS            63
Linux            57
Windows 7        41
Chrome OS        27
macOS            12
Mac OS X          8
Windows 10 S      8
Android           1
Name: OpSys, dtype: int64
```

In [51]:
```
df['OpSys'].unique()
```

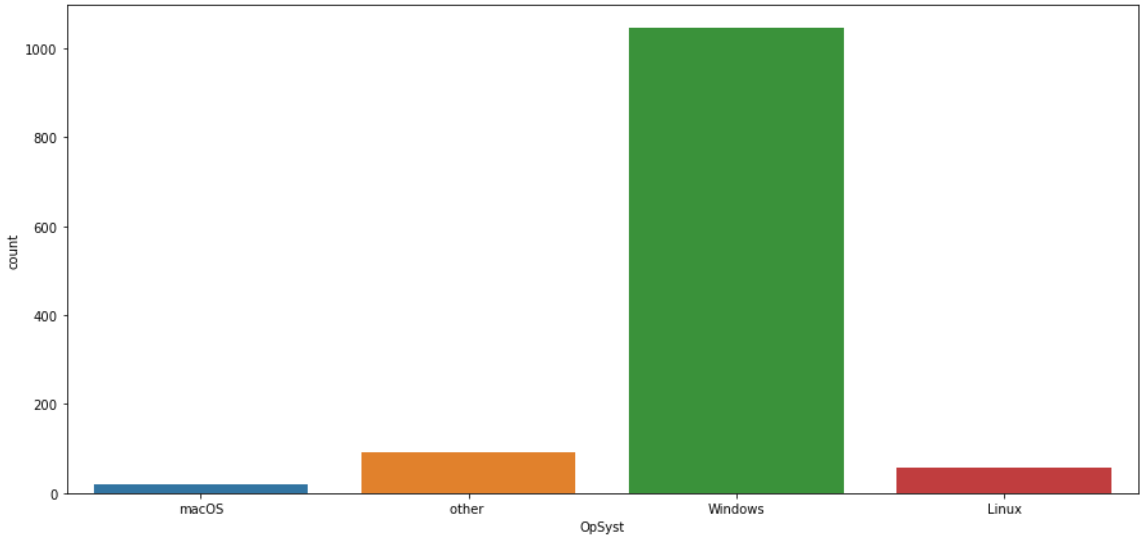
Out[51]:
```
array(['macOS', 'No OS', 'Windows 10', 'Mac OS X', 'Linux',
       'Windows 10 S', 'Chrome OS', 'Windows 7', 'Android'], dtype=object)
```

In [52]:
```
for i in df.index:
    if 'Windows' in  df.loc[i,'OpSys']:
        df.loc[i,'OpSyst']='Windows'
    elif 'macOS'  in  df.loc[i,'OpSys'] or 'Mac OS X'  in  df.loc[i,'OpSys'
        df.loc[i,'OpSyst']='macOS'
    elif 'Linux' in  df.loc[i,'OpSys']:
        df.loc[i,'OpSyst']='Linux'
    else:
        df.loc[i,'OpSyst']='other '
```

In [53]:
```
df['OpSyst'].value_counts()
```

Out[53]:
```
Windows    1045
other        91
Linux        57
macOS        20
Name: OpSyst, dtype: int64
```

In [54]:
```
plt.figure(figsize=(15,7))
sns.countplot(data=df,x='OpSyst')
plt.show()
```



In [55]:
```
df.drop(columns=['OpSys'],inplace=True)
```

In [ ]:

In [ ]:

cleaning and Analysis on Opsys column

In [56]:
```
df['Memory'].value_counts()
```

```
180GB SSD                          4
128GB Flash Storage                4
512GB SSD +  2TB HDD               3
16GB SSD                           3
512GB Flash Storage                2
256GB SSD +  500GB HDD             2
128GB SSD +  2TB HDD               2
256GB SSD +  256GB SSD             2
512GB SSD +  512GB SSD             1
512GB SSD +  256GB SSD             1
64GB Flash Storage +  1TB HDD      1
64GB SSD                           1
1TB HDD +  1TB HDD                 1
32GB HDD                           1
128GB HDD                          1
8GB SSD                            1
508GB Hybrid                       1
1.0TB HDD                          1
256GB SSD +  1.0TB Hybrid          1
Name: Memory, dtype: int64
```

In [57]:
```
# creating ssd column
for i in df.index:
    if 'SSD' in df.loc[i,'Memory'] or 'Hybrid' in df.loc[i,'Memory']:
        df.loc[i,'SSD']=1
    else:
        df.loc[i,'SSD']=0
```

In [58]:
```
# creating hdd coluumn
for i in df.index:
    if 'HDD' in df.loc[i,'Memory'] or 'Hybrid' in df.loc[i,'Memory']:
        df.loc[i,'HDD']=1
    else:
        df.loc[i,'HDD']=0
```

In [59]:
```
# creating flash storage column
for i in df.index:
    if 'Flash Storage' in df.loc[i,'Memory'] :
        df.loc[i,'Flash Storage']=1
    else:
        df.loc[i,'Flash Storage']=0
```

In [ ]:

In [60]:
```
# creating the storage column

for i in df.index:
    k=df.loc[i,'Memory'].split()
    if 'SSD' in k:
        k.remove('SSD')
    if 'HDD' in k:
        k.remove('HDD')
    if 'Hybrid' in k:
        k.remove('Hybrid')
    if 'Flash' in k:
        k.remove('Flash')
    if 'Storage' in k:
        k.remove('Storage')
    if 'SSD' in k:
        k.remove('SSD')
    if 'HDD' in k:
        k.remove('HDD')
    if '+' in k:
        k.remove('+')
    storage=0
    if len(k)==1:
        if 'GB' in k[0]:
            k[0]=k[0].replace("GB",'')
        if 'TB' in k[0]:
            if '.0TB' in k[0]:
                k[0]=k[0].replace(".0TB","000")
            else:
                k[0]=k[0].replace("TB","000")
        storage=int(k[0])
        df.loc[i,'Storage']=storage
    if len(k)==2:
        if 'GB' in k[0]:
            k[0]=k[0].replace("GB",'')
        if 'TB' in k[0]:
            if '.0TB' in k[0]:
                k[0]=k[0].replace(".0TB","000")
            else:
                k[0]=k[0].replace("TB","000")

