MACHINE LEARNING PROJECT REPORT

on

Laptop Price Prediction

Submitted by

Jaismeen

Registration Number: 12015083

Program Name: B. Tech Data Science (ML and AI)

Under the Guidance of

Mr. Ved Prakash Chaubey

School of Computer Science & Engineering

Lovely Professional University, Phagwara

I**,** Jaismeen, certify that this project is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this project has not previously been submitted for assessment in any academic capacity, and that I have not copied in part or whole or otherwise plagiarised the work of other persons. I confirm that I have identified and declared all possible conflicts that I may have.

Signature: Jaismeen

Acknowledgement

I would like to express my gratitude to **Mr. Ved Prakash Chaubey** my project supervisor, for their guidance and support throughout the project. I would also like to thank **Lovely Professional University** for providing me with the necessary resources and infrastructure to complete this project.

INTRODUCTION:

A laptop price prediction dataset typically contains a variety of features and attributes of laptops that can be used to train machine learning models for predicting laptop prices. The laptop price prediction dataset is a real-world dataset that contains information about various laptops and their specifications.

The dataset includes features such as brand, model, display size, processor type, RAM, storage type, graphics card, weight, price, etc. This dataset can be used for predicting the price of a laptop based on its features. The dataset can be useful for individuals, retailers, and manufacturers who are interested in understanding the factors that influence laptop prices. It can also be used in data analysis and machine learning.

WHY THIS DATASET?

The purpose of choosing the laptop price prediction dataset for performing machine learning is to develop a model that can accurately predict the prices of

laptops based on various features such as brand, processor, memory, screen size, etc. This can be useful for both consumers and sellers in the laptop market. Consumers can use the model to determine if they are getting a fair price for a laptop, while sellers can use it to set competitive prices for their products. Additionally, this dataset provides a good opportunity to explore various machine learning techniques such as linear regression, lasso regression, ridge regression, SVM, decision trees and random forest.

DOMAIN:

Computer Hardware and Electronics:

Computer hardware and electronics domain deals with the design, development, manufacturing, and maintenance of various computer components and electronic devices.

This dataset belongs to the domain of computer hardware and electronics. The dataset contains information about various features of laptops such as brand, screen size, RAM, processor, graphics card, operating system, weight and price. These features are important factors that determine the performance and price of a laptop.

DESCRIPTION OF DATASET:

This Dataset consists of 1303 rows and 12 columns:

1. Company: The name of the laptop brand, such as Acer, Apple, Asus, Dell, HP, Lenovo or MSI.

2. Product: The specific model of the laptop, such as Aspire 3, MacBook Air, ROG Strix G15 or ThinkPad E14.

3. TypeName: The general category of the laptop, such as Ultrabook, Notebook, Gaming, or Workstation.

4. Inches: The size of the laptop screen, measured diagonally in inches.

5. Screen Resolution: The screen resolution of the laptop, such as 1920x1080, 1366x768, or 3840x2160.

6. CPU: The processor type and speed, such as Intel Core i5-7200U or AMD Ryzen 5 2500U.

7. RAM: The amount of random-access memory (RAM) in the laptop, measured in gigabytes (GB).

8. Memory: The storage capacity of the laptop, measured in gigabytes (GB).

9. GPU: The graphics card type and memory, such as NVIDIA GeForce GTX 1650 with 4GB GDDR5.

10. OpSys: The operating system installed on the laptop, such as Windows 10, macOS or Linux.

11. Weight: The weight of the laptop, measured in kilograms (kg).

12. Price: The price of the laptop in Euros (€) at the time of data collection.

This dataset can be used to train machine learning models for predicting laptop prices based on the given features. The target variable in this case is the laptop price and the other features can be used as predictors.

QUESTIONS THAT CAN BE SOLVED USING THIS DATA:

- Which features have the most significant impact on the price of a laptop?
- Are customers willing to pay a premium for laptops with better graphics cards or processors?
- Is there a particular brand that tends to be priced higher or lower than its competitors?
- Are laptops with larger screen sizes generally more expensive than those with smaller screens?
- Can we predict the price range for a laptop based on its features and if so, how accurate are our predictions?
- Are there any combinations of features that result in a laptop being priced significantly higher or lower than expected?
- Can we identify any trends in the laptop market, such as the increasing popularity of certain features or brands?

- How well do different machine learning algorithms perform in predicting laptop prices and which one is the most accurate?

- Can we use the insights gained from this project to inform pricing strategies for laptop manufacturers and retailers?

- What additional data could we collect to improve the accuracy of our price predictions and how feasible would it be to collect this data?

OBJECTIVE:

Given the various specifications and features of laptops, the task is to predict the price of a laptop accurately. This can help customers in making informed decisions while purchasing a laptop and can also assist companies in setting prices for their products based on their features and specifications. Additionally, this can aid in identifying the important features that contribute the most towards the price of a laptop.

The objective of this project is to develop a machine learning model that can accurately predict the prices of laptops based on their hardware specifications and other features.

This will enable businesses in the computer hardware and electronics domain to offer competitive prices to their customers and optimize their inventory management. By accurately predicting the prices of laptops, businesses can ensure that they are not overpricing or under-pricing their products, which can lead to loss of sales or profit.

By understanding which features have the greatest impact on laptop prices, businesses can make informed decisions about which products to stock and how to price them. The goal is to provide better customer satisfaction and increase revenue for the business.

