Laptop Price Prediction for SmartTech Co. (Machine Learning Capstone Project)

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.

Questions to Explore:

- Which features have the most significant impact on laptop prices?
- Can the model accurately predict the prices of laptops from lesser-known brands?
- Does the brand of the laptop significantly influence its price?
- How well does the model perform on laptops with high-end specifications compared to budget laptops?
- What are the limitations and challenges in predicting laptop prices accurately?
- How does the model perform when predicting the prices of newly released laptops not present in the training dataset?

Work Overview :

- This project uses supervised ML algorithm in which Gradient Boosting Regressor has performed best among all regression models by scoring an accuracy rate of 88.6% with hyper parameter tuning.
- Here Price is the target variable and all the other columns are the features affecting the Price of the laptops.
- I've used various graphs showing EDA of the features and the distribution of actual vs predicted values.

Lets begin by importing libraries !

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Data Exploration and Understanding

In [6]:

```
df = pd.read_csv(r"C:\Users\Parth\Downloads\laptop\laptop.csv")
```

In [7]:

```
df.head()
```

Out[7]:

	Unnamed: 0.1	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	\
0	0	0.0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	
1	1	1.0	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	
2	2	2.0	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	
3	3	3.0	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	
4	4	4.0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Unnamed: 0.1      1303 non-null   int64  
 1   Unnamed: 0        1273 non-null   float64
 2   Company           1273 non-null   object 
 3   TypeName          1273 non-null   object 
 4   Inches            1273 non-null   object 
 5   ScreenResolution  1273 non-null   object 
 6   Cpu               1273 non-null   object 
 7   Ram               1273 non-null   object 
 8   Memory            1273 non-null   object 
 9   Gpu               1273 non-null   object 
 10  OpSys             1273 non-null   object 
 11  Weight            1273 non-null   object 
 12  Price             1273 non-null   float64
dtypes: float64(2), int64(1), object(10)
memory usage: 132.5+ KB
```

In [9]:

```
df.isnull().sum()
```

```
Out[9]: Unnamed: 0.1         0
        Unnamed: 0          30
        Company             30
        TypeName            30
        Inches              30
        ScreenResolution    30
        Cpu                 30
        Ram                 30
        Memory              30
        Gpu                 30
        OpSys               30
        Weight              30
        Price               30
        dtype: int64
```

Note : There are 30 missing values which do not have any information

Data Preprocessing

```
In [10]:
# Removing useless columns

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

```
In [11]:
# Filling null values

df['Price'] = df['Price'].fillna(df['Price'].median())
```

```
In [12]:
categorical_columns = ['Company', 'TypeName', 'Inches', 'ScreenResolution', 'Cpu', 'Ram', 'Memory', 'Gpu', '
for column in categorical_columns:
 df[column] = df[column].fillna(df[column].mode()[0])
```

```
In [13]:
# Replacing extra strings

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

```
In [14]:
# changing dtypes

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

```
In [15]:
df['Weight'] = df['Weight'].replace('?', np.nan)
df['Weight'] = df['Weight'].astype('float32')
df['Inches'] = df['Inches'].replace('?', np.nan)
df['Inches'] = df['Inches'].astype('float64')
```

In [16]:
```
df['Weight']= df['Weight'].fillna(df['Weight'].median())
df['Inches']= df['Inches'].fillna(df['Inches'].median())
```