        if 'GB' in k[1]:
            k[1]=k[1].replace("GB",'')
        if 'TB' in k[1]:
            if '.0TB' in k[1]:
                k[1]=k[1].replace(".0TB","000")
            else:
                k[1]=k[1].replace("TB","000")
        storage=int(k[0])+int(k[1])
        df.loc[i,'Storage']=storage









    df.loc[i,'New']=" ".join(k)
```

In [61]:
```
df['New'].value_counts()
```

```
256 1000      69
32            42
2000          16
64            13
16            10
512 1000      10
256 2000      10
180            4
512 2000       3
256 500        2
128 2000       2
256 256        2
512 512        1
512 256        1
64 1000        1
1000 1000      1
8              1
508            1
Name: New, dtype: int64
```

In [62]:
```
df.head()
```

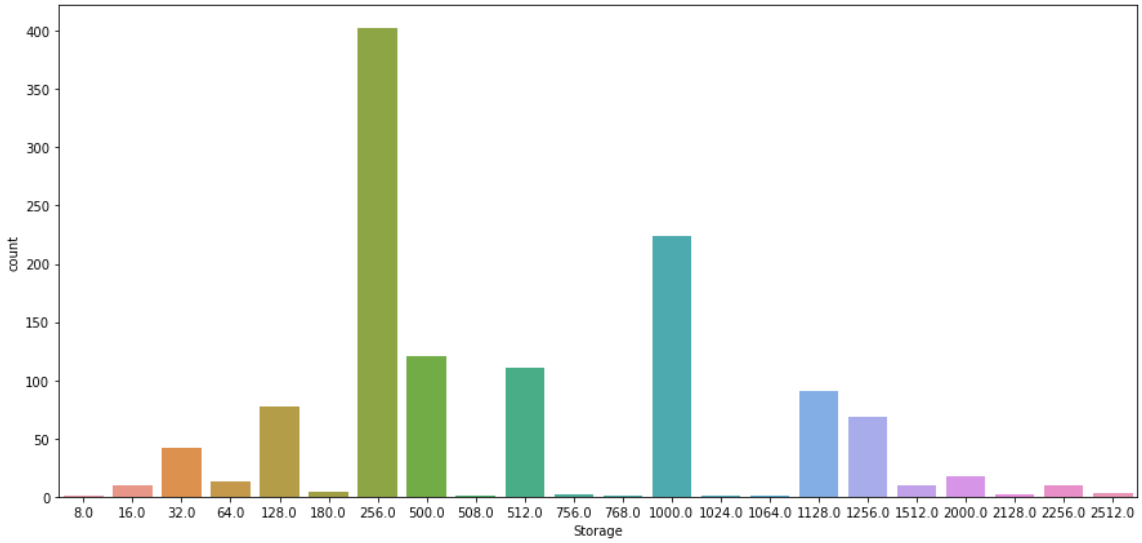
Out[62]:

	Company	TypeName	Inches	Ram	Memory	Weight	Price	Touchscreen	IPS	
0	Apple	Ultrabook	13.3	8	128GB SSD	1.37	71378.6832	0	1	226
1	Apple	Ultrabook	13.3	8	128GB Flash Storage	1.34	47895.5232	0	0	127
2	HP	Notebook	15.6	8	256GB SSD	1.86	30636.0000	0	0	141
3	Apple	Ultrabook	15.4	16	512GB SSD	1.83	135195.3360	0	1	220
4	Apple	Ultrabook	13.3	8	256GB SSD	1.37	96095.8080	0	1	226

In [63]:
```
df['Storage'].value_counts()
```

Out[63]:
```
256.0     402
1000.0    224
500.0     121
512.0     111
1128.0     91
128.0      77
1256.0     69
32.0       42
2000.0     17
64.0       13
16.0       10
1512.0     10
2256.0     10
180.0       4
2512.0      3
756.0       2
2128.0      2
1024.0      1
768.0       1
```

In [64]:
```
plt.figure(figsize=(15,7))
sns.countplot(data=df,x='Storage')
plt.show()
```



In [65]:
```
df.drop(columns=["New","Memory"],inplace=True)
```

In [66]:
```
df.head()
```

Out[66]:

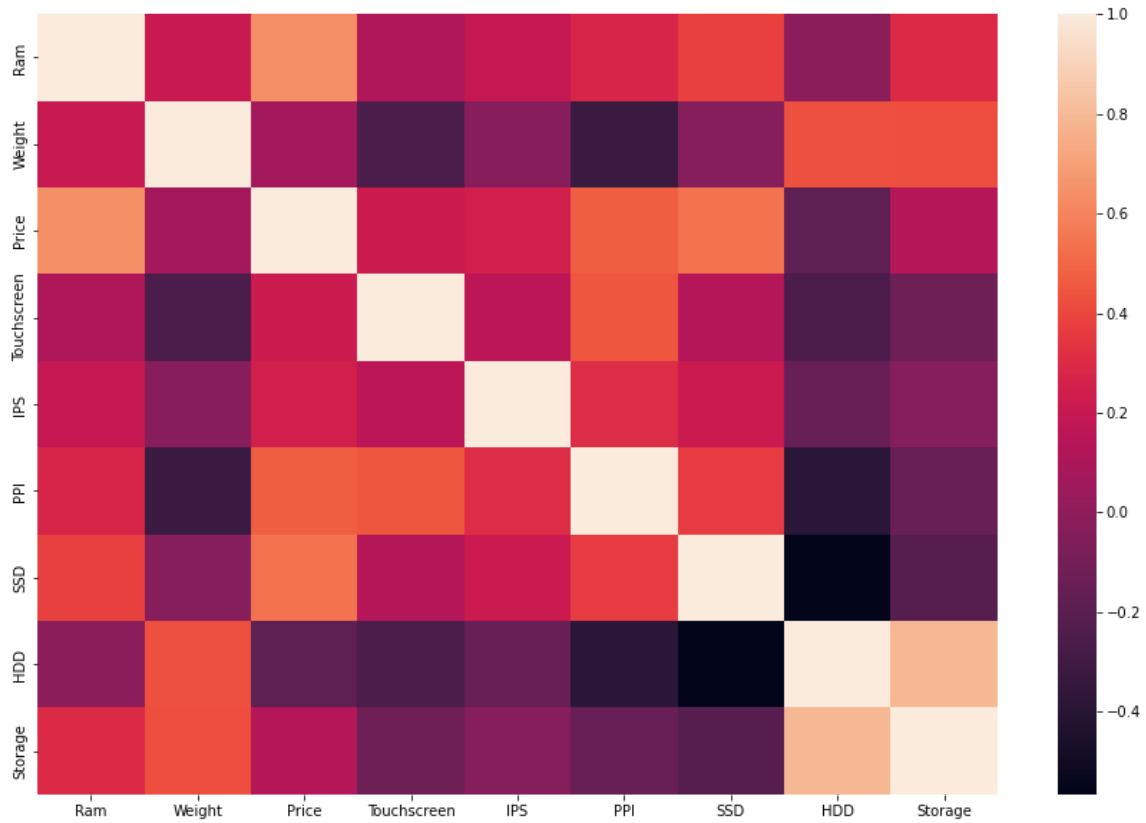
	Company	TypeName	Inches	Ram	Weight	Price	Touchscreen	IPS	PPI
0	Apple	Ultrabook	13.3	8	1.37	71378.6832	0	1	226.983001
1	Apple	Ultrabook	13.3	8	1.34	47895.5232	0	0	127.677938
2	HP	Notebook	15.6	8	1.86	30636.0000	0	0	141.211995
3	Apple	Ultrabook	15.4	16	1.83	135195.3360	0	1	220.534629
4	Apple	Ultrabook	13.3	8	1.37	96095.8080	0	1	226.983001

In [67]:
```
df.drop(columns=["Flash Storage","Inches"],inplace=True)
```

In [ ]:
```

```

In [68]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr())
```

