A PROJECT REPORT

On

Laptop Price Prediction

Submitted by

Subham Shaw (10871022004)

Nishant Yadav (10871022029)

Angelina Deepshikha Hans (10871022009)

Puja Verma (10871022011)

Gaurav Kumar (10871022016)

Under the Guidance of

Mr. Arnab Chakraborty
Professor

Computer Applications

Asansol Engineering College

Asansol

Affiliated to

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

Contents

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty, **Prof. Arnab Chakraborty** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Gaurav Kumar

Shubham Shaw

Angelina Deepshikha Hans

Puja Verma

Nishant Yadav

Project Objective

In this project we have a shortened 'Laptop Price' Dataset from Kaggle. In this dataset, the target attribute is the Price . So, in this project we need to do Regression based on the attributes present in our dataset and predict the price of laptops according to its specifications.

Our methodology for solving the problems in the given project is described below:

- Load the required dataset.
- Study the dataset.
- Describe the dataset.
- Visualise the dataset.
- Find out if the dataset needs to be pre-processed.
    - It will be determined on the basis of whether the dataset has null values or outliers or any such discrepancy that might affect the output of the models to be trained.
- If the dataset is required to be pre-processed, take the necessary steps to pre-process the data.
- Find out the principal attributes for training.
- Split the given dataset for training and testing purpose.
- Fit the previously split train data in the aforementioned 4 models.
- Calculate the accuracy of the 4 models and find out the classification reports.
- Plot the necessary graphs.
- Use each trained model to predict the outcomes of the given test dataset.
- Choose the best model among the 4 trained models bases on the R2 Score and Mean Absolute Error.

Project Scope

The Scope of the project is to develop a machine learning model that can predict the price of a laptop based on its specifications. This model will be helpful for various purposes such as:

- **Consumers:** get a better understanding of the fair market value of a laptop they are considering purchasing.
- **Retailers:** optimize their pricing strategies for laptops in their inventory.
- **Manufacturers:** gain insights into factors that influence laptop prices and make informed decisions about product development and pricing.

Data Description

Source of the data: Kaggle. The given dataset is a shortened version of the original dataset in Kaggle.
Data Description: The given train dataset has 1303 rows and 12 columns

Attributes	Description	Data Type
Company	The manufacturer of the laptop (e.g., Apple, HP, Asus)	Categorical
TypeName	The type of laptop (e.g., Ultrabook, Notebook)	Categorical
Inches	The size of the laptop screen in inches (e.g., 13.3, 15.6)	Numeric
Screen Resolution	The resolution of the laptop screen (e.g., 1920x1080, 2560x1600)	Categorical
CPU	The central processing unit (CPU) of the laptop (e.g., Intel Core i5, AMD A9-Series)	Categorical
RAM	The amount of random access memory (RAM) in the laptop (e.g., 4GB, 8GB, 16GB)	Numeric
Memory	The type and storage capacity of the laptop's memory (e.g., 128GB SSD, 500GB HDD)	Categorical
GPU	The graphics processing unit (GPU) of the laptop (e.g., Intel Iris Plus Graphics, Nvidia GeForce MX150)	Categorical
OpSys	The operating system installed on the laptop (e.g., macOS, Windows 10)	Categorical
Weight	The weight of the laptop in kilograms (kg)	Numeric
Price	The price of the laptop (presumably in INR)	Numeric

Table 1: Data Description

Now we will pre-process the data. The methodology followed is given below:

- Checking for null values.
  - If null values are present, we will fill them or drop the row containing the null value based on the dataset.
- Checking for outliers.
  - If outliers are present, they will either be removed or replaced by following a suitable method depending on the dataset.

Null Values:

```
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
```

Data Pre-processing

As the given dataset had Categorical and Non-categorical data mixed, we converted the categorical data into non-categorical data accordingly. We converted the String into Features. We converted the other categorical attributes into suitable numerical values. The following table shows the conversion record:

Non-Numeric to Numeric Change table		
Column	Initial Value	Replaced Value
Ram	8 GB	8
Weight	2 Kg	2
Touchscreen	NA	0
	NA	1
PPI	NA	Continuous Variable
Processor	NA	Categorical Values

Table 3: Feature Engineering

Now we have successfully handled Null values and converted non-numeric values to Numeric values. We didn't drop the rows with null values as we have a small dataset (only 1300 entries).

So, we are moving on to find if there are any outliers in our data and find the correlations of different attributes to our target i.e. 'Price' column in the dataset.

The following table gives the correlation value of each attribute with our target attribute i.e. ‘Price’:

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

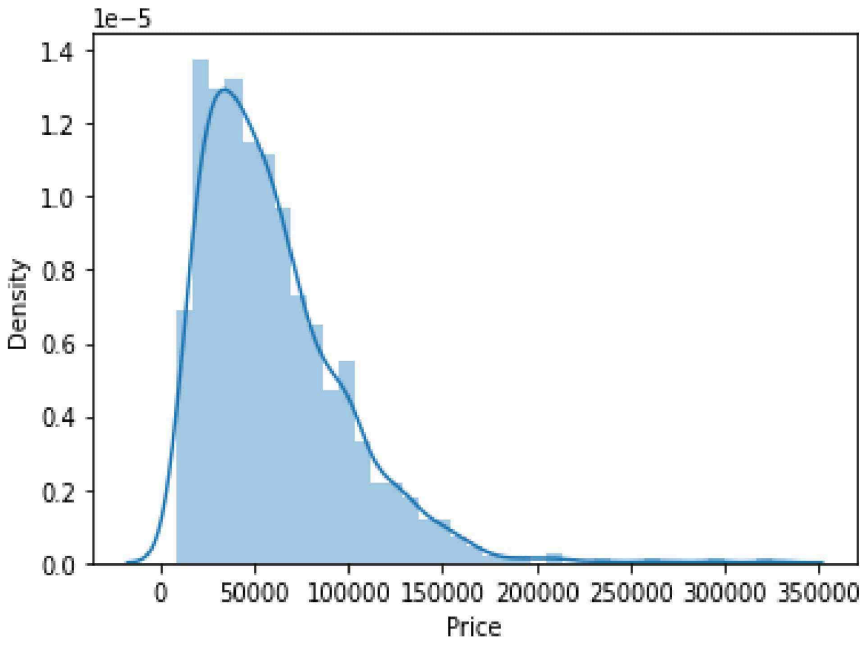
Table 5: Correlation values with Target Attribute

EDA

Exploratory analysis is a process to explore and understand the data and data relationship in a complete depth so that it makes feature engineering and machine learning modeling steps smooth and streamlined for prediction. EDA involves Univariate, Bivariate, or Multivariate analysis. EDA helps to prove our assumptions true or false. In other words, it helps to perform hypothesis testing. We will start from the first column and explore each column and understand what impact it creates on the target column. At the required step, we will also perform preprocessing and feature engineering tasks. our aim in performing in-depth EDA is to prepare and clean data for better machine learning modeling to achieve high performance and generalized models. so let's get started with analyzing and preparing the dataset for prediction.

1) Distribution of target column

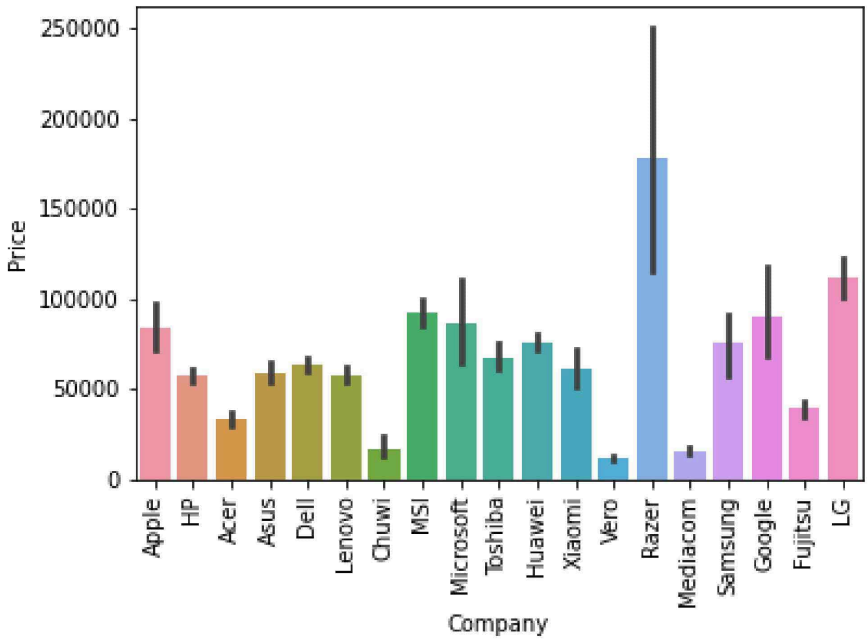
Working with regression problem statement target column distribution is important to understand.



2) Company column

we want to understand how does brand name impacts the laptop price or what is the average price of each laptop brand? If you plot a count plot(frequency plot) of a company then the major categories present are Lenovo, Dell, HP, Asus, etc.

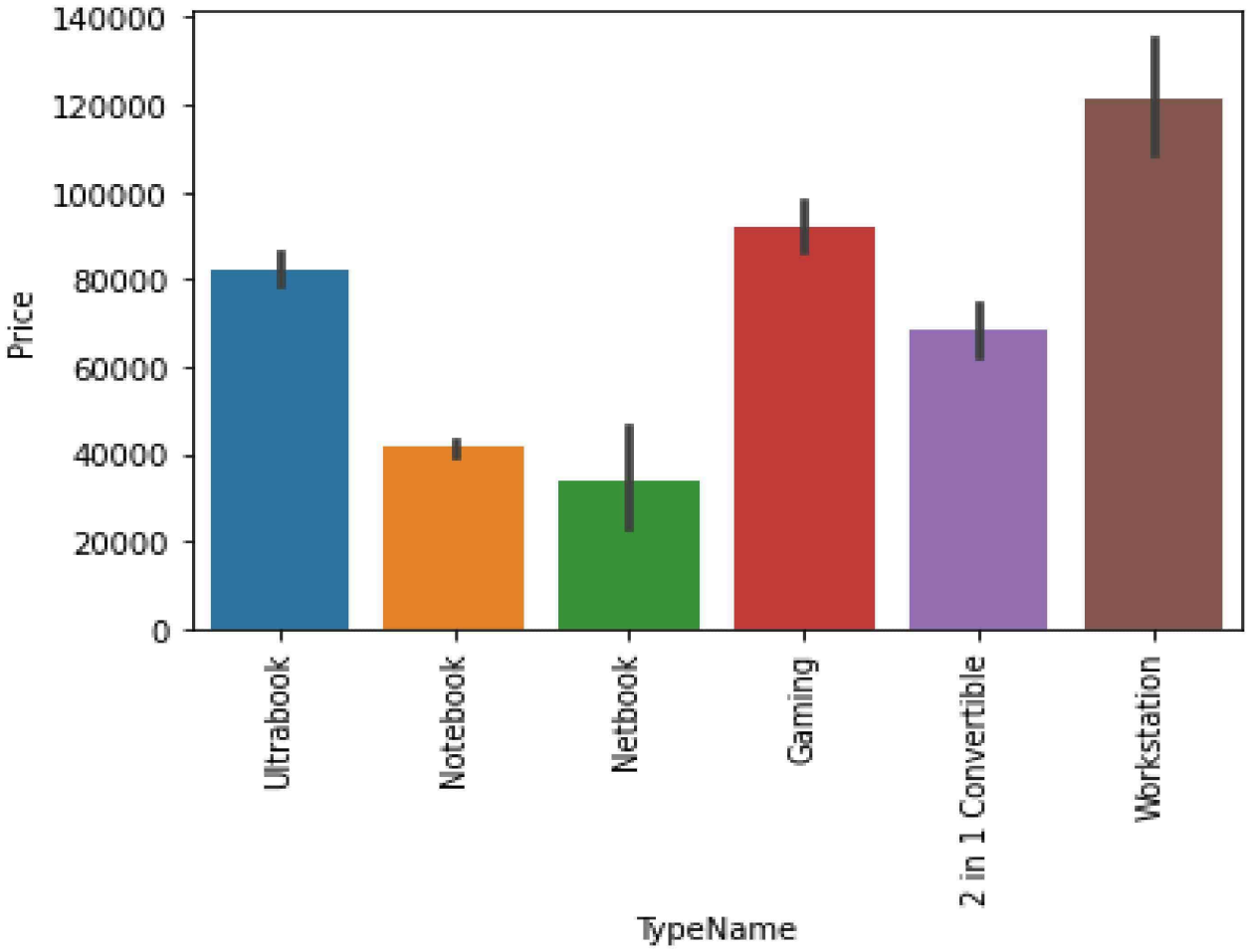
Now if we plot the company relationship with price then you can observe that how price varies with different brands.



Razer, Apple, LG, Microsoft, Google, MSI laptops are expensive, and others are in the budget range

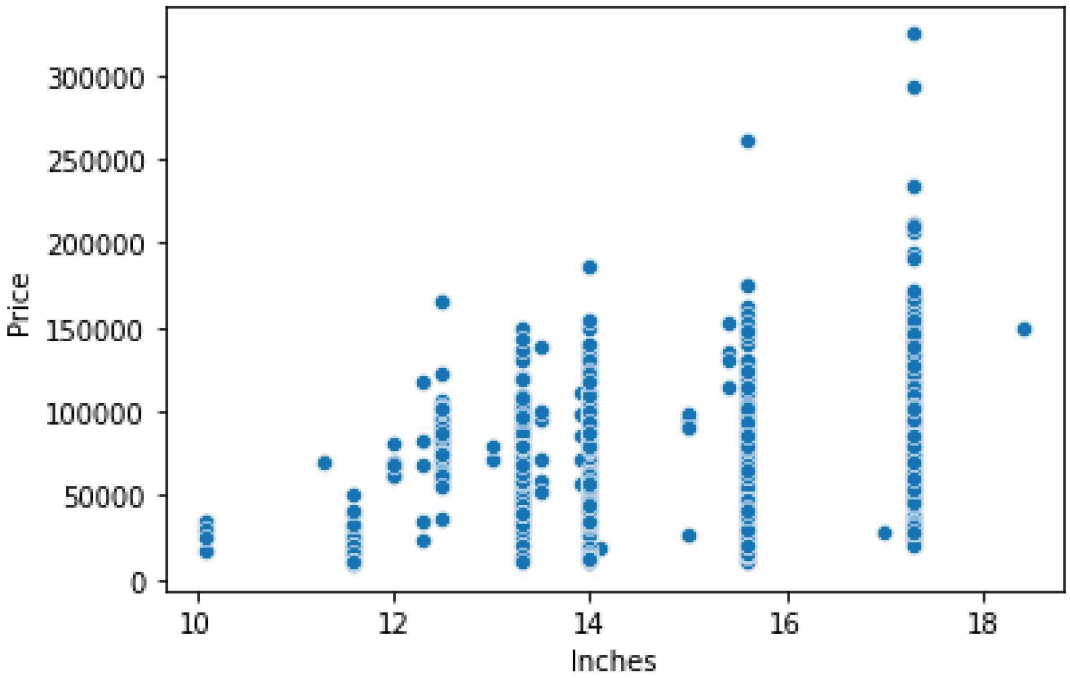
3) Type of laptop

Which type of laptop you are looking for like a gaming laptop, workstation, or notebook. As major people prefer notebook because it is under budget range and the same can be concluded from our data.



4) Does the price vary with laptop size in inches?

A Scatter plot is used when both the columns are numerical and it answers our question in a better way. From the below plot we can conclude that there is a relationship but not a strong relationship between the price and size column.



Feature Engineering

Feature engineering is a process to convert raw data to meaningful information. there are many methods that come under feature engineering like transformation, categorical encoding, etc. Now the columns we have are noisy so we need to perform some feature engineering steps.

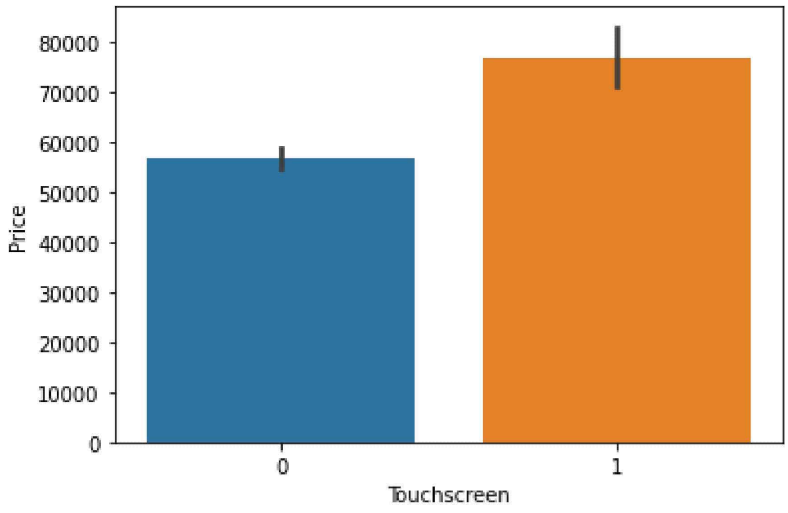
Screen Resolution

screen resolution contains lots of information. before any analysis first, we need to perform feature engineering over it. If you observe unique values of the column then we can see that all value gives information related to the presence of an IPS panel, are a laptop touch screen or not, and the X-axis and Y-axis screen resolution. So, we will extract the column into 3 new columns in the dataset.

Extract Touch screen information

It is a binary variable so we can encode it as 0 and 1. one means the laptop is a touch screen and zero indicates not a touch screen.

If we plot the touch screen column against price then laptops with touch screens are expensive which is true in real life.



Extract IPS Channel presence information

It is a binary variable and the code is the same we used above. The laptops with IPS channel are present less in our data but by observing relationship against the price of IPS channel laptops are high.

Extract X-axis and Y-axis screen resolution dimensions

Now both the dimension are present at end of a string and separated with a cross sign. So first we will split the string with space and access the last string from the list. then split the string with a cross sign and access the zero and first index for X and Y-axis dimensions.

Replacing inches, X and Y resolution to PPI

If you find the correlation of columns with price using the **corr** method then we can see that inches do not have a strong correlation but X and Y-axis resolution have a very strong resolution so we can take advantage of it and convert these three columns to a single column that is known as Pixel per inches(PPI). In the end, our goal is to improve the performance by having fewer features.

```
data.corr()['Price'].sort_values(ascending=False)
```

```
Price          1.000000
Ram            0.743007
X_res          0.556529
Y_res          0.552809
ppi            0.473487
Ips            0.252208
Weight         0.210370
Touchscreen    0.191226
Inches         0.068197
Name: Price, dtype: float64
```

So now we can drop the extra columns which are not of use. At this point, we have started keeping the important columns in our dataset.

