



# LAPTOP PRICE PREDICTION

MACHINE LEARNING SE-204

BY:

LOVENDRA KUMAR 2K22/SE/108

LALIT KUMAR YADAV 2K22/SE/105

RAJAT 2K22/SE/133



# Outline of our Projects

1. Introduction
2. Motivation
3. Dataset
4. Research Methodology
5. Result
6. Limitation
7. Conclusion



# Introduction:

Our project aims to develop a machine learning model that can predict the prices of laptops based on their features and help consumers make more informed purchasing decisions and assist in pricing their products competitively.

# Motivation

**Consumer Empowerment:** Empowering consumers with transparent price information for

**Market Competitiveness:** Assisting businesses in competitive pricing strategies for laptops

**Technological Advancement:** Leveraging machine learning to elevate pricing accuracy and efficiency

## Why We have selected this Project.

We have selected this project because the laptop market is highly competitive and prices can vary depending on the brand, model, and specifications. By developing a machine learning model, we tool that can accurately predict laptop prices, taking into account various factors that affect pricing. ultimately benefit both consumers and businesses in the laptop industry.

# Problem Statements

## The Challenge

Laptop prices can vary widely widely depending on a complex complex set of factors, making it making it difficult for consumers consumers to determine a fair fair price. Our project seeks to to address this challenge by creating a predictive model that that can analyze these factors factors and provide accurate price estimates.

## The Goal

The goal of this project is to develop a machine learning model that can accurately predict the prices of laptops based on their brand, and other relevant features. This will empower consumers to make more informed purchasing and help businesses price products competitively.

## The Approach

We will use a dataset of specifications and prices to a machine learning model can identify the key drivers laptop prices. This model then be used to provide predictions for new laptop configurations, helping both consumers and businesses better-informed decisions.

# The Dataset

1

## Comprehensive

Our dataset includes detailed information on over 1,000 laptop models, covering a wide range of brands, specifications, and price points.

2

## Diverse

The dataset includes laptops targeting different market segments, from entry-level to level to high-end gaming and professional professional models.

3

## Reliable

The data has been carefully curated from reputable online sources and cross- for accuracy and consistency.

4

## Relevant

The dataset includes all the key features and specifications that are known to influence laptop prices, such as processor, RAM, storage, display, and more.



## Our dataset:

1	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
2		13.3IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
3		13.3 1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
4		15.6 Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0
5		15.4IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.336
6		13.3IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.808
7		15.6 1366x768	AMD A9-Series 9420 3GHz	4GB	500GB HDD	AMD Radeon R5	Windows 10	2.1kg	21312.0
8		15.4IPS Panel Retina Display 2880x1800	Intel Core i7 2.2GHz	16GB	256GB Flash Storage	Intel Iris Pro Graphics	Mac OS X	2.04kg	114017.6016
9		13.3 1440x900	Intel Core i5 1.8GHz	8GB	256GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	61735.536
10		14.0 Full HD 1920x1080	Intel Core i7 8550U 1.8GHz	16GB	512GB SSD	Nvidia GeForce MX150	Windows 10	1.3kg	79653.6
11		14.0IPS Panel Full HD 1920x1080	Intel Core i5 8250U 1.6GHz	8GB	256GB SSD	Intel UHD Graphics 620	Windows 10	1.6kg	41025.6
12		15.6 1366x768	Intel Core i5 7200U 2.5GHz	4GB	500GB HDD	Intel HD Graphics 620	No OS	1.86kg	20986.992
13		15.6 Full HD 1920x1080	Intel Core i3 6006U 2GHz	4GB	500GB HDD	Intel HD Graphics 520	No OS	1.86kg	18381.0672
14		15.4IPS Panel Retina Display 2880x1800	Intel Core i7 2.8GHz	16GB	256GB SSD	AMD Radeon Pro 555	macOS	1.83kg	130001.6016
15		15.6 Full HD 1920x1080	Intel Core i3 6006U 2GHz	4GB	256GB SSD	AMD Radeon R5 M430	Windows 10	2.2kg	26581.392
16		12.0IPS Panel Retina Display 2304x1440	Intel Core M m3 1.2GHz	8GB	256GB SSD	Intel HD Graphics 615	macOS	0.92kg	67260.672
17		13.3IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	256GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	80908.344
18		15.6 Full HD 1920x1080	Intel Core i7 7500U 2.7GHz	8GB	256GB SSD	AMD Radeon R5 M430	Windows 10	2.2kg	39693.6
19		15.4IPS Panel Retina Display 2880x1800	Intel Core i7 2.8GHz	16GB	512GB SSD	AMD Radeon Pro 560	macOS	1.83kg	152274.24

link for full dataset:

[Click here](#)

# Preprocessing

```
[21]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[2]: df = pd.read_csv('laptop_data.csv')

[3]: df.head()

[3]:
```

	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

Check for size of dataset and datatype of independent variable

```
[4]: df.shape

[4]: (1303, 12)

[5]: df.info()
```

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



Check for duplicate and null rows

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

[6]: 0

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

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

Drop Unnamed column

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

[9]: 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	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
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	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	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	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

Remove GB from Ram and Kg from weight

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

Changing the datatype for Ram and Weight to int

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

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Company               1303 non-null  object
1   TypeName              1303 non-null  object
2   Inches               1303 non-null  float64
3   ScreenResolution     1303 non-null  object
4   Cpu                  1303 non-null  object
5   Ram                  1303 non-null  int32
6   Memory              1303 non-null  object
7   Gpu                  1303 non-null  object
8   OpSys                1303 non-null  object
9   Weight               1303 non-null  float32
10  Price                1303 non-null  float64
dtypes: float32(1), float64(2), int32(1), object(7)
memory usage: 101.9+ KB
```

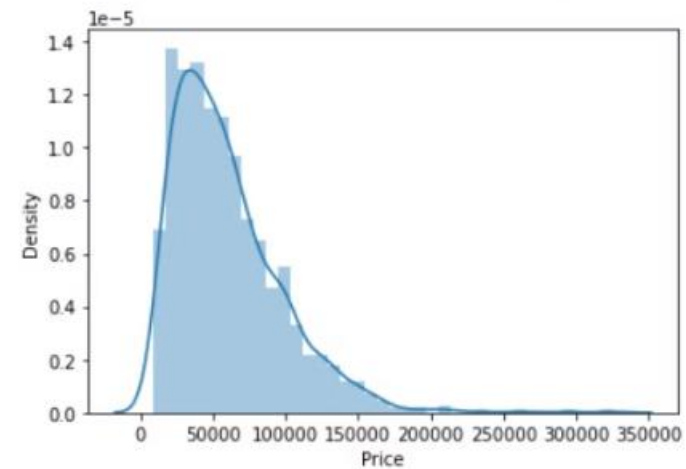
# Feature Selection

Importing seaborn for statistical graphics

```
[17]: import seaborn as sns
```

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

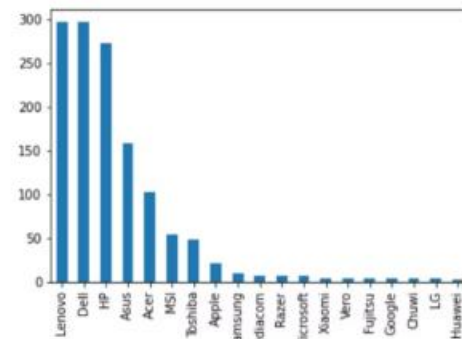
```
[18]: <AxesSubplot:xlabel='Price', ylabel='Density'>
```



- no. of laptops for different company

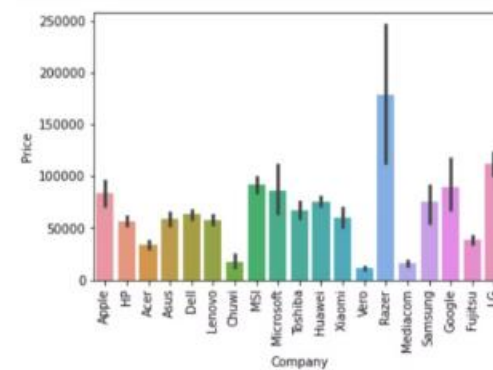
```
[19]: df['Company'].value_counts().plot(kind='bar')
```

```
[19]: <AxesSubplot:>
```

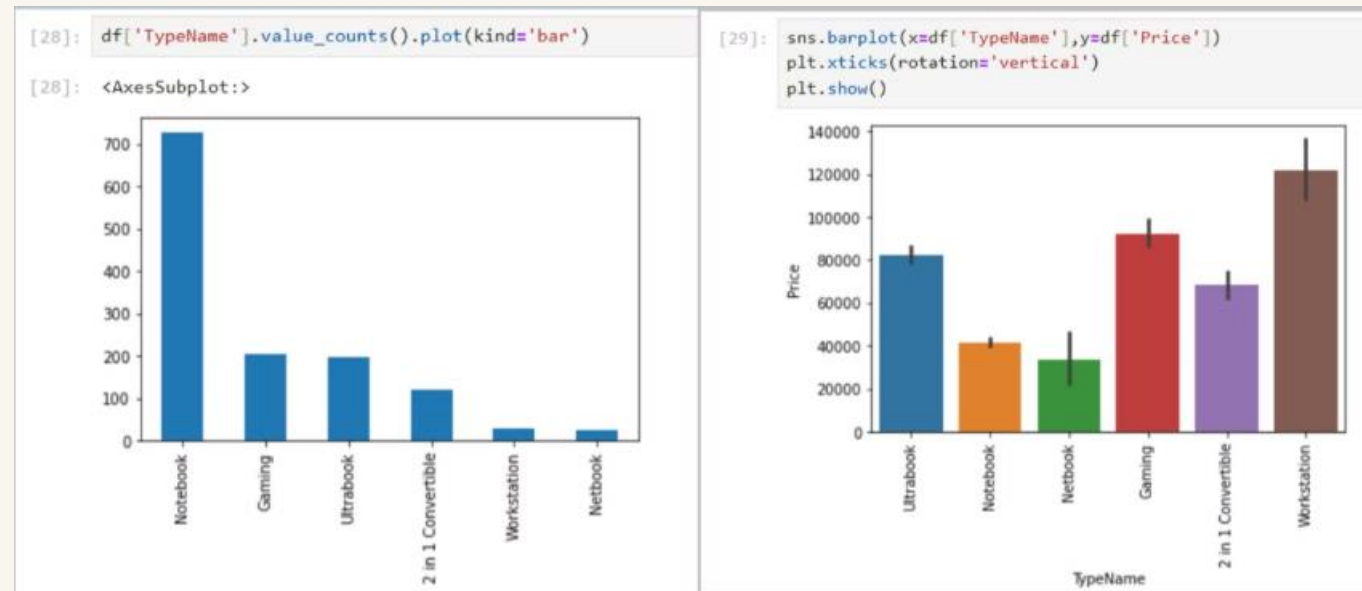


- price for corresponding company

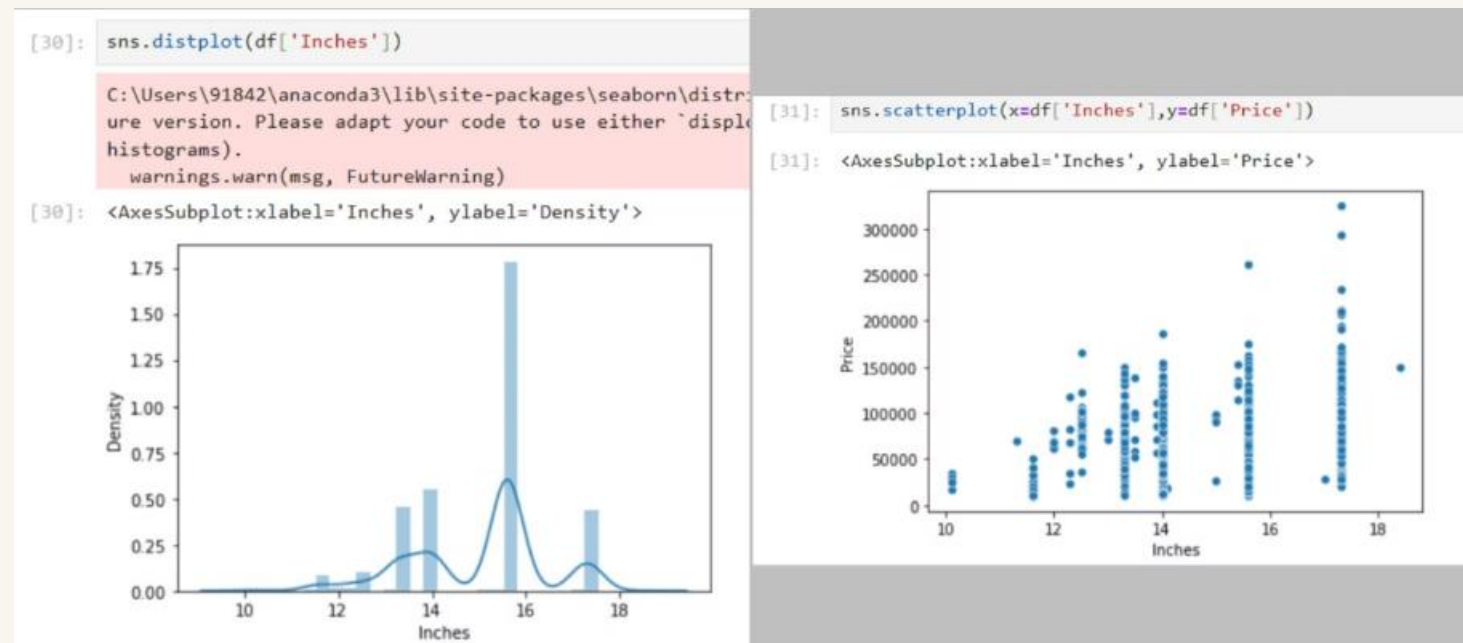
```
[27]: sns.barplot(x=df['Company'], y=df['Price'])  
plt.xticks(rotation='vertical')  
plt.show()
```



-no. of laptops for types of series of laptops    -price for corresponding series of laptops



-density for corresponding laptop inches    -price for different laptop size in inches



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

[32]: 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
      Touchscreen 2560x1440          7
      IPS Panel 1366x768          7
      4K Ultra HD 3840x2160          7
      IPS Panel Quad HD+ / Touchscreen 3200x1800      6
      Touchscreen 2256x1504          6
      IPS Panel Retina Display 2304x1440      6
      IPS Panel Retina Display 2560x1600      6
      IPS Panel Touchscreen 2560x1440      5
      IPS Panel 2560x1440          4
      IPS Panel Retina Display 2880x1800      4
      IPS Panel Touchscreen 1920x1200      4
      1440x900              4
      Quad HD+ 3200x1800              3
      IPS Panel Quad HD+ 2560x1440      3
      1920x1080              3
      Touchscreen 2400x1600          3
      IPS Panel Touchscreen 1366x768          3
      2560x1440              3
      IPS Panel Full HD 2160x1440          2
      IPS Panel Touchscreen / 4K Ultra HD 3840x2160      2
      IPS Panel Quad HD+ 3200x1800          2
      Touchscreen / Full HD 1920x1080          1
      IPS Panel Retina Display 2736x1824          1
      IPS Panel Full HD 1920x1200          1
      IPS Panel Full HD 1366x768          1
      Touchscreen / 4K Ultra HD 3840x2160          1
      IPS Panel Touchscreen 2400x1600          1
      IPS Panel Full HD 2560x1440          1
      Touchscreen / Quad HD+ 3200x1800          1
      Name: ScreenResolution, dtype: int64
```

```
]:
```

```
# Create the boxplot
plt.figure(figsize=(12, 8)) # Set the figure size
sns.boxplot(x='Company', y='Price', data=data)
plt.title('Price Distribution by Company')
plt.xlabel('Company')
plt.ylabel('Price')
plt.xticks(rotation=45) # Rotate x-axis labels for better readability
plt.tight_layout() # Adjust layout to prevent clipping of labels
plt.show()
```

Adding new column named Touchscreen

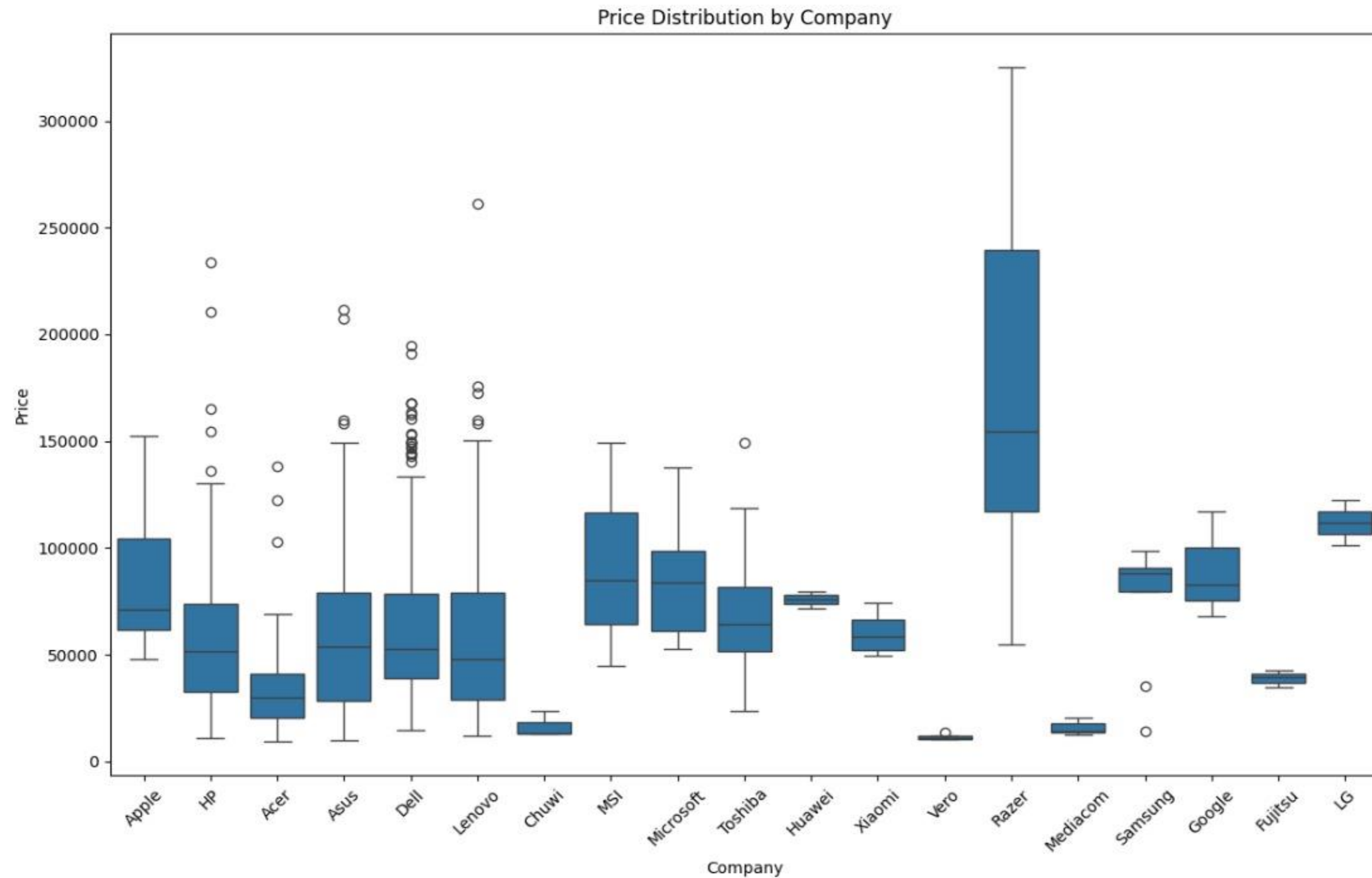
```
[34]: df['Touchscreen'] = df['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in x else 0)

[37]: df.sample(5)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen
1154	Dell	Notebook	15.6	IPS Panel Touchscreen / 4K Ultra HD 3840x2160	Intel Core i5 6300HQ 2.3GHz	8	256GB SSD	Nvidia GeForce 960M	Windows 10	2.04	119916.2304	1
750	Lenovo	Netbook	11.6	Touchscreen 1366x768	Intel Celeron Dual Core N3060 1.6GHz	4	128GB SSD	Intel HD Graphics 400	Windows 10	1.40	25308.0000	1
1246	Dell	Notebook	14.0	1366x768	Intel Core i5 7200U 2.5GHz	4	500GB HDD	Intel HD Graphics 620	Windows 10	1.60	46620.0000	0
879	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	4	256GB SSD	Intel HD Graphics 620	Windows 10	2.04	44701.9200	0
1021	Toshiba	Ultrabook	13.3	Full HD 1920x1080	Intel Core i5 6200U 2.3GHz	8	256GB SSD	Intel HD Graphics 520	Windows 10	1.20	84715.2000	0



# BOXPLOT OF OUR DATA





Adding new column named IPS for laptop panels and

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

[41]: 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

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

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

[50]: df.sample(5)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	X_res	Y_res
141	Lenovo	Notebook	14.0	IPS Panel Full HD 1920x1080	Intel Core i5 8250U 1.6GHz	8	256GB SSD	AMD Radeon RX 550	Windows 10	1.75	59461.5456	0	1	IPS Panel Full HD 1920	1080
1055	HP	Notebook	15.6	1366x768	Intel Core i3 6100U 2.3GHz	4	500GB HDD	Intel HD Graphics 520	Windows 10	2.31	37570.3920	0	0	1366	768
75	Asus	Gaming	15.6	Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	8	1TB HDD	Nvidia GeForce GTX 1050	Windows 10	2.20	50562.7200	0	0	Full HD 1920	1080
984	Toshiba	Notebook	14.0	1366x768	Intel Core i5 6200U 2.3GHz	4	500GB HDD	Intel HD Graphics 520	Windows 10	1.75	48751.2000	0	0	1366	768
337	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	1.84	60952.3200	0	0	Full HD 1920	1080

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

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

```
[60]:
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	X_res	Y_res
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	2560	1600
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	1440	900
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	1920	1080
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	2880	1800
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	2560	1600

changing the datatype for X\_res and Y\_res

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

```
[63]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 15 columns):
#   Column              Non-Null Count  Dtype
---  -
0   Company              1303 non-null   object
1   TypeName             1303 non-null   object
2   Inches               1303 non-null   float64
3   ScreenResolution     1303 non-null   object
4   Cpu                  1303 non-null   object
5   Ram                  1303 non-null   int32
6   Memory               1303 non-null   object
7   Gpu                  1303 non-null   object
8   OpSys                1303 non-null   object
9   Weight               1303 non-null   float32
10  Price                1303 non-null   float64
11  Touchscreen          1303 non-null   int64
12  Ips                  1303 non-null   int64
13  X_res                1303 non-null   int32
14  Y_res                1303 non-null   int32
dtypes: float32(1), float64(2), int32(3), int64(2), object(7)
memory usage: 132.5+ KB
```

Co-relation of X\_res and Y\_res on price and making a new column for ppi(pixel per inch) using X\_res and Y\_res

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

```
[65]:
```

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	

Drop Screen resolution , Inches, X\_res and Y\_res

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

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

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

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005

Now check for CPU

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

```
[74]: 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 Celeron Quad Core N3710 1.6GHz    1
Intel Core i5 7200U 2.7GHz             1
Intel Pentium Dual Core N4200 1.1GHz     1
AMD FX 8800P 2.1GHz                    1
Intel Atom x5-Z8300 1.44GHz             1
Name: Cpu, Length: 118, dtype: int64
```

Making a new column for CPU Name

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

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

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu Name
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

Adding CPU brand column using CPU Name

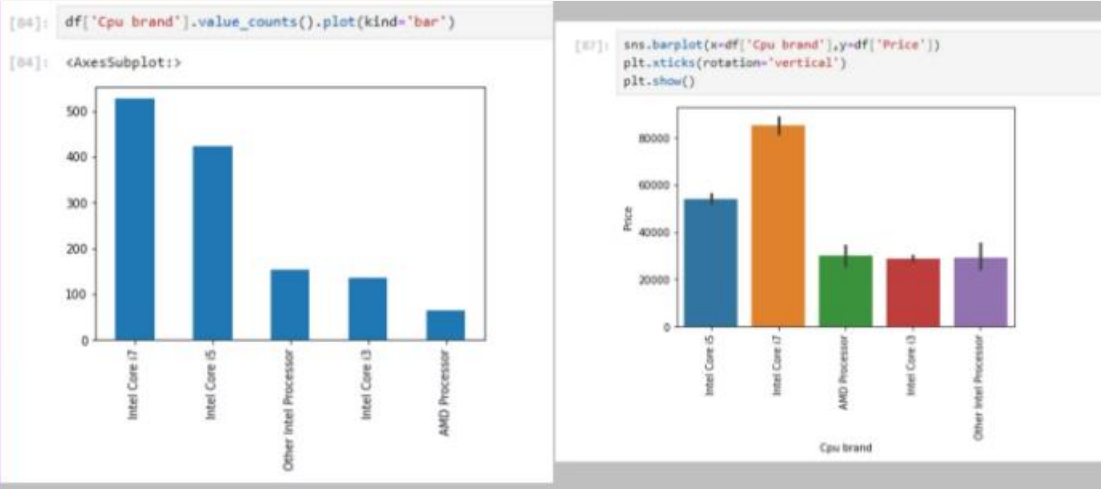
```
[81]: def fetch_processor(text):
      if text == 'Intel Core i7' or text == 'Intel Core i5' or text == 'Intel Core i3':
          return text
      else:
          if text.split()[0] == 'Intel':
              return 'Other Intel Processor'
          else:
              return 'AMD Processor'

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

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

```
[83]:
```

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu Name	Cpu brand
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	Intel Core i5
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	Intel Core i5
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	Intel Core i5
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	Intel Core i7
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	Intel Core i5



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

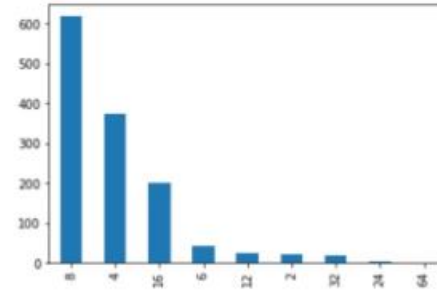
[89]: df.head()
```

```
[89]:
```

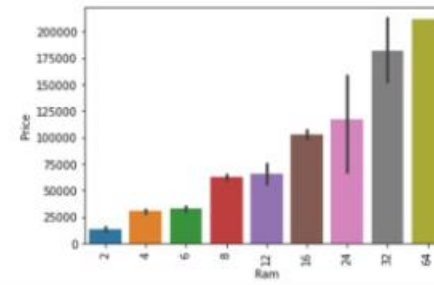
	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5

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

```
[90]: <AxesSubplot:>
```



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



Check for memory

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

```
[92]: 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
      16GB SSD            3
      512GB SSD + 2TB HDD  3
      256GB SSD + 256GB SSD  2
      128GB SSD + 2TB HDD  2
      256GB SSD + 500GB HDD  2
      512GB Flash Storage  2
      1TB SSD + 1TB HDD    2
      32GB HDD            1
      64GB SSD            1
      1.0TB HDD           1
      512GB SSD + 256GB SSD  1
      512GB SSD + 1.0TB Hybrid  1
      8GB SSD             1
      240GB SSD           1
      128GB HDD           1
      1TB HDD + 1TB HDD    1
      512GB SSD + 512GB SSD  1
      256GB SSD + 1.0TB Hybrid  1
      500GB Hybrid         1
      64GB Flash Storage + 1TB HDD  1
      Name: Memory, dtype: int64
```

```
[93]: 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()
```



```
[98]: df.sample(5)
```

```
[98]:
```

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Hybrid	Flash_Storage
1247	Asus	Gaming	16	256 SSD + 1000 HDD	Nvidia GeForce GTX 1070	Windows 10	2.34	123876.000	0	1	141.211998	Intel Core i7	1000	256	0	0
505	Lenovo	Notebook	8	256 SSD	Intel HD Graphics 620	Windows 10	1.44	50562.720	0	0	165.632118	Intel Core i5	0	256	0	0
820	Lenovo	Notebook	4	500 HDD	Intel HD Graphics 520	Windows 10	2.10	26101.872	0	0	100.454670	Intel Core i3	500	0	0	0
21	Lenovo	Gaming	8	128 SSD + 1000 HDD	Nvidia GeForce GTX 1050	Windows 10	2.50	53226.720	0	1	141.211998	Intel Core i5	1000	128	0	0
301	Asus	Gaming	16	256 SSD + 1000 HDD	Nvidia GeForce GTX 1070	Windows 10	2.90	113060.160	0	0	127.335675	Intel Core i7	1000	256	0	0

### Drop Memory Column

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

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

```
[100]:
```

	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

### Co-relation of variables with price

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

```
[101]:
```

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



Drop hybrid and flash storage(as their change does not effect price that enough)

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

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

```
[103]:
```

	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

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

```
[104]: 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
...
Intel HD Graphics 540        1
AMD FirePro W6150M          1
AMD Radeon R5 M315          1
AMD Radeon R7 M360          1
AMD FirePro W5130M          1
Name: Gpu, Length: 110, dtype: int64
```

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

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

```
[107]:
```

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

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

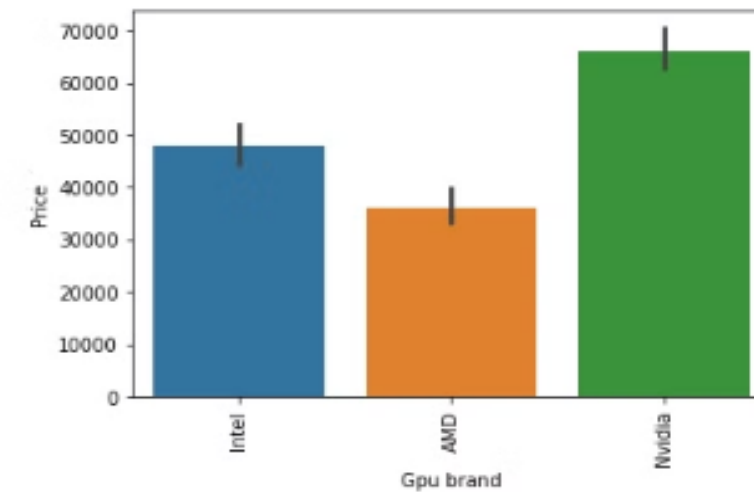
```
[108]: Intel      722  
      Nvidia    400  
      AMD       180  
      ARM         1  
      Name: Gpu brand, dtype: int64
```

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

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

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

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

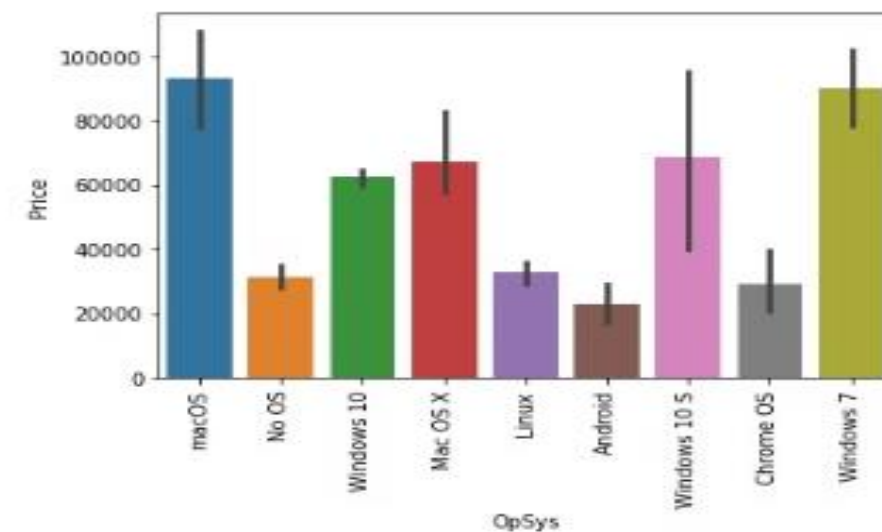


Check for operating system

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

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

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



```
[121]: 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'
```

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

<ipython-input-122-38671a3c07bd>:1: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

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

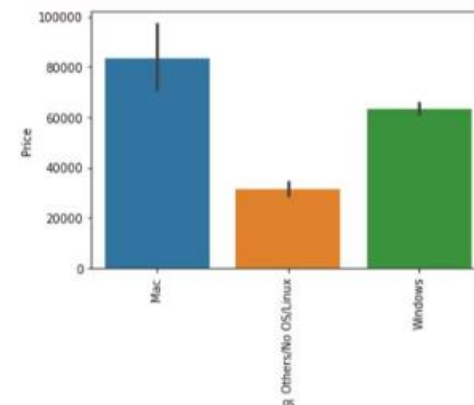
```
[123]: df.head()
```

```
[123]:
```

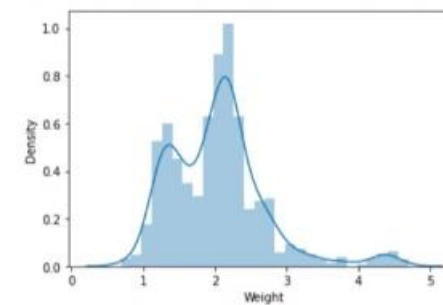
	Company	TypeName	Ram	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Gpu brand	os
0	Apple	Ultrabook	8	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	Intel	Mac
1	Apple	Ultrabook	8	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	Intel	Mac
2	HP	Notebook	8	No OS	1.86	30636.0000	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

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

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



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



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

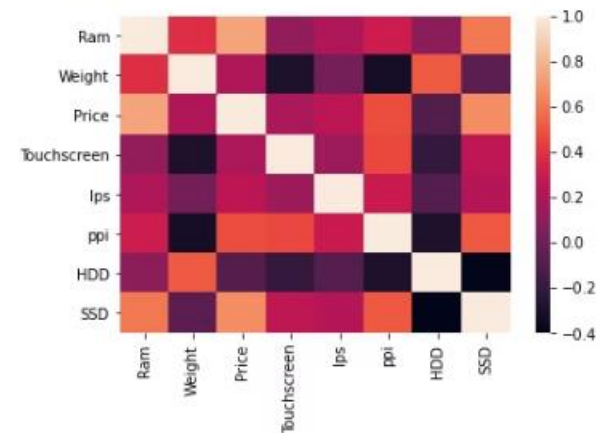
```
[127]: <AxesSubplot:xlabel='Weight', ylabel='Price'>
```

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

```
[128]: 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
```

```
[130]: sns.heatmap(df.corr())
```

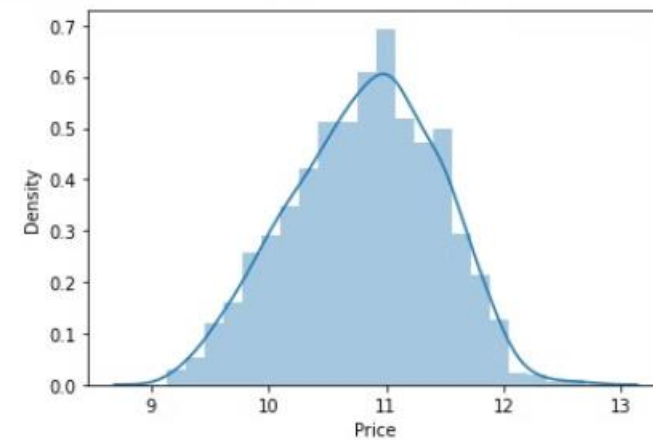
```
[130]: <AxesSubplot:>
```



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

```
C:\Users\91842\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: Future version. Please adapt your code to use either 'displot' (a figure-level function for histograms).  
warnings.warn(msg, FutureWarning)
```

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



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

```
[135]:
```

X

```
[135]:
```

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

1302 rows × 12 columns

```
[136]:
```

y

```
[136]:
```

```
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
```

```
[137]: 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)
```

```
[138]:
```

X\_train

```
[138]:
```

	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



# Performance Analysis

## Techniques

```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegressor
```

### ▼ Linear regression

```
[145]: 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 0.8073277448418521

MAE 0.21017827976429174



## KNN

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

step2 = KNeighborsRegressor(n_neighbors=3)

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 0.8021984604448553
MAE 0.19319716721521116
```

## Random Forest

```
[306]: 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))

R2 score 0.8873402378382488
MAE 0.15860130110457718
```

## Decision Tree

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

step2 = DecisionTreeRegressor(max_depth=8)

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

R2 score 0.8466456692979233
MAE 0.1806340977609143
```

## AdaBoost

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

step2 = AdaBoostRegressor(n_estimators=15, learning_rate=1.0)

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 0.7929652659237908
MAE 0.23296532406396742
```

## SVM

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

step2 = SVR(kernel='rbf', C=10000, epsilon=0.1)

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

R2 score 0.8083180902257614
MAE 0.20239059427481307
```

## XgBoost

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

step2 = XGBRegressor(n_estimators=45, max_depth=5, learning_rate=0.5)

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

R2 score 0.8811773435850243
MAE 0.16496203512600974
```

## Stacking

```
] from sklearn.ensemble import VotingRegressor, StackingRegressor

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

estimators = [
    ('rf', RandomForestRegressor(n_estimators=350, random_state=3, max_samples=0.5, max_features=0.75, max_depth=15)),
    ('gbdt', GradientBoostingRegressor(n_estimators=100, max_features=0.5)),
    ('xgb', XGBRegressor(n_estimators=25, learning_rate=0.3, max_depth=5))
]

step2 = StackingRegressor(estimators=estimators, final_estimator=Ridge(alpha=100))

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

R2 score 0.8816958647512341
MAE 0.1663048975120589
```

## Gradient Boost

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

step2 = GradientBoostingRegressor(n_estimators=500)

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

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

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

R2 score 0.8823244736036472
MAE 0.15929506744611283
```





# Results and Performance

1

## Model Training

We trained a variety of machine learning models, including linear regression, regression, decision trees, and random forests, to predict laptop prices based on the based on the dataset.

2

## Model Evaluation

The models were evaluated using common performance metrics such as root mean squared error (RMSE), and mean absolute error (MAE).

3

## Final Model

The random forest model demonstrated the best overall performance, with an R- with an R-squared of 0.91 and a RMSE of \$120, making it a highly accurate predictor accurate predictor of laptop prices.

## Performance of our Model with all Algorithm:

### 1. Linear Regression

R2 Score: 0.8073277448418

MAE: 0.21017827976429

### 2.. K N N

R2 Score: 0.8021984604448

MAE: 0.193197167215211

### 3. Decision Tree

R2 Score: 0.846645669297

MAE: 0.18063409776091

### 4.Random Forest

R2 Score: 0.887340237838

MAE: 0.15860130110457

### 5. Gradient boost

R2 Score: 0.8823244736036

MAE: 0.15929506744611

### 6. Ada boost

R2 Score: 0.7929652659237

MAE: 0.23296532406396

### 7. SVM

R2 Score:0.80831

MAE: 0.202390

### 8. XG Boost

R2score: 0.8811

MAE : 0.1649

### 9. Stacking Regressor

R2 Score:0.881695

MAE: 0.166304

Random Forest have the best result with accuracy of 88.23%. So Random Forest Algo will be Used for our Model.  
for our Model.

# Limitations and Future Improvements

## Limited Data Scope

Our dataset, while comprehensive, may not capture the full diversity of the laptop market, particularly for newer or more specialized models.

## Changing Market Conditions

Laptop prices and features can evolve rapidly, so the model's accuracy may degrade over time without regular retraining and updates.

## Unexplained Factors

There may be additional factors, such as brand reputation or customer reviews, that influence laptop prices but are not captured in the current dataset.

## Future Enhancements

Future work could include expanding the dataset to include real-time market data, and exploring more advanced machine learning techniques to further improve the model's accuracy and robustness.

# Conclusion

1

## Accurate Predictions

Our machine learning model has demonstrated the ability to accurately predict laptop prices based on their key features and specifications.

2

## Empowered

This tool can help consumers make more informed purchasing decisions by providing reliable price estimates for different laptop configurations.

3

## Competitive Pricing

Businesses can also leverage leverage the model to price their laptop offerings more competitively and better understand market trends.

Overall, this project has the potential to transform the way consumers and businesses approach leading to more informed decisions and a more efficient laptop market.

Comparing all the algo, Random Forest with accuracy of 88.23% will be best fit for our Model.



# References:

1. Books
2. Youtube
3. Github
4. kaggle,.com
5. Wikipedia
6. Geeks for geeks