Out[68]: <AxesSubplot:>

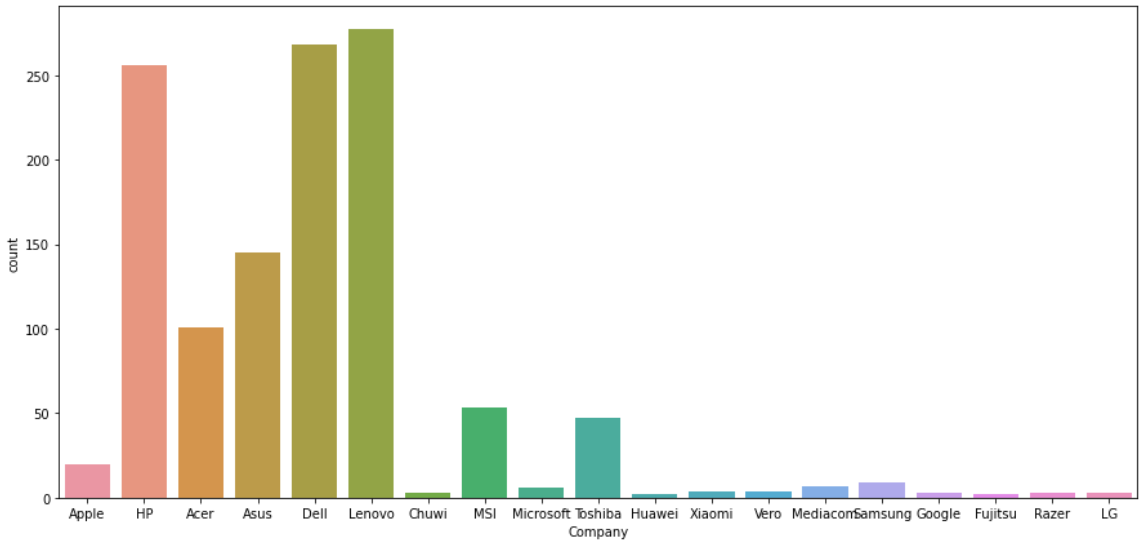


In [69]:
```
df.head()
```

Out[69]:

	Company	TypeName	Ram	Weight	Price	Touchscreen	IPS	PPI	Cp procerss
0	Apple	Ultrabook	8	1.37	71378.6832	0	1	226.983001	Intel Co
1	Apple	Ultrabook	8	1.34	47895.5232	0	0	127.677938	Intel Co
2	HP	Notebook	8	1.86	30636.0000	0	0	141.211995	Intel Co
3	Apple	Ultrabook	16	1.83	135195.3360	0	1	220.534629	Intel Co
4	Apple	Ultrabook	8	1.37	96095.8080	0	1	226.983001	Intel Co