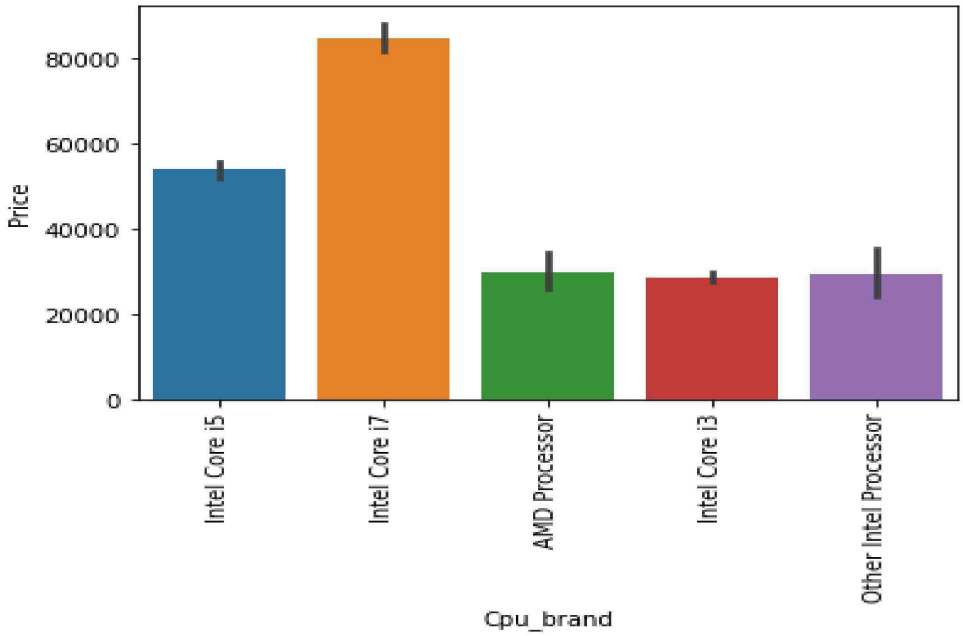
CPU column

If you observe the CPU column then it also contains lots of information. If you again use a unique function or value counts function on the CPU column then we have 118 different categories. The information it gives is about preprocessors in laptops and speed.

To extract the preprocessor we need to extract the first three words from the string. we are having an Intel preprocessor and AMD preprocessor so we are keeping 5 categories in our dataset as i3, i5, i7, other intel processors, and AMD processors.

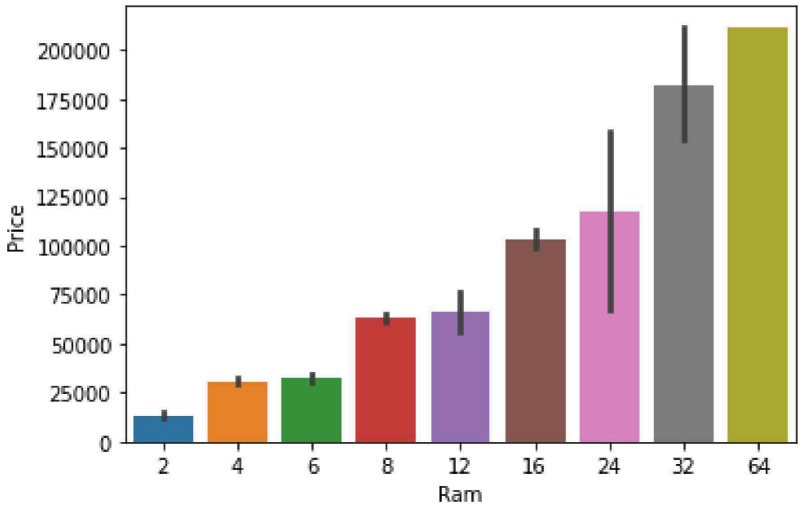
How does the price vary with processors?

we can again use our bar plot property to answer this question. And as obvious the price of i7 processor is high, then of i5 processor, i3 and AMD processor lies at the almost the same range. Hence price will depend on the preprocessor



Price with Ram

Again Bivariate analysis of price with Ram. If you observe the plot then Price is having a very strong positive correlation with Ram or you can say a linear relationship.



Memory column

memory column is again a noisy column that gives an understanding of hard drives. many laptops came with HHD and SSD both, as well in some there is an external slot present to insert after purchase. This column can disturb your analysis if not feature engineer it properly. So If you use value counts on a column then we are having 4 different categories of memory as HHD, SSD, Flash storage, and hybrid.

First, we have cleaned the memory column and then made 4 new columns which are a binary column where each column contains 1 and 0 indicate that amount four is present and which is not present. Any laptop has a single type of memory or a combination of two. so in the first column, it consists of the first memory size and if the second slot is present in the laptop then the second column contains it else we fill the null values with zero. After that in a particular column, we have multiplied the values by their binary value. It means that if in any laptop particular memory is present then it contains binary value as one and the first value will be multiplied by it, and same with the second combination. For the laptop which does have a second slot, the value will be zero multiplied by zero is zero.

Now when we see the correlation of price then Hybrid and flash storage have very less or no correlation with a price. We will drop this column with CPU and memory which is no longer required.

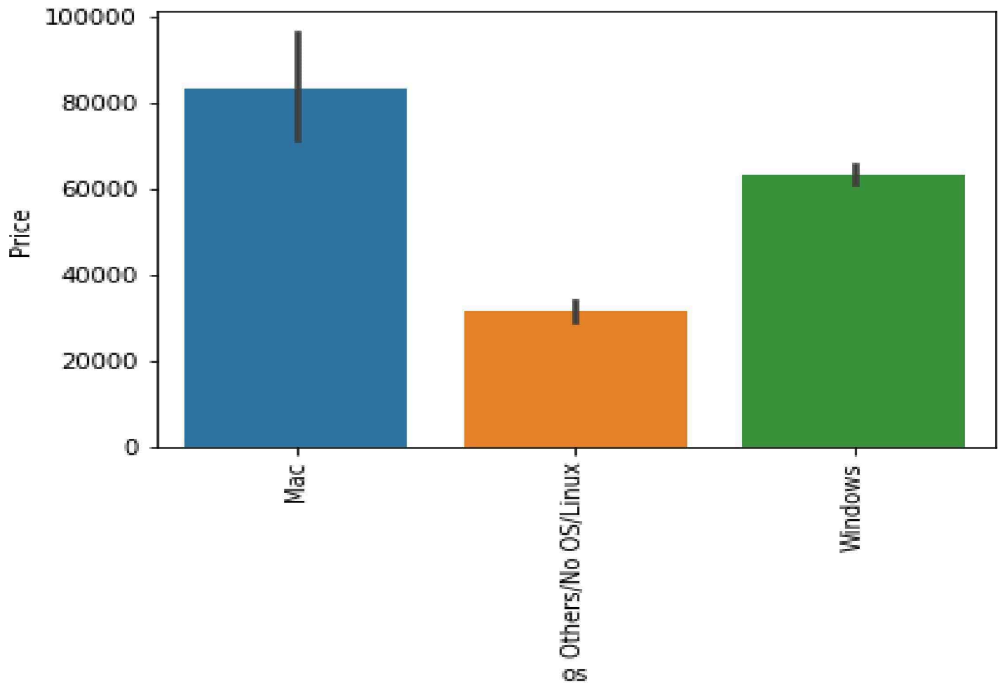
GPU Variable

GPU(Graphical Processing Unit) has many categories in data. We are having which brand graphic card is there on a laptop. we are not having how many capacities like (6Gb, 12 Gb) graphic card is present. so we will simply extract the name of the brand.

Operating System Column

There are many categories of operating systems. we will keep all windows categories in one, Mac in one, and remaining in others. This is a simple and most used feature engineering method, you can try something else if you find more correlation with price.

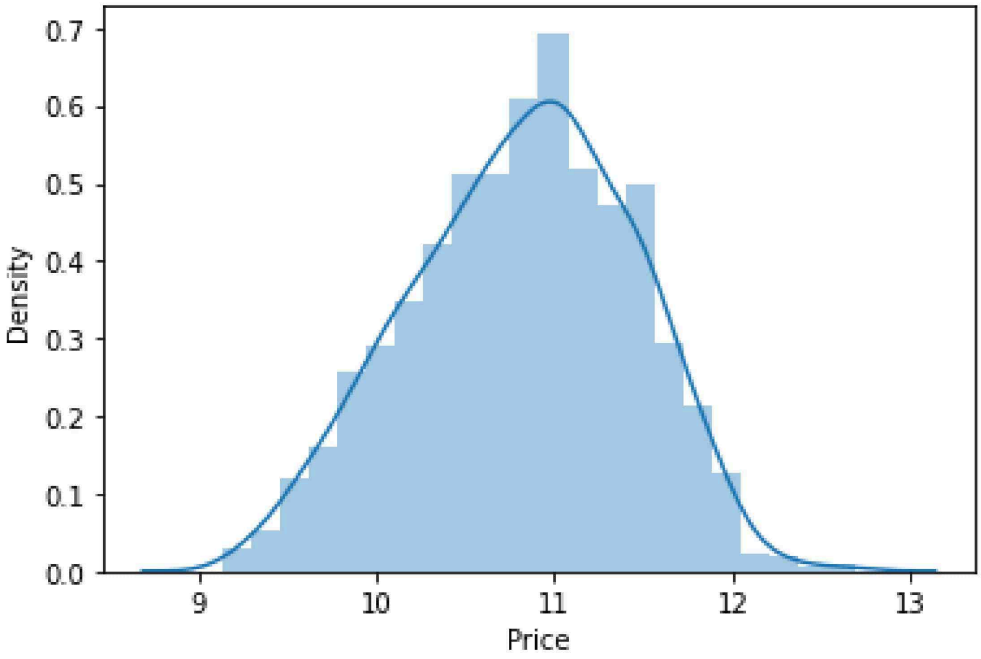
when you plot price aginst operating system then as usual Mac is most expensive.



Log-Normal Transformation

we saw the distribution of the target variable above which was right-skewed. By transforming it to normal distribution performance of the algorithm will increase. we take the log of values that transform to the normal

distribution which you can observe below. So while separating dependent and independent variables we will take a log of price, and in displaying the result perform exponent of it.



Model Building

Splitting data for training and testing purpose

We split the given train dataset into two parts for training and testing purpose. The split ratio we used is 0.75 which indicates we used 75% data for training purpose and 25% data for testing purpose. We will be using the same split ratio for all the models trained.

Random Forest Regressor Model

The object description of the Random Forest Classifier used is given below:

Object Name	RandomForestRegressor
Parameters	**Value**
max_depth	15
max_features	0.75
min_sample	0.5
n_estimators	100
random_state	100

Table 6: Object Parameter Table for Random Forest Regressor

Now we calculated R2 Score and MAE:

```
R2 score 0.8873402378382488
MAE 0.15860130110457718
```

Now we will be training our required models. Our project goal requires us to train specific 4 classifier models viz.

1. KNN Regressor
2. Random Forest Regression
3. Decision Tree and
4. Regression (As the target attribute is Continous, We will use Linear Regression)

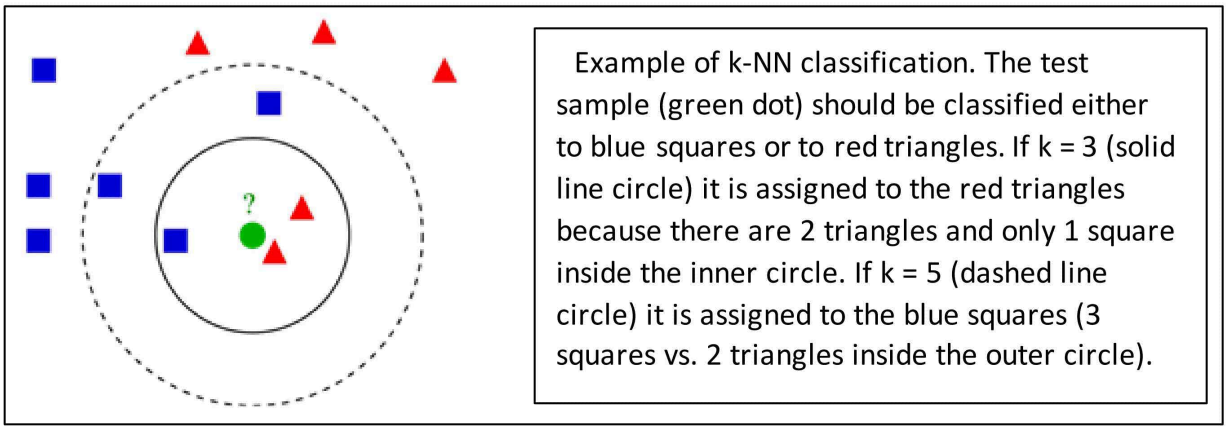
We will be using the final dataset obtained after pre-processing the given train dataset to train our required models.

KNN Regression

k-NN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression.

K-Nearest Neighbors (KNN) is a non-parametric machine learning algorithm that can be used for both classification and regression tasks. In the context of regression, KNN is often referred to as "K-Nearest Neighbors Regression" or "KNN Regression." It's a simple and intuitive algorithm that makes predictions by finding the K nearest data points to a given input and averaging their target values (for numerical regression) or selecting the majority class (for classification).

An example of k-NN classification:



Example of k-NN classification. The test sample (green dot) should be classified either to blue squares or to red triangles. If k = 3 (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle).

The object description of the k-NN Classifier used is given below:

Object Name	KNeighborsClassifier
Parameters	**Value**
algorithm	'auto'
leaf_size	30
metric	'minkowski'
metric_params	None
n_neighbors	3
weights	'uniform'

Table 11: Object Parameter Table for k-NN Regressor

Now we calculate R2 Score and MAE for our trained k-NN model which is given below:

```
R2 score 0.803148868705085
MAE 0.19264883332948868
```

Fig 7: R2 Score and MAE of k-NN Regressor

Decision Tree

A Decision Tree Regressor is a supervised learning technique used for regression tasks. It works by creating a tree-like model that predicts a continuous value (in this case, price) based on a set of features (laptop specifications). Here's a breakdown of how it functions:

1. Building the Tree:

- The algorithm starts with the entire dataset and identifies the feature that best splits the data into subsets with the most significant difference in the target variable (price). This "best" feature is typically the one that minimizes a cost function, like variance in price.
- Each split creates a new decision node in the tree. The algorithm continues splitting the data at each node based on the best feature and its corresponding threshold value. This process repeats recursively until a stopping criteria is met, like reaching a maximum depth for the tree or having a sufficiently homogeneous subset of data points (similar prices) at a particular node.

2. Prediction:

- When presented with a new unseen laptop with its specifications, the model traverses the decision tree starting from the root node.
- At each node, the model compares the new data point's feature value with the threshold value associated with that node.
- Based on the comparison, the model follows a specific branch of the tree leading to a leaf node.
- The leaf node contains the predicted price for the new laptop. The prediction is typically the average price of the data points that reached that particular leaf node during training.

Advantages of Decision Tree Regressor:

- Easy to interpret: The tree structure provides a clear visualization of how features influence the predicted price.
- Handles both categorical and numerical features: No need for separate data preprocessing for different feature

types.

- Robust to outliers: Less sensitive to outliers in the data compared to some other models.

Disadvantages of Decision Tree Regressor:

- Prone to overfitting: If the tree is allowed to grow too deep, it may become overly complex and capture noise in the training data, leading to poor performance on unseen data.
- High variance: Small changes in the training data can lead to significant variations in the tree structure and predictions.

The object description of the Decision Tree Regressor used is given below:

Object Name	**DecisionTreeRegressorr**
Parameters	**Value**
max_depth	8

Table 17: Object Parameter Table for Decision Tree Regressor

Now, We Calculated the R2 Score and MAE of the Decision Tree Model:

```
R2 score 0.8411948861111318
MAE 0.18163378208451647
```

Logistic Regression

Linear regression is a statistical method used to model the relationship between a dependent variable (also known as response variable or target variable) and one or more independent variables (also known as explanatory variables or predictor variables). It creates a linear equation that attempts to estimate the dependent variable based on the values of the independent variables.

Here's a breakdown of the key aspects of linear regression:

- **Equation:** The linear equation in linear regression typically takes the form of $y = mx + b$, where:
    - y represents the dependent variable (e.g., laptop price in your project).
    - x represents the independent variable (e.g., RAM size).
    - m is the slope of the line, which indicates the change in y for a unit change in x.
    - b is the y-intercept, which represents the predicted value of y when x is zero (not always meaningful in real-world applications).
- **Assumptions:** Linear regression makes certain assumptions about the data, including linearity (the relationship between variables can be modeled by a straight line), normality (errors are normally distributed), homoscedasticity (variance of errors is constant across all values of x), and independence (errors are independent of each other).

The object description of the Logistic Regression Classifier used is given below:

Parameter	Description	Values
fit_intercept	(boolean, default=True) Whether to calculate the intercept (b) for the linear equation.	TRUE
copy_X	(boolean, default=True) Controls whether to copy the input data (X) before fitting the model. Copying prevents accidental modification of the original data.	TRUE
n_jobs	(int, default=None) The number of CPU cores to use for parallel computations during fitting.	None
positive	(boolean, default=False) When set to True, forces the coefficients to be positive. This is only applicable for dense arrays and might not be used in your code.	FALSE

Table 20: Object Parameter Table for Linear Regression

Comparison of the Models trained

We trained 4 models using the 4 algorithms viz.