Importing the Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import statsmodels.api as sm
import pickle as pkcvb
warnings.filterwarnings('ignore')
from scipy.stats import norm, boxcox
from scipy import stats
from collections import Counter
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
```

Reading and Understanding Data:

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

```
df.head()
```

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

```
df.tail()
```

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
1298	1298	Lenovo	2 in 1 Convertible	14.0	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4GB	128GB SSD	Intel HD Graphics 520	Windows 10	1.8kg	33992.64
1299	1299	Lenovo	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16GB	512GB SSD	Intel HD Graphics 520	Windows 10	1.3kg	79866.72
1300	1300	Lenovo	Notebook	14.0	1366x768	Intel Celeron Dual Core N3050 1.6GHz	2GB	64GB Flash Storage	Intel HD Graphics	Windows 10	1.5kg	12201.12
1301	1301	HP	Notebook	15.6	1366x768	Intel Core i7 6500U 2.5GHz	6GB	1TB HDD	AMD Radeon R5 M330	Windows 10	2.19kg	40705.92
1302	1302	Asus	Notebook	15.6	1366x768	Intel Celeron Dual Core N3050 1.6GHz	4GB	500GB HDD	Intel HD Graphics	Windows 10	2.2kg	19660.32

Checking the Datatypes of Columns:

```
df.info()
```

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

Checking the Null – Values:

```
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
```

Removing Categorical features from Numerical Columns:

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

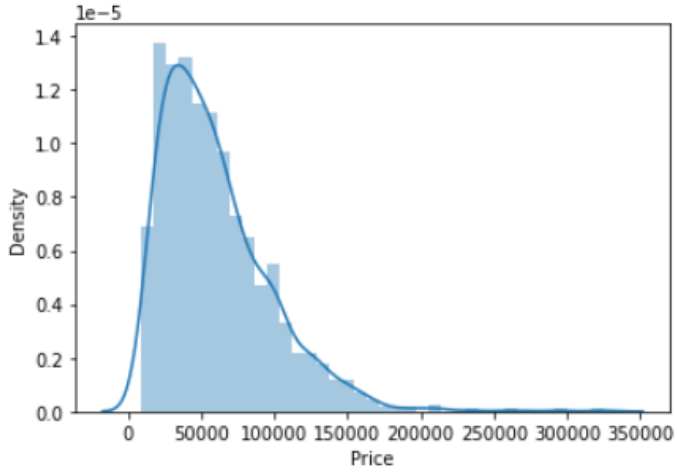
```
df.head()
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
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

Plotting the Prices of Laptops on a Distplot:

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

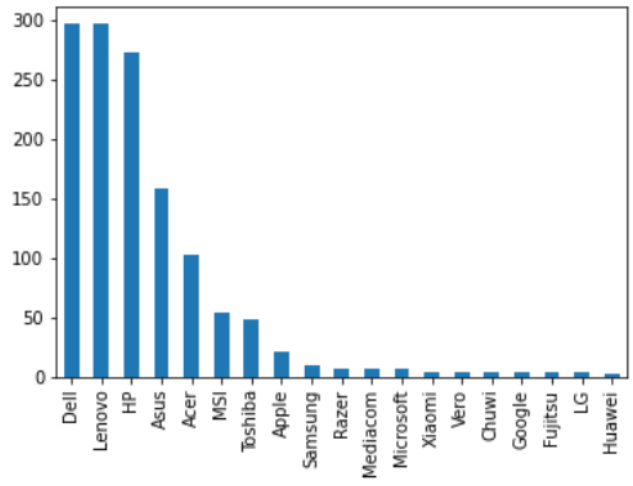
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



Visualizing the Laptop Companies on a Bar plot:

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

```
<AxesSubplot:>
```

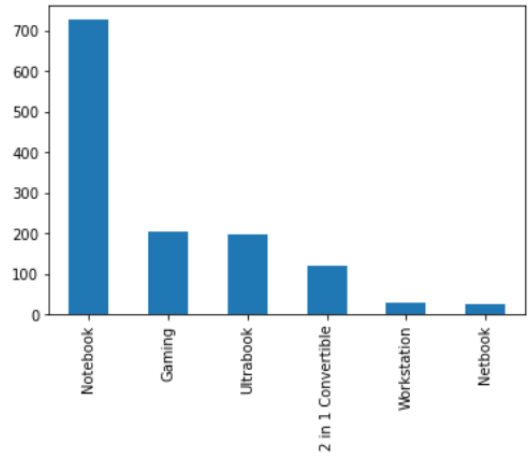


It can be visualized that number of Dell and Lenovo laptops is high.

Plotting the Type of Laptop on a Bar Plot:

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

```
<AxesSubplot:>
```

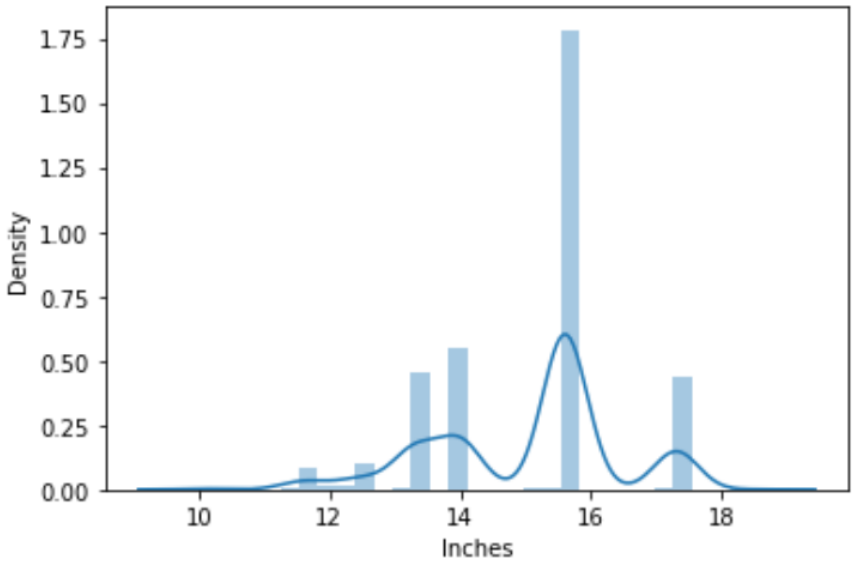


It can be inferred that there are more number of notebooks and there are least number of Netbooks.

Distplot that represents Number of inches the Laptop is:

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

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

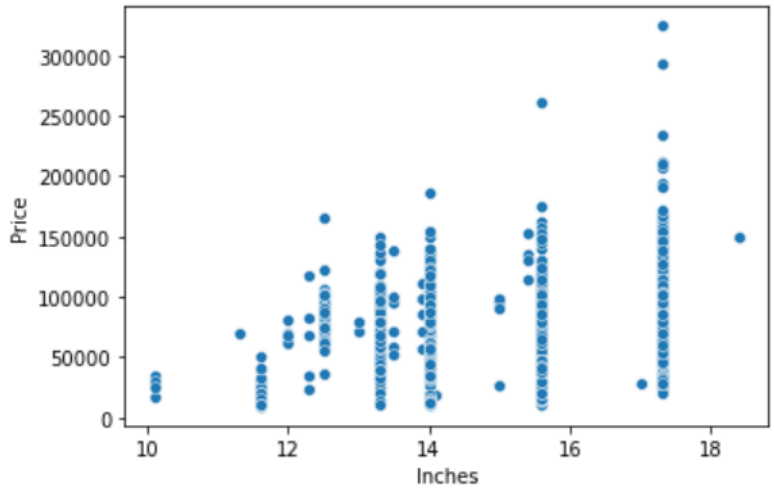


It can be inferred that more number of laptops are between 15 to 16 inches.

Scatterplot that represents the relationship between Inches and Prices.

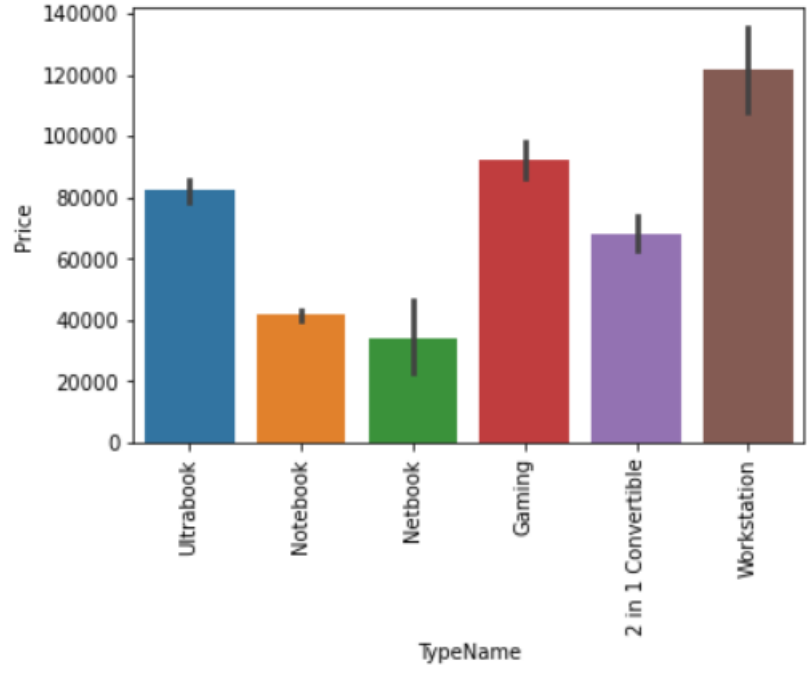
```
sns.scatterplot(x=df['Inches'],y=df['Price'])
```

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



Plotting the Type of Laptop and its price on a Bar Plot:

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



This barplot visualises the relationship between Laptop type and 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
```

It can be deduced that there are 507 laptops with Resolution **Full HD 1920x1080**

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

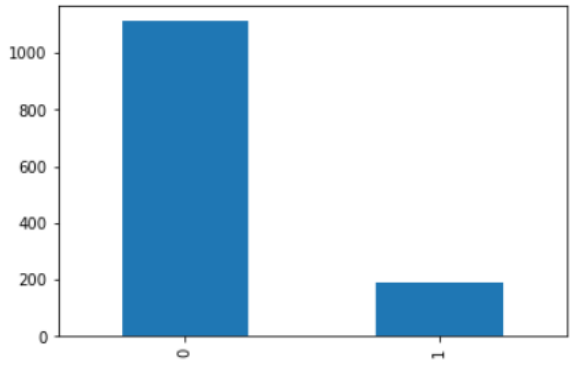
```
df.sample(5)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen
629	Dell	Notebook	15.6	Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	8	256GB SSD	Nvidia GeForce GTX 1050	Windows 10	2.00	96969.60	0
174	HP	Notebook	17.3	Full HD 1920x1080	Intel Core i5 8250U 1.6GHz	8	256GB SSD	Nvidia GeForce 930MX	Windows 10	2.50	49177.44	0
707	Lenovo	Workstation	15.6	Full HD 1920x1080	Intel Core i7 6500U 2.5GHz	16	512GB SSD	Nvidia Quadro M520M	Windows 7	2.18	98834.40	0
1079	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 6200U 2.3GHz	4	500GB HDD	Intel HD Graphics 520	Windows 7	2.31	50083.20	0
938	Dell	Ultrabook	14.0	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	1.36	87325.92	0

Here, Touchscreen column is added. It is depended on Screen Resolution. If it is mentioned Touchscreen, then assigned 1. Else,0.

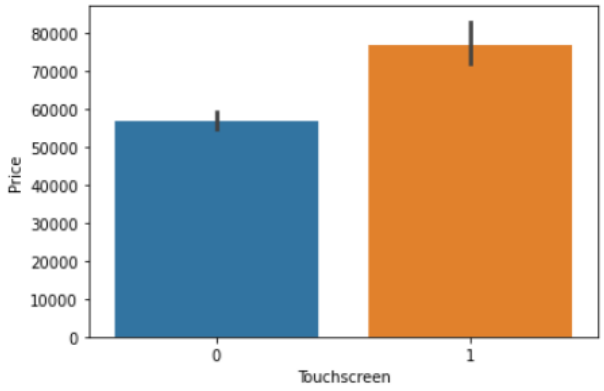
```
df['Touchscreen'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```




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

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



It can be inferred that there are more number of Touchscreen laptops.

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

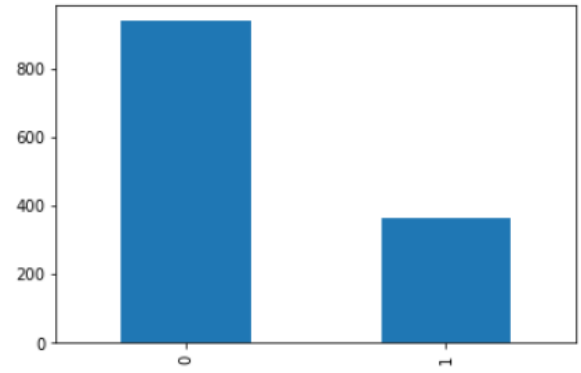
```
df.head()
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips
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	0	1
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0
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	0	0
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	0	1
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	0	1

Here, IPS column is added. It is also depended on Screen Resolution. If it is mentioned IPS , then assigned 1. Else,0.

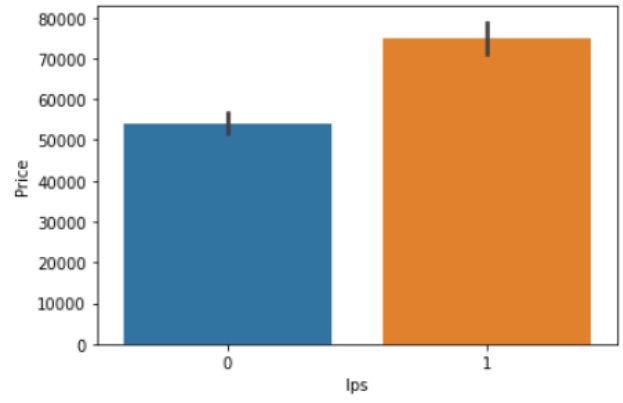
```
df['Ips'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```




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

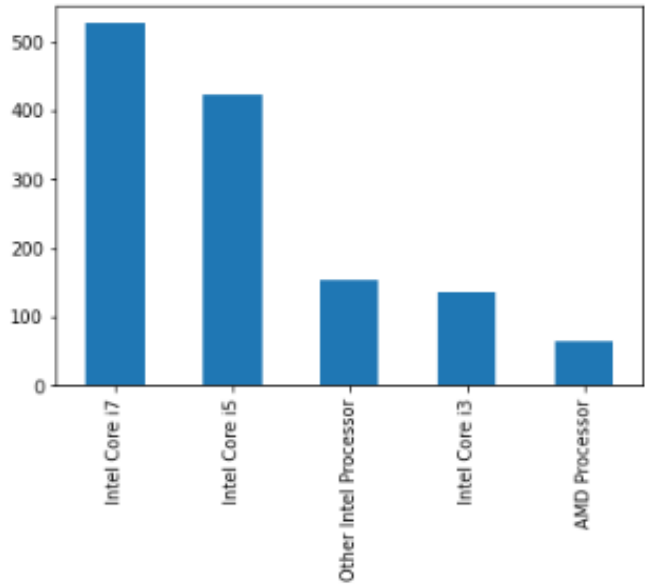
```
<AxesSubplot:xlabel='Ips', ylabel='Price'>
```



It can be inferred that there are less number of IPS display laptops.

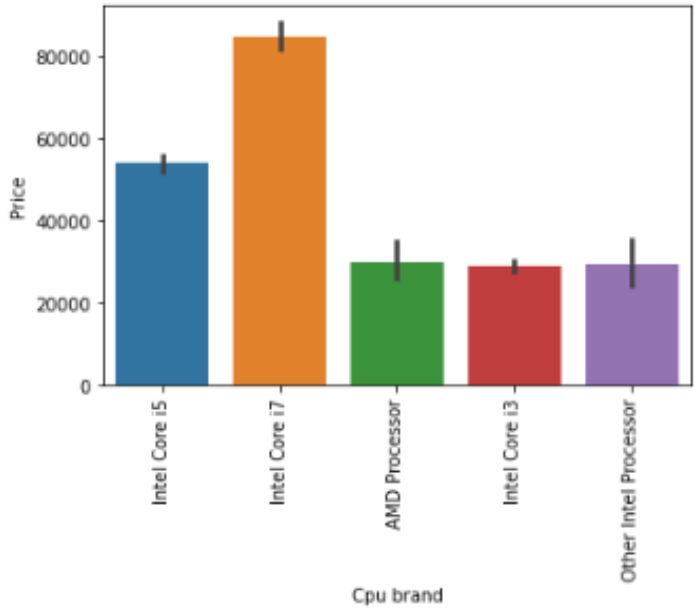
```
df['Cpu brand'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```



It can be inferred that there are more number of intel- core i7 CPU processors.

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



It can be inferred that intel- core i7 CPU processors are with high price.

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

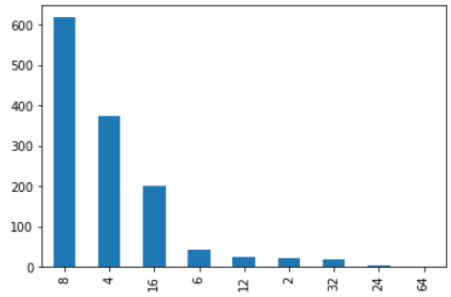
Removing the columns CPU, CPU Name.

```
df.head()
```

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

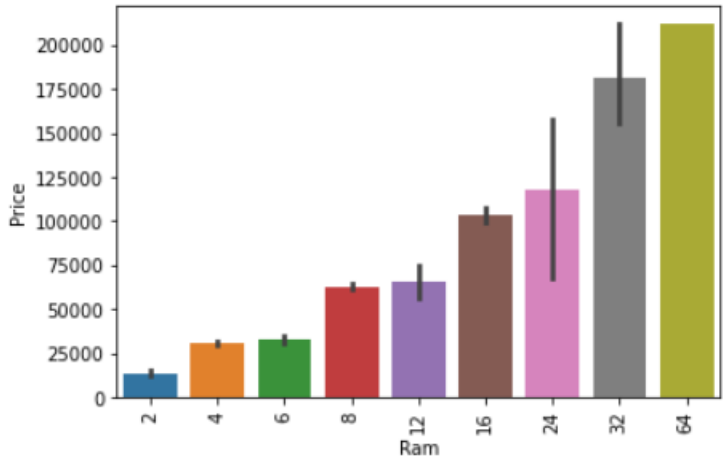
```
df['Ram'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```



It can be deduced that there are more number of 8GB ram laptops.

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



This plot shows the relation between RAM and Price.

```
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
```

There are 412 laptops with 256GB SSD Storage.

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

Dropping the Memory Column

```
df.head()
```

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Hybrid	Flash_Storage
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	0	0
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	0	128
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256	0	0
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512	0	0
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256	0	0

Dropping the Redundant Columns:

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

```
df.head()
```

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	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
```

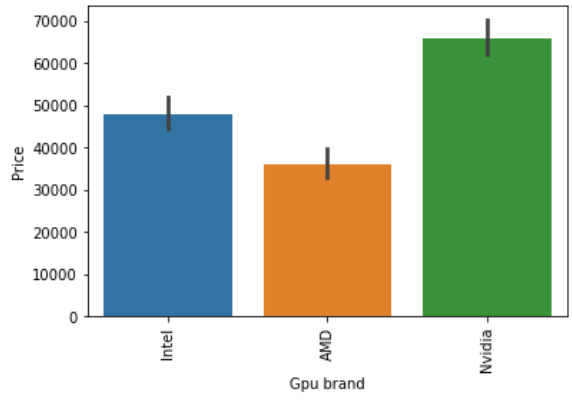
```
df = df[df['Gpu brand'] != 'ARM']
```

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

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

There are More number of Intel GPU's present.

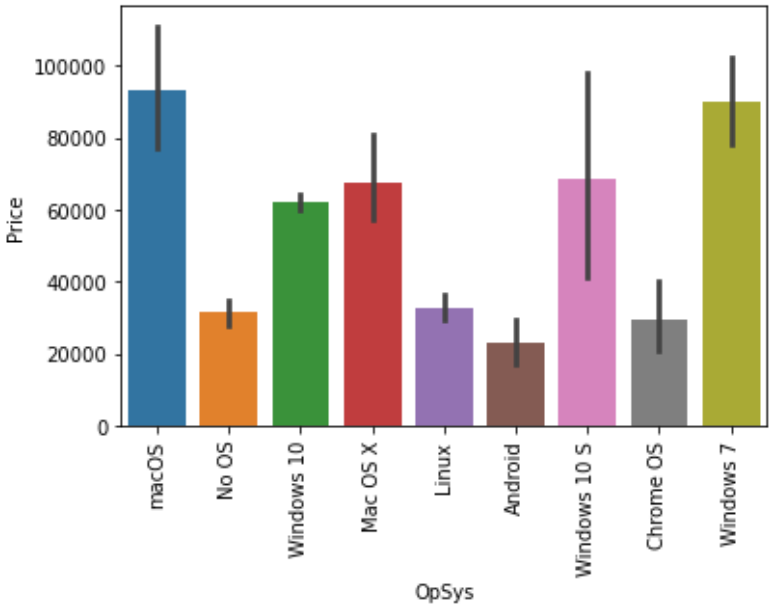
```
sns.barplot(x=df['Gpu brand'],y=df['Price'],estimator=np.median)
plt.xticks(rotation='vertical')
plt.show()
```



It can be deduced that laptops containing Nvidia GPU have more price when compared to laptops having Intel and AMD GPUs.

Visualising the relation between OS and Price

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



The above graph infers the relation between the Operating System and the Price Columns.

```
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'
```

Converted the data into Simpler form i.e,

If the OS is windows10/windows7/windows10S -- It is renamed as 'Windows'.

Similarly, If the OS is macOS/macOSX -- It is renamed as 'Mac'.

If the OS is neither Windows nor Mac then it is renamed as 'Others/No OS/Linux'.

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

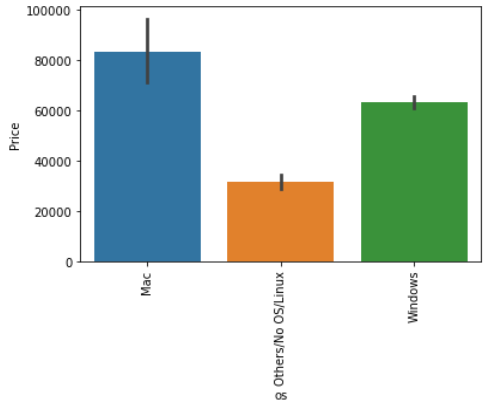
```
df.head()
```

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

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

Visualising the relation between OS and Price Columns after Simplifying

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



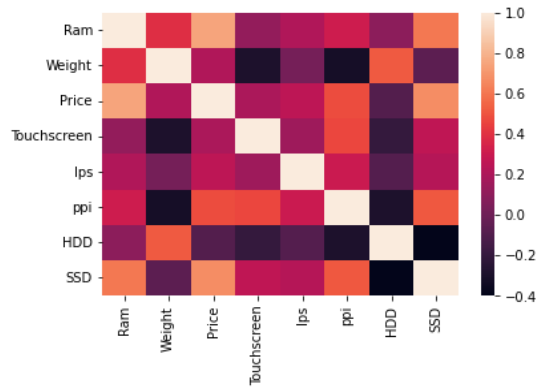
Heatmap Representation:

```
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())
```

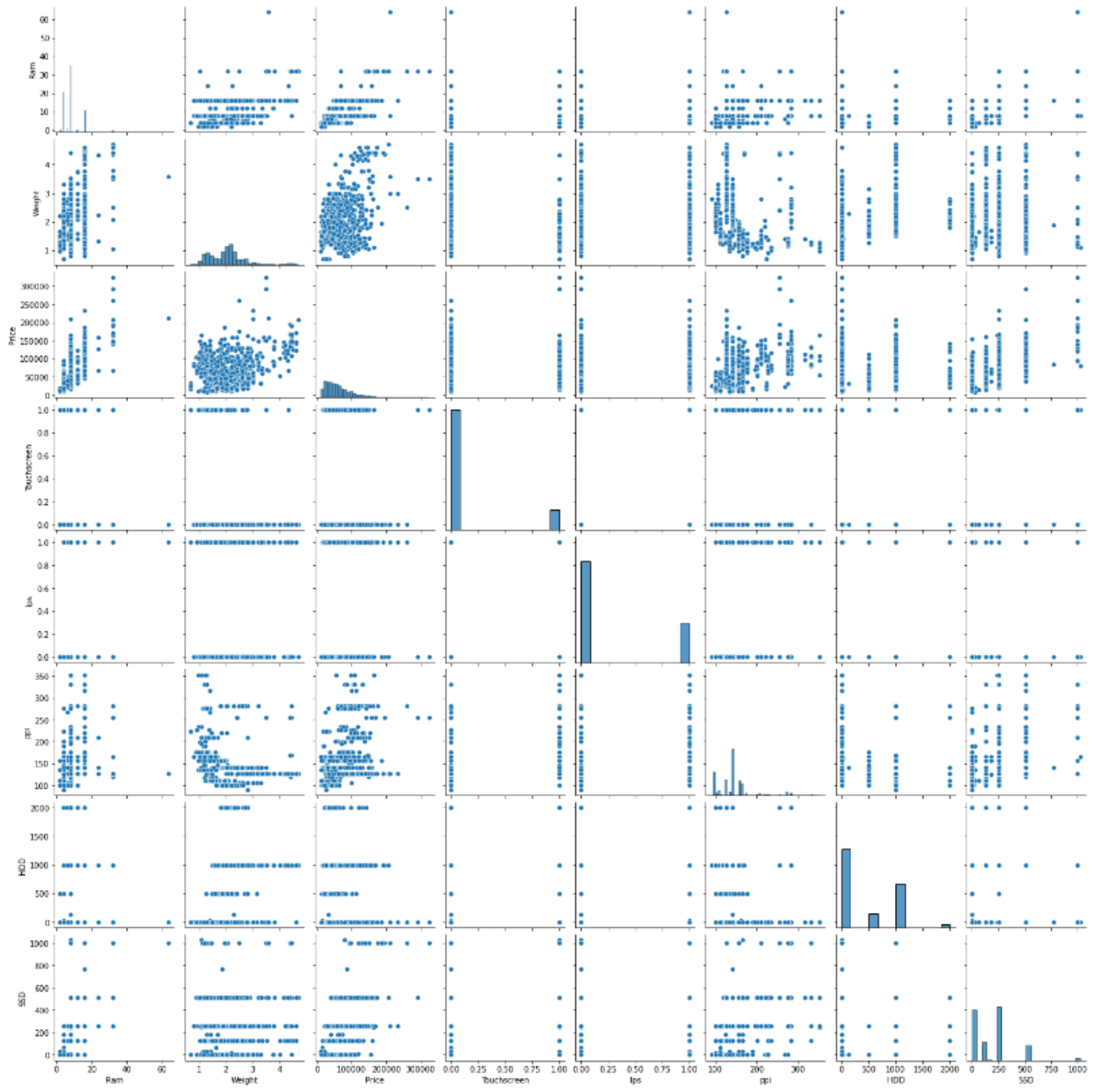
```
<AxesSubplot:>
```



Pairplot Representation:

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x2dcbdd10a90>
```



This pairplot shows that there is a strong positive correlation between the laptop's price and its specifications such as the processor speed, RAM and hard disk capacity. This suggests that laptops with higher specifications tend to have a higher price. Additionally, we observe a positive correlation between the price and the weight of the laptop, indicating that lighter laptops are typically more expensive.

Splitting Dataset into Train and Test Data:

```
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	Cpu brand	HDD	SSD	Gpu brand	os
183	Toshiba	Notebook	8	2.00	0	0	100.454670	Intel Core i5	0	128	Intel	Windows
1141	MSI	Gaming	8	2.40	0	0	141.211998	Intel Core i7	1000	128	Nvidia	Windows
1049	Asus	Netbook	4	1.20	0	0	135.094211	Other Intel Processor	0	0	Intel	Others/No OS/Linux
1020	Dell	2 in 1 Convertible	4	2.08	1	1	141.211998	Intel Core i3	1000	0	Intel	Windows
878	Dell	Notebook	4	2.18	0	0	141.211998	Intel Core i5	1000	128	Nvidia	Windows
...	...	...	...	...	...	...	...	...	...	...	...	...
466	Acer	Notebook	4	2.20	0	0	100.454670	Intel Core i3	500	0	Nvidia	Windows
299	Asus	Ultrabook	16	1.63	0	0	141.211998	Intel Core i7	0	512	Nvidia	Windows
493	Acer	Notebook	8	2.20	0	0	100.454670	AMD Processor	1000	0	AMD	Windows
527	Lenovo	Notebook	8	2.20	0	0	100.454670	Intel Core i3	2000	0	Nvidia	Others/No OS/Linux
1193	Apple	Ultrabook	8	0.92	0	1	226.415547	Other Intel Processor	0	0	Intel	Mac

1106 rows × 12 columns

Linear Regression:

Linear regression is a statistical technique that is commonly used in data science and machine learning to model the relationship between a dependent variable and one or more independent variables. In the context of the laptop price prediction dataset, linear regression can be used to build a model that predicts the price of a laptop based on its specifications and brand.

The laptop price prediction dataset contains information about various laptops, including their brand, processor speed, RAM, hard disk capacity, weight, and graphics card type. By applying linear regression to this dataset, we can build a model that estimates the price of a laptop based on its specifications.

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

step2 = LinearRegression()

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

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

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

```
R2 score 0.807327744841852
MAE 0.21017827976429213
```

The "ColumnTransformer" step applies one-hot encoding to certain columns of the dataset, converting categorical variables into numerical values that can be used in the regression model.

The "LinearRegression" step fits a linear regression model to the transformed data.

The "Pipeline" step combines these two steps into a single process, allowing for easy application to new data

The R2 score measures the proportion of the variance in the target variable that is explained by the model, with a higher score indicating a better fit. The MAE measures the average difference between the predicted and actual values, with a lower score indicating better performance.

Ridge Regression:

Ridge regression is a regularization technique used in linear regression models to prevent overfitting. It is particularly useful when dealing with datasets with a large number of features.

Ridge regression can be used to predict the price of a laptop based on various features such as the brand, processor type, RAM size, storage capacity, screen size, and others.

The amount of regularization in ridge regression is controlled by a hyperparameter called the regularization strength, which balances the trade-off between model complexity and data fitting.

```
steprr1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

steprr2 = Ridge(alpha=10)

pipe = Pipeline([
    ('step1',steprr1),
    ('step2',steprr2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

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

```
R2 score 0.812733103131181
MAE 0.20926802242582962
```

The R2 score is a measure of how well the Ridge Regression model fits the data, with a higher score indicating a better fit. In this case, the R2 score is not provided, so it is not possible to evaluate the model's performance based on this metric.

The MAE (mean absolute error) is a measure of the average difference between the predicted and actual values, with a lower value indicating better performance. The MAE value is also not provided in the code, so it is not possible to evaluate the model's performance based on this metric either.

Therefore, without the R2 score and MAE values, it is not possible to draw a conclusion about the effectiveness of the Ridge Regression model in predicting laptop prices.

Lasso Regression:

Lasso Regression is a type of linear regression that involves adding a penalty term to the cost function in order to reduce the complexity of the model and avoid overfitting. It is particularly useful when dealing with high-dimensional datasets, where the number of features is much larger than the number of observations.

Lasso Regression can be used to identify the most important features that contribute to the price of a laptop. By shrinking the coefficients of less important features towards zero, Lasso Regression can help to improve the accuracy of the model and make it more interpretable.

```
steplsr1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

steplsr2 = Lasso(alpha=0.001)

pipe = Pipeline([
    ('step1',steplsr1),
    ('step2',steplsr2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

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

```
R2 score 0.8071853945317105
MAE 0.21114361613472565
```

The alpha value for Lasso Regression is set to 0.001. The code fits the model on the training set, makes predictions on the test set and calculates the R2 score and Mean Absolute Error (MAE) between the predicted and actual values.

The output of the code indicates that the Lasso regression model has achieved an R2 score of 0.8071 and an MAE of 0.21114 on the test dataset. This means that the model can explain around 80.71% of the variance in the dependent variable and the average absolute difference between the predicted and actual values is around 0.21114.

Decision Tree:

Decision Trees are a type of supervised learning algorithm used in machine learning for both regression and classification tasks. It builds a model in the form of a tree structure, where each internal node represents a feature or attribute, each branch represents a decision rule, and each leaf node represents a prediction or outcome.

Decision trees can be used to predict the price of a laptop based on a set of input features such as the brand, processor type, memory, and so on.

The decision tree algorithm builds a tree-like model of decisions and their possible consequences, with each internal node representing a decision based on the value of a feature, and each leaf node representing a predicted output value. The result is a tree-like model that can be used to make predictions on new, unseen data.

```
stepdt1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

stepdt2 = DecisionTreeRegressor(max_depth=8)

pipe = Pipeline([
    ('step1',stepdt1),
    ('step2',stepdt2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

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

```
R2 score 0.8433319878038295
MAE 0.1809830045203912
```

The decision tree model has achieved a good R2 score of 0.843331 and a low MAE of 0.180983 on the laptop price prediction dataset. This indicates that the decision tree model is able to explain a significant amount of variance in the target variable and the average difference between the actual and predicted prices is low.

SVM (Support Vector Machines):

Support Vector Machines (SVM) is a popular machine learning algorithm used for classification and regression analysis. SVM works by creating a hyperplane or a decision boundary that separates the data into different classes or groups. SVM can be used for both linear and non-linear classification and regression problems

```
stepsvm1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

stepsvm2 = SVR(kernel='rbf',C=10000,epsilon=0.1)

pipe = Pipeline([
    ('step1',stepsvm1),
    ('step2',stepsvm2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

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

```
R2 score 0.8083180902257614
MAE 0.20239059427481307
```

The SVM model with the RBF kernel and the specified hyperparameters has achieved an R2 score of 0.808 and a mean absolute error (MAE) of 0.202 on the test set. The R2 score indicates that the model explains 80.8% of the variance in the target variable, while the MAE suggests that the average absolute difference between the predicted and true values of the target variable is 0.202. These metrics suggest that the SVM model is performing reasonably well in predicting the laptop prices.

Random Forest:

Random forest is a machine learning algorithm used for classification and regression tasks. It is an ensemble learning method that creates multiple decision trees and combines their outputs to make a final prediction. Random forest is a powerful algorithm that is commonly used for both classification and regression tasks, and is particularly well-suited for handling high-dimensional datasets with complex interactions between variables.

Random forest can be used to predict laptop prices based on multiple input features such as brand, model, display size, processor type, RAM, storage, etc. It works by constructing multiple decision trees using different subsets of the input features and data samples and then aggregating their predictions to get the final prediction. Random forest is known for its high accuracy and ability to handle non-linear relationships between the input features and target variable.

```
steprf1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

steprf2 = RandomForestRegressor(n_estimators=100,
                                random_state=3,
                                max_samples=0.5,
                                max_features=0.75,
                                max_depth=15)

pipe = Pipeline([
    ('step1',steprf1),
    ('step2',steprf2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

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

```
R2 score 0.8873402378382488
MAE 0.15860130110457718
```

The output suggests that the Random Forest model performs well on the laptop price prediction dataset. The R2 score of 0.8873 indicates that the model explains about 88.73% of the variation in the dependent variable, while the MAE of 0.1586 indicates that, on average, the predicted prices are within $0.16 of the true prices. Therefore, we can conclude that the Random Forest model is a good fit for this dataset and can accurately predict laptop prices.

Saving the Model:

```
import pickle

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

This code saves the df and pipe objects as pickle files with the names [df.pkl] and [pipe.pkl] respectively.

Pickle is a Python module that allows you to save and load Python objects in a binary format.

Saving these objects as pickle files allows you to reuse them later without having to recreate them from scratch, which can save time and computational resources.

```
df
```

	Company	TypeName	Ram	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Gpu brand	os
0	Apple	Ultrabook	8	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	Intel	Mac
1	Apple	Ultrabook	8	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	Intel	Mac
2	HP	Notebook	8	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256	Intel	Others/No OS/Linux
3	Apple	Ultrabook	16	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512	AMD	Mac
4	Apple	Ultrabook	8	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256	Intel	Mac
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	4	1.80	33992.6400	1	1	157.350512	Intel Core i7	0	128	Intel	Windows
1299	Lenovo	2 in 1 Convertible	16	1.30	79866.7200	1	1	276.053530	Intel Core i7	0	512	Intel	Windows
1300	Lenovo	Notebook	2	1.50	12201.1200	0	0	111.935204	Other Intel Processor	0	0	Intel	Windows
1301	HP	Notebook	6	2.19	40705.9200	0	0	100.454670	Intel Core i7	1000	0	AMD	Windows
1302	Asus	Notebook	4	2.20	19660.3200	0	0	100.454670	Other Intel Processor	500	0	Intel	Windows

1302 rows × 13 columns

Conclusion:

The data contains various types of features, including categorical and numerical features. The data was cleaned by removing the duplicates and filling in the missing values using mean or median of the respective columns.

Exploratory Data Analysis (EDA) revealed several insights, such as HP and Dell are the top two brands in terms of the number of laptops sold, and the average price of laptops with discrete graphics cards is higher than that of laptops without them.

Machine learning models were built using various algorithms such as linear regression, lasso regression, ridge regression, SVM, decision trees and random forest to predict the laptop prices.

Among all the models, the random forest model performed the best, with an R2 score of 0.887 and MAE of 0.159.

After performing various regression algorithms on the laptop price prediction dataset, we can conclude that Random Forest Regression algorithm gave the best results.

- The Random Forest Regression model has an R2 score of 0.8873 and MAE of 0.1586 which means the model can explain almost 88.7% of the variance in the target variable and the average difference between the actual and predicted values is 0.1586.
- After Random Forest, Decision Tree gave a good R2 score of 0.8433 and MAE of 0.1809, and Lasso Regression gave an R2 score of 0.8071 and MAE of 0.211.
- Ridge Regression gave a slightly better result than Lasso Regression with an R2 score of 0.8127 and MAE of 0.209.
- Linear Regression and SVM gave R2 score of 0.8073 and 0.8083 respectively.
- The Decision Tree and Random Forest models gave lower R2 scores compared to other models with R2 scores of 0.18 and 0.20 respectively.

On a conclusive note, the Random Forest Regression model could be used for predicting laptop prices, when given relevant features. The accuracy of the model could be improved by gathering more data and fine-tuning hyperparameters.

The insights derived from the machine learning models can help businesses in making data-driven decisions, such as deciding on the price of new laptops or optimizing the inventory levels of different brands and configurations of laptops.

Dataset Reference:

https://www.kaggle.com/datasets/mohidabdulrehman/laptop-price-dataset