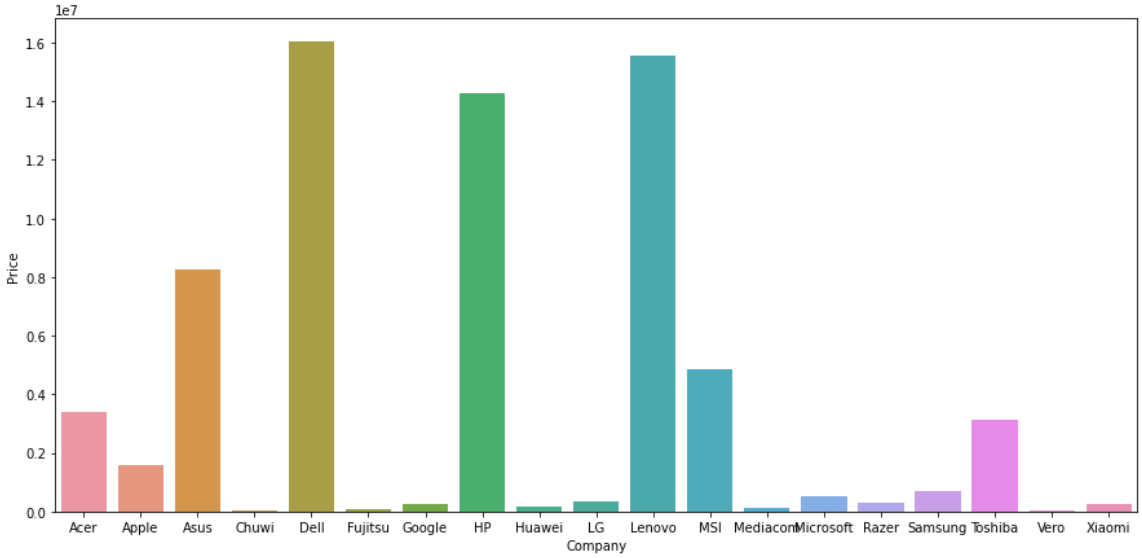
In [73]:
```
plt.figure(figsize=(15,7))
sns.countplot(data=df,x='Company')
plt.show()
```



In [ ]:

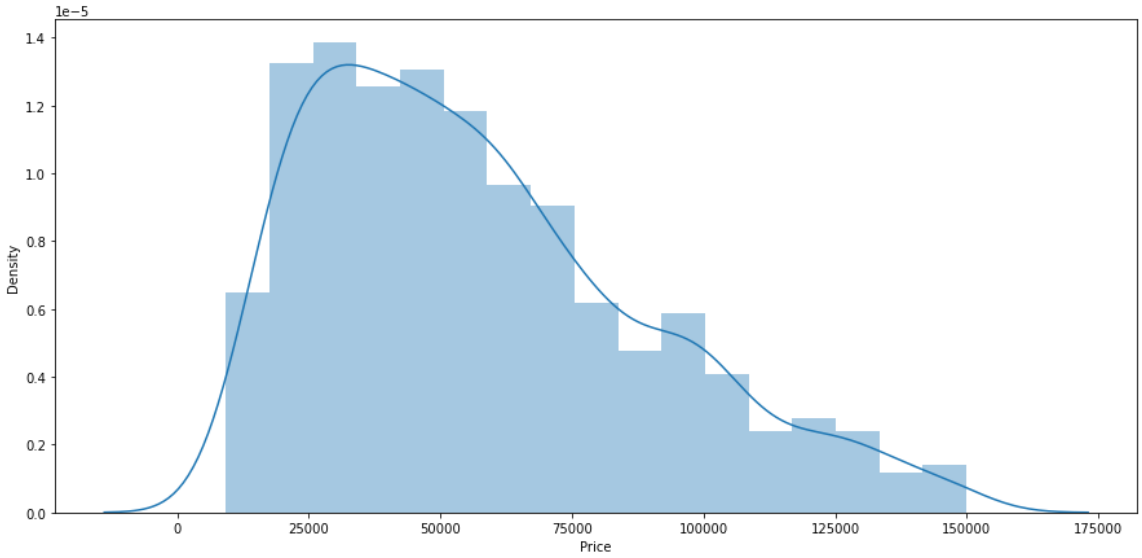
In [100]:
```
plt.figure(figsize=(15,7))

sns.barplot(data=k,x='Company',y='Price')
```

Out[100]:
```
<AxesSubplot:xlabel='Company', ylabel='Price'>
```



In [223]:
```
plt.figure(figsize=(15,7))
sns.distplot(df['Price'])
plt.show()
```

```
C:\Users\91630\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```



Encoding the categorical values

In [101]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
ohe = OneHotEncoder(handle_unknown='ignore', drop='first')
```

In [102]:
```
transformer = ColumnTransformer(
    transformers = [('encoder', OneHotEncoder(),[0,1,7,8,9])],
    remainder='passthrough')

x=df.drop(columns='Price')
y=df['Price']
x=transformer.fit_transform(x)
```

In [103]:
```
for i in x[29]:
    print(i)
print(x[0].shape)
```

```
  (0, 2)        1.0
  (0, 22)       1.0
  (0, 25)       1.0
  (0, 30)       1.0
  (0, 35)       1.0
  (0, 38)       2.0
  (0, 39)       1.649999976158142
  (0, 42)       111.93520355955211
  (0, 45)       32.0
(1, 46)
```

In [104]:
```
df.nunique()
```

Out[104]:
```
Company            19
TypeName            6
Ram                10
Weight            176
Price             750
Touchscreen         2
IPS                 2
PPI                47
Cpu procerssor      5
Gpu brand           4
OpSyst              4
SSD                 2
HDD                 2
Storage            22
dtype: int64
```

In [ ]:

splitting the data

In [ ]:

In [105]:
```
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=42)
```

In [106]:
```
y_test
```

Out[106]:
```
382     16303.6800
787     32980.3200
43      75604.3200
155     38787.8400
493     64961.1072
           ...
184     53274.6720
424     76137.1200
1159    67772.1600
259     43263.3600
778     26101.8720
Name: Price, Length: 182, dtype: float64
```

Multi linaer regression

In [107]:
```
# importing the module and creating a model
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr
```