1. k-Nearest Neighbour
2. Gaussian Naive Bayes
3. Decision Tree and
4. Logistic Regression

The 4 models had different accuracy. The comparison of the accuracies of the models are given below:

Model	R2 Score
k-Nearest Neighbour	0.8031
Random Forest	0.8873
Decision Tree	0.8411
Linear Regression	0.8073

Table 24: Accuracy Comparison table

The following bar graph shows the accuracy comparison in graphical way:

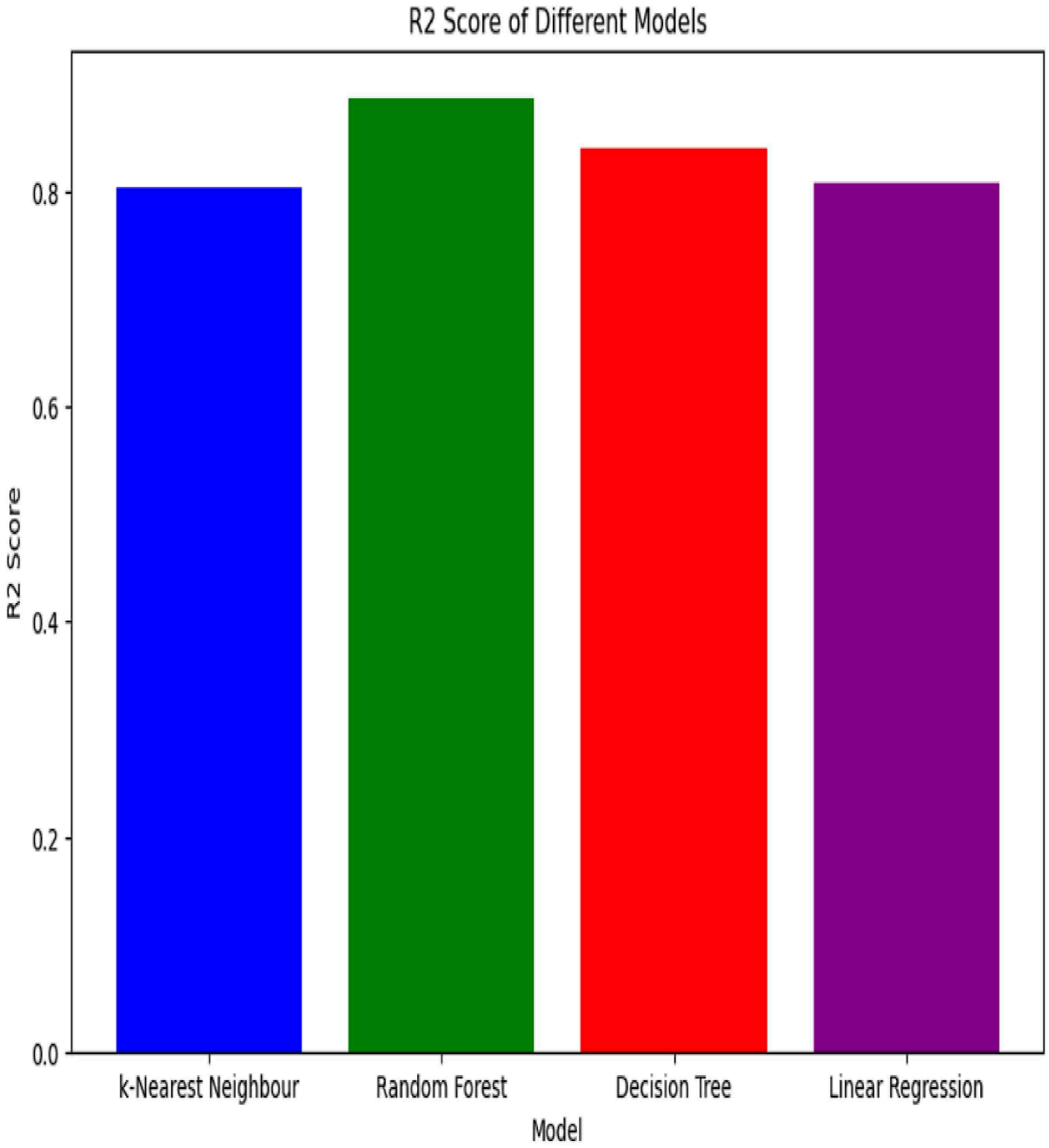


Fig 12: Comparison of accuracy of the 4 different models trained

Thus, from the above comparison we can see Random Forest Regressor has the highest R2 Score. So, our selected model is Random Forest

Test Dataset

Testing the given Test Dataset

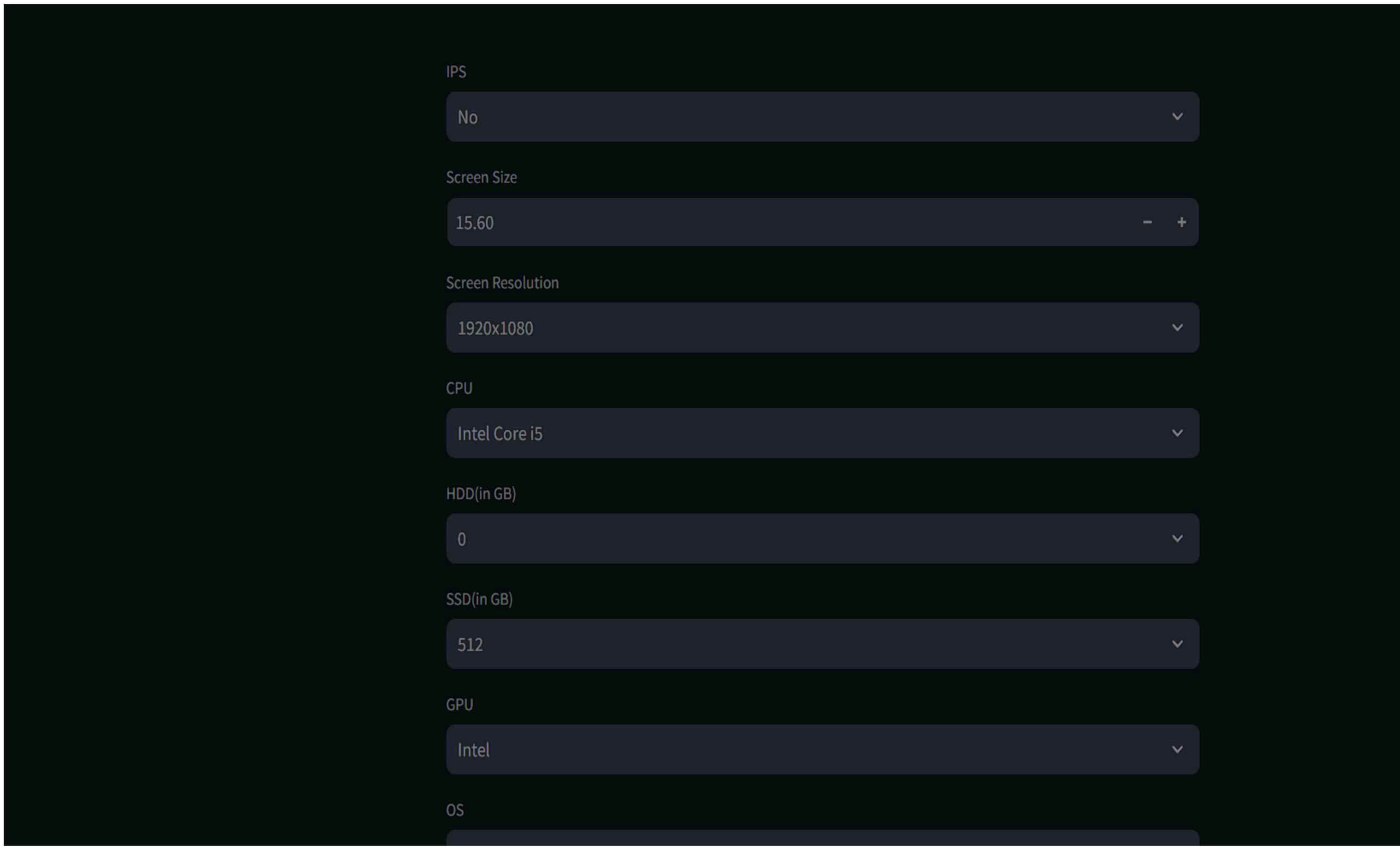
We were given a test dataset for this laptop prediction problem. We pre-process the given test dataset in a similar way we pre-processed our train dataset. The methodology followed is given below:

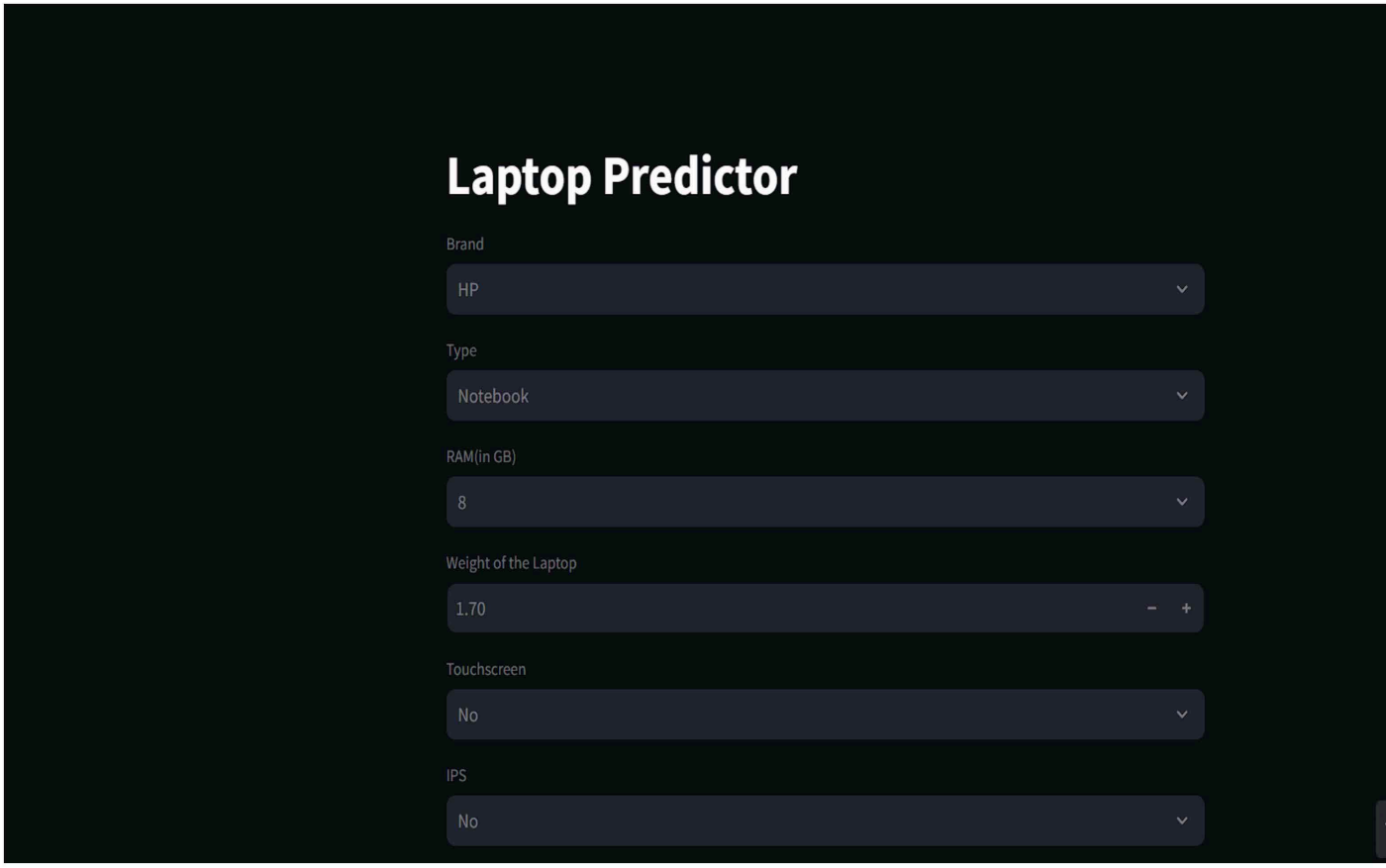
- Change the categorical values to numeric values using the same process we used during changing values in our train dataset.

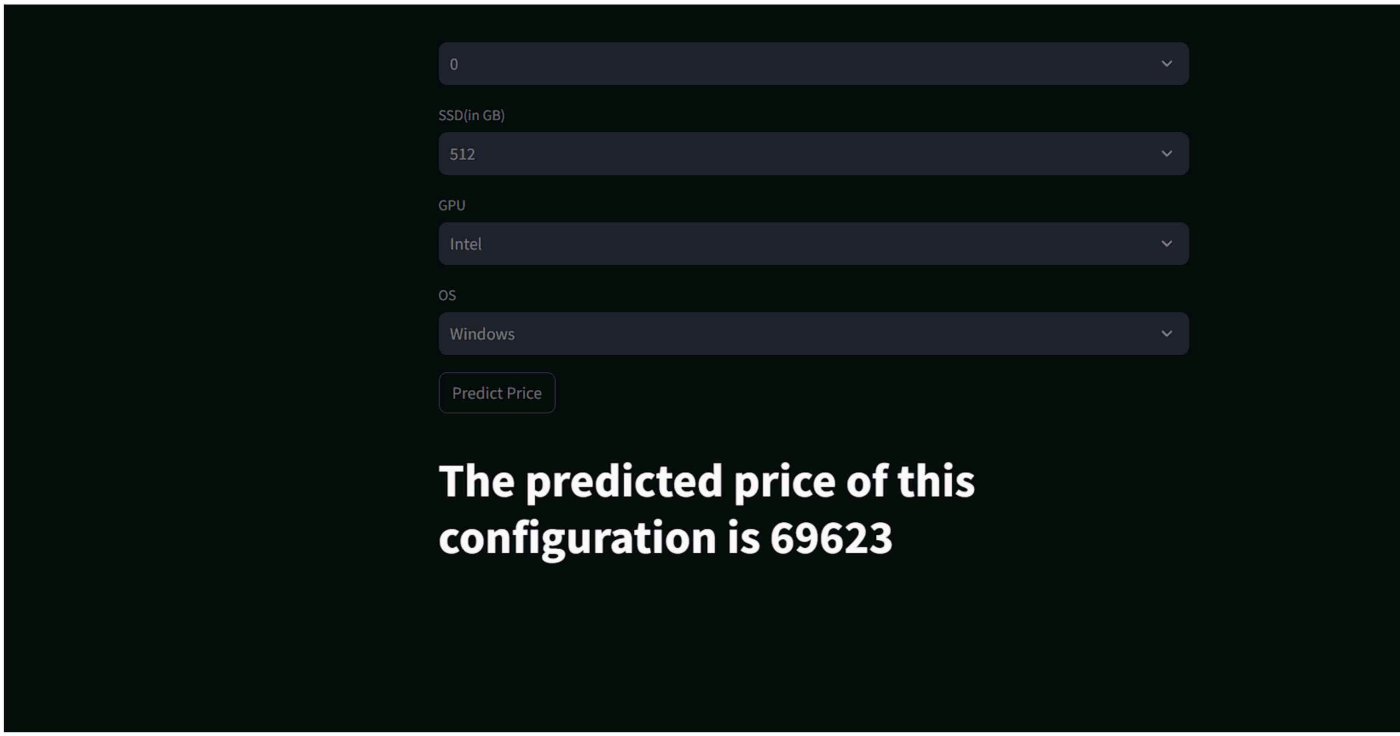
- Checking for null values.
    - If null values are present, we will fill them or drop the row containing the null value based on the dataset.
- Checking for outliers.
    - If outliers are present, they will either be removed or replaced by following a suitable method depending on the dataset.

After pre-processing of the test dataset had been done, we retrained the models with 100% of the train dataset given and used those trained models to predict the outcome for each input in the test dataset.

SCREENSHOT


IPS
No
Screen Size
15.60
Screen Resolution
1920x1080
CPU
Intel Core i5
HDD(in GB)
0
SSD(in GB)
512
GPU
Intel
OS


Laptop Predictor
Brand
HP
Type
Notebook
RAM(in GB)
8
Weight of the Laptop
1.70
Touchscreen
No
IPS
No


0
SSD(in GB)
512
GPU
Intel
OS
Windows
Predict Price
The predicted price of this configuration is 69623

Code

For GUI:

```
import streamlit as st
import pickle
import numpy as np

# import the model
pipe = pickle.load(open('pipe.pkl','rb'))
df = pickle.load(open('df.pkl','rb'))

st.title("Laptop Predictor")

# brand
company = st.selectbox('Brand',df['Company'].unique())

# type of laptop
type = st.selectbox('Type',df['TypeName'].unique())

# Ram
ram = st.selectbox('RAM(in GB)',[2,4,6,8,12,16,24,32,64])

# weight
weight = st.number_input('Weight of the Laptop')
```

```
# Touchscreen

touchscreen = st.selectbox('Touchscreen',['No','Yes'])


# IPS

ips = st.selectbox('IPS',['No','Yes'])


# screen size

screen_size = st.number_input('Screen Size')


# resolution

resolution = st.selectbox('Screen Resolution',['1920x1080','1366x768','1600x900','3840x2160','3200x1800','2880x1800','2560x1600','2560x1440','2304x1440'])


#cpu

cpu = st.selectbox('CPU',df['Cpu brand'].unique())


hdd = st.selectbox('HDD(in GB)',[0,128,256,512,1024,2048])


ssd = st.selectbox('SSD(in GB)',[0,8,128,256,512,1024])


gpu = st.selectbox('GPU',df['Gpu brand'].unique())


os = st.selectbox('OS',df['os'].unique())
```

```
if st.button('Predict Price'):
    # query
    ppi = None
    if touchscreen == 'Yes':
        touchscreen = 1
    else:
        touchscreen = 0

    if ips == 'Yes':
        ips = 1
    else:
        ips = 0

    X_res = int(resolution.split('x')[0])
    Y_res = int(resolution.split('x')[1])
    ppi = ((X_res*2) + (Y_res2))*0.5/screen_size
    query = np.array([company,type,ram,weight,touchscreen,ips,ppi
,cpu,hdd,ssd,gpu,os])

    query = query.reshape(1,12)
    st.title("The predicted price of this configuration is " +
str(int(np.exp(pipe.predict(query)[0]))))
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('laptop_data.csv')

df.head()
```

```
   Unnamed: 0 Company   TypeName  Inches
ScreenResolution  \
0           0   Apple  Ultrabook    13.3  IPS Panel Retina Display
2560x1600
1           1   Apple  Ultrabook    13.3
1440x900
2           2      HP   Notebook    15.6                   Full HD
1920x1080
3           3   Apple  Ultrabook    15.4  IPS Panel Retina Display
2880x1800
4           4   Apple  Ultrabook    13.3  IPS Panel Retina Display
2560x1600

                           Cpu   Ram               Memory  \
0        Intel Core i5 2.3GHz   8GB            128GB SSD
1        Intel Core i5 1.8GHz   8GB  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz   8GB            256GB SSD
3        Intel Core i7 2.7GHz  16GB            512GB SSD
4        Intel Core i5 3.1GHz   8GB            256GB SSD

                            Gpu  OpSys  Weight        Price
0  Intel Iris Plus Graphics 640  macOS  1.37kg   71378.6832
1        Intel HD Graphics 6000  macOS  1.34kg   47895.5232
2         Intel HD Graphics 620  No OS  1.86kg   30636.0000
3            AMD Radeon Pro 455  macOS  1.83kg  135195.3360
4  Intel Iris Plus Graphics 650  macOS  1.37kg   96095.8080
```

```
df.shape
```

```
(1303, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        1303 non-null   int64
 1   Company           1303 non-null   object
 2   TypeName          1303 non-null   object
 3   Inches            1303 non-null   float64
 4   ScreenResolution  1303 non-null   object
```

```
 5   Cpu               1303 non-null   object
 6   Ram               1303 non-null   object
 7   Memory            1303 non-null   object
 8   Gpu               1303 non-null   object
 9   OpSys             1303 non-null   object
 10  Weight            1303 non-null   object
 11  Price             1303 non-null   float64
dtypes: float64(2), int64(1), object(9)
memory usage: 76.4+ KB
```

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

```
0
```

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

```
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
```

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

```
df.head()
```

```
  Company   TypeName  Inches                    ScreenResolution  \
0   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Ultrabook    13.3                        1440x900
2      HP   Notebook    15.6                  Full HD 1920x1080
3   Apple  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu   Ram               Memory  \
0        Intel Core i5 2.3GHz   8GB            128GB SSD
1        Intel Core i5 1.8GHz   8GB  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz   8GB            256GB SSD
3        Intel Core i7 2.7GHz  16GB            512GB SSD
4        Intel Core i5 3.1GHz   8GB            256GB SSD

                           Gpu  OpSys  Weight        Price
0  Intel Iris Plus Graphics 640  macOS  1.37kg   71378.6832
1        Intel HD Graphics 6000  macOS  1.34kg   47895.5232
```

```
2          Intel HD Graphics 620  No OS  1.86kg   30636.0000
3             AMD Radeon Pro 455  macOS  1.83kg  135195.3360
4  Intel Iris Plus Graphics 650  macOS  1.37kg   96095.8080
```

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

df.head()
```

```
  Company   TypeName  Inches                    ScreenResolution  \
0   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Ultrabook    13.3                            1440x900
2      HP   Notebook    15.6                   Full HD 1920x1080
3   Apple  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu Ram               Memory  \
0        Intel Core i5 2.3GHz   8            128GB SSD
1        Intel Core i5 1.8GHz   8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz   8            256GB SSD
3        Intel Core i7 2.7GHz  16            512GB SSD
4        Intel Core i5 3.1GHz   8            256GB SSD

                           Gpu  OpSys Weight        Price
0  Intel Iris Plus Graphics 640  macOS   1.37   71378.6832
1        Intel HD Graphics 6000  macOS   1.34   47895.5232
2         Intel HD Graphics 620  No OS   1.86   30636.0000
3            AMD Radeon Pro 455  macOS   1.83  135195.3360
4  Intel Iris Plus Graphics 650  macOS   1.37   96095.8080
```

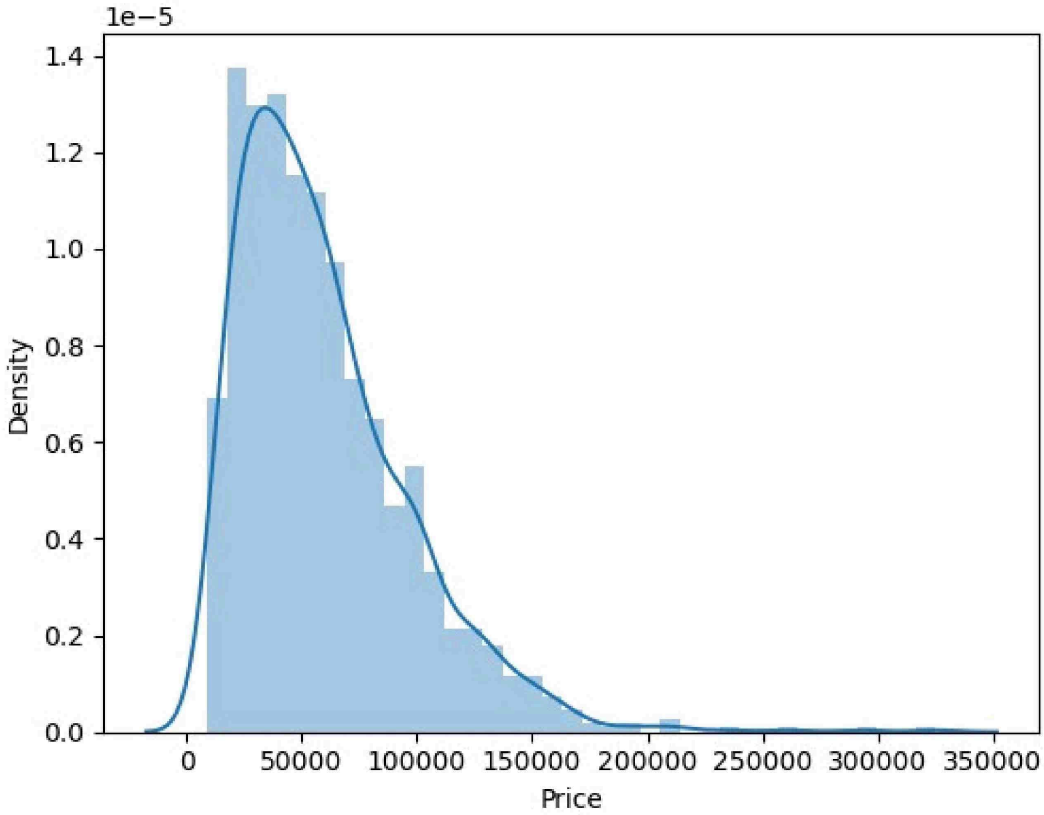
```
df['Ram'] = df['Ram'].astype('int32')
df['Weight'] = df['Weight'].astype('float32')

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: 66.2+ KB
```

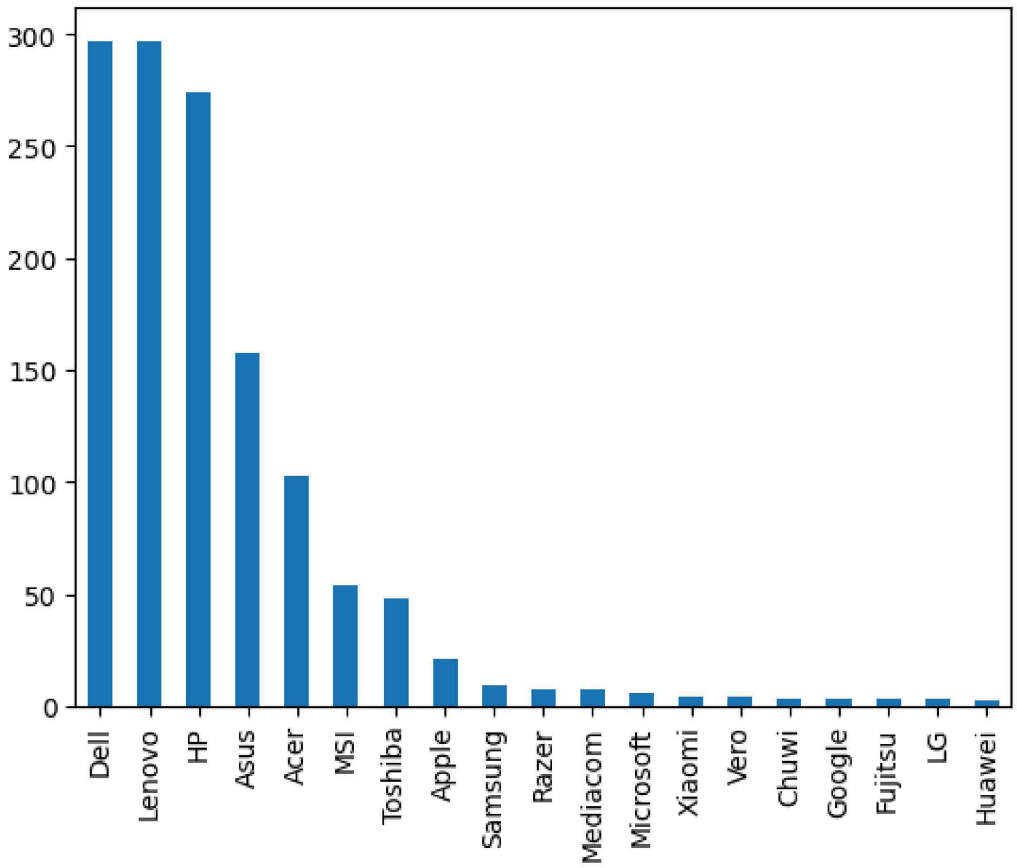
```
%pip install -q seaborn
import seaborn as sns

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

```
<ipython-input-18-87e11caeb2c4>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Price'])

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

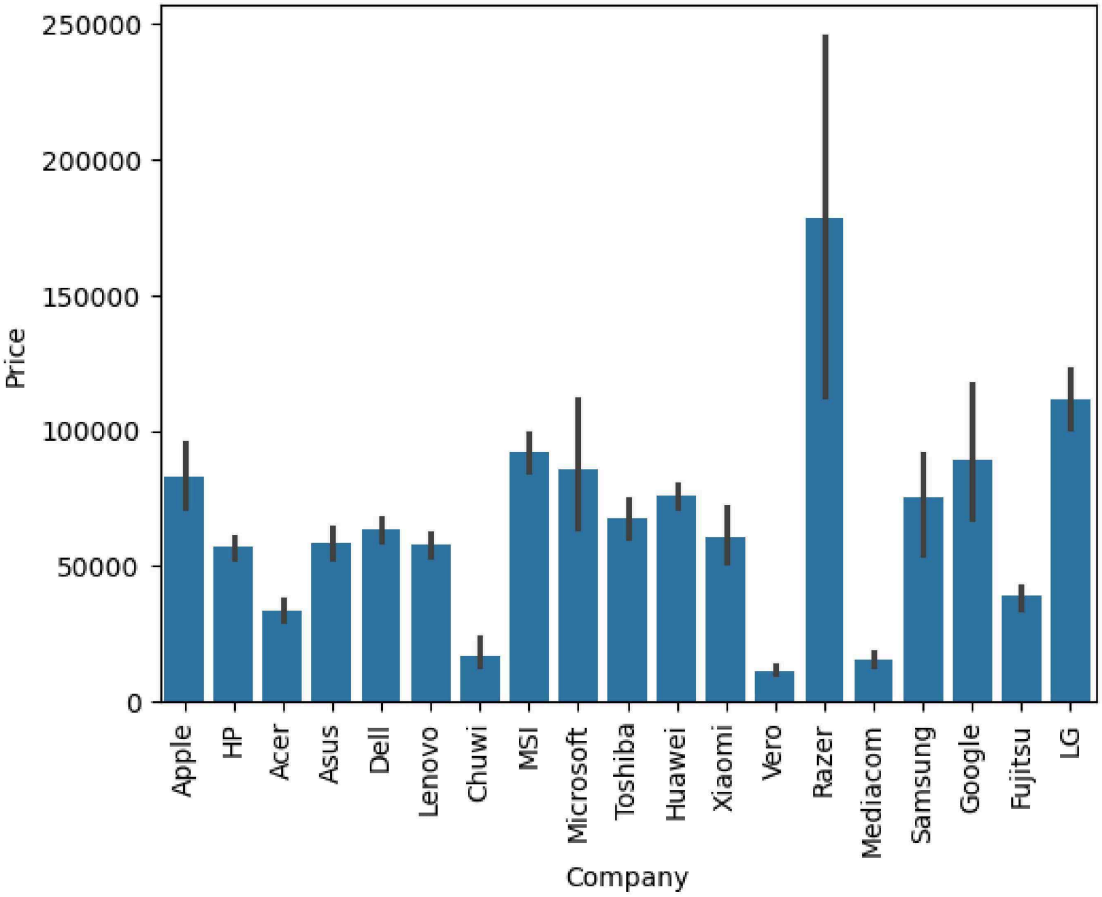


```
df['Company'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```

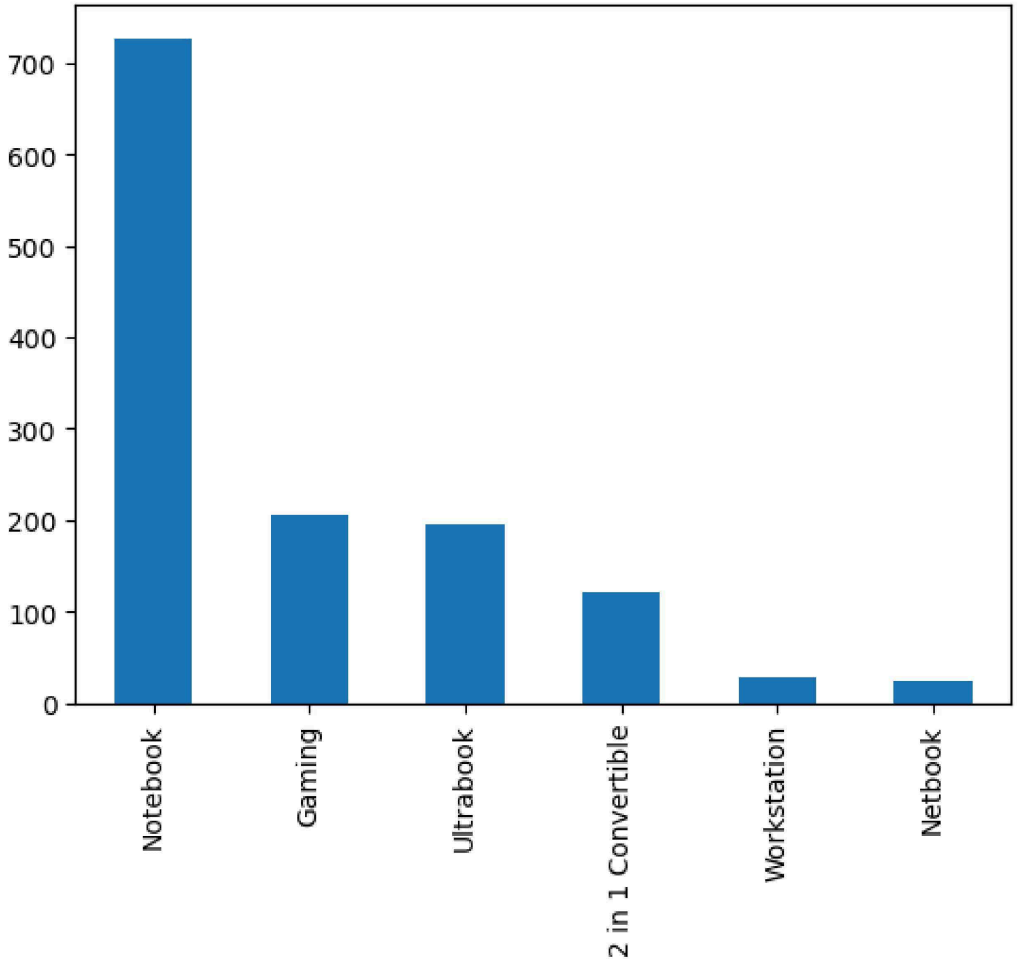



```
sns.barplot(x=df['Company'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

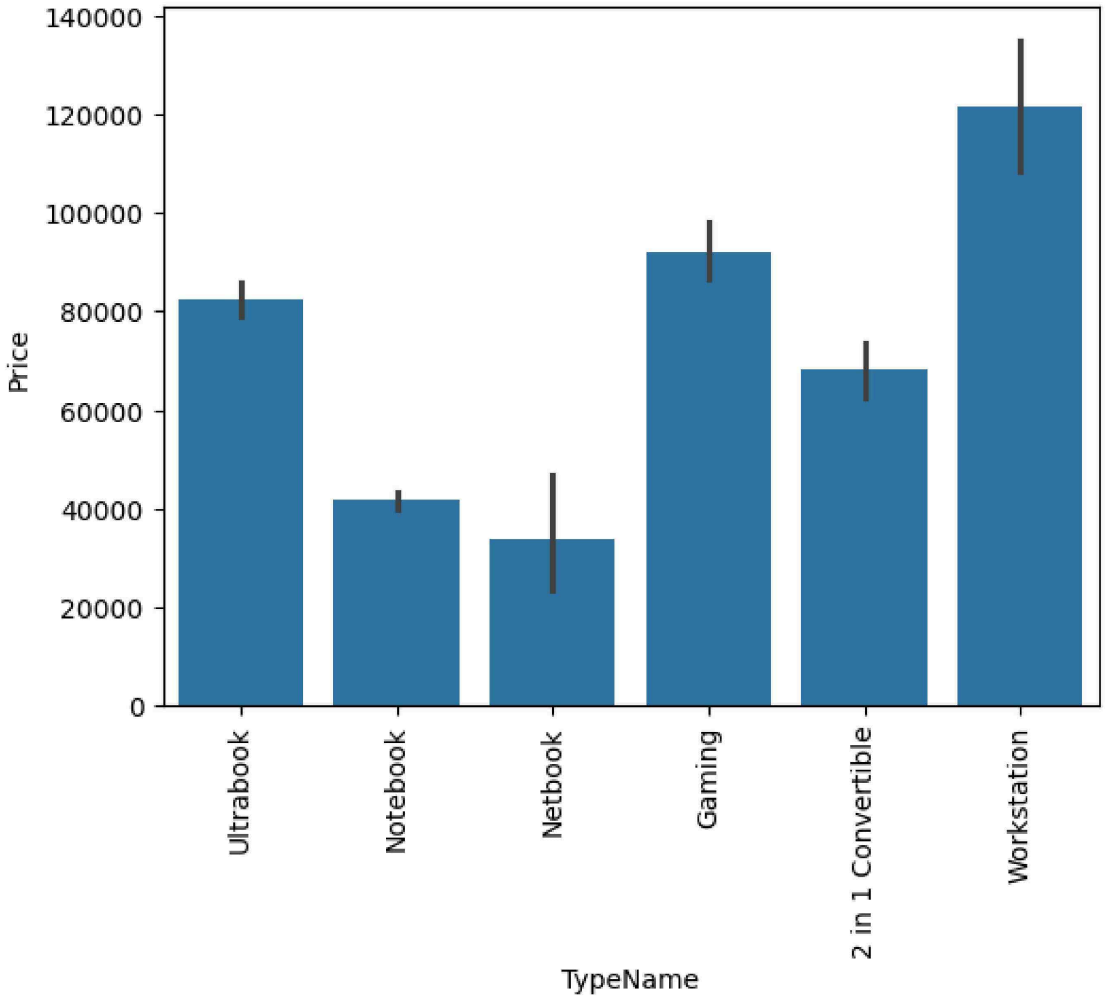



```
df['TypeName'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```



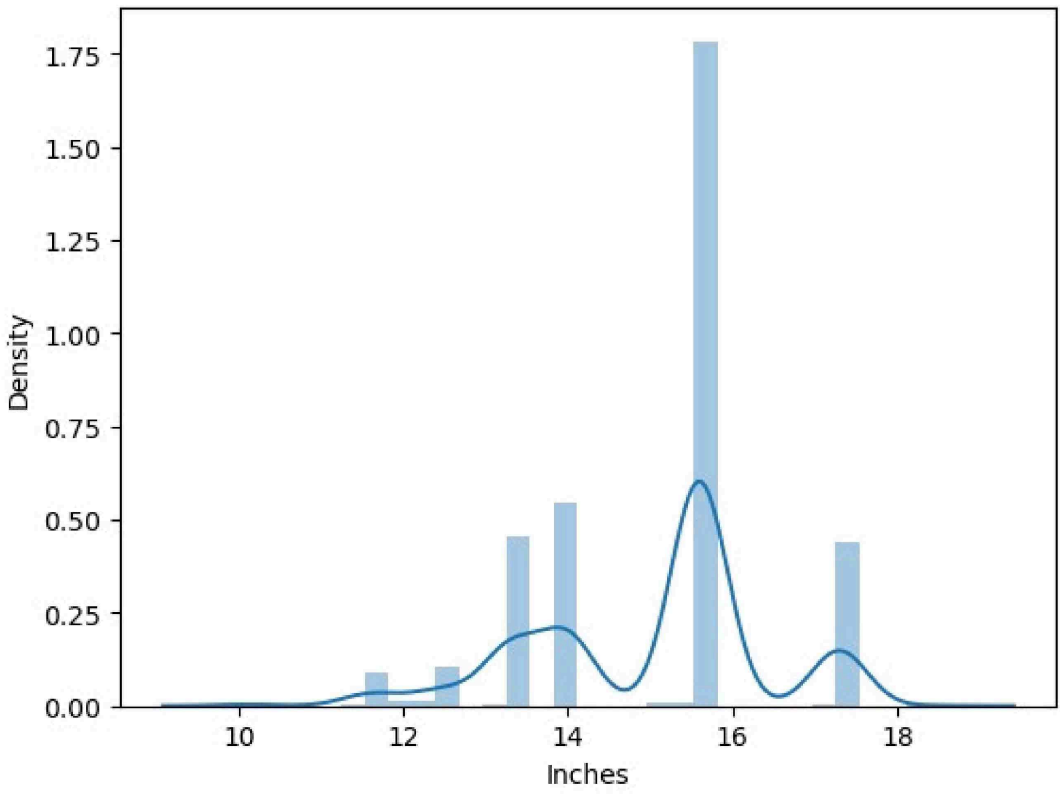

```
sns.barplot(x=df['TypeName'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```




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

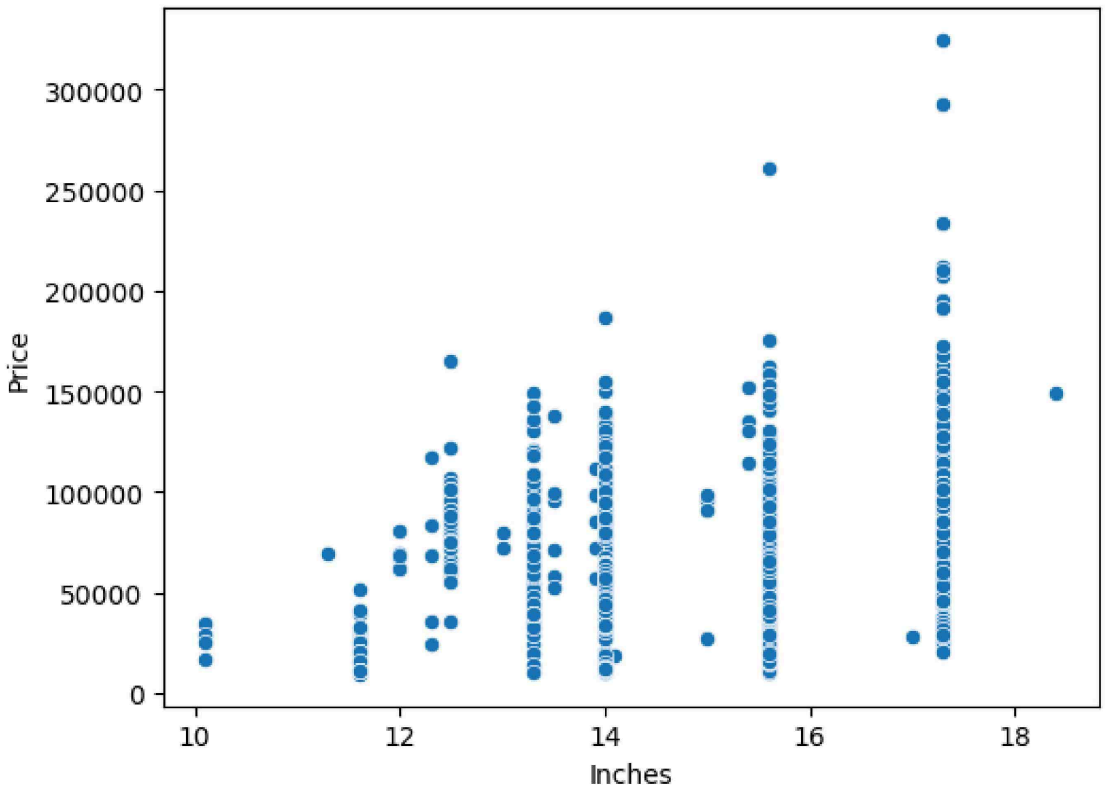
```
<ipython-input-23-51888cb550e6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

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

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




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

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




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

```
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
4K Ultra HD 3840x2160                                 7
Touchscreen 2560x1440                                 7
IPS Panel 1366x768                                    7
IPS Panel Quad HD+ / Touchscreen 3200x1800            6
IPS Panel Retina Display 2560x1600                    6
IPS Panel Retina Display 2304x1440                    6
Touchscreen 2256x1504                                 6
IPS Panel Touchscreen 2560x1440                       5
IPS Panel Retina Display 2880x1800                    4
IPS Panel Touchscreen 1920x1200                       4
1440x900                                              4
```

```
IPS Panel 2560x1440                            4
IPS Panel Quad HD+ 2560x1440                   3
Quad HD+ 3200x1800                             3
1920x1080                                      3
Touchscreen 2400x1600                          3
2560x1440                                      3
IPS Panel Touchscreen 1366x768                 3
IPS Panel Touchscreen / 4K Ultra HD 3840x2160  2
IPS Panel Full HD 2160x1440                    2
IPS Panel Quad HD+ 3200x1800                   2
IPS Panel Retina Display 2736x1824             1
IPS Panel Full HD 1920x1200                    1
IPS Panel Full HD 2560x1440                    1
IPS Panel Full HD 1366x768                     1
Touchscreen / Full HD 1920x1080                1
Touchscreen / Quad HD+ 3200x1800               1
Touchscreen / 4K Ultra HD 3840x2160            1
IPS Panel Touchscreen 2400x1600                1
Name: ScreenResolution, dtype: int64
```

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

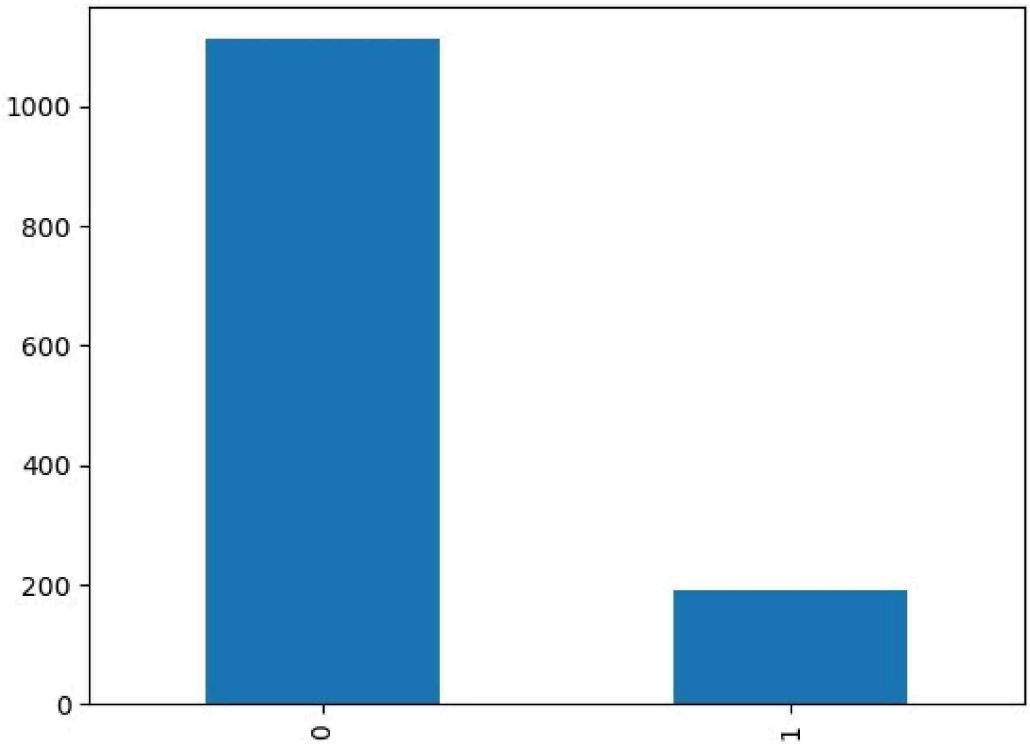
```
df.sample(5)
```

```
     Company  TypeName  Inches              ScreenResolution  \
553       HP  Notebook    17.3                     1600x900
239     Acer    Gaming    15.6  IPS Panel Full HD 1920x1080
47      Asus    Gaming    17.3            Full HD 1920x1080
1034      HP  Notebook    15.6            Full HD 1920x1080
917      MSI    Gaming    17.3            Full HD 1920x1080

                              Cpu  Ram                Memory  \
553      Intel Core i3 6006U 2GHz    8              1TB HDD
239   Intel Core i5 7300HQ 2.5GHz    8             256GB SSD
47            AMD Ryzen 1700 3GHz    8  256GB SSD +  1TB HDD
1034   Intel Core i7 6500U 2.5GHz    8             256GB SSD
917   Intel Core i7 7700HQ 2.8GHz   16  256GB SSD +  1TB HDD

                             Gpu       OpSys  Weight       Price
Touchscreen
553        Intel HD Graphics 520  Windows 10    2.65  28992.3120
0
239      Nvidia GeForce GTX 1050  Windows 10    2.50  45074.8800
0
47             AMD Radeon RX 580  Windows 10    3.20  69210.7200
0
1034       Intel HD Graphics 520  Windows 10    1.84  65480.5872
0
```

```
917   Nvidia GeForce GTX 1050 Ti  Windows 10    2.70  85194.7200
0
```

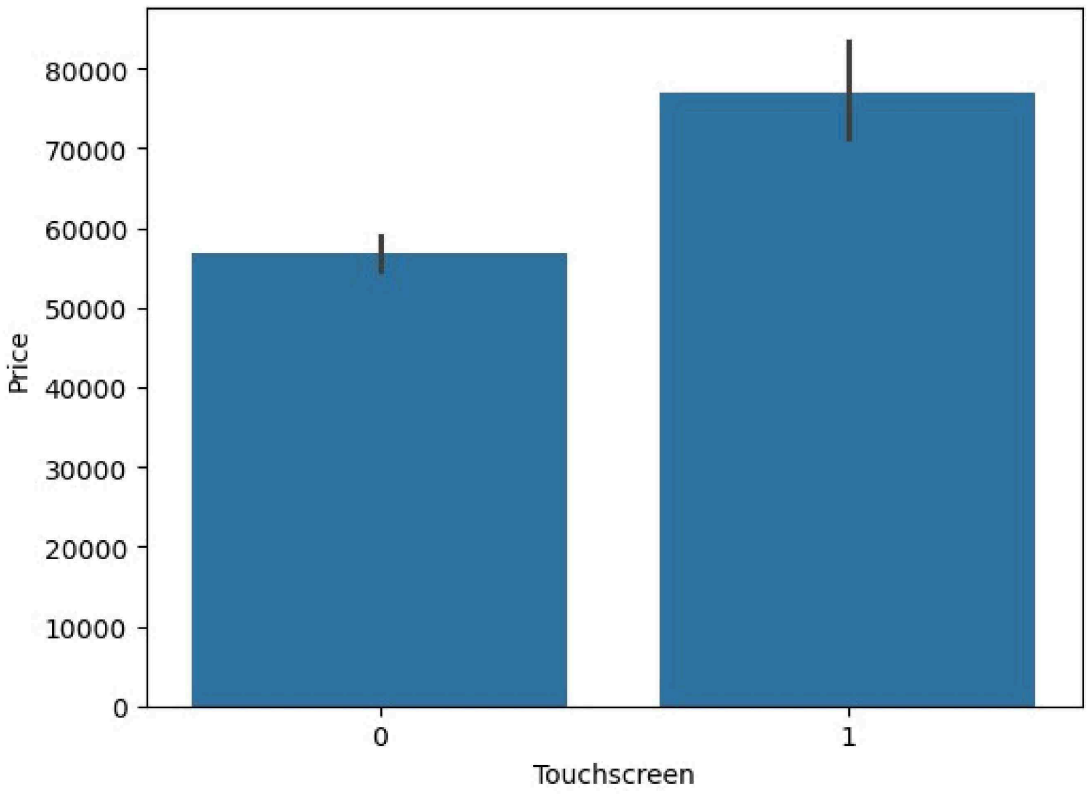
```
df['Touchscreen'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```




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

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



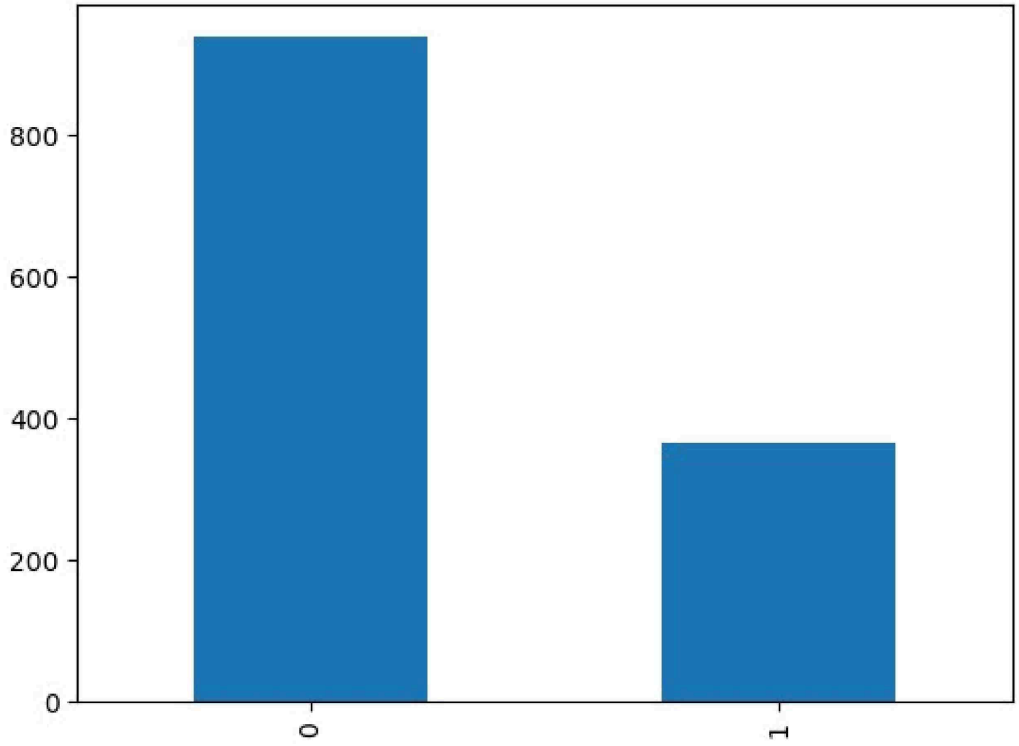

```
df['Ips'] = df['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else
0)

df.head()
```

```
  Company   TypeName  Inches                    ScreenResolution  \
0   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Ultrabook    13.3                            1440x900
2      HP   Notebook    15.6                   Full HD 1920x1080
3   Apple  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu  Ram               Memory  \
0        Intel Core i5 2.3GHz    8            128GB SSD
1        Intel Core i5 1.8GHz    8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz    8            256GB SSD
3        Intel Core i7 2.7GHz   16            512GB SSD
4        Intel Core i5 3.1GHz    8            256GB SSD

                           Gpu  OpSys  Weight        Price
Touchscreen  Ips
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
```

```
2          Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3             AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1
```

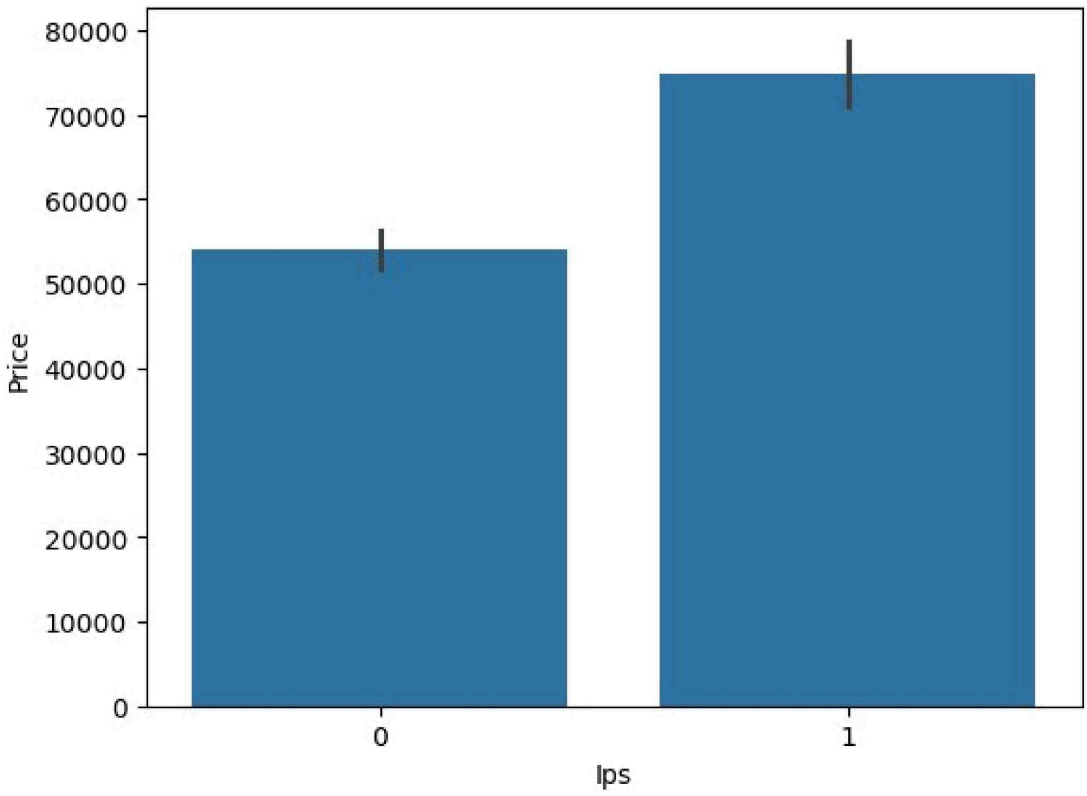
```
df['Ips'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```




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

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




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

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

df.sample(5)
```

```
     Company   TypeName  Inches                ScreenResolution  \
536     Dell   Notebook    15.6              Full HD 1920x1080
1007      HP  Ultrabook    14.0              Full HD 1920x1080
1244      HP   Notebook    14.0              Full HD 1920x1080
469   Lenovo  Ultrabook    14.0  IPS Panel Quad HD+ 2560x1440
363       HP   Notebook    15.6              Full HD 1920x1080

                                Cpu  Ram     Memory
Gpu  \
536    Intel Core i5 8250U 1.6GHz    8  256GB SSD         AMD Radeon
530
1007   Intel Core i7 6600U 2.6GHz    8  256GB SSD  Intel HD Graphics
520
1244   Intel Core i5 6200U 2.3GHz    4  256GB SSD  Intel HD Graphics
520
469    Intel Core i7 6600U 2.6GHz   12  256GB SSD  Intel HD Graphics
520
363    Intel Core i5 7200U 2.5GHz    8    1TB HDD  Intel HD Graphics
```

```
620
```

```
           OpSys  Weight        Price  Touchscreen  Ips  \
536   Windows 10    2.20  42486.0048            0    0
1007   Windows 7    1.43  77202.7200            0    0
1244   Windows 7    1.54  58607.4672            0    0
469   Windows 10    1.40  98994.2400            0    1
363   Windows 10    1.86  34045.9200            0    0

                        X_res Y_res
536               Full HD 1920  1080
1007              Full HD 1920  1080
1244              Full HD 1920  1080
469   IPS Panel Quad HD+ 2560  1440
363               Full HD 1920  1080
```

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

```
df.head()
```

```
  Company   TypeName  Inches                    ScreenResolution  \
0   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Ultrabook    13.3                            1440x900
2      HP   Notebook    15.6                   Full HD 1920x1080
3   Apple  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu  Ram               Memory  \
0        Intel Core i5 2.3GHz    8            128GB SSD
1        Intel Core i5 1.8GHz    8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz    8            256GB SSD
3        Intel Core i7 2.7GHz   16            512GB SSD
4        Intel Core i5 3.1GHz    8            256GB SSD

                           Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

  X_res Y_res
0  2560  1600
```

```
1  1440   900
2  1920  1080
3  2880  1800
4  2560  1600
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Company           1303 non-null   object 
 1   TypeName          1303 non-null   object 
 2   Inches            1303 non-null   float64
 3   ScreenResolution  1303 non-null   object 
 4   Cpu               1303 non-null   object 
 5   Ram               1303 non-null   int32  
 6   Memory            1303 non-null   object 
 7   Gpu               1303 non-null   object 
 8   OpSys             1303 non-null   object 
 9   Weight            1303 non-null   float32
 10  Price             1303 non-null   float64
 11  Touchscreen       1303 non-null   int64  
 12  Ips               1303 non-null   int64  
 13  X_res             1303 non-null   int32  
 14  Y_res             1303 non-null   int32  
dtypes: float32(1), float64(2), int32(3), int64(2), object(7)
memory usage: 96.8+ KB
```

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

```
<ipython-input-41-9447c1bc3d29>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()['Price']
```

```
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
```

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

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

```
<ipython-input-43-9447c1bc3d29>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  df.corr()['Price']

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
```

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

df.head()
```

```
  Company   TypeName  Inches                         Cpu  Ram  \
0   Apple  Ultrabook    13.3        Intel Core i5 2.3GHz    8
1   Apple  Ultrabook    13.3        Intel Core i5 1.8GHz    8
2      HP   Notebook    15.6  Intel Core i5 7200U 2.5GHz    8
3   Apple  Ultrabook    15.4        Intel Core i7 2.7GHz   16
4   Apple  Ultrabook    13.3        Intel Core i5 3.1GHz    8

                Memory                           Gpu  OpSys  Weight  \
0            128GB SSD  Intel Iris Plus Graphics 640  macOS    1.37
1  128GB Flash Storage        Intel HD Graphics 6000  macOS    1.34
2            256GB SSD         Intel HD Graphics 620  No OS    1.86
3            512GB SSD            AMD Radeon Pro 455  macOS    1.83
4            256GB SSD  Intel Iris Plus Graphics 650  macOS    1.37

         Price  Touchscreen  Ips  X_res  Y_res         ppi
0   71378.6832            0    1   2560   1600  226.983005
1   47895.5232            0    0   1440    900  127.677940
2   30636.0000            0    0   1920   1080  141.211998
3  135195.3360            0    1   2880   1800  220.534624
4   96095.8080            0    1   2560   1600  226.983005
```

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

df.head()
```

```
  Company   TypeName                          Cpu  Ram
Memory  \
0   Apple  Ultrabook        Intel Core i5 2.3GHz    8            128GB
SSD
1   Apple  Ultrabook        Intel Core i5 1.8GHz    8  128GB Flash
Storage
2      HP   Notebook  Intel Core i5 7200U 2.5GHz    8            256GB
SSD
3   Apple  Ultrabook        Intel Core i7 2.7GHz   16            512GB
SSD
4   Apple  Ultrabook        Intel Core i5 3.1GHz    8            256GB
SSD

                            Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

          ppi
0  226.983005
1  127.677940
2  141.211998
3  220.534624
4  226.983005
```

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

```
Intel Core i5 7200U 2.5GHz       190
Intel Core i7 7700HQ 2.8GHz      146
Intel Core i7 7500U 2.7GHz       134
Intel Core i7 8550U 1.8GHz        73
Intel Core i5 8250U 1.6GHz        72
                                ...
Intel Core M M3-6Y30 0.9GHz        1
AMD A9-Series 9420 2.9GHz          1
Intel Core i3 6006U 2.2GHz         1
AMD A6-Series 7310 2GHz            1
Intel Xeon E3-1535M v6 3.1GHz      1
Name: Cpu, Length: 118, dtype: int64
```

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

```
df.head()
```

```
  Company   TypeName                          Cpu  Ram
Memory  \
0   Apple  Ultrabook        Intel Core i5 2.3GHz    8            128GB
SSD
1   Apple  Ultrabook        Intel Core i5 1.8GHz    8  128GB Flash
Storage
2      HP   Notebook  Intel Core i5 7200U 2.5GHz    8            256GB
SSD
3   Apple  Ultrabook        Intel Core i7 2.7GHz   16            512GB
SSD
4   Apple  Ultrabook        Intel Core i5 3.1GHz    8            256GB
SSD

                            Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

          ppi       Cpu Name
0  226.983005  Intel Core i5
1  127.677940  Intel Core i5
2  141.211998  Intel Core i5
3  220.534624  Intel Core i7
4  226.983005  Intel Core i5
```

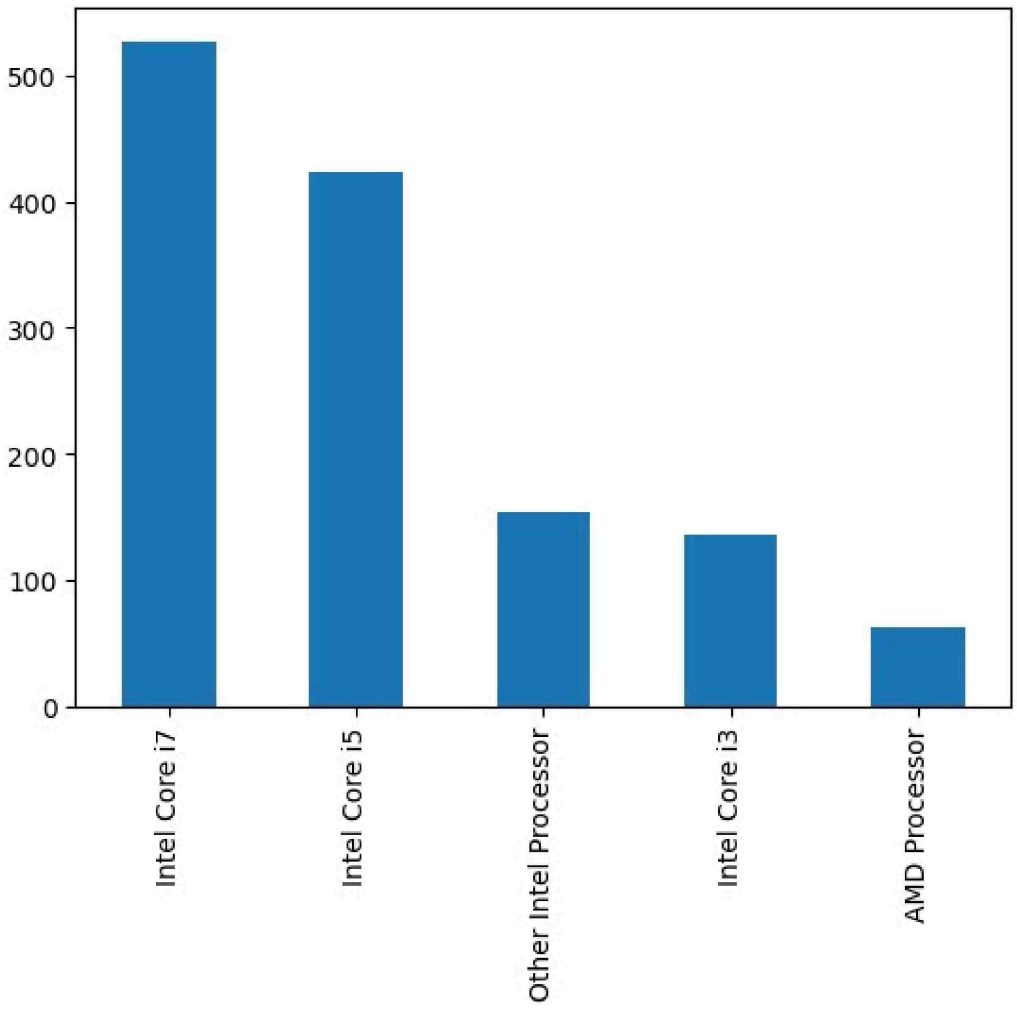
```
def fetch_processor(text):
    if text == 'Intel Core i7' or text == 'Intel Core i5' or text ==
'Intel Core i3':
        return text
    else:
        if text.split()[0] == 'Intel':
            return 'Other Intel Processor'
        else:
            return 'AMD Processor'

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

df.head()
```

```
  Company   TypeName                          Cpu  Ram
Memory  \
```

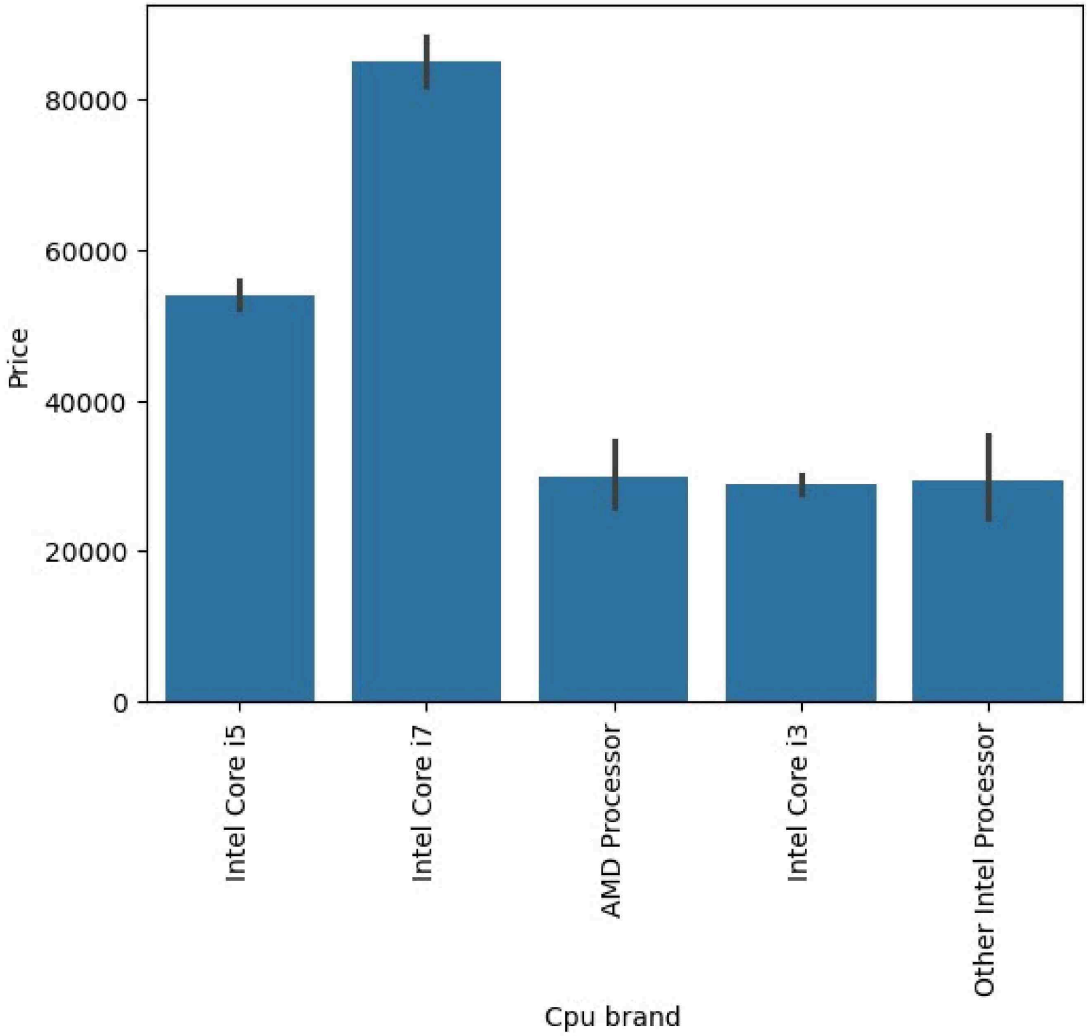
```
0   Apple  Ultrabook        Intel Core i5 2.3GHz    8            128GB
SSD
1   Apple  Ultrabook        Intel Core i5 1.8GHz    8  128GB Flash
Storage
2      HP   Notebook  Intel Core i5 7200U 2.5GHz    8            256GB
SSD
3   Apple  Ultrabook        Intel Core i7 2.7GHz   16            512GB
SSD
4   Apple  Ultrabook        Intel Core i5 3.1GHz    8            256GB
SSD

                             Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

          ppi       Cpu Name      Cpu brand
0  226.983005  Intel Core i5  Intel Core i5
1  127.677940  Intel Core i5  Intel Core i5
2  141.211998  Intel Core i5  Intel Core i5
3  220.534624  Intel Core i7  Intel Core i7
4  226.983005  Intel Core i5  Intel Core i5
```

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

```
<AxesSubplot:>
```




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



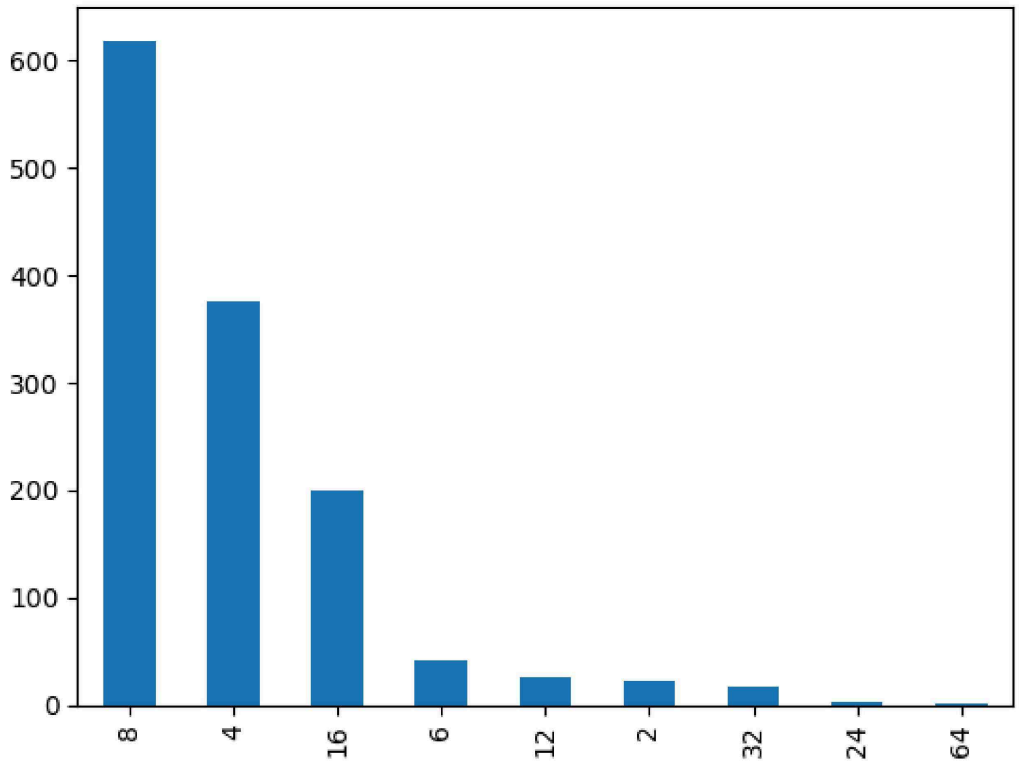

```
df.drop(columns=['Cpu','Cpu Name'],inplace=True)

df.head()
```

```
  Company   TypeName  Ram                Memory
Gpu  \
0   Apple  Ultrabook    8             128GB SSD  Intel Iris Plus
Graphics 640
1   Apple  Ultrabook    8  128GB Flash Storage        Intel HD
Graphics 6000
2      HP   Notebook    8             256GB SSD         Intel HD
Graphics 620
3   Apple  Ultrabook   16             512GB SSD            AMD Radeon
Pro 455
4   Apple  Ultrabook    8             256GB SSD  Intel Iris Plus
Graphics 650
```

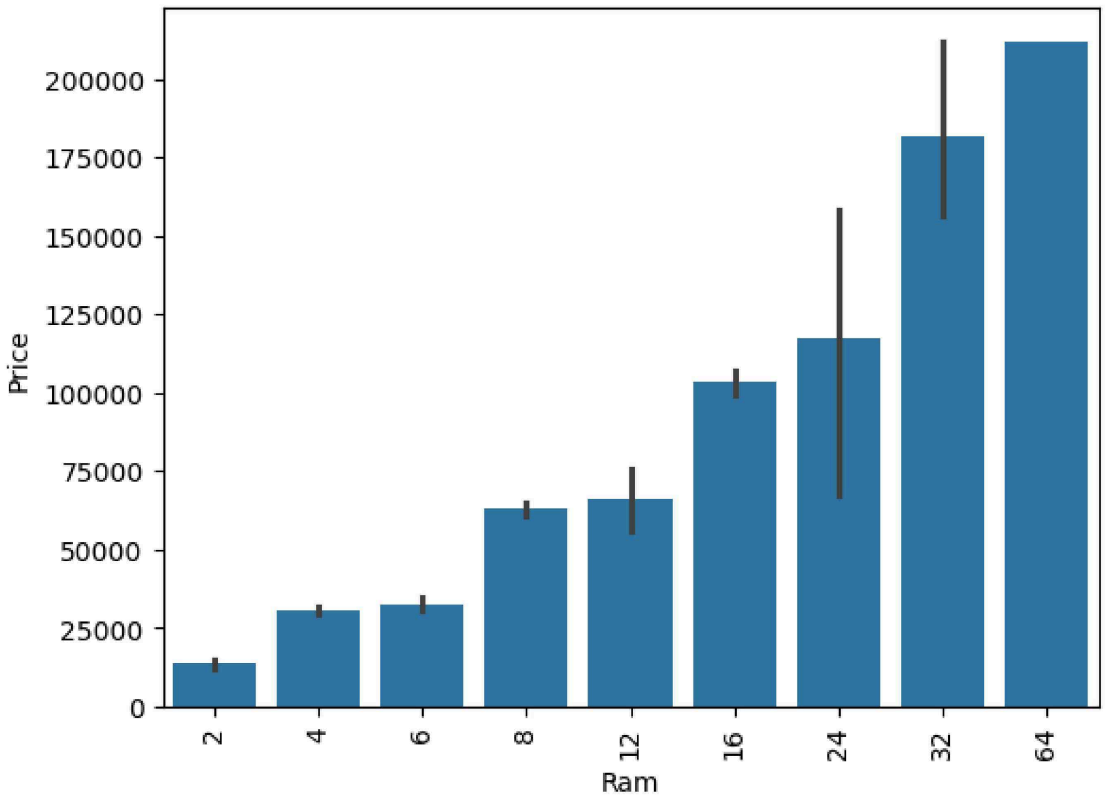
```
   OpSys  Weight        Price  Touchscreen  Ips         ppi      Cpu
brand
0  macOS    1.37   71378.6832            0    1  226.983005  Intel
Core i5
1  macOS    1.34   47895.5232            0    0  127.677940  Intel
Core i5
2  No OS    1.86   30636.0000            0    0  141.211998  Intel
Core i5
3  macOS    1.83  135195.3360            0    1  220.534624  Intel
Core i7
4  macOS    1.37   96095.8080            0    1  226.983005  Intel
Core i5
```

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

```
<AxesSubplot:>
```




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




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

```
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
```

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

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

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

df["second"]= new[1]

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

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

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

df["Layer2HDD"] = df["second"].apply(lambda x: 1 if "HDD" in x else 0)
df["Layer2SSD"] = df["second"].apply(lambda x: 1 if "SSD" in x else 0)
df["Layer2Hybrid"] = df["second"].apply(lambda x: 1 if "Hybrid" in x
else 0)
df["Layer2Flash_Storage"] = df["second"].apply(lambda x: 1 if "Flash
Storage" in x else 0)

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

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

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

df["HDD"]=(df["first"]*df["Layer1HDD"]+df["second"]*df["Layer2HDD"])
df["SSD"]=(df["first"]*df["Layer1SSD"]+df["second"]*df["Layer2SSD"])
df["Hybrid"]=(df["first"]*df["Layer1Hybrid"]
+df["second"]*df["Layer2Hybrid"])
df["Flash_Storage"]=(df["first"]*df["Layer1Flash_Storage"]
+df["second"]*df["Layer2Flash_Storage"])

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

```
<ipython-input-61-10829db803de>:16: FutureWarning: The default value
of regex will change from True to False in a future version.
  df['first'] = df['first'].str.replace(r'\D', '')
<ipython-input-61-10829db803de>:25: FutureWarning: The default value
of regex will change from True to False in a future version.
  df['second'] = df['second'].str.replace(r'\D', '')
```

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

```
      Company   TypeName  Ram               Memory
Gpu  \
1155       HP  Notebook    4              256 SSD           AMD Radeon
520
538        HP    Gaming   12             1000 HDD  Nvidia GeForce GTX
1060
837   Toshiba  Notebook    4              500 HDD    Intel HD Graphics
620
445      Dell  Notebook    8              256 SSD     Nvidia GeForce
930MX
955      Dell    Gaming   16  512 SSD +  1000 HDD  Nvidia GeForce GTX
1070

           OpSys  Weight        Price  Touchscreen  Ips         ppi  \
1155  Windows 10    1.91   25515.2592            0    0  141.211998
538   Windows 10    2.62   95850.7200            0    0  127.335675
837   Windows 10    2.00   35644.3200            0    0  100.454670
445   Windows 10    1.64   62817.1200            0    0  157.350512
955   Windows 10    4.36  168045.1200            0    1  254.671349

          Cpu brand   HDD  SSD  Hybrid  Flash_Storage
1155  AMD Processor     0  256       0              0
538   Intel Core i7  1000    0       0              0
837   Intel Core i3   500    0       0              0
445   Intel Core i5     0  256       0              0
955   Intel Core i7  1000  512       0              0
```

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

df.head()
```

```
  Company   TypeName  Ram                           Gpu  OpSys  Weight
\
0   Apple  Ultrabook    8  Intel Iris Plus Graphics 640  macOS    1.37

1   Apple  Ultrabook    8        Intel HD Graphics 6000  macOS    1.34

2      HP   Notebook    8         Intel HD Graphics 620  No OS    1.86

3   Apple  Ultrabook   16            AMD Radeon Pro 455  macOS    1.83

4   Apple  Ultrabook    8  Intel Iris Plus Graphics 650  macOS    1.37


         Price  Touchscreen  Ips         ppi      Cpu brand  HDD  SSD
Hybrid  \
0   71378.6832            0    1  226.983005  Intel Core i5    0  128
0
1   47895.5232            0    0  127.677940  Intel Core i5    0    0
0
2   30636.0000            0    0  141.211998  Intel Core i5    0  256
0
3  135195.3360            0    1  220.534624  Intel Core i7    0  512
0
4   96095.8080            0    1  226.983005  Intel Core i5    0  256
0

   Flash_Storage
0              0
1            128
2              0
3              0
4              0
```

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

```
<ipython-input-66-9447c1bc3d29>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  df.corr()['Price']

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
```

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

```
df.head()
```

```
  Company   TypeName  Ram                           Gpu  OpSys  Weight
\
0   Apple  Ultrabook    8  Intel Iris Plus Graphics 640  macOS    1.37

1   Apple  Ultrabook    8        Intel HD Graphics 6000  macOS    1.34

2      HP   Notebook    8         Intel HD Graphics 620  No OS    1.86

3   Apple  Ultrabook   16            AMD Radeon Pro 455  macOS    1.83

4   Apple  Ultrabook    8  Intel Iris Plus Graphics 650  macOS    1.37


         Price  Touchscreen  Ips         ppi      Cpu brand  HDD  SSD

0   71378.6832            0    1  226.983005  Intel Core i5    0  128

1   47895.5232            0    0  127.677940  Intel Core i5    0    0

2   30636.0000            0    0  141.211998  Intel Core i5    0  256

3  135195.3360            0    1  220.534624  Intel Core i7    0  512

4   96095.8080            0    1  226.983005  Intel Core i5    0  256
```

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

```
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 R5 520             1
AMD Radeon R7                 1
Intel HD Graphics 540         1
AMD Radeon 540                1
ARM Mali T860 MP4             1
Name: Gpu, Length: 110, dtype: int64
```

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

```
df.head()
```

```
  Company   TypeName  Ram                           Gpu  OpSys  Weight
\
0   Apple  Ultrabook    8  Intel Iris Plus Graphics 640  macOS    1.37

1   Apple  Ultrabook    8        Intel HD Graphics 6000  macOS    1.34

2      HP   Notebook    8         Intel HD Graphics 620  No OS    1.86

3   Apple  Ultrabook   16            AMD Radeon Pro 455  macOS    1.83

4   Apple  Ultrabook    8  Intel Iris Plus Graphics 650  macOS    1.37


         Price  Touchscreen  Ips         ppi      Cpu brand  HDD  SSD
\
0   71378.6832            0    1  226.983005  Intel Core i5    0  128

1   47895.5232            0    0  127.677940  Intel Core i5    0    0

2   30636.0000            0    0  141.211998  Intel Core i5    0  256

3  135195.3360            0    1  220.534624  Intel Core i7    0  512

4   96095.8080            0    1  226.983005  Intel Core i5    0  256


  Gpu brand
0     Intel
1     Intel
2     Intel
3       AMD
4     Intel
```

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

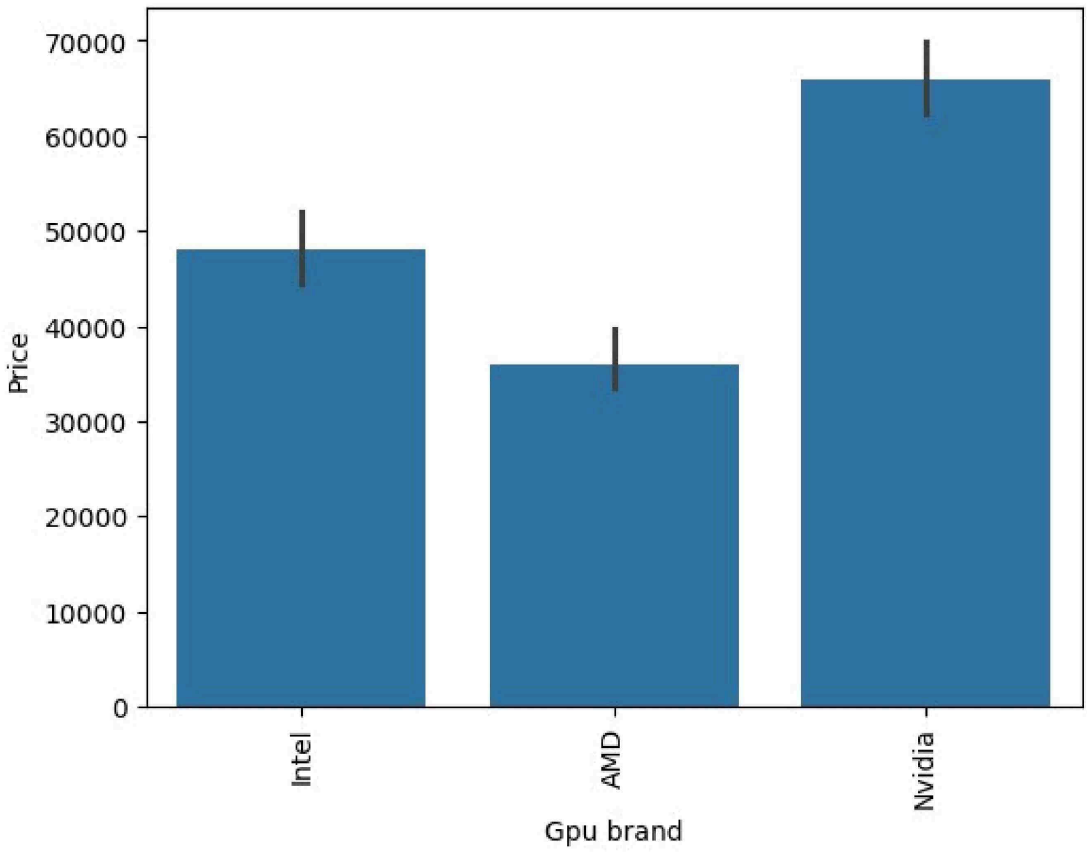
```
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: Gpu brand, dtype: int64
```

```
df = df[df['Gpu brand'] != 'ARM']
```

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

```
Intel     722
Nvidia    400
AMD       180
Name: Gpu brand, dtype: int64
```

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




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

df.head()
```

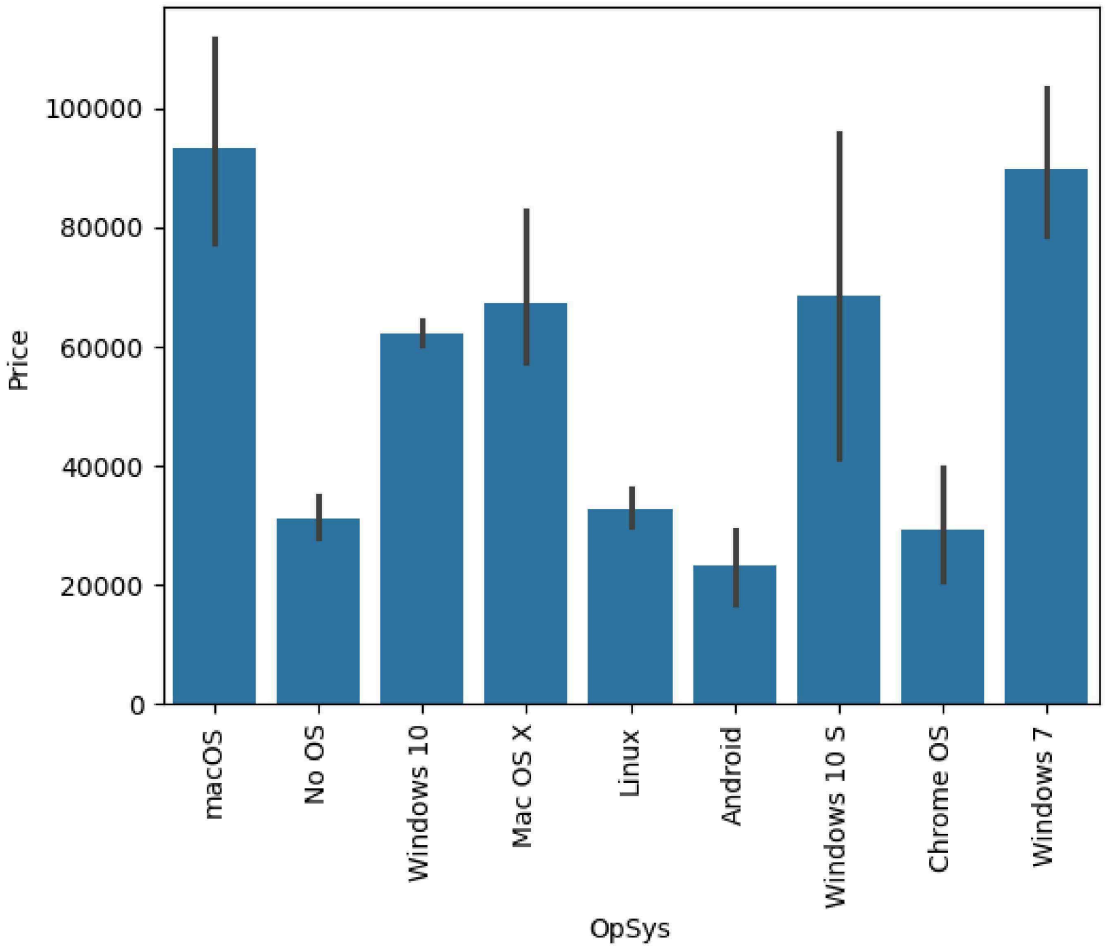
```
  Company   TypeName  Ram  OpSys  Weight        Price  Touchscreen
Ips  \
0   Apple  Ultrabook    8  macOS    1.37   71378.6832            0
1
1   Apple  Ultrabook    8  macOS    1.34   47895.5232            0
0
2      HP   Notebook    8  No OS    1.86   30636.0000            0
0
3   Apple  Ultrabook   16  macOS    1.83  135195.3360            0
1
4   Apple  Ultrabook    8  macOS    1.37   96095.8080            0
1

          ppi      Cpu brand  HDD  SSD Gpu brand
```

```
0  226.983005  Intel Core i5    0  128     Intel
1  127.677940  Intel Core i5    0    0     Intel
2  141.211998  Intel Core i5    0  256     Intel
3  220.534624  Intel Core i7    0  512       AMD
4  226.983005  Intel Core i5    0  256     Intel
```

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

```
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
```

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



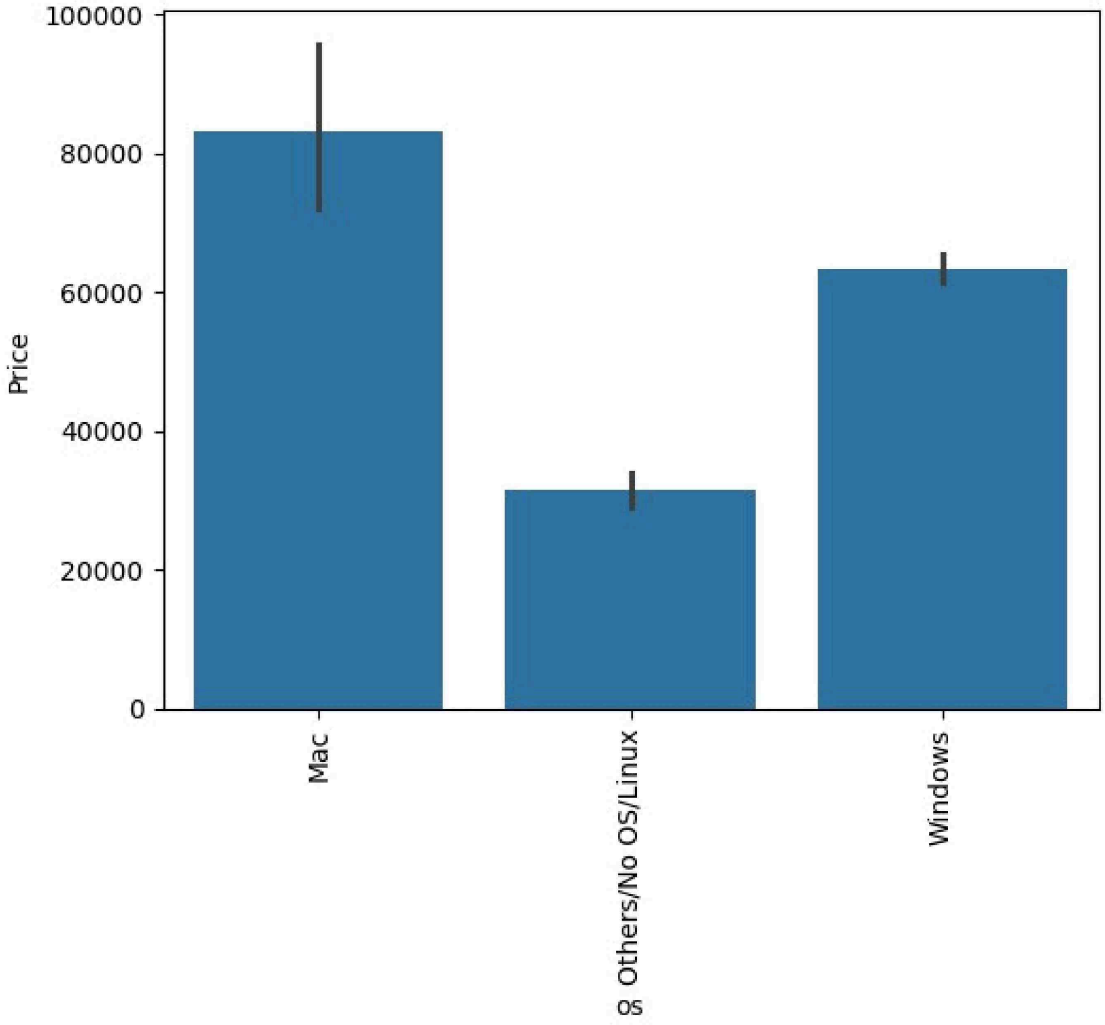

```
def cat_os(inp):
    if inp == 'Windows 10' or inp == 'Windows 7' or inp == 'Windows 10
S':
        return 'Windows'
    elif inp == 'macOS' or inp == 'Mac OS X':
        return 'Mac'
    else:
        return 'Others/No OS/Linux'

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

df.head()
```

```
  Company   TypeName  Ram  OpSys  Weight        Price  Touchscreen
Ips  \
0   Apple  Ultrabook    8  macOS    1.37   71378.6832            0
1
1   Apple  Ultrabook    8  macOS    1.34   47895.5232            0
0
2      HP   Notebook    8  No OS    1.86   30636.0000            0
```

```
0
3   Apple  Ultrabook   16  macOS    1.83  135195.3360            0
1
4   Apple  Ultrabook    8  macOS    1.37   96095.8080            0
1
```

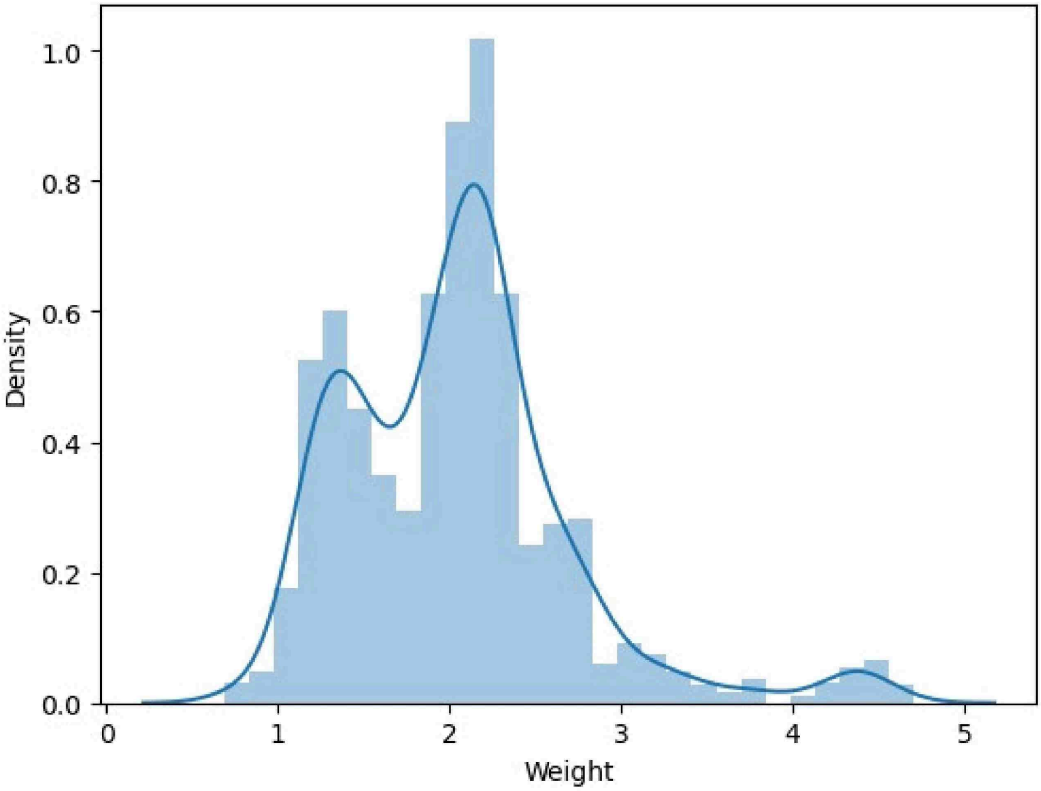
	ppi	Cpu brand	HDD	SSD	Gpu brand	os
0	226.983005	Intel Core i5	0	128	Intel	Mac
1	127.677940	Intel Core i5	0	0	Intel	Mac
2	141.211998	Intel Core i5	0	256	Intel	Others/No OS/Linux
3	220.534624	Intel Core i7	0	512	AMD	Mac
4	226.983005	Intel Core i5	0	256	Intel	Mac

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

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



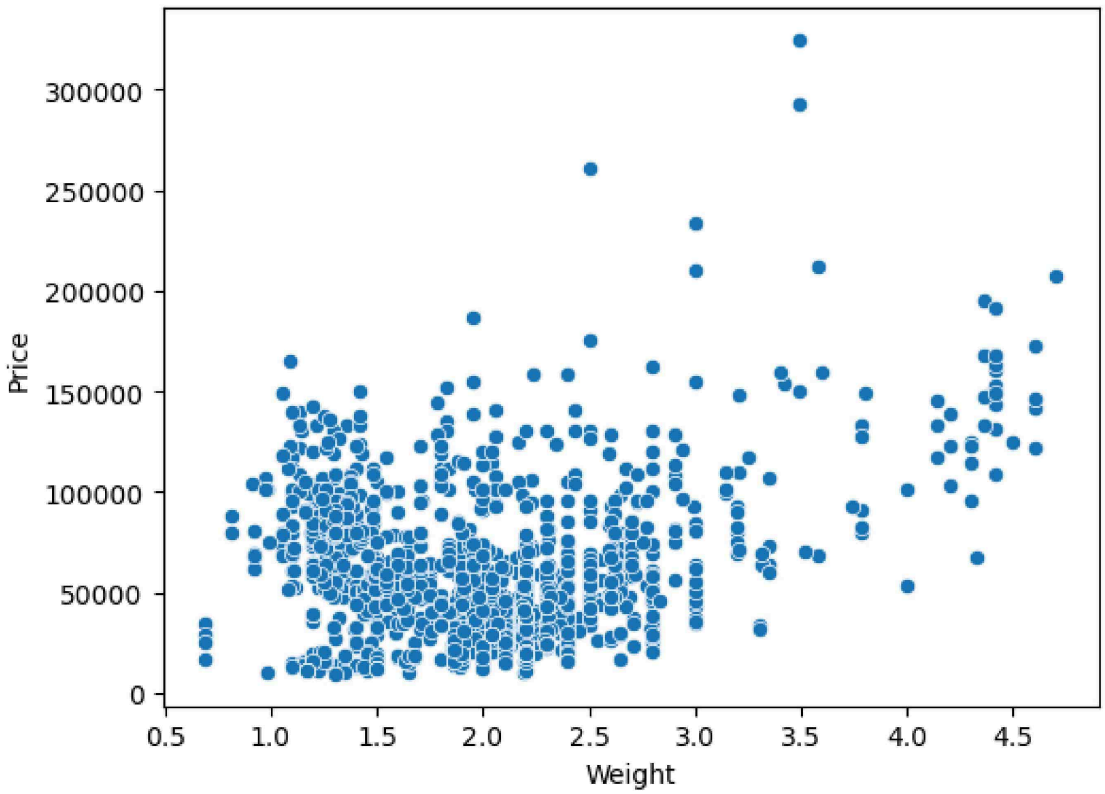

```
sns.distplot(df['Weight'])
```

```
<ipython-input-85-05ee4c8848be>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Weight'])

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




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

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



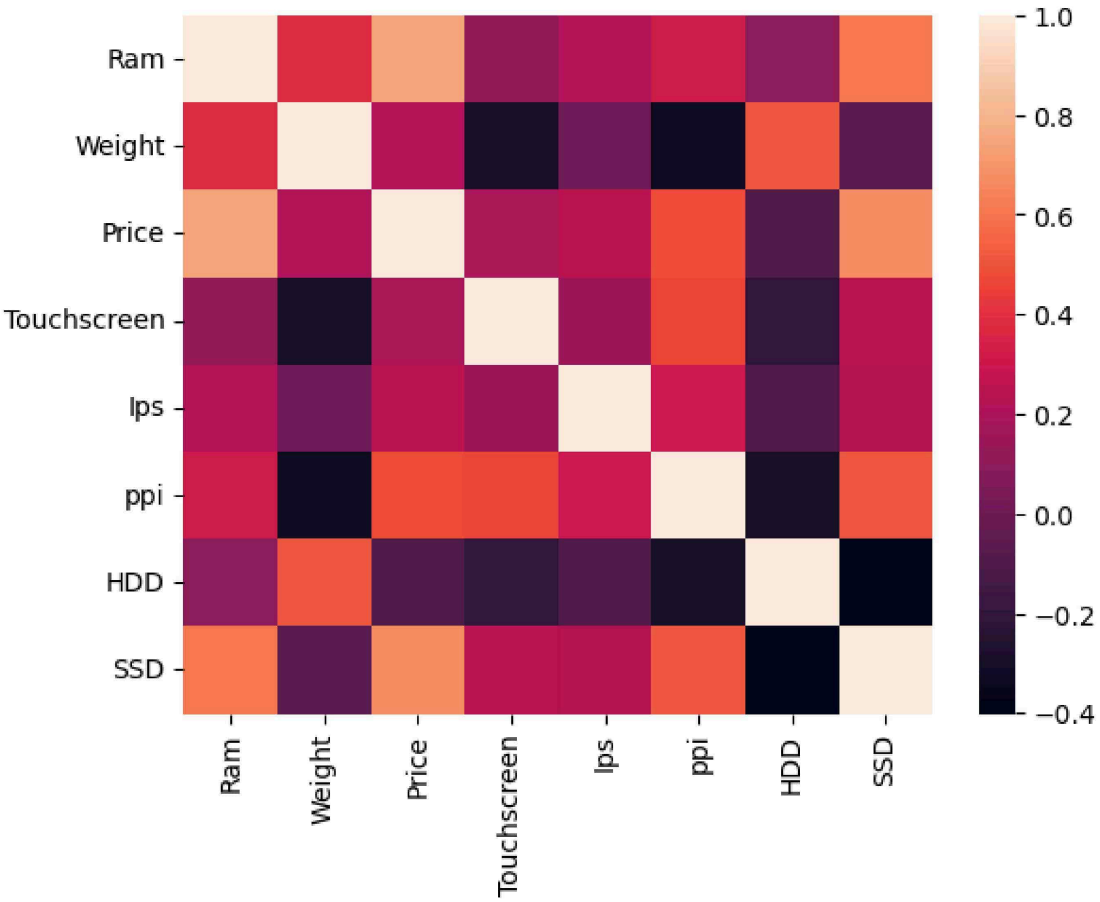

```
df.corr()['Price']
```

```
<ipython-input-87-9447c1bc3d29>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  df.corr()['Price']

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
```

```
sns.heatmap(df.corr())
```

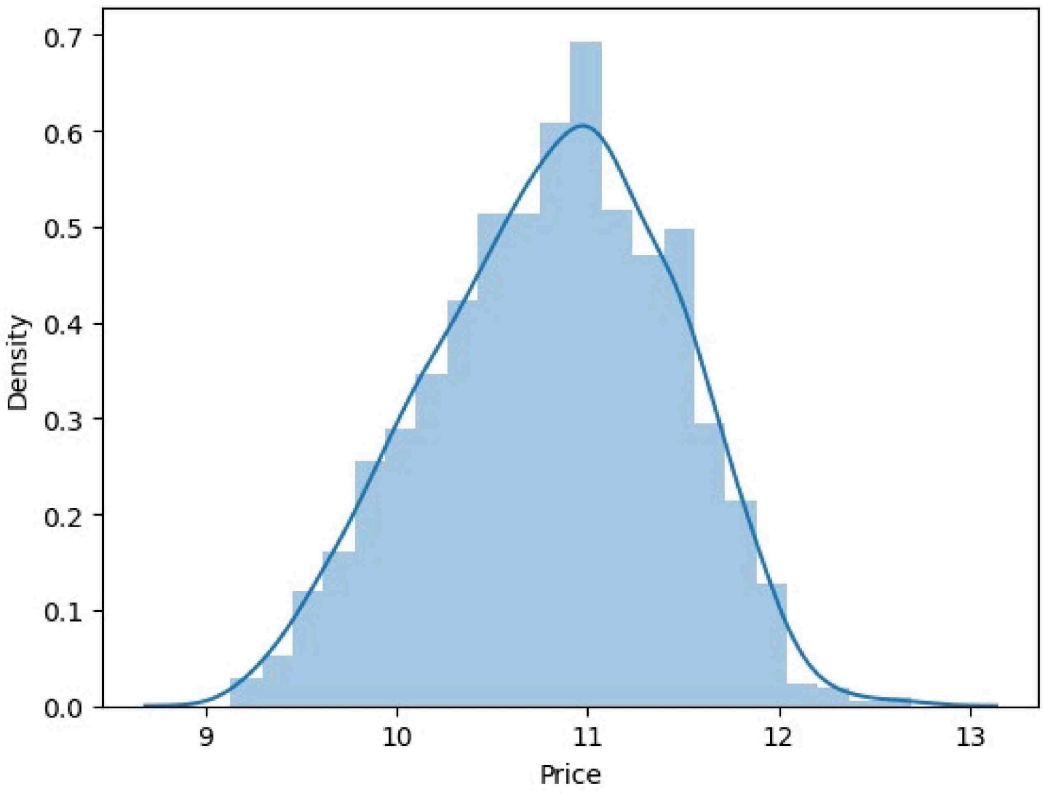
```
<ipython-input-88-aa4f4450a243>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```




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

```
<ipython-input-89-c1a82a4801f0>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

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

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




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

X
```

	Company	TypeName	Ram	Weight	Touchscreen	Ips	ppi
0	Apple	Ultrabook	8	1.37	0	1	226.983005
1	Apple	Ultrabook	8	1.34	0	0	127.677940
2	HP	Notebook	8	1.86	0	0	141.211998
3	Apple	Ultrabook	16	1.83	0	1	220.534624
4	Apple	Ultrabook	8	1.37	0	1	226.983005
...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	4	1.80	1	1	157.350512
1299	Lenovo	2 in 1 Convertible	16	1.30	1	1	276.053530
1300	Lenovo	Notebook	2	1.50	0	0	

```
111.935204
1301      HP            Notebook    6    2.19            0    0
100.454670
1302    Asus            Notebook    4    2.20            0    0
100.454670
```

```
                  Cpu brand   HDD  SSD Gpu brand                  os
0             Intel Core i5     0  128     Intel                 Mac
1             Intel Core i5     0    0     Intel                 Mac
2             Intel Core i5     0  256     Intel  Others/No OS/Linux
3             Intel Core i7     0  512       AMD                 Mac
4             Intel Core i5     0  256     Intel                 Mac
...                     ...   ...  ...       ...                 ...
1298          Intel Core i7     0  128     Intel             Windows
1299          Intel Core i7     0  512     Intel             Windows
1300  Other Intel Processor     0    0     Intel             Windows
1301          Intel Core i7  1000    0       AMD             Windows
1302  Other Intel Processor   500    0     Intel             Windows

[1302 rows x 12 columns]
```

```
y

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
```

```
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
```

```
      Company            TypeName  Ram  Weight  Touchscreen  Ips
ppi  \
183   Toshiba            Notebook    8    2.00            0    0
100.454670
1141      MSI              Gaming    8    2.40            0    0
141.211998
1049     Asus             Netbook    4    1.20            0    0
135.094211
1020     Dell  2 in 1 Convertible    4    2.08            1    1
```

```
141.211998
878      Dell            Notebook    4    2.18            0    0
141.211998
...       ...                ...  ...     ...          ...  ...
...
466      Acer            Notebook    4    2.20            0    0
100.454670
299      Asus           Ultrabook   16    1.63            0    0
141.211998
493      Acer            Notebook    8    2.20            0    0
100.454670
527    Lenovo            Notebook    8    2.20            0    0
100.454670
1193    Apple           Ultrabook    8    0.92            0    1
226.415547
```

	Cpu brand	HDD	SSD	Gpu brand	os
183	Intel Core i5	0	128	Intel	Windows
1141	Intel Core i7	1000	128	Nvidia	Windows
1049	Other Intel Processor	0	0	Intel	Others/No OS/Linux
1020	Intel Core i3	1000	0	Intel	Windows
878	Intel Core i5	1000	128	Nvidia	Windows
...	...	...	...	...	...
466	Intel Core i3	500	0	Nvidia	Windows
299	Intel Core i7	0	512	Nvidia	Windows
493	AMD Processor	1000	0	AMD	Windows
527	Intel Core i3	2000	0	Nvidia	Others/No OS/Linux
1193	Other Intel Processor	0	0	Intel	Mac

```
[1106 rows x 12 columns]
```

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

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

Linear regression

```
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))
```

```
/lib/python3.11/site-packages/sklearn/preprocessing/_encoders.py:975:
FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2
and will be removed in 1.4. `sparse_output` is ignored unless you
leave `sparse` to its default value.
  warnings.warn(

R2 score 0.8073277448418492
MAE 0.21017827976428904
```

Random Forest

```
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))
```

```
/lib/python3.11/site-packages/sklearn/preprocessing/_encoders.py:975:
FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2
and will be removed in 1.4. `sparse_output` is ignored unless you
```

```
leave `sparse` to its default value.
  warnings.warn(

R2 score 0.8873402378382488
MAE 0.15860130110457718
```

Exporting the Model

```
import pickle

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

df
```

```
     Company            TypeName  Ram  Weight        Price
Touchscreen  Ips  \
0      Apple           Ultrabook    8    1.37   71378.6832
0    1
1      Apple           Ultrabook    8    1.34   47895.5232
0    0
2         HP            Notebook    8    1.86   30636.0000
0    0
3      Apple           Ultrabook   16    1.83  135195.3360
0    1
4      Apple           Ultrabook    8    1.37   96095.8080
0    1
...      ...                 ...  ...     ...          ...           ..
.  ...
1298  Lenovo  2 in 1 Convertible    4    1.80   33992.6400
1    1
1299  Lenovo  2 in 1 Convertible   16    1.30   79866.7200
1    1
1300  Lenovo            Notebook    2    1.50   12201.1200
0    0
1301      HP            Notebook    6    2.19   40705.9200
0    0
1302    Asus            Notebook    4    2.20   19660.3200
0    0

              ppi              Cpu brand   HDD  SSD Gpu brand  \
0     226.983005          Intel Core i5     0  128     Intel
1     127.677940          Intel Core i5     0    0     Intel
2     141.211998          Intel Core i5     0  256     Intel
3     220.534624          Intel Core i7     0  512       AMD
4     226.983005          Intel Core i5     0  256     Intel
...          ...                   ...   ...  ...       ...
1298  157.350512          Intel Core i7     0  128     Intel
1299  276.053530          Intel Core i7     0  512     Intel
1300  111.935204  Other Intel Processor     0    0     Intel
```

```
1301  100.454670          Intel Core i7  1000    0       AMD
1302  100.454670  Other Intel Processor   500    0     Intel

                       os
0                     Mac
1                     Mac
2     Others/No OS/Linux
3                     Mac
4                     Mac
...                   ...
1298              Windows
1299              Windows
1300              Windows
1301              Windows
1302              Windows

[1302 rows x 13 columns]
```

X_train

```
      Company             TypeName  Ram  Weight  Touchscreen  Ips
ppi  \
183   Toshiba             Notebook    8    2.00            0    0
100.454670
1141      MSI               Gaming    8    2.40            0    0
141.211998
1049     Asus              Netbook    4    1.20            0    0
135.094211
1020     Dell  2 in 1 Convertible    4    2.08            1    1
141.211998
878      Dell             Notebook    4    2.18            0    0
141.211998
...       ...                  ...  ...     ...          ...  ...
...
466      Acer             Notebook    4    2.20            0    0
100.454670
299      Asus           Ultrabook   16    1.63            0    0
141.211998
493      Acer             Notebook    8    2.20            0    0
100.454670
527    Lenovo             Notebook    8    2.20            0    0
100.454670
1193    Apple           Ultrabook    8    0.92            0    1
226.415547

                  Cpu brand   HDD  SSD Gpu brand                  os
183           Intel Core i5     0  128     Intel             Windows
1141          Intel Core i7  1000  128    Nvidia             Windows
1049  Other Intel Processor     0    0     Intel  Others/No OS/Linux
1020          Intel Core i3  1000    0     Intel             Windows
```

```
878           Intel Core i5  1000  128    Nvidia             Windows
...                     ...   ...  ...       ...                 ...
466           Intel Core i3   500    0    Nvidia             Windows
299           Intel Core i7     0  512    Nvidia             Windows
493           AMD Processor  1000    0       AMD             Windows
527           Intel Core i3  2000    0    Nvidia  Others/No OS/Linux
1193  Other Intel Processor     0    0     Intel                 Mac

[1106 rows x 12 columns]
```

Future Scope of Improvement

The future of your laptop price predictor model can be expanded in various ways to improve its accuracy, functionality, and user experience. Here are some potential areas for exploration:

Data Acquisition and Preprocessing:

- Real-time Price Data: Integrate real-time pricing information from online retailers using web scraping techniques or APIs. This can help capture dynamic price fluctuations and provide more up-to-date predictions.

- Image Recognition: Explore using image recognition models to extract information from laptop product images, such as brand logos, design elements, or physical features that might influence price.

Model Development and Training:

- Deep Learning Architectures: Implement deep learning models like Convolutional Neural Networks (CNNs) to learn complex patterns from laptop specification data and potentially achieve higher prediction accuracy.

- Transfer Learning: Leverage pre-trained deep learning models on image datasets (e.g., for image recognition of laptop features) and fine-tune them for your specific laptop price prediction task.

- Multi-output Models: Develop models that predict not just the overall price, but also component-wise prices (CPU, RAM, storage etc.). This can provide more granular insights into the price breakdown of a laptop.

Model Deployment and User Interface:

- Web Application: Develop a user-friendly web application where users can easily input laptop specifications and receive predicted prices. This can make the model more accessible and user-friendly.

- API Integration: Create an API (Application Programming Interface) that allows other applications to integrate your price prediction functionality. This can expand the reach of your model to other platforms or services.

- Price Alerts: Implement a feature that allows users to set price alerts for specific laptops based on their predicted price range. This can be helpful for deal hunters who want to be notified when a desired laptop reaches their budgets

Certificate

This is to certify that Mr. Gaurav Kumar of Asansol Engineering College, roll number: 10871022016, has successfully completed a project on *Prediction of Laptop Prices* using *Machine Learning with Python* under the guidance of Prof. Arnab Chakraborty.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Prof. Arnab Chakraborty
Globsyn Finishing School

Certificate

This is to certify that Mr. Subham Shaw of Asansol Engineering College, roll number: 10871022004, has successfully completed a project on *Prediction of Laptop Prices* using *Machine Learning with Python* under the guidance of Prof. Arnab Chakraborty.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Prof. Arnab Chakraborty
Globsyn Finishing School

Certificate

This is to certify that Mr. Nishant Yadav of Asansol Engineering College, roll number: 10871022029, has successfully completed a project on *Prediction of Laptop Prices* using *Machine Learning with Python* under the guidance of Prof. Arnab Chakraborty.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Prof. Arnab Chakraborty
Globsyn Finishing School

Certificate

This is to certify that Ms. Puja Verma of Asansol Engineering College, roll number: 10871022011, has successfully completed a project on *Prediction of Laptop Prices* using *Machine Learning with Python* under the guidance of Prof. Arnab Chakraborty.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Prof. Arnab Chakraborty
Globsyn Finishing School

Certificate

This is to certify that Ms. Angelina Deepshikha Hans of Asansol Engineering College, roll number: 10871022009, has successfully completed a project on *Prediction of Laptop Prices* using *Machine Learning with Python* under the guidance of Prof. Arnab Chakraborty.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Prof. Arnab Chakraborty
Globsyn Finishing School