In [17]:
```
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 [18]:
```
df.isnull().sum()
```

In [18]:
```
df.isnull().sum()
```

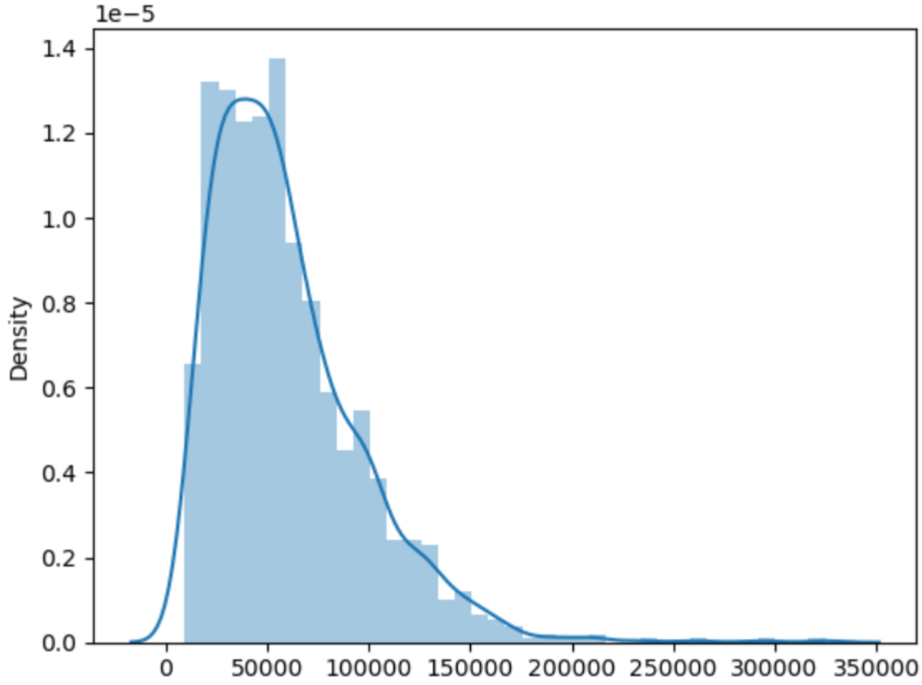
Out[18]:
```
Company             0
TypeName            0
Inches              0
ScreenResolution    0
Cpu                 0
Ram                 0
Memory              0
Gpu                 0
OpSys               0
Weight              0
Price               0
dtype: int64
```

Now data is ready for EDA

Data Analysis for Insights

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