Out[107]:
```
LinearRegression()
```

In [108]:
```
# training a model
lr.fit(x_train,y_train)
```

Out[108]:
```
LinearRegression()
```

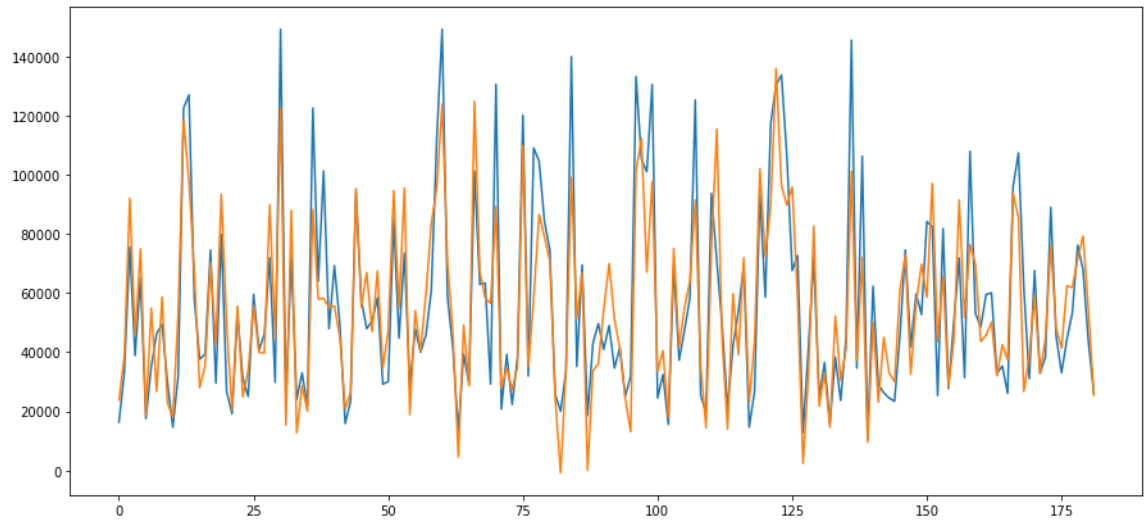
In [109]:
```
# testing the model on testing data
y_pred=lr.predict(x_test)
```

In [110]:
```
# finding the accuracy of the model
from  sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
print(r2_score(y_test,y_pred))
```

```
0.7653176339757078
```

In [111]:
```
# visualizing the performance of the model
plt.figure(figsize=(15,7))
plt.plot(y_test.values)
plt.plot(y_pred)
```

Out[111]:
```
[<matplotlib.lines.Line2D at 0x25b97d04160>]
```



In [ ]:

Ridge regression

In [112]:
```
# importing and creating the model
from sklearn.linear_model import Ridge
rig = Ridge(alpha=10)
```

In [113]:
```
# training th emodel
rig.fit(x_train,y_train)
```

Out[113]:
```
Ridge(alpha=10)
```

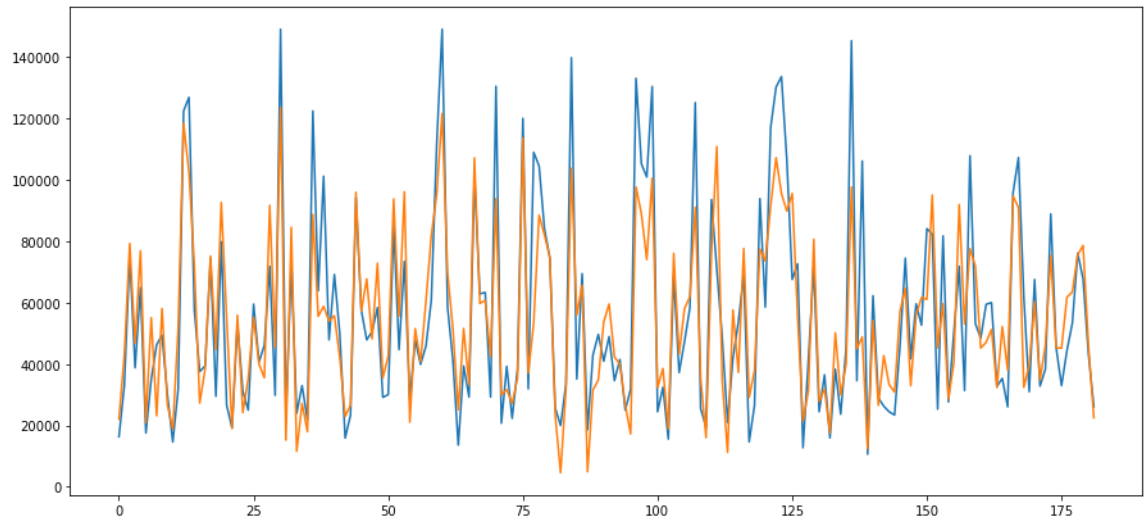
In [114]:
```
# testing and predicting the output on x_test
y_pred=rig.predict(x_test)
```

In [115]:
```
# finding the accuray of the model
from  sklearn.metrics import r2_score
print(r2_score(y_test,y_pred))
```

```
0.7623373672657158
```

In [116]:
```
# visualizing the performance of the model
plt.figure(figsize=(15,7))
plt.plot(y_test.values)
plt.plot(y_pred)
```

Out[116]:
```
[<matplotlib.lines.Line2D at 0x25b97d79df0>]
```



In [ ]:

Decision Tree regressor

In [117]:
```
# importing and creating the model
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor(max_depth=10,
                         random_state=15,
                         ccp_alpha=0.0085)
```

In [118]:
```
# training a model
dt.fit(x_train,y_train)
```

Out[118]:
```
DecisionTreeRegressor(ccp_alpha=0.0085, max_depth=10, random_state=15)
```

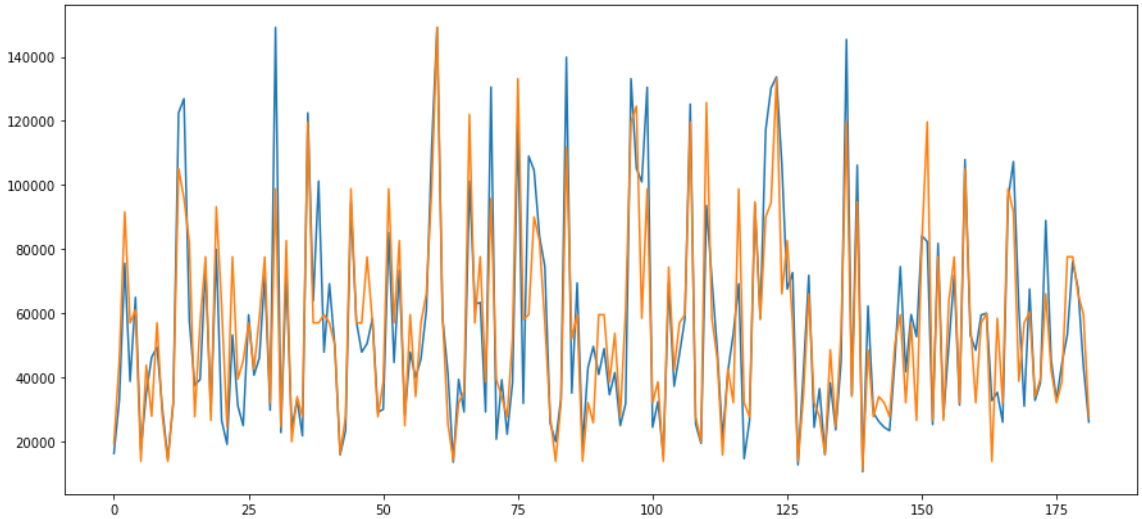
In [119]:
```
# testing and predicting th eoutput on x_test
y_pred=dt.predict(x_test)
```

In [120]:
```
# finding the accuray of the model
from  sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
print(r2_score(y_test,y_pred))
```

```
0.7881831030087202
```

In [121]:
```
# visualizing the performance of the model

plt.figure(figsize=(15,7))
plt.plot(y_test.values)
plt.plot(y_pred)
```

Out[121]:
```
[<matplotlib.lines.Line2D at 0x25b9822c430>]
```



Random Forest regressor

In [ ]:
```

```

In [122]:
```
# importing and creating the model
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=150,
                           random_state=3,
                           max_samples=0.5,
                           max_features=0.75,
                           max_depth=30
                          )
```

In [123]:
```
# training a model

rf.fit(x_train,y_train)
```

Out[123]:
```
RandomForestRegressor(max_depth=30, max_features=0.75, max_samples=0.5,
                      n_estimators=150, random_state=3)
```

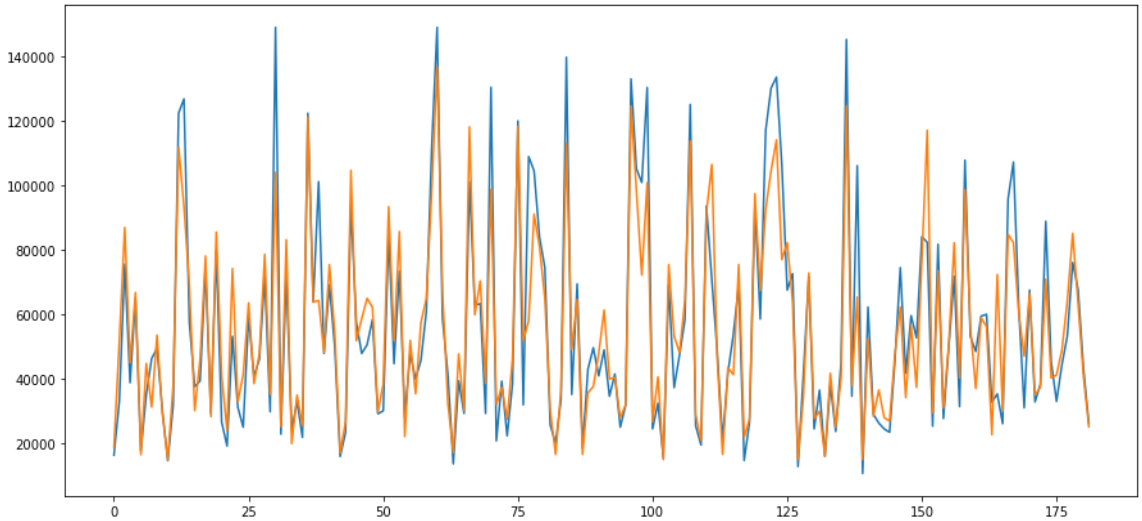
In [124]:
```
# testing and predicting th eoutput on x_test

y_pred=rf.predict(x_test)
```

In [125]:
```
# finding the accuray of the model

from  sklearn.metrics import r2_score,mean_squared_error,mean_absolute_erro
print(r2_score(y_test,y_pred))
```

```
0.8508199098128405
```

In [126]:
```
# visualizing the performance of the model
plt.figure(figsize=(15,7))
plt.plot(y_test.values)
plt.plot(y_pred)
```

Out[126]:
```
[<matplotlib.lines.Line2D at 0x25b983ae970>]
```



In [127]:
```
dfm=pd.DataFrame({"Actual":y_test.values,"Pred":y_pred})
dfm
```

Out[127]:

	Actual	Pred
0	16303.6800	18671.747488
1	32980.3200	51615.738816
2	75604.3200	86989.959704
3	38787.8400	45089.140688
4	64961.1072	66832.624032
...	...	...
177	53274.6720	64124.173120
178	76137.1200	85198.916688
179	67772.1600	63444.490816
180	43263.3600	40762.080672
181	26101.8720	25095.508704

182 rows × 2 columns

In [128]:
```
df.loc[361,]
```

Out[128]:
```
Company                    Acer
TypeName               Notebook
Ram                           8
Weight                      2.4
Price                   45074.88
Touchscreen                    0
IPS                            0
PPI                   141.211995
Cpu procerssor    Intel Core i7
Gpu brand                 Nvidia
OpSyst                     Linux
SSD                          0.0
HDD                          1.0
Storage                   1000.0
Name: 361, dtype: object
```

In [175]:
```
a=rf.feature_importances_
```

In [ ]:
```
com_imp=
```

XG - Boost regressor

In [129]:
```
# importing and creating the model

from xgboost import XGBRegressor
xgb_r = XGBRegressor(objective ='reg:linear',
                     booster="gblinear",
                     max_depth=200, eta=0.1,
                     subsample=0.7,
                     colsample_bytree=0.8,
                     n_estimators = 1000,
                     seed = 200)
```

In [130]:
```
# training a model

xgb_r.fit(x_train, y_train)
```

```
C:\Users\91630\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarnin
g: [10:51:22] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-aut
oscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\object
ive\regression_obj.cu:209: reg:linear is now deprecated in favor of reg:sq
uarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\91630\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarnin
g: [10:51:22] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-aut
oscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\learne
r.cc:742:
Parameters: { "colsample_bytree", "max_depth", "subsample" } are not used.

  warnings.warn(smsg, UserWarning)
```

Out[130]:
```
XGBRegressor(base_score=None, booster='gblinear', callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.8, device=None, early_stopping_rounds=Non
e,
             enable_categorical=False, eta=0.1, eval_metric=None,
             feature_types=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=None, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=200,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=
1000,
             n_jobs=None, num_parallel_tree=None, ...)
```

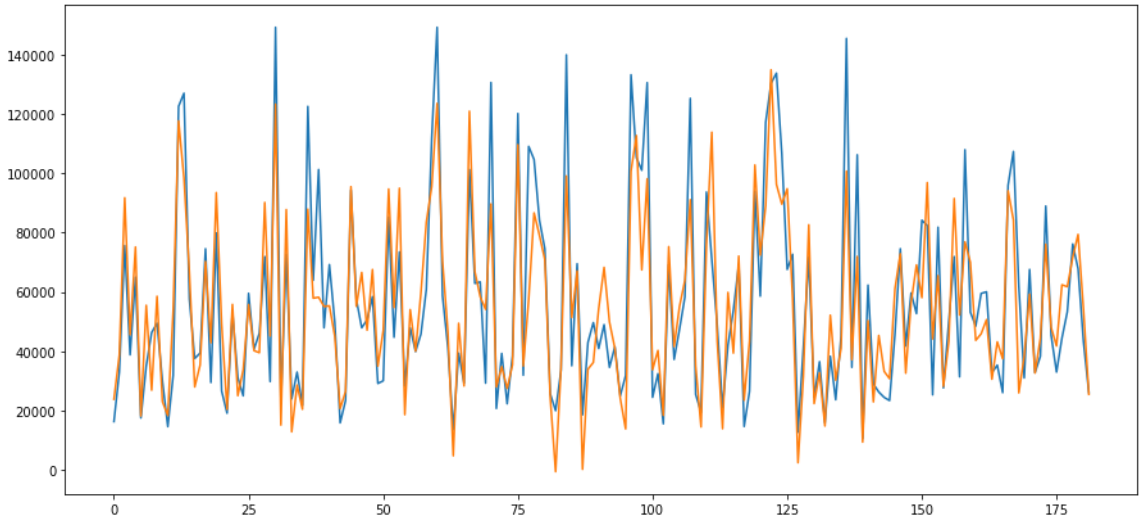
In [131]:
```
# testing and predicting th eoutput on x_test

y_pred=xgb_r.predict(x_test)
```

In [132]:
```
# finding the accuray of the model

from  sklearn.metrics import r2_score,mean_squared_error,mean_absolute_erro
print(r2_score(y_test,y_pred))
```

```
0.7671961358906454
```

In [133]:
```
# visualizing the performance of the model

plt.figure(figsize=(15,7))
plt.plot(y_test.values)
plt.plot(y_pred)
```

Out[133]:
```
[<matplotlib.lines.Line2D at 0x25b8f530670>]
```



In [ ]:

In [ ]:

feature importance

In [201]:
```
a=rf.feature_importances_
```

In [202]:
```
com_imp=a[0:19].sum()
Type_imp=a[19:25].sum()
cpu_imp=a[25:30].sum()
ppu_imp=a[30:34].sum()
os_imp=a[34:38].sum()
rem=list(a[38:])
```

In [203]:
```
print(com_imp)
print(Type_imp)
print(cpu_imp)
print(ppu_imp)
print(os_imp)
rem
```

```
0.043532462964768114
0.1274843843515954
0.14129184445673426
0.015291080972095271
0.010546925674236655
```

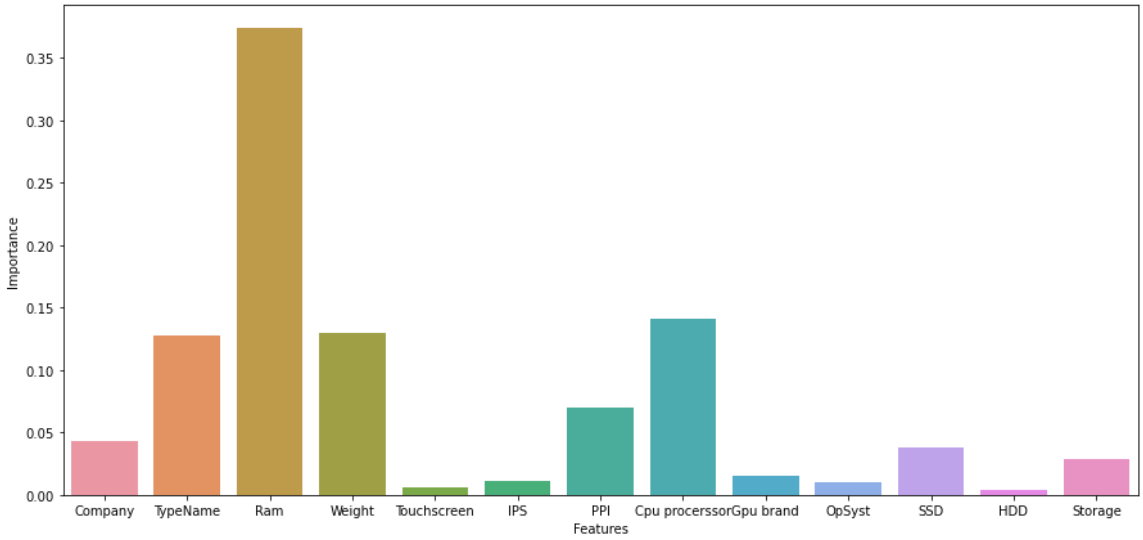
Out[203]:
```
[0.37355150186278374,
 0.12960576707213342,
 0.006167806506549625,
 0.010767164847454465,
 0.07034817701621192,
 0.03830288390185594,
 0.003983294924680326,
 0.029126705448901086]
```

In [204]:
```
b=np.zeros(13)
b[0]=com_imp
b[1]=Type_imp
b[2]=a[38]
b[3]=a[39]
b[4]=a[40]
b[5]=a[41]
b[6]=a[42]
b[7]=cpu_imp
b[8]=ppu_imp
b[9]=os_imp
b[10]=a[43]
b[11]=a[44]
b[12]=a[45]

b
```

Out[204]:
```
array([0.04353246, 0.12748438, 0.3735515 , 0.12960577, 0.00616781,
       0.01076716, 0.07034818, 0.14129184, 0.01529108, 0.01054693,
       0.03830288, 0.00398329, 0.02912671])
```

In [205]:
```
imp_df1=pd.DataFrame({"Features":x.columns,'Importance':b})
```

In [206]:
```
plt.figure(figsize=(15,7))
sns.barplot(data=imp_df1,x='Features',y='Importance')
plt.show()
```



input scanner

In [207]:

```
comp=int(input('''Choose the company
0: Acer 1:Apple
2:Asus  3:Chuwi
4:Dell  5:Fujitsu
6:Google  7:HP
8:Huawei  9:Lenovo
10:LG     11:Mediacom
12:Microsoft  13:MSI
14:Razer  15:Samsung
16:Toshiba  17:Vero
18:Xiaomi                ''']))

Type=int(input('''Choose the Type
0: 2 in 1 Convertible  1:Gaming
2:Netbook  3:Notebook
4:Ultrabook  5:Workstation   '''))

ram=int (input("Enter the ram"))
wt=float(input("Enter the weight of laptop"))
Touchscr=int(input(''' Enter type of laptop
1: Touch screen  0: Not touch screen''' ))
IPS=int(input(''' Enter type of laptop
1: IPS  0: Not IPS''' ))
x_res=float(input("Enter x_resolurion"))
y_res=float(input("Enter y_resolurion"))
processor=int(input('''Choose the cpu processor
0: AMD procerssor  1:Intel Core i3
2:Intel Core i5  3:Intel Core i7
4:other proecrssor '''))
inch=float(input("Enter the inches"))
gpu=int(input('''Choose  GPU Brand
0: AMD   1:ARM
2:Intel  3:Nvidia '''))

ops=int(input('''Choose  operating system
0: Linux   1:Windows
2:macOS  3:other '''))
ssd=int(input(''' choose the below
1: SSD  0: Not SSD''' ))
hdd=int(input(''' choose the below
1: HDD  0: Not HDD''' ))
storage=float(input("Enter the storage "))
```

```
Choose the company
0: Acer 1:Apple
2:Asus  3:Chuwi
4:Dell  5:Fujitsu
6:Google  7:HP
8:Huawei  9:Lenovo
10:LG     11:Mediacom
12:Microsoft  13:MSI
14:Razer  15:Samsung
16:Toshiba  17:Vero
18:Xiaomi                2
Choose the Type
0: 2 in 1 Convertible  1:Gaming
2:Netbook  3:Notebook
4:Ultrabook  5:Workstation   1
Enter the ram16
Enter the weight of laptop2.3
 Enter type of laptop
1: Touch screen  0: Not touch screen0
 Enter type of laptop
1: IPS  0: Not IPS0
Enter x_resolurion1920
Enter y_resolurion1080
Choose the cpu processor
0: AMD procerssor  1:Intel Core i3
2:Intel Core i5  3:Intel Core i7
4:other proecrssor 2
Enter the inches15.6
Choose  GPU Brand
0: AMD   1:ARM
2:Intel  3:Nvidia 3
Choose  operating system
0: Linux   1:Windows
2:macOS  3:other 1
 choose the below
1: SSD  0: Not SSD1
 choose the below
1: HDD  0: Not HDD0
Enter the storage 512
```

In [208]:
```
lst=[]
company=np.zeros(19)
company[comp]=1
lst.extend(company)

lap_type=np.zeros(6)
lap_type[Type]=1
lst.extend(lap_type)

cpu_processor=np.zeros(5)
cpu_processor[processor]=1
lst.extend(cpu_processor)

Gpu_brand=np.zeros(4)
Gpu_brand[gpu]=1
lst.extend(Gpu_brand)

op_sys=np.zeros(4)
op_sys[ops]=1
lst.extend(op_sys)

z=math.sqrt(x_res**2+y_res**2)
ppi=z/inch
lst.extend([ram,wt,Touchscr,IPS,ppi,ssd,hdd,storage])
```

In [ ]:

In [209]:
```
price=rf.predict(np.array(lst).reshape(1,46))
```

In [210]:
```
price
```

Out[210]:
```
array([85899.66768])
```

In [ ]:
```
# Actual price for above laptop is 89,000
# our model prediction is 86,000 approximately
```

In [ ]: