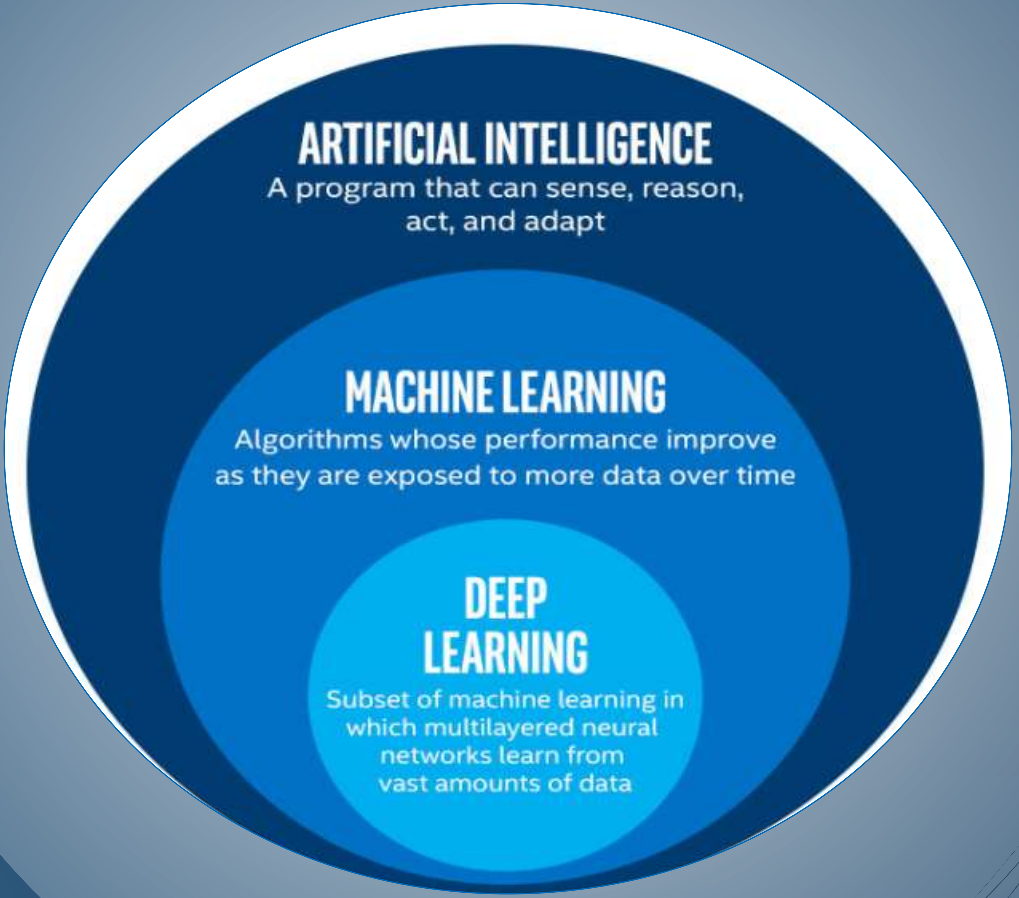
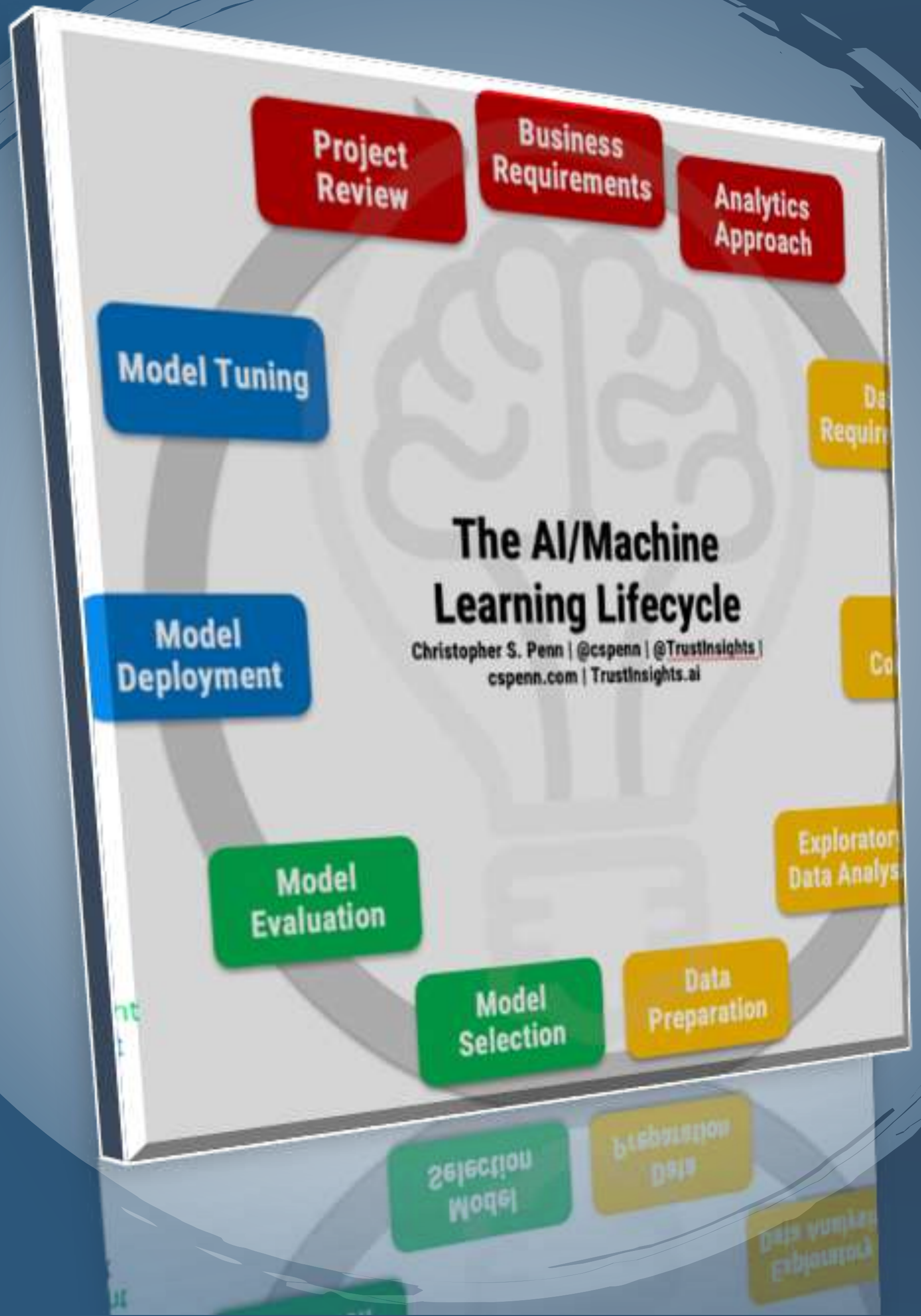


LAPTOP PRICE PREDICTION

FOR SMART TECH CO



WHAT IS MACHINE LEARNING

MACHINE LEARNING IS A BRANCH OF ARTIFICIAL INTELLIGENCE THAT DEALS WITH IMPLEMENTING APPLICATIONS THAT CAN MAKE A FUTURE PREDICTION BASED ON PAST DATA.

THERE ARE VARIOUS STEPS INVOLVED IN BUILDING A MACHINE LEARNING PROJECT BUT NOT ALL THE STEPS ARE MANDATORY TO USE IN A SINGLE PROJECT, AND IT ALL DEPENDS ON THE DATA.

- **WHY WE NEED LAPTOP PRIDICTION MODEL ?**

- Consumers often face the challenge of navigating through various options to find a laptop that meets their requirements and budget constraints. Additionally, market fluctuations and the rapid pace of technological advancements contribute to the complexity of understanding and predicting laptop prices.

LAPTOP PREDICTION MODEL

Contoso

Project Overview:

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

Table of Contents

Problem Statement

The problem statement is that if any user wants to buy a laptop then our application should be compatible to provide a tentative price of laptop according to the user configurations.

SOLUTION

WE WILL MAKE A PROJECT FOR LAPTOP

PRICE PREDICTION.

BASIC UNDERSTANDING OF LAPTOP PRICE PREDICTION DATA

LAPTOP PRICE PRICTION MODEL

```
# importing libraries
```

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')
```

[**df.shape**] # Checking no of rows and no columns.

In this dataset we have; ### ROWS = 1308 ### COLUMNS = 13

```
##Importing files
```

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

```
# Understanding the dataset.
```

```
df.head()
```

	Unnamed: 0.1	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	0	0.0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	1	1.0	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232

Descriptive statistical analysis.
df.describe(include='all')

```
## Descriptive statistical analysis.
df.describe(include='all')
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
count	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273.000000
unique	19	6	25	40	118	10	40	110	9	189	NaN
top	Lenovo	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	Windows 10	2.2kg	NaN
freq	290	710	640	495	183	601	401	271	1047	111	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	59955.814073
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	37332.251005
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9270.720000
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	31914.720000
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	52161.120000
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	79333.387200
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	324954.720000

df.info()

```
df.info()

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

Checking no of rows and no columns.
[**df.shape**]

In this dataset we have;
ROWS = 1308
COLUMNS = 13

# remove nan values
df.dropna(inplace=True)

```
[177]: # remove nan values
       df.dropna(inplace=True)

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

[178]: Unnamed: 0.1        0
       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 cleaning & Feature Engineering

```python
## Drop unwanted columns from the dataframe
df.drop(columns=['Unnamed: 0.1','Unnamed: 0'],axis=1,inplace=True)
```

```python
## Now, check the dataframe again
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

Drop unwanted columns from the dataframe df.drop(columns=['Unnamed: 0.1','Unnamed: 0'],axis=1,inplace=True)

Now, check the data frame again df.head()

Now, you can se that ram is object, we will change it to **[int].**
Weight is also object i will change it to **[float].**
Change price dtpe into **[float].**

```
# Replace the 'empty' string and '?' with NaN and then drop rows with NaN in 'Ram' and 'Weight' columns
df.replace(['', '?'], np.nan, inplace=True)
df.dropna(subset=['Ram', 'Weight'], inplace=True)

# Now convert to integer and float
df['Ram'] = df['Ram'].str.replace('GB','').astype('int32')
df['Weight'] = df['Weight'].str.replace('kg', '').astype('float32')
```

```
# Remove non-numeric characters from 'Inches' column
df['Inches']=df['Inches'].replace('?',float('nan'))

# convert 'Inches' column to float
df['Inches']=df['Inches'].astype(float)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1272 entries, 0 to 1302
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1272 non-null   object
 1   TypeName          1272 non-null   object
 2   Inches            1271 non-null   float64
 3   ScreenResolution  1272 non-null   object
 4   Cpu               1272 non-null   object
 5   Ram               1272 non-null   int32
```

```
[183]: print(df['Ram'].dtype)
object
[184]: print(df['Weight'].dtype)
object
[185]: print(df['Price'].dtype)
float64
```

df['Company'].value_counts()

```
[191]: df['Company'].value_counts()

[191]: Company
       Lenovo       290
       Dell         286
       HP           266
       Asus         156
       Acer         103
       MSI           53
       Toshiba       47
       Apple         21
       Samsung        9
       Razer          7
       Mediacom       7
       Microsoft      6
       Xiaomi         4
       Vero           4
       Chuwi          3
       Google         3
       LG             3
       Huawei         2
       Fujitsu        2
       Name: count, dtype: int64
```

df['TypeName'].value_count

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

TypeName
Notebook              710
Gaming                203
Ultrabook             190
2 in 1 Convertible    116
Workstation            29
Netbook                24
Name: count, dtype: int64
```

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

ScreenResolution
Full HD 1920x1080                                494
1366x768                                         274
IPS Panel Full HD 1920x1080                      226
IPS Panel Full HD / Touchscreen 1920x1080         52
Full HD / Touchscreen 1920x1080                   45
1600x900                                          23
Touchscreen 1366x768                              16
Quad HD+ / Touchscreen 3200x1800                  14
IPS Panel 4K Ultra HD 3840x2160                   12
IPS Panel 4K Ultra HD / Touchscreen 3840x2160     11
4K Ultra HD / Touchscreen 3840x2160                9
4K Ultra HD 3840x2160                              7
IPS Panel 1366x768                                 7
IPS Panel Retina Display 2560x1600                 6
IPS Panel Quad HD+ / Touchscreen 3200x1800         6
Touchscreen 2560x1440                              6
IPS Panel Retina Display 2304x1440                 6
```

df['Screen Resolution'].value_counts()

Checking data type after dtype modification.

```
## Checking data type after dtype modification.

df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 1272 entries, 0 to 1302
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1272 non-null   object
 1   TypeName          1272 non-null   object
 2   Inches            1271 non-null   float64
 3   ScreenResolution  1272 non-null   object
 4   Cpu               1272 non-null   object
 5   Ram               1272 non-null   int32
 6   Memory            1271 non-null   object
 7   Gpu               1272 non-null   object
 8   OpSys             1272 non-null   object
 9   Weight            1272 non-null   float32
 10  Price             1272 non-null   float64
dtypes: float32(1), float64(2), int32(1), object(7)
memory usage: 109.3+ KB
```

Feature Engineering over screen resolution column

There are some hidden specifications in Screen resolution column which we have to find out and create a separate column for it.

IPS PANEL RETINA SISPLAY

df['IPS'] = df['Screen Resolution'].apply(lambda x: 1 if 'IPS' in x else 0)

TOUCH SCREEN

df['Screen Resolution'].apply(lambda x:1 if 'Touchscreen' in x else 0)

df.sample(10)

[200]:
```
df.sample(10)
```

[200]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	IPS	Touchscreen
635	Asus	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7300HQ 2.5GHz	8	1TB HDD	Nvidia GeForce GTX 1050	Windows 10	1.99	48304.7136	0	0
1057	Acer	Notebook	15.6	1366x768	AMD A8-Series 7410 2.2GHz	8	1TB HDD	AMD Radeon R5	Windows 10	2.40	23922.7200	0	0
1260	Lenovo	2 in 1 Convertible	14.0	Full HD / Touchscreen 1920x1080	Intel Core i5 6200U 2.3GHz	4	128GB SSD	Intel HD Graphics 520	Windows 10	1.80	44382.7728	0	1
1297	Asus	Notebook	15.6	1366x768	Intel Core i7 6500U 2.5GHz	4	500GB HDD	Nvidia GeForce 920M	Windows 10	2.20	38378.6496	0	0
1109	Asus	Gaming	15.6	IPS Panel Full HD 1920x1080	Intel Core i7 6700HQ 2.6GHz	16	128GB SSD + 1TB HDD	Nvidia GeForce GTX 960M	Windows 10	2.59	71341.9200	1	0
1249	Dell	2 in 1 Convertible	13.3	Quad HD+ / Touchscreen 3200x1800	Intel Core i5 7Y54 1.2GHz	8	256GB SSD	Intel HD Graphics 615	Windows 10	1.24	96596.6400	0	1
164	Acer	Notebook	15.6	1366x768	Intel Celeron Dual Core N3350 1.1GHz	4	1TB HDD	Intel HD Graphics 500	Windows 10	2.10	18541.4400	0	0
578	MSI	Gaming	17.3	Full HD 1920x1080	Intel Core i7 7820HK 2.9GHz	16	512GB SSD + 1TB HDD	Nvidia GeForce GTX	Windows 10	4.14	145401.1200	0	0

Create two new columns :

```
df['Screen Resolution'].str.split('x',n=1,expand=True)
##Creating two new columns..x_res,y_res

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

Screen Resolution

We split it into X_res and Y_res

X_res

X resolution

Y_res

Y resolution

df.head()

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

```
df.head()
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	IPS	Touchscreen	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	1	0	IPS Panel Retina Display 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	Full HD 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	1	0	IPS Panel Retina Display 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	1	0	IPS Panel Retina Display 2560	1600

we have to change X-res.## with the help of regular expression we will extract required info from X_res.

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

```
## we have to change X-res.
## with the help of regular expression we will extract required info from X_res.

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

0       2560
1       1440
2       1920
3       2880
4       2560
        ...
1298    1920
1299    3200
1300    1366
1301    1366
1302    1366
Name: X_res, Length: 1272, dtype: object

## We successfully extracted the values.
```

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

df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 1272 entries, 0 to 1302
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1272 non-null   object
 1   TypeName          1272 non-null   object
 2   Inches            1271 non-null   float64
 3   ScreenResolution  1272 non-null   object
 4   Cpu               1272 non-null   object
 5   Ram               1272 non-null   int32
 6   Memory            1271 non-null   object
 7   Gpu               1272 non-null   object
 8   OpSys             1272 non-null   object
 9   Weight            1272 non-null   float32
 10  Price             1272 non-null   float64
 11  IPS               1272 non-null   int64
 12  Touchscreen       1272 non-null   int64
 13  X_res             1272 non-null   object
 14  Y_res             1272 non-null   object
dtypes: float32(1), float64(2), int32(1), int64(2), object(9)
memory usage: 149.1+ KB
```

Converting X_res and Y_res into int dtype.

```
## Converting X_res and Y_res into int dtype.

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

df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 1272 entries, 0 to 1302
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1272 non-null   object
 1   TypeName          1272 non-null   object
 2   Inches            1271 non-null   float64
 3   ScreenResolution  1272 non-null   object
 4   Cpu               1272 non-null   object
 5   Ram               1272 non-null   int32
 6   Memory            1271 non-null   object
 7   Gpu               1272 non-null   object
 8   OpSys             1272 non-null   object
 9   Weight            1272 non-null   float32
 10  Price             1272 non-null   float64
 11  IPS               1272 non-null   int64
 12  Touchscreen       1272 non-null   int64
 13  X_res             1272 non-null   int32
 14  Y_res             1272 non-null   int32
dtypes: float32(1), float64(2), int32(3), int64(2), object
```

Now we will create a new column i.e, PPI

What is pixel per inch (PPI)?# The term Pixels Per Inch (PPI) commonly refers to the measurement of **pixel density in display screens, including those of computers, laptops, TVs, and smartphones.** # This metric helps you determine the sharpness and clarity of the image you see on the screens of these devices. # PPI is a metric used for all kinds of screens

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

```
# Select only numeric columns for correlation calculation
numeric_df = df.select_dtypes(include=['number'])

# Calculate correlations with 'Price'
correlations = numeric_df.corr()['Price']


print(correlations)
```

```
Inches         0.045042
Ram            0.685737
Weight         0.175928
Price          1.000000
IPS            0.255140
Touchscreen    0.189172
X_res          0.557584
Y_res          0.554104
ppi            0.469329
Name: Price, dtype: float64
```

We check correlation to understand more about numeric columns and which are least important.

Now we will remove some columns which are of no use in future.

we will drop Inches,ScreenResolution,X_res,Y_res.

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

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

```
df.head()
```

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	IPS	Touchscreen	ppi
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	1	0	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	1	0	220.534624
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	1	0	226.983005

Now we will focus on cpu and gpu

cpu

df['Cpu'].value_counts()

df['Cpu'].apply(lambda x:x.split()[0:3])

I will create a new column [Cpu Name]

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

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

```
Cpu
Intel Core i5 7200U 2.5GHz     183
Intel Core i7 7700HQ 2.8GHz    142
Intel Core i7 7500U 2.7GHz     128
Intel Core i7 8550U 1.8GHz      71
Intel Core i5 8250U 1.6GHz      68
                               ...
AMD A9-Series 9420 2.9GHz        1
Intel Core i7 2.2GHz             1
AMD A6-Series 7310 2GHz          1
Intel Atom Z8350 1.92GHz         1
AMD E-Series 9000e 1.5GHz        1
Name: count, Length: 118, dtype: int64
```

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

```
0              [Intel, Core, i5]
1              [Intel, Core, i5]
2              [Intel, Core, i5]
3              [Intel, Core, i7]
4              [Intel, Core, i5]
                     ...
1298           [Intel, Core, i7]
1299           [Intel, Core, i7]
1300       [Intel, Celeron, Dual]
1301           [Intel, Core, i7]
1302       [Intel, Celeron, Dual]
Name: Cpu, Length: 1272, dtype: object
```

df.head()

[228]:

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	IPS	Touchscreen	ppi	Cpu Name
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	1	0	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	1	0	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	1	0	226.983005	Intel Core i5

Gpu

```
df['Gpu'].value_counts()
df['Gpu brand'] = df['Gpu'].apply(lambda x:x.split()[0])
df.head()
df['Gpu brand'].value_counts()
df.drop(columns=['Gpu'],inplace=True)
df.head()
```

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

```
Gpu brand
Intel     702
Nvidia    393
AMD       176
ARM         1
Name: count, dtype: int64
```

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

```
df.head()
```

	Company	TypeName	Cpu	Ram	OpSys	Weight	Price	IPS	Touchscreen	ppi	Cpu Name	Cpu brand	SSD	HDD	Gpu brand
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	macOS	1.37	71378.6832	1	0	226.983005	Intel Core i5	Intel Core i5	128	0	Intel
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	Intel Core i5	0	0	Intel
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	Intel Core i5	256	0	Intel
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	macOS	1.83	135195.3360	1	0	220.534624	Intel Core i7	Intel Core i7	512	0	AMD

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

```
Gpu
Intel HD Graphics 620      271
Intel HD Graphics 520      181
Intel UHD Graphics 620      65
Nvidia GeForce GTX 1050     64
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: count, Length: 110, dtype: int64
```

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

```
df.head()
```

	Company	TypeName	Cpu	Ram	Gpu	OpSys	Weight	Price	IPS	Touchscreen	ppi	Cpu Name	Cpu brand	SSD	HDD	Gpu brand
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	1	0	226.983005	Intel Core i5	Intel Core i5	128	0	Intel
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	Intel Core i5	0	0	Intel
2	HP	Notebook	Intel Core i5	8	Intel HD	No OS	1.86	30636.0000	0	0	141.211998	Intel	Intel	256	0	Intel

Now we will focus on memory

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

```
import re
# Extract SSD and HDD values using regular expressions
df['SSD'] = df['Memory'].apply(lambda x: int(re.findall(r'(\d+)GB SSD', str(x))[0]) if re.findall(r'(\d+)GB SSD', str(x)) else 0)
df['HDD'] = df['Memory'].apply(lambda x: int(re.findall(r'(\d+)GB HDD', str(x))[0]) if re.findall(r'(\d+)GB HDD', str(x)) else 0)
```

df.head()

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

```
Memory
256GB SSD                  400
1TB HDD                    217
500GB HDD                  130
512GB SSD                  116
128GB SSD +  1TB HDD        92
128GB SSD                   74
256GB SSD +  1TB HDD        71
32GB Flash Storage          37
2TB HDD                     16
64GB Flash Storage          14
512GB SSD +  1TB HDD        14
1TB SSD                     13
256GB SSD +  2TB HDD        10
1.0TB Hybrid                 9
256GB Flash Storage          8
16GB Flash Storage           7
32GB SSD                     6
128GB Flash Storage          4
180GB SSD                    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
```

```
# Extract SSD and HDD values using regular expressions
df['SSD'] = df['Memory'].apply(lambda x: int(re.findall(r'(\d+)GB SSD', str(x))[0]) if re.findall(r'(\d+)GB SSD', str(x)) else 0)
df['HDD'] = df['Memory'].apply(lambda x: int(re.findall(r'(\d+)GB HDD', str(x))[0]) if re.findall(r'(\d+)GB HDD', str(x)) else 0)
```

```
df.head()
```

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	IPS	Touchscreen	ppi	Cpu Name	Cpu brand	SSD	HDD
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	1	0	226.983005	Intel Core i5	Intel Core i5	128	0

df.drop('Memory', axis=1, inplace=True)

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

Operating system

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

```
OpSys
Windows 10      1046
No OS             63
Linux             61
Windows 7         45
Chrome OS         27
macOS             13
Mac OS X           8
Windows 10 S       8
Android            1
Name: count, dtype: int64
```

```
## we will club it
```

```
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()
```

```
df.head()
```

	Company	TypeName	Cpu	Ram	OpSys	Weight	Price	IPS	Touchscreen	ppi	Cpu Name	Cpu brand	SSD	HDD	Gpu brand	os
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	macOS	1.37	71378.6832	1	0	226.983005	Intel Core i5	Intel Core i5	128	0	Intel	Mac
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	Intel Core i5	0	0	Intel	Mac
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	Intel Core i5	256	0	Intel	Others/No OS/Linux
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	macOS	1.83	135195.3360	1	0	220.534624	Intel Core i7	Intel Core i7	512	0	AMD	Mac
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	macOS	1.37	96095.8080	1	0	226.983005	Intel Core i5	Intel Core i5	256	0	Intel	Mac

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

EDA of Laptop Price Prediction Dataset

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.

import the neccesary libraries

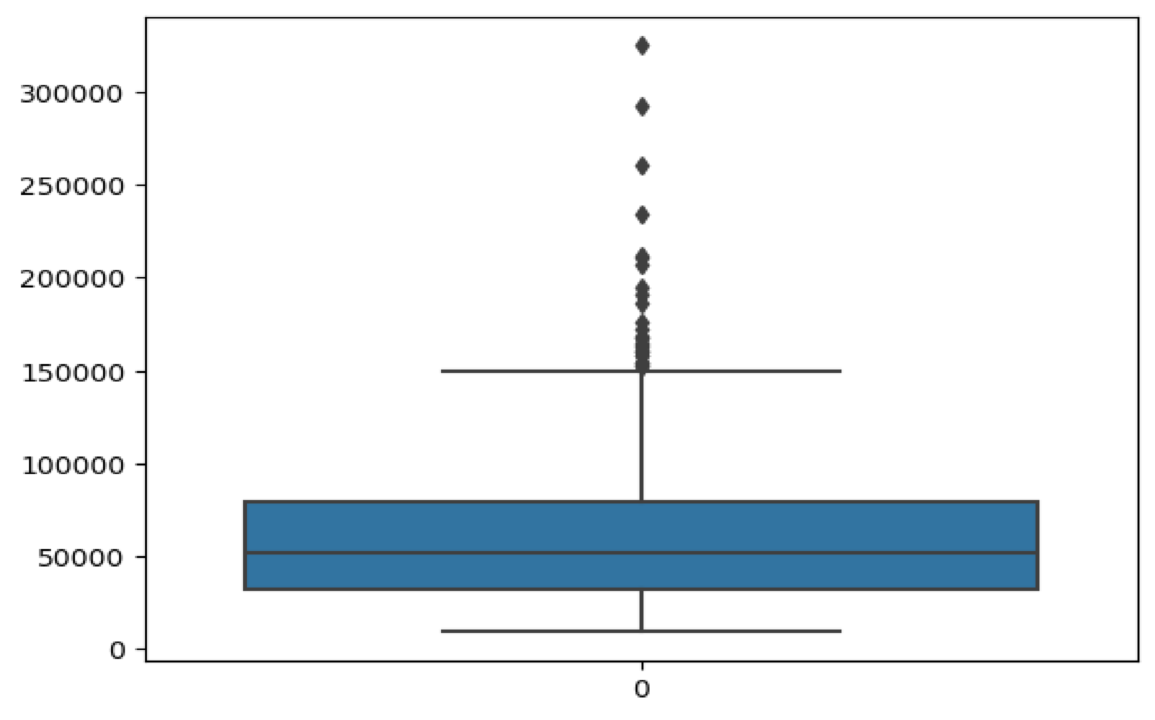
```
import seaborn as sns
import matplotlib.pyplot as plt
```

THE EPICENTRE OF EDA IS PRICE

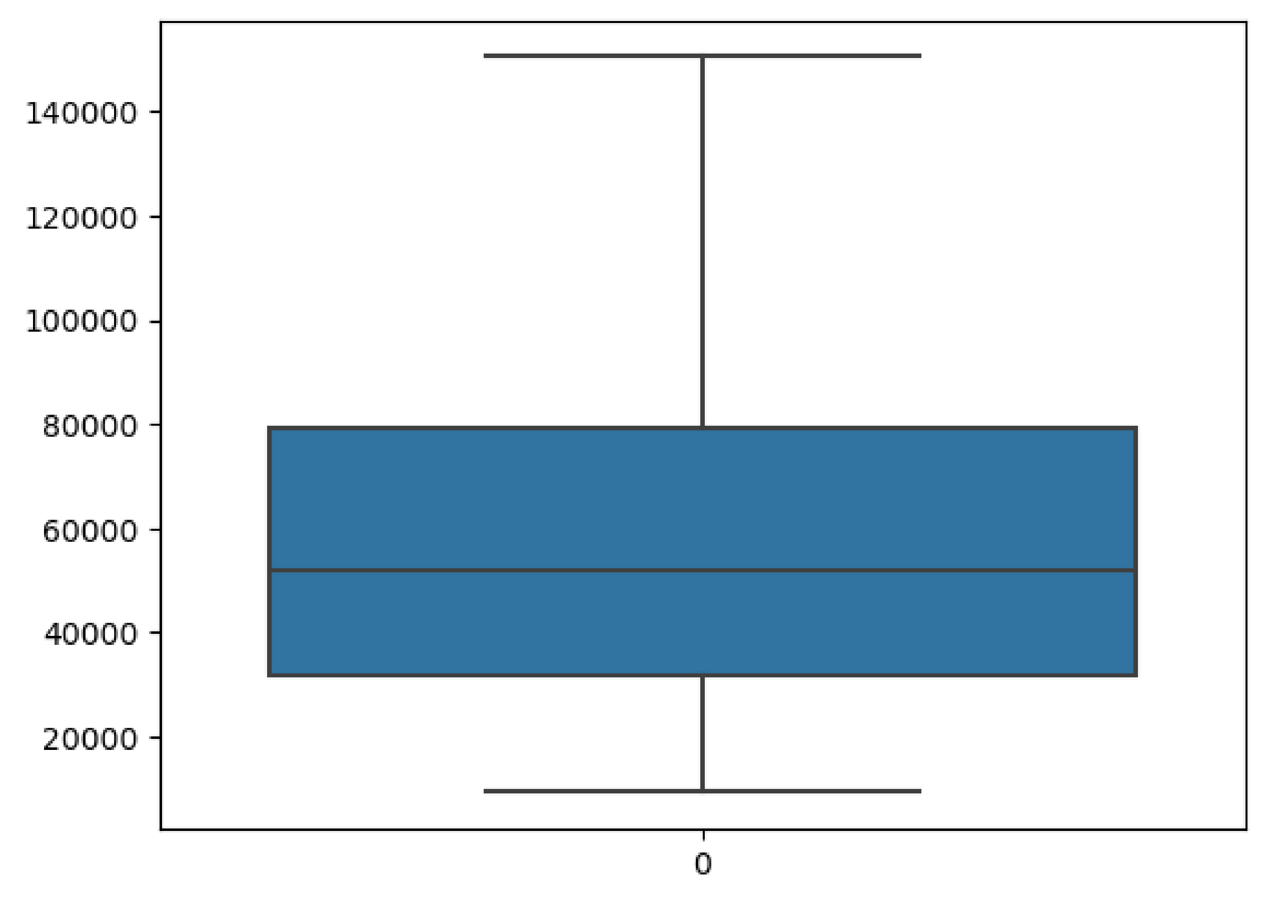
Treating outliers

```
# Filling the null values with median of each
varibledf['Ram']=df['Ram'].fillna(df['Ram'].median())
df['Weight']=df['Weight'].fillna(df['Weight'].media)))
df['Price']=df['Price'].fillna(df['Price'].median())


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



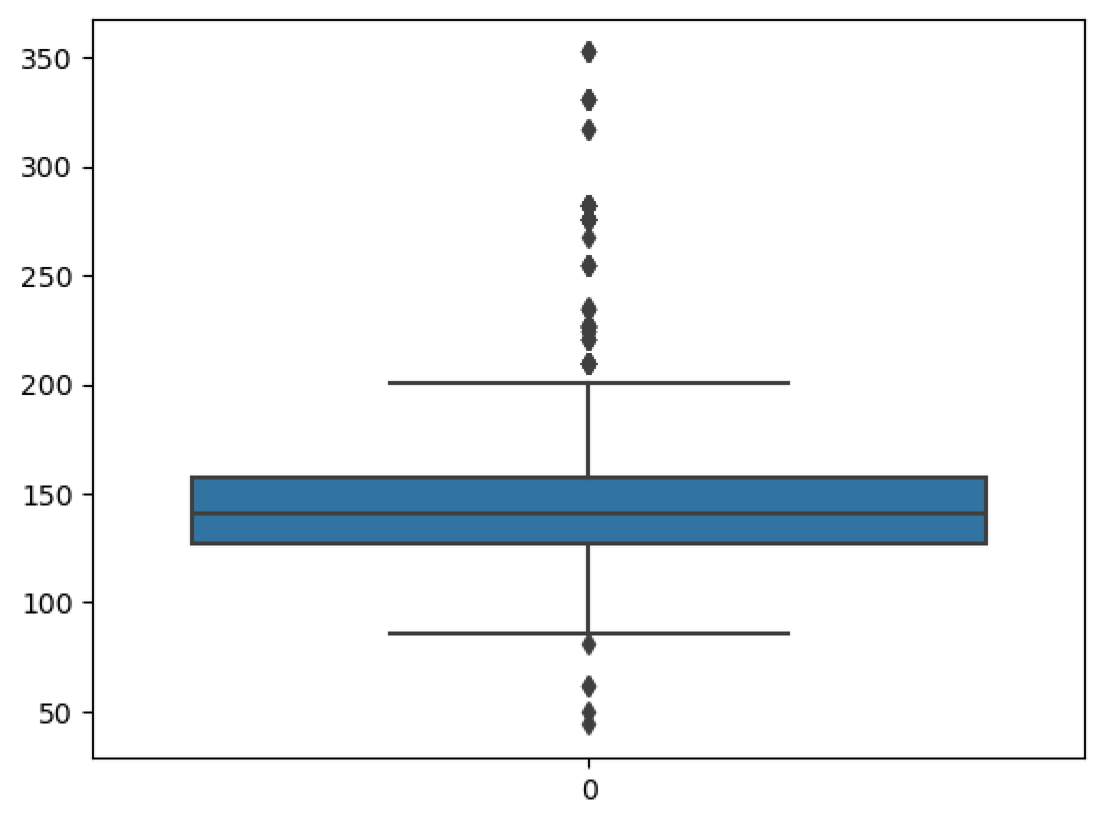

```
q3=df.describe()['Price']['75%']q1=df.describe()['Price']['25%’]
iqr=q3-q1upper=q3+1.5*iqr
lower=q1-1.5*iqr
df['Price']=df['Price'].clip(upper,lower)
```



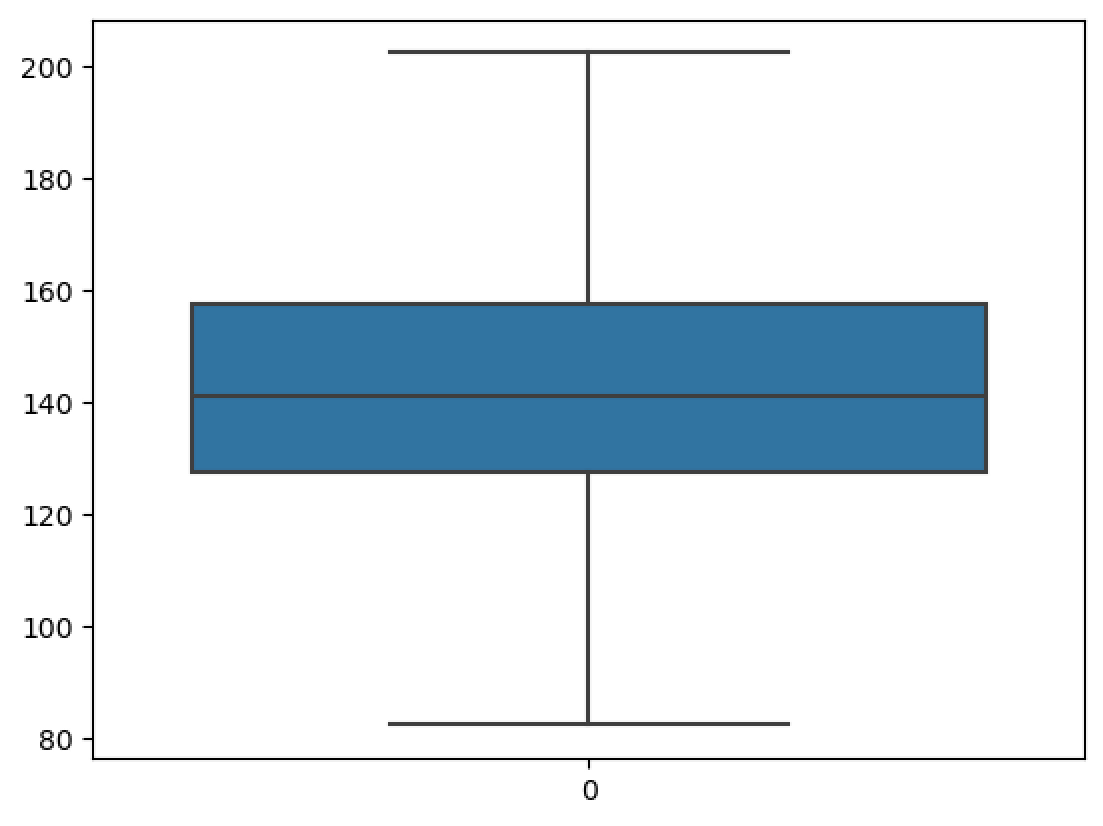
Treating outliers

```
q3=df.describe()['ppi']['75%’]
q1=df.describe()['ppi']['25%’]
iqr=q3-q1upper=q3+1.5*iqr
lower=q1-1.5*iqrdf['ppi']=df['ppi'].clip(upper,lower)
```

```
import seaborn as snssns.boxplot(df['ppi'])
```



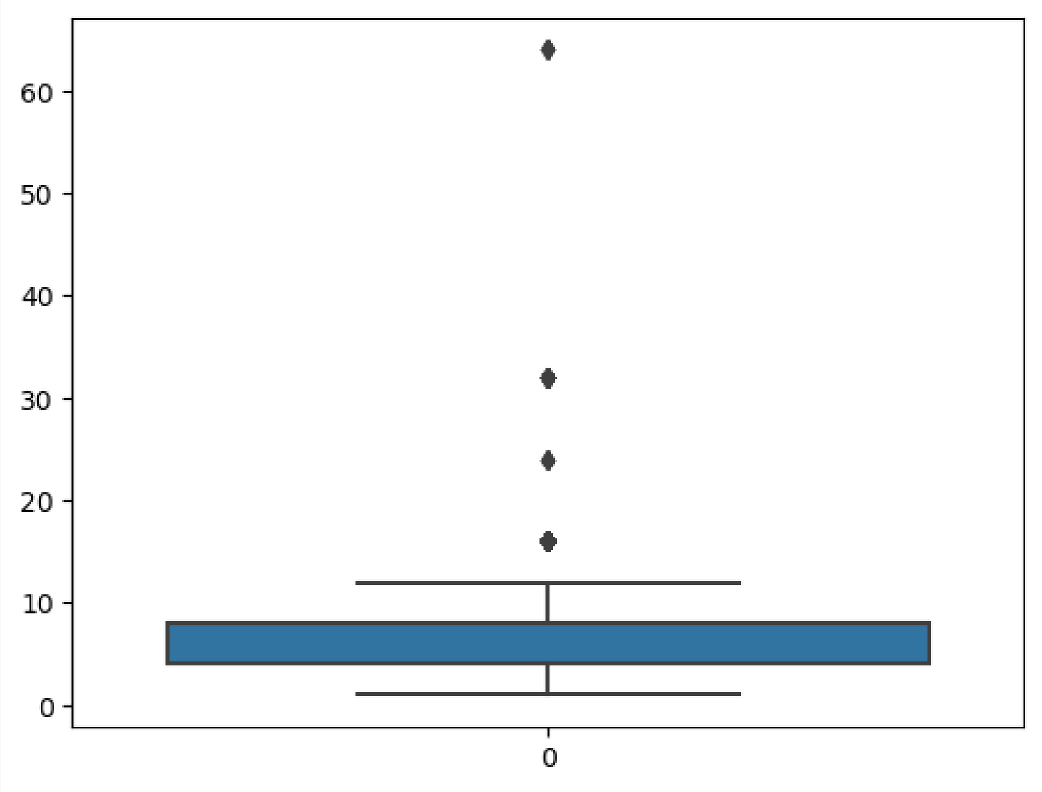

```
sns.boxplot(df['ppi'])
```



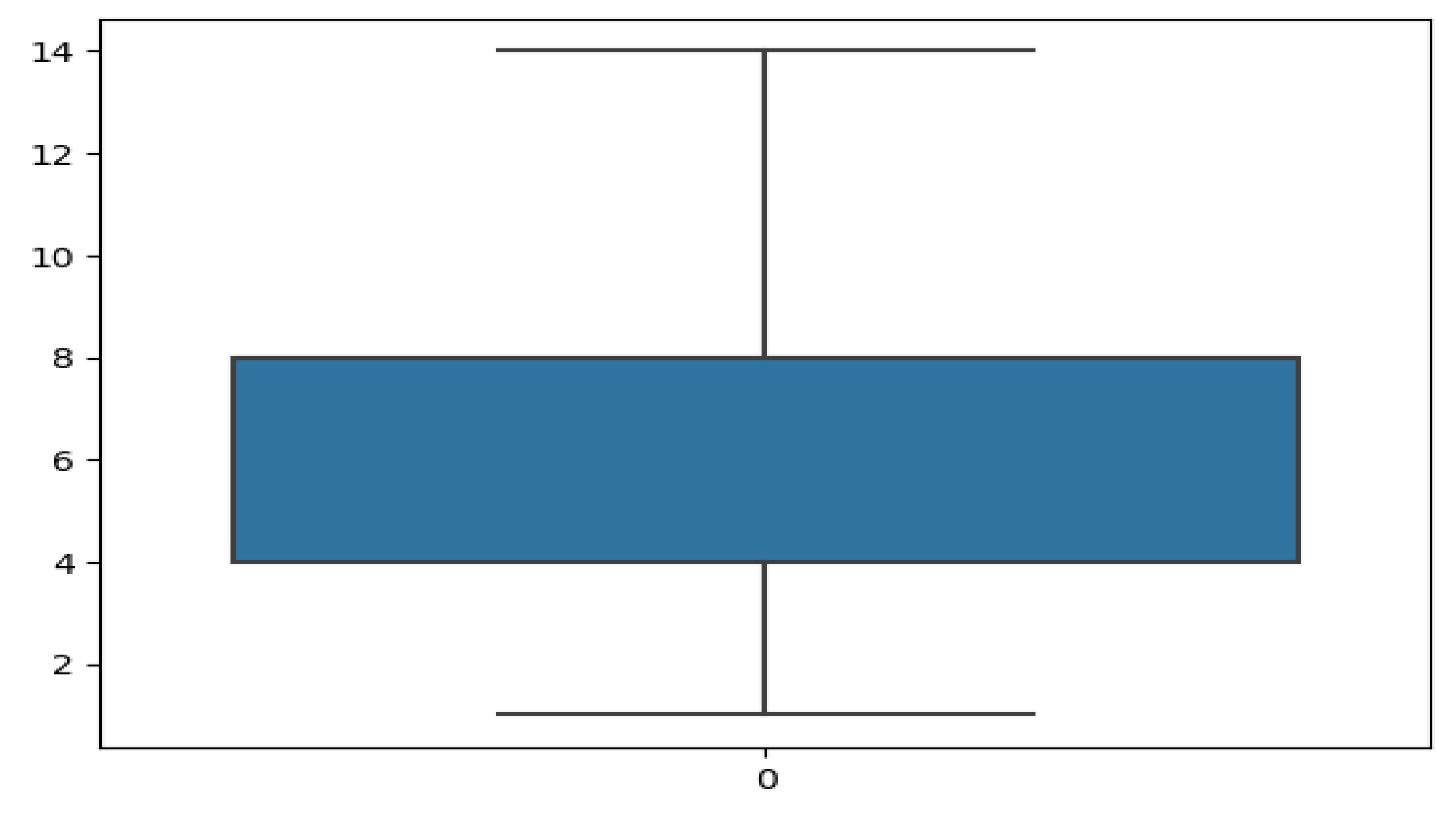
Treating outliers

q3=df.describe()['Ram']['75%’]
q1=df.describe()['Ram']['25%’]
iqr=q3-q1upper=q3+1.5*iqr
lower=q1-1.5*iqrdf['Ram']=df['Ram'].clip(upper,lower)

sns.boxplot(df['Ram'])

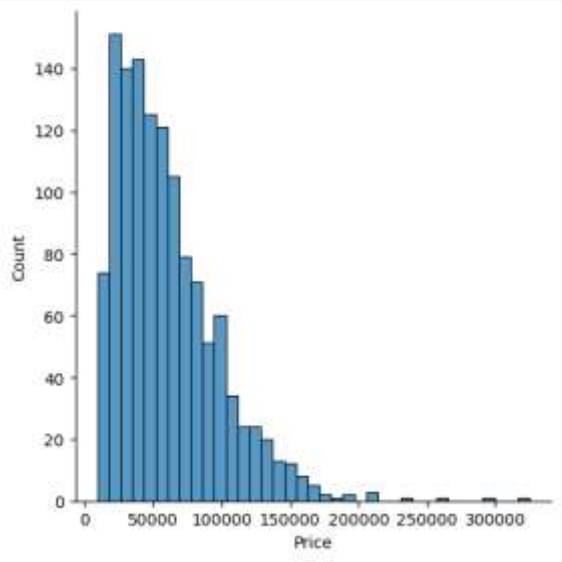


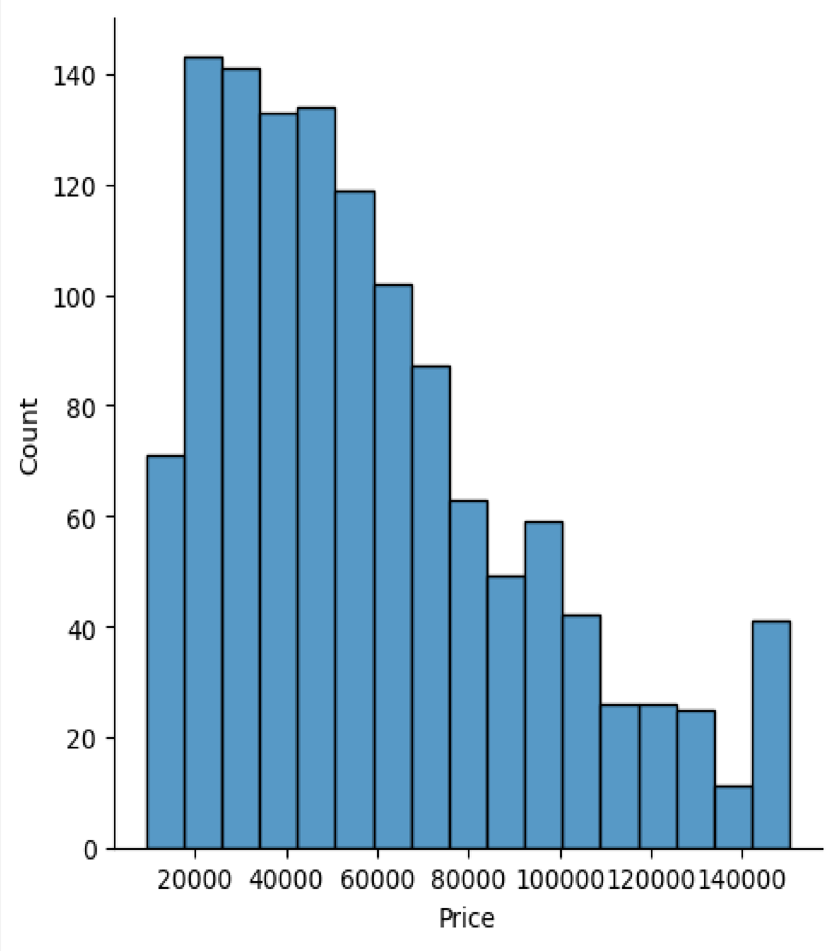
sns.boxplot(df['Ram'])



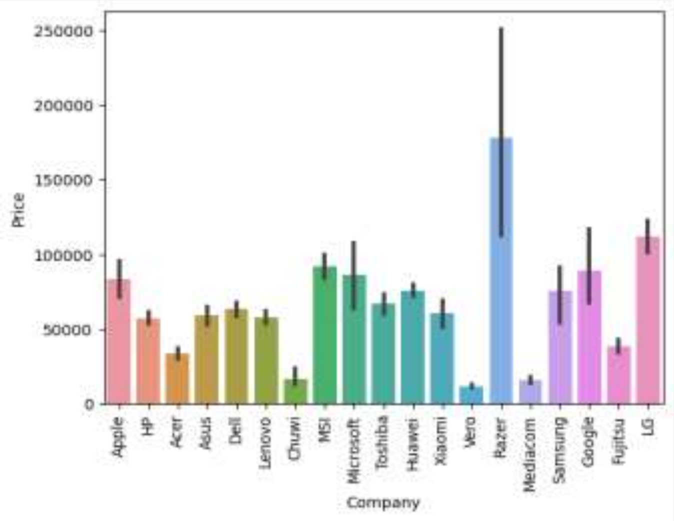
PRICE ANALYSIS

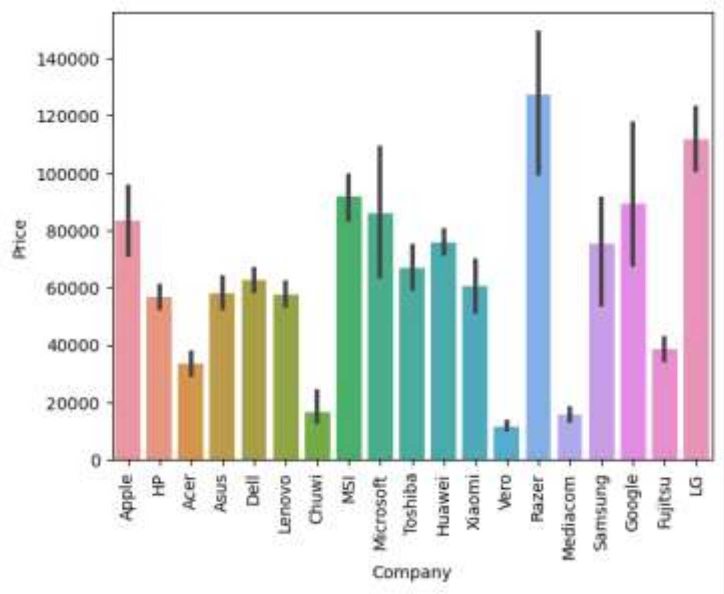
sns.displot(df['Price'])





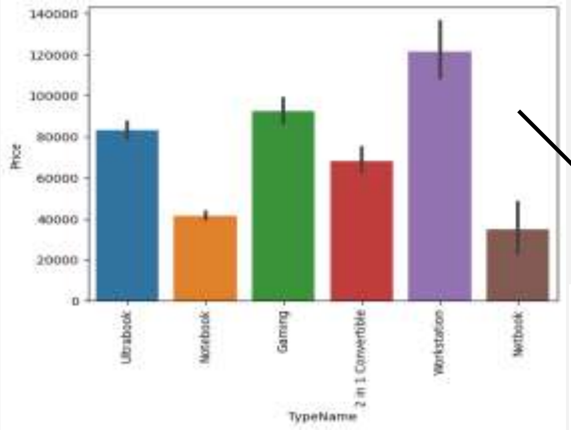
sns.barplot(x=df['Company'],y=df['Price'])plt.xticks(rotation='vertical')plt.show

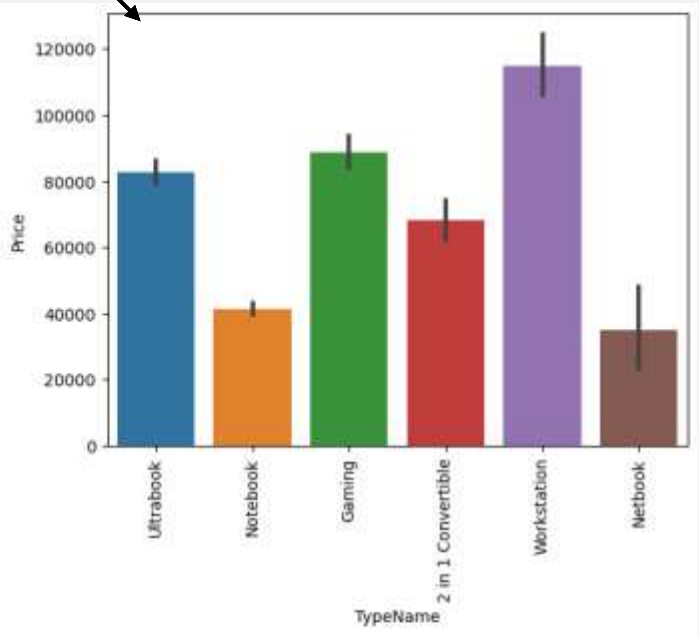




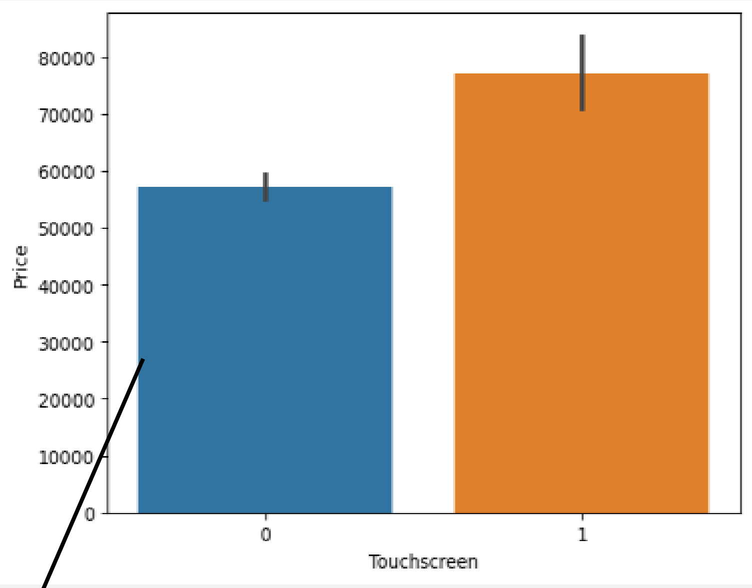
PRICE ANALYSIS

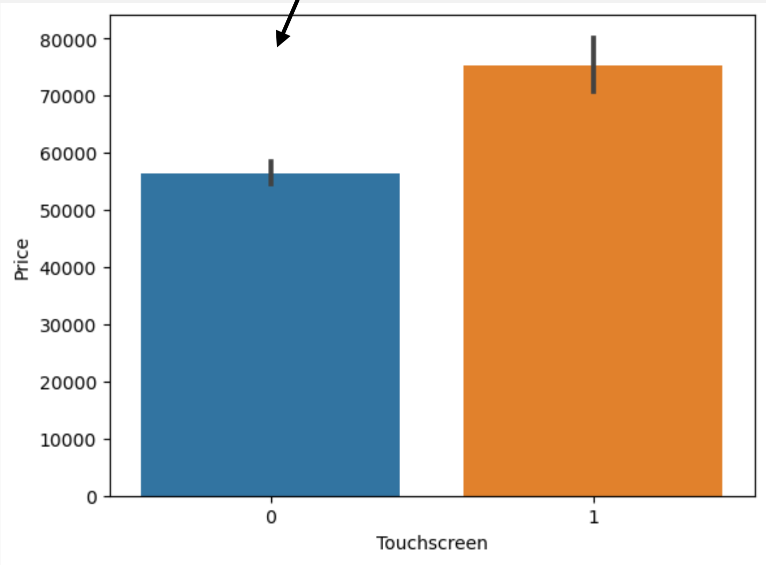
sns.barplot(x=df['TypeName'],y=df['Price'])plt.xticks(rotation='vertical')
plt.show





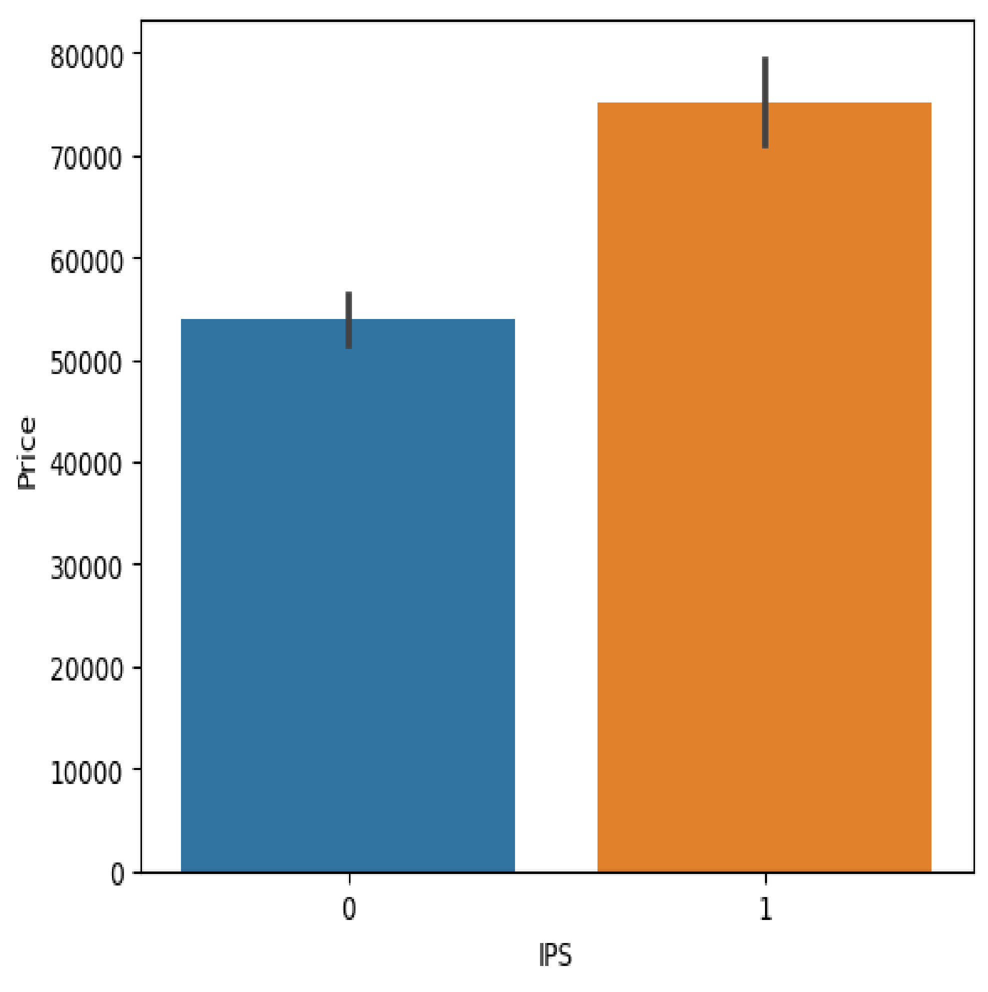
sns.barplot(x=df['Touchscreen'],y=df['Price'])



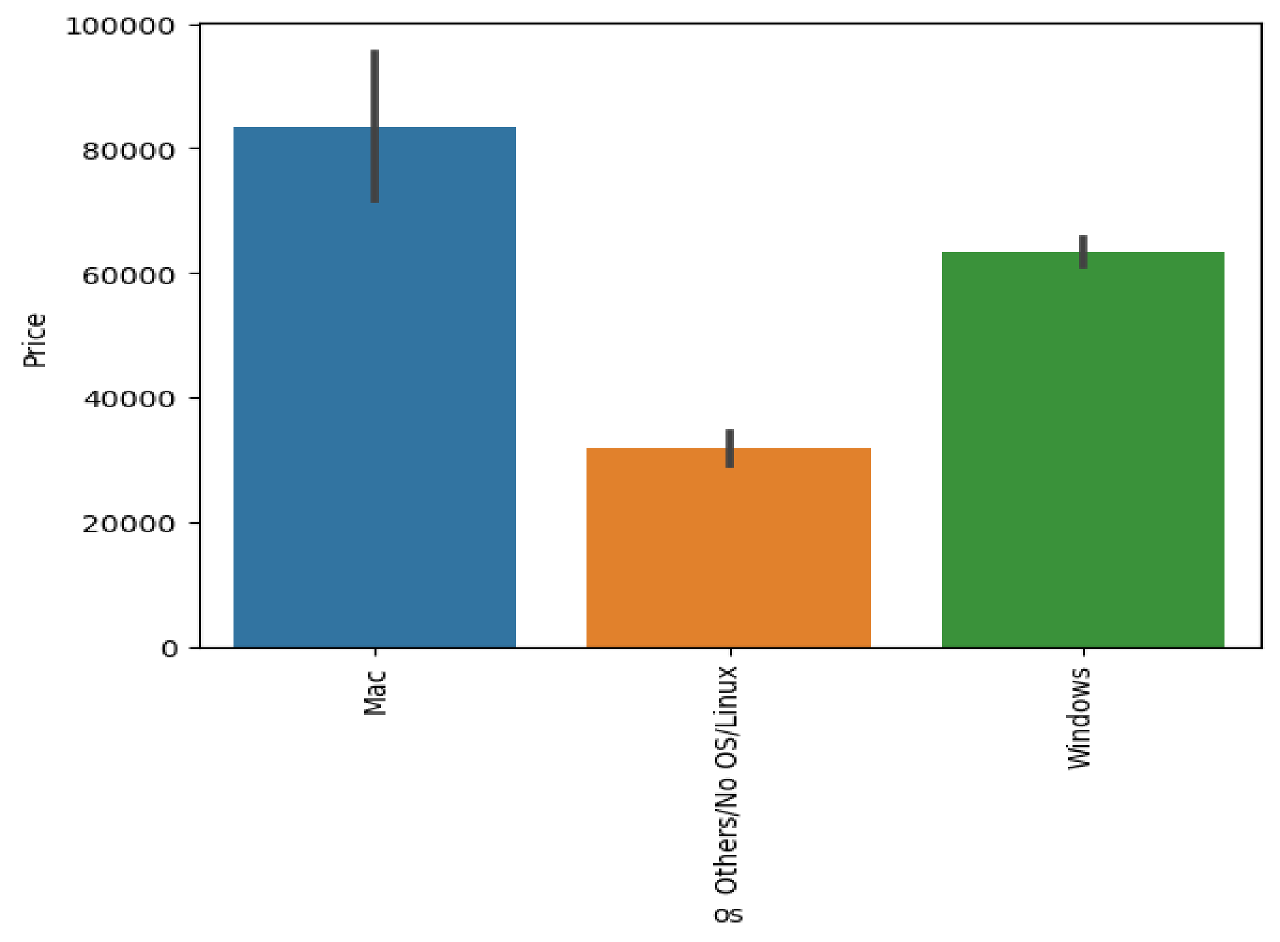


PRICE ANALYSIS

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

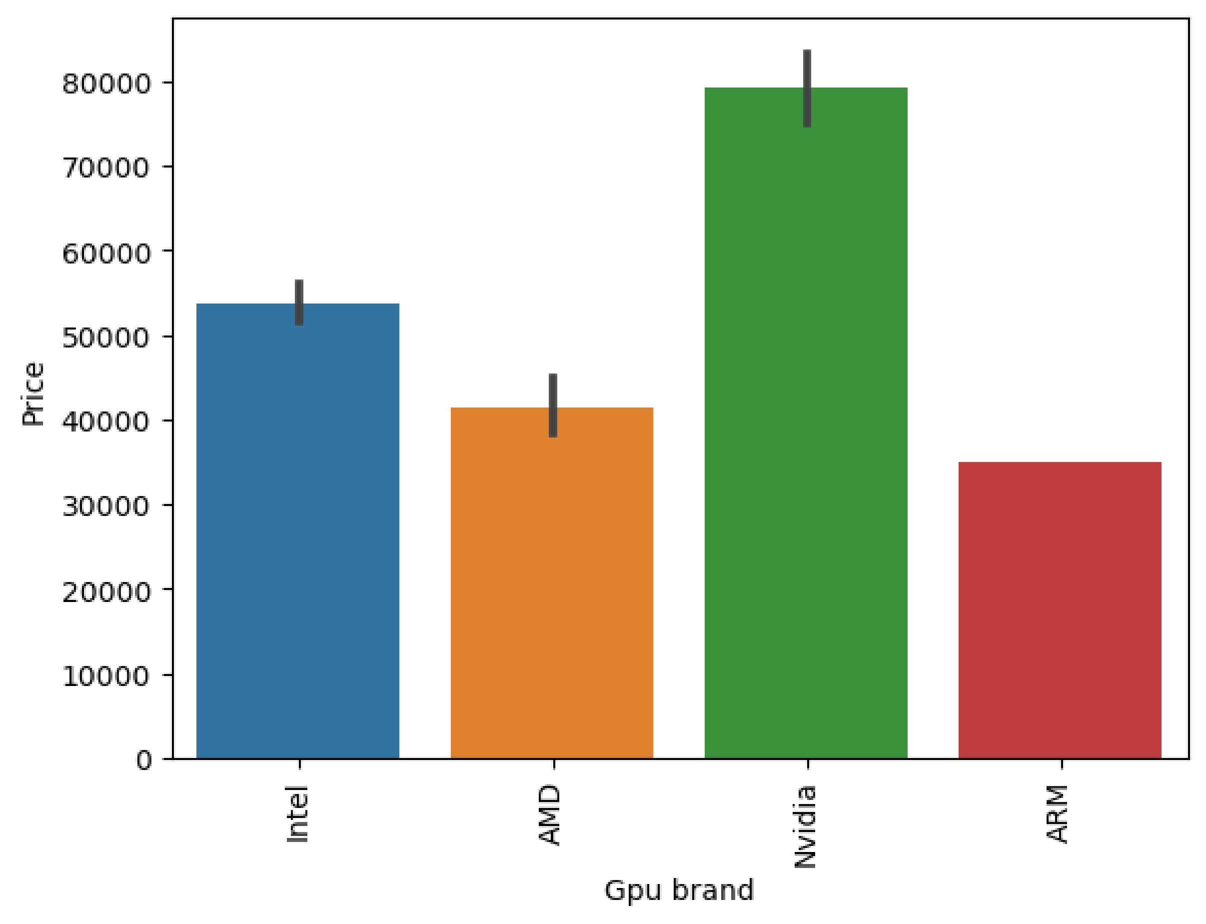


sns.barplot(x=df[‘os'],y=df['Price'])plt.xticks(rotation='vertical’)
plt.show()

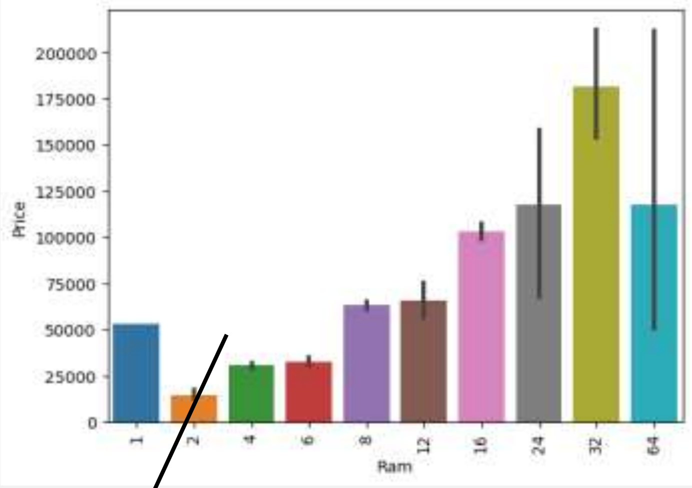


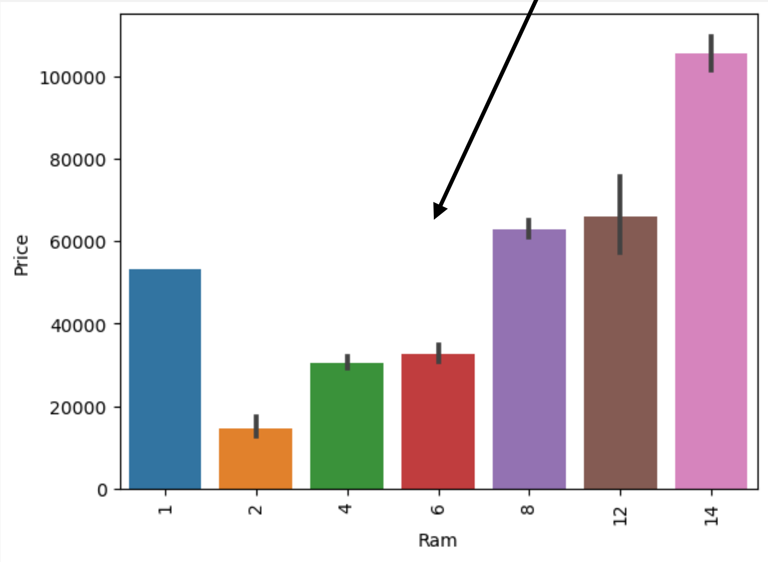
PRICE ANALYSIS

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



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





MACHINE LEARNING : Model Training & Model Testing

Machine Leaning Problem

It is a *Regression problem*, for a given columns we need to predict the price of laptop.

Performance Metric

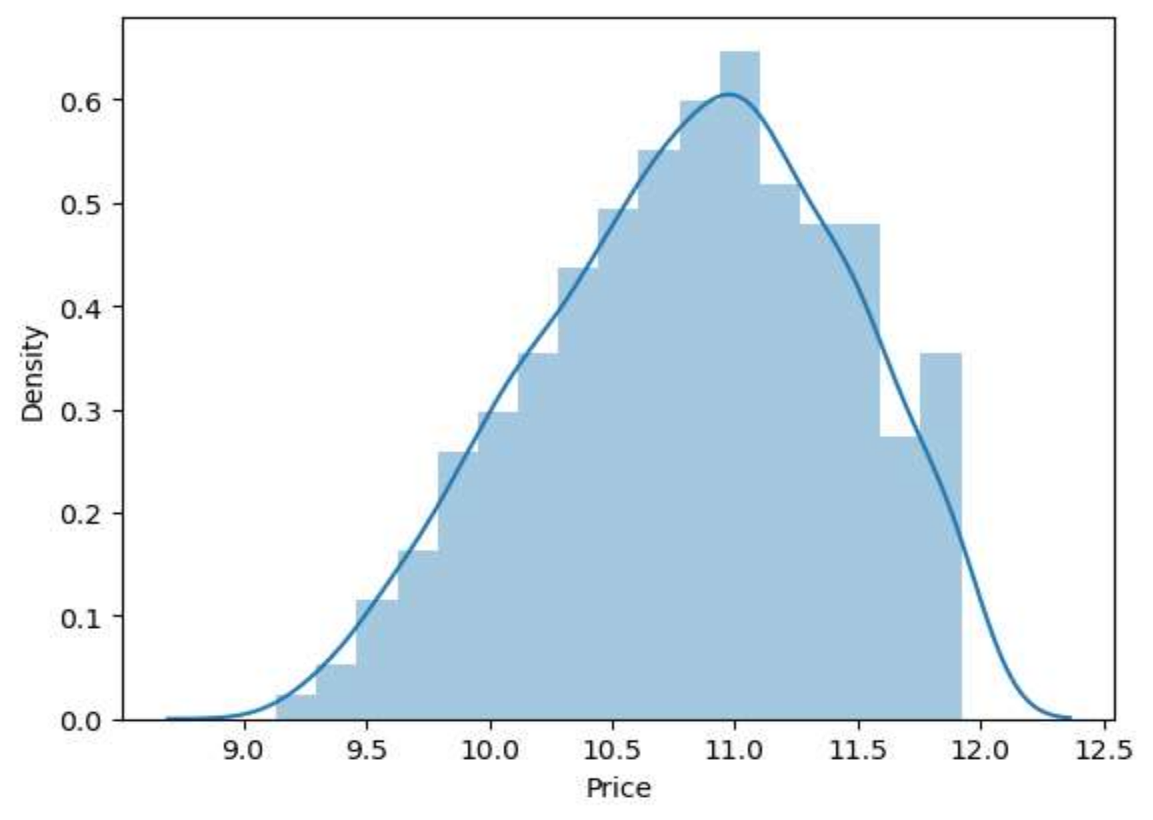
1). R2 Score

2).Mean Absolute Error

MODEL TRAINING

Normalisation
extract x and y by applying log transformation.
It will increase the value of R^2.

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



from sklearn.model_selection import train_test_splitX_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.15,random_state=2)

X_train

```
[163]: 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)

[164]: X_train
```

[164]:

	Company	TypeName	Ram	Weight	IPS	Touchscreen	ppi	Cpu brand	SSD	HDD	Gpu brand	os
735	Lenovo	Notebook	4	1.85	0	0	141.211998	Intel Core i7	0	0	Intel	Windows
22	HP	Notebook	4	1.86	0	0	100.454670	AMD Processor	0	500	AMD	Others/No OS/Linux
811	MSI	Gaming	14	2.90	0	0	127.335675	Intel Core i7	512	0	Nvidia	Windows
283	Lenovo	Notebook	6	2.20	0	0	141.211998	Intel Core i5	256	0	Intel	Windows
765	Acer	Notebook	4	1.60	0	0	117.826530	Intel Core i5	128	0	Intel	Windows
...	...	...	...	...	...	...	...	...	...	...	...	...
479	Toshiba	Notebook	8	1.05	1	0	165.632118	Intel Core i5	256	0	Intel	Windows
309	HP	Notebook	4	1.86	0	0	141.211998	Intel Core i3	0	0	Intel	Windows
506	Asus	Notebook	8	2.00	0	0	141.211998	Intel Core i7	256	0	Intel	Windows
540	Dell	Ultrabook	8	1.20	0	1	202.372769	Intel Core i7	256	0	Intel	Windows
1222	HP	Notebook	6	2.10	0	0	141.211998	AMD Processor	0	0	AMD	Windows

1081 rows × 12 columns

Machine Learning Modeling for Laptop Price Prediction

Now we have prepared our data and hold a better understanding of the dataset. so let's get started with Machine learning modeling and find the best algorithm with the best hyperparameters to achieve maximum accuracy.

Import Libraries

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

```
!pip install xgboost==1.7.5
```

```
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,ExtraTreesRegressor
from sklearn.svm import SVRfrom xgboost import XGBRegressor
from sklearn.cluster import AffinityPropagation
```

Implement Pipeline for training and testing

Now we will implement a pipeline to streamline the training and testing process. first, we use a column transformer to encode categorical variables which is step one. After that, we create an object of our algorithm and pass both steps to the pipeline. using pipeline objects we predict the score on new data and display the accuracy.

RANDOM FOREST