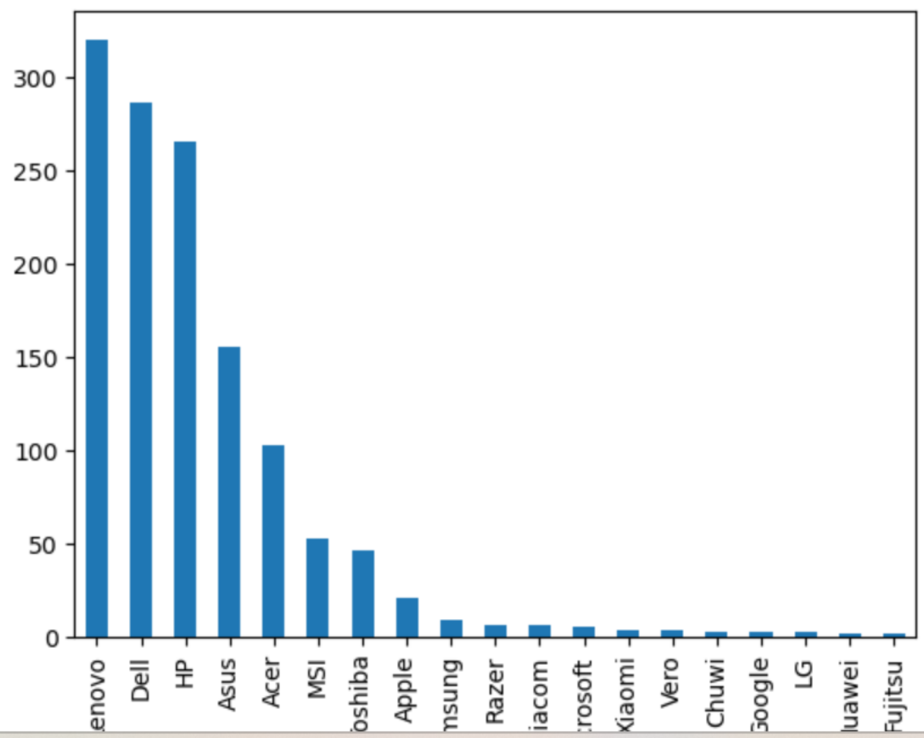
Out[19]: <Axes: xlabel='Price', ylabel='Density'>



From the above diagram we can see the price is right skewed as there are many laptops available within a budget range starting from 10000-150000 and more, hence we can say it is rightly skewed.

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

Out[20]: <Axes: xlabel='Company'>



The top 5 companies having more number of laptops :

- Lenovo

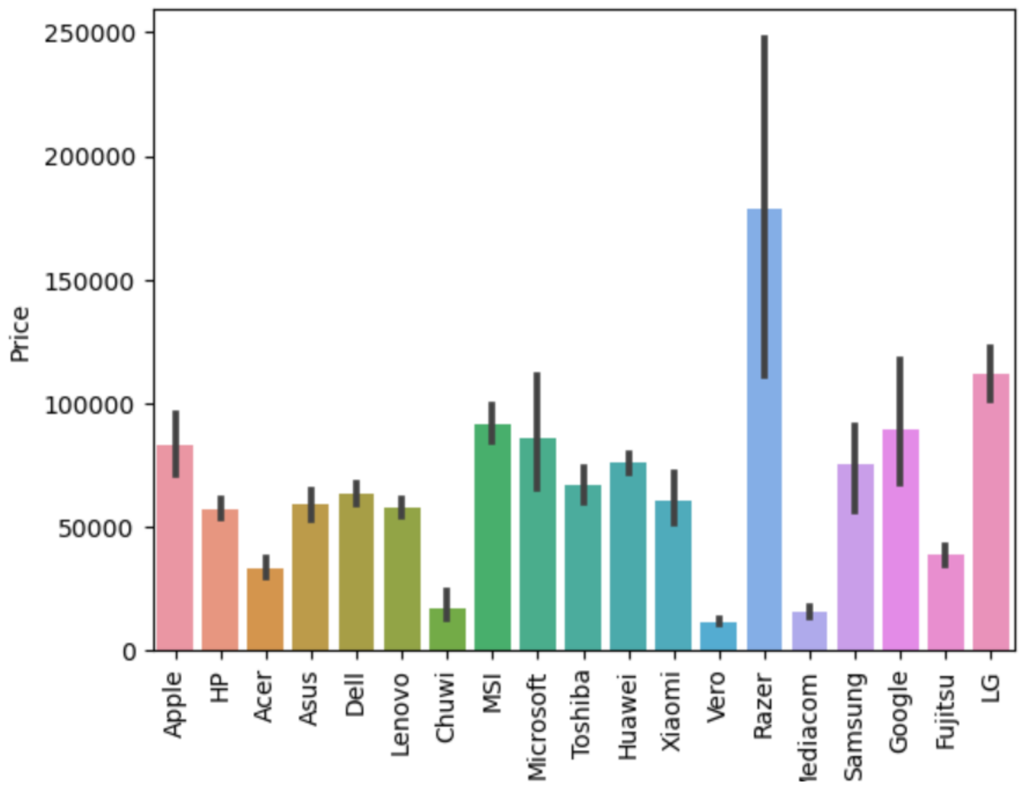
- Dell

- HP

- Asus

- Acer

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



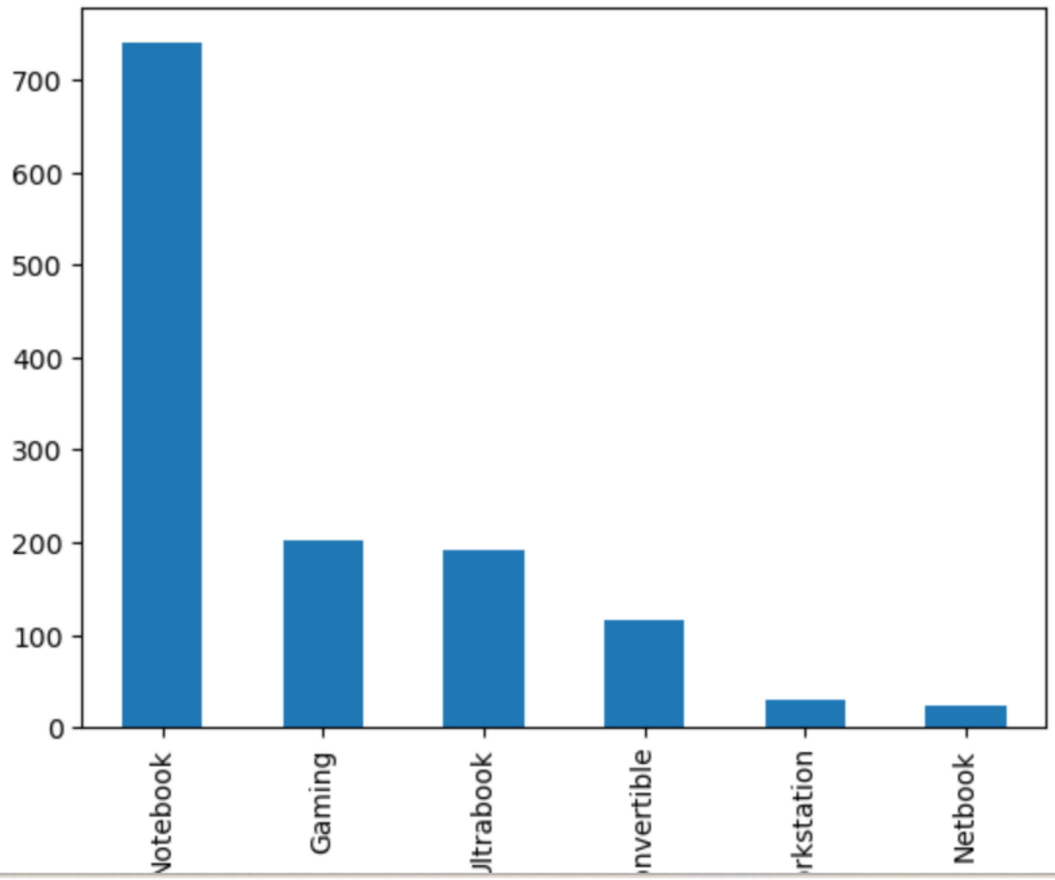
Correlation between the price of laptops and company

Mark point :

Razer laptops seems to be amongst the highest priced possible for their gaming performance.

In [22]:

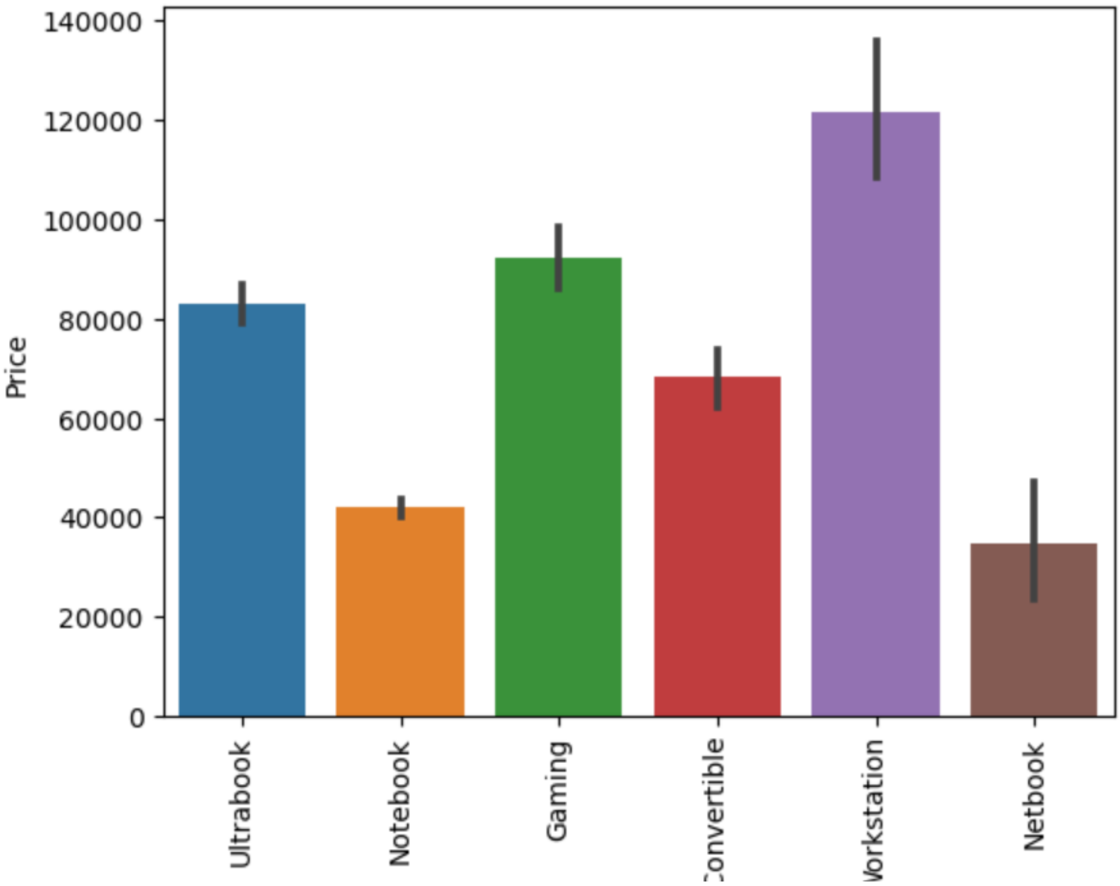
```
df['TypeName'].value_counts().plot(kind='bar')
```



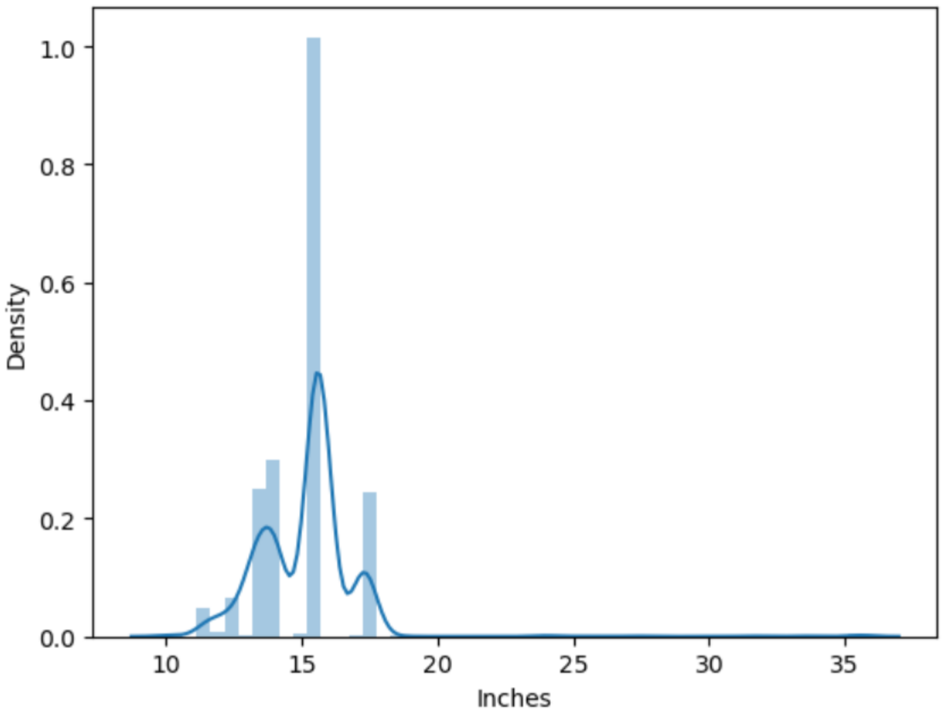
- **Notebooks are highly sold laptops followed by gaming, ultrabook, etc**

In [23]:

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

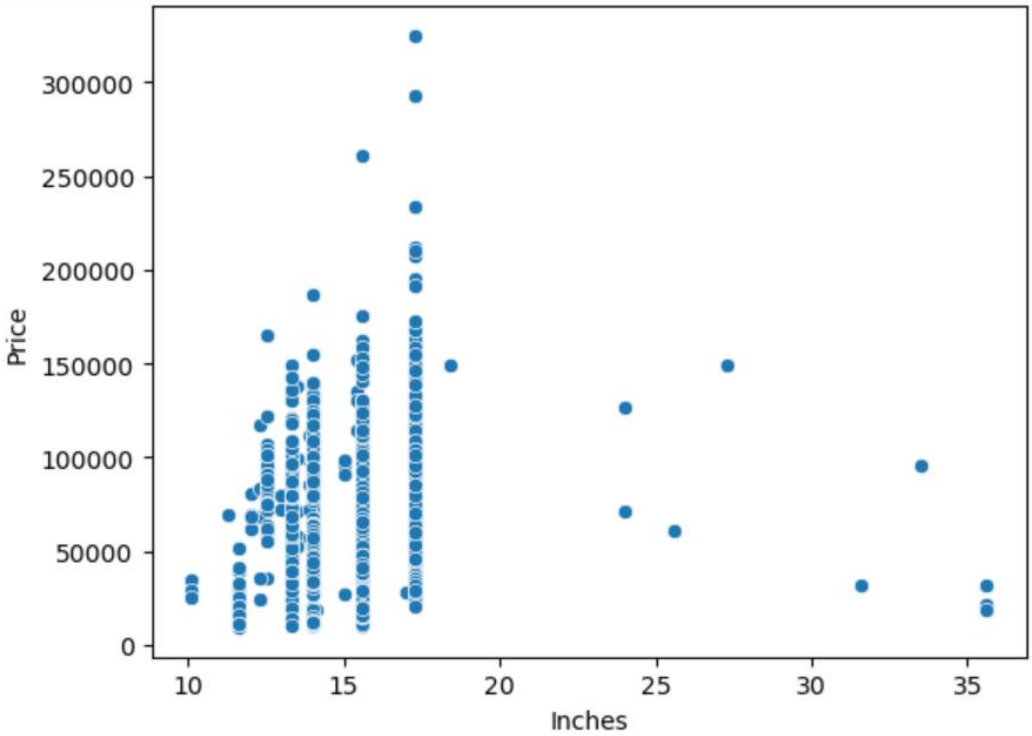



```
sns.distplot(df['Inches'])
```



Maximum size around : 15 inches

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




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

```
ScreenResolution
Full HD 1920x1080                                  525
1366x768                                           274
IPS Panel Full HD 1920x1080                        226
IPS Panel Full HD / Touchscreen 1920x1080           52
Full HD / Touchscreen 1920x1080                     45
1600x900                                            23
Touchscreen 1366x768                                16
Quad HD+ / Touchscreen 3200x1800                    14
IPS Panel 4K Ultra HD 3840x2160                     12
IPS Panel 4K Ultra HD / Touchscreen 3840x2160       11
4K Ultra HD / Touchscreen 3840x2160                  9
4K Ultra HD 3840x2160                                7
IPS Panel 1366x768                                   7
IPS Panel Retina Display 2560x1600                   6
IPS Panel Quad HD+ / Touchscreen 3200x1800           6
Touchscreen 2560x1440                                6
IPS Panel Retina Display 2304x1440                   6
Touchscreen 2256x1504                                6
IPS Panel Touchscreen 2560x1440                      5
1440x900                                             4
IPS Panel 2560x1440                                  4
IPS Panel Retina Display 2880x1800                   4
1920x1080                                            3
IPS Panel Touchscreen 1920x1200                      3
Quad HD+ 3200x1800                                   3
Touchscreen 2400x1600                                3
2560x1440                                            3
IPS Panel Quad HD+ 2560x1440                         3
IPS Panel Touchscreen 1366x768                       3
```

```
IPS Panel Full HD 2160x1440                        2
IPS Panel Touchscreen / 4K Ultra HD 3840x2160      2
IPS Panel Quad HD+ 3200x1800                       2
IPS Panel Full HD 1920x1200                        1
IPS Panel Retina Display 2736x1824                 1
IPS Panel Full HD 2560x1440                        1
Touchscreen / Full HD 1920x1080                    1
IPS Panel Full HD 1366x768                         1
Touchscreen / Quad HD+ 3200x1800                   1
Touchscreen / 4K Ultra HD 3840x2160                1
IPS Panel Touchscreen 2400x1600                    1
Name: count, dtype: int64
```

Feature Engineering With Insights

```
# checking each row in the 'ScreenResolution' column and assigns a value of
1 if the string 'Touchscreen' is found in it, and 0 otherwise.
df['Touchscreen'] = df['ScreenResolution'].apply(lambda x:1 if
'Touchscreen' in x else 0)
```

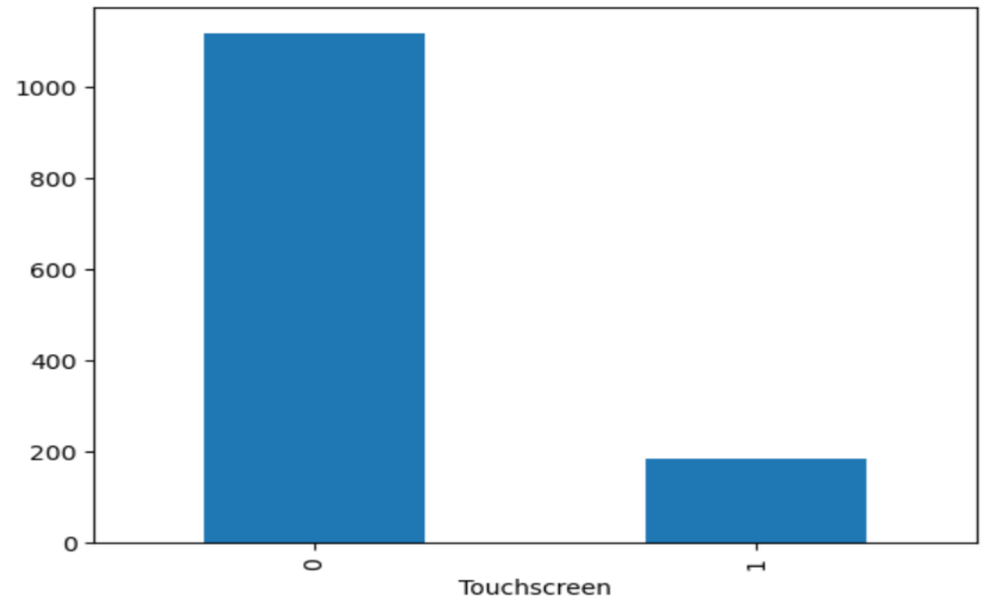
In [28]:

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

Out[28]:

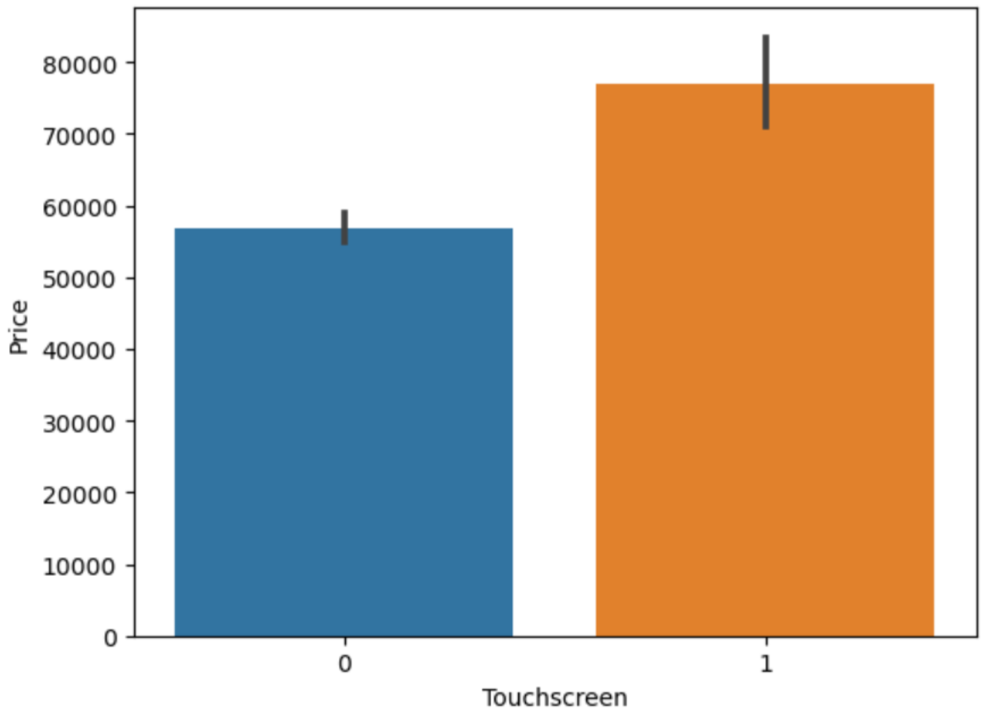
	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
89	Dell	Ultrabook	13.3	IPS Panel Full HD 1920x1080	Intel Core i7 8550U 1.8GHz	8	256GB SSD	Intel UHD Graphics 620	Windows 10	1.210	87858.72
1213	Dell	2 in 1 Convertible	15.6	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	2.191	53226.72
1285	Lenovo	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16	512GB SSD	Intel HD Graphics 520	Windows 10	1.300	79866.72
972	Dell	Gaming	17.3	Full HD 1920x1080	Intel Core i7 6700HQ 2.6GHz	32	256GB SSD + 1TB HDD	Nvidia GeForce GTX 1070	Windows 10	4.420	149184.00
836	Asus	Gaming	17.3	Full HD 1920x1080	Intel Core i7 7700HQ	16	256GB SSD + 1TB	Nvidia GeForce GTX	Windows 10	2.900	128884.32

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

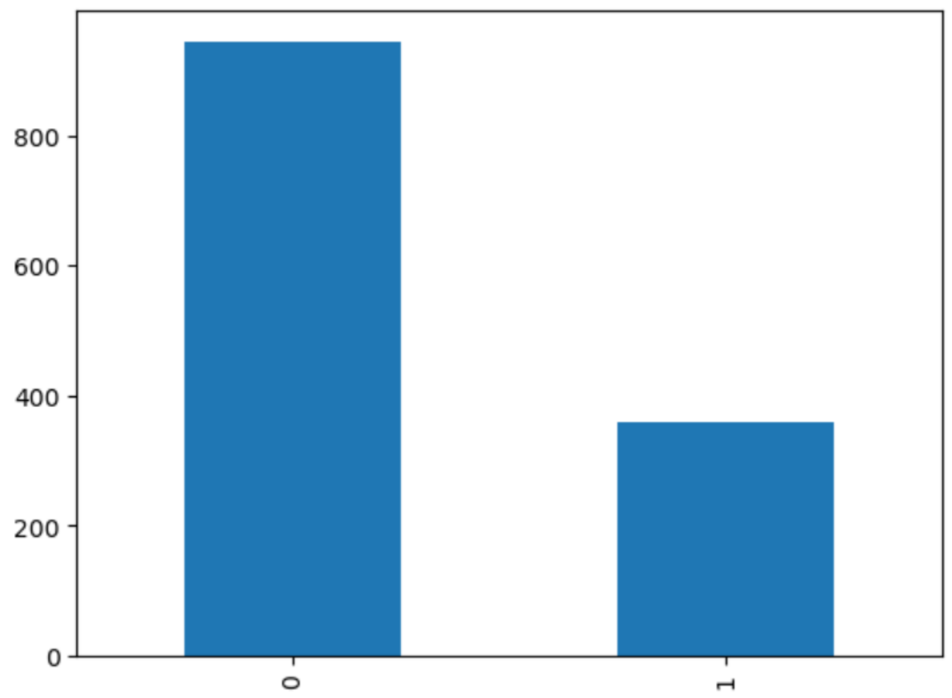


Number of touchscreen is less

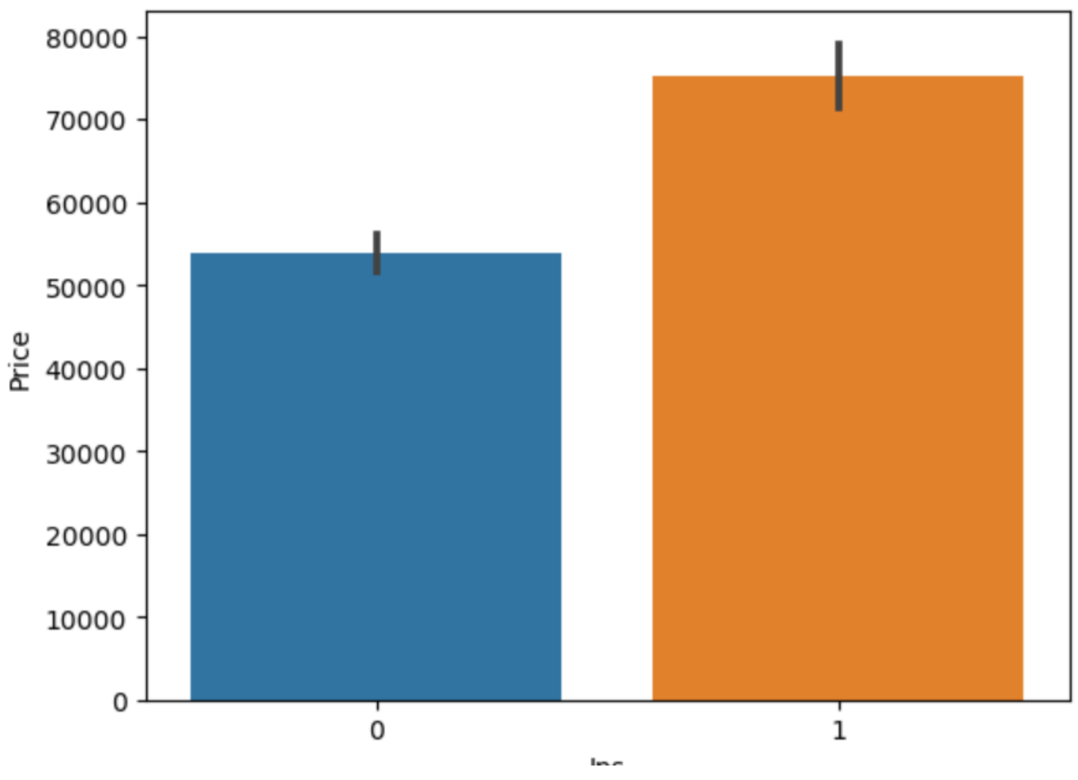
```
sns.barplot(x=df['Touchscreen'],y=df['Price'])
```




```
# similarly for screen resolution
df['Ips'] = df['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)
df['Ips'].value_counts().plot(kind='bar')
```




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




```
# splitting screenresoltion from 'X'
new = df['ScreenResolution'].str.split('x',n=1,expand=True)
```

In [36]:

```
df['X_res'] = new[0]
df['Y_res'] = new[1]
```

In [37]:

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

Out[37]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
557	Lenovo	Notebook	17.3	1600x900	Intel Core i7 7500U 2.7GHz	6	128GB SSD + 1TB HDD	Nvidia GeForce 940MX	Windows 10	2.80	50562.72
903	Lenovo	Ultrabook	14.0	IPS Panel Full HD 1920x1080	Intel Core i7 7500U 2.7GHz	8	256GB Flash Storage	Intel HD Graphics 620	Windows 10	1.13	109170.72
1250	Dell	Notebook	15.6	1366x768	Intel Pentium Quad Core N3710 1.6GHz	4	500GB HDD	Intel HD Graphics	Linux	2.20	17262.72
357	Dell	Gaming	15.6	Full HD 1920x1080	Intel Core i5 7300HQ 2.5GHz	8	1TB HDD	Nvidia GeForce GTX 1050	Windows 10	2.65	53226.72
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Intel Core i3	[illegible]	1TB	AMD Radeon	Windows	[illegible]	[illegible]

collecting the digits

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

In [39]:

```
df.head()
```

Out[39]:

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

Need to change the dtypes

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

In [41]:

```
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 [42]:

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

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

In [ ]:

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

Adding PPI for more information because PPI plays Positive role with correlation of price.

In [ ]:

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

In [ ]:

```
df.head()
```

In [ ]:

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

In [ ]:

```
df.head()
```

In [ ]:

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

In [ ]:

```
# collecting processors separately
df['Cpu Name'] = df['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))
```

In [ ]:

```
df.head()
```

In [ ]:

```
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 [ ]:

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

In [ ]:

```
df.head()
```

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

we can see most demanded processor in market is Intel Core i7.

In [ ]:

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

In [ ]:

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

In [ ]:

```
df.head()
```

In [ ]:

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

8 Gb ram laptops were highly sold in market

In [ ]:

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

Price of 32 GB ram laptops are more than 64 GB ram laptops

In [ ]:

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

In [ ]:

```
# so here  I changed the dtypes of columns, removed strings, and assigning
columns as per requirements

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 str(x) else 0)
```

```
df["Layer2SSD"] = df["second"].apply(lambda x: 1 if "SSD" in str(x) else 0)
df["Layer2Hybrid"] = df["second"].apply(lambda x: 1 if "Hybrid" in str(x)
else 0)
df["Layer2Flash_Storage"] = df["second"].apply(lambda x: 1 if "Flash
Storage" in str(x) else 0)

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

# Convert to integers with error handling
df["first"] = pd.to_numeric(df["first"],
errors='coerce').fillna(0).astype(int)
df["second"] = pd.to_numeric(df["second"],
errors='coerce').fillna(0).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"])

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

Here we can see Ram , SSD , PPI plays major role for price fluctuation.

In [ ]:

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

In [ ]:

```
df.head()
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
df.head()
```

In [ ]:

```
df['Gpu brand'].value_counts()
df = df[df['Gpu brand'] != 'ARM']
```

In [ ]:

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

In [ ]:

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

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

Nvidia's price is maximum because of their graphics performance

In [ ]:

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

In [ ]:

```
df.head()
```

In [ ]:

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

In [ ]:

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

In case of OpSys macOS price is highest followed by windows 7, mac OS X, etc

In [ ]:

```
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 [ ]:

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

In [ ]:

```
df.head()
```

In [ ]:

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

In [ ]:

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

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

Majority of weights are around 2-3 kg

In [ ]:

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

In [ ]:

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

In [ ]:

```
# Used log for thr right skewed price distribution

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

In [ ]:

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

In [ ]:

```
X
```

In [ ]:

```
y
```

Model Development

In [ ]:

```
# importing ML libraries

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 [ ]:

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

In [ ]:

```
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)
X_train
```

LINEAR REGRESSION

In [ ]:

```
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))
```

DECISION TREE

In [ ]:

```
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))
```

RANDOM FOREST

In [ ]:

```
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))
```

GRADIENT BOOST

In [ ]:

```
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))
```

EXPORTING THE MODEL

In [ ]:

```
import pickle

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

In [ ]:

```
df
```

Questions as conclusion

Which features have the most significant impact on laptop prices?

- RAM > SSD > PPI

Can the model accurately predict the prices of laptops from lesser-known brands?

- Yes for that we have to export the model it will predict with accuracy of 88.6%.

Does the brand of the laptop significantly influence its price?

- yes this is the sequence RAZER > LG > MSI > APPLE > MICROSOFT > GOOGLE > SAMSUNG ...

How well does the model perform on laptops with high-end specifications compared to budget laptops?

- It will give closer accurate prices around 88.6% as the given data is not so big.

What are the limitations and challenges in predicting laptop prices accurately?

- Small data to train. Require more data for good accuracy.

How does the model perform when predicting the prices of newly released laptops not present in the training dataset?

- Closer to accurate prices as the Gradient boost performance is 88.6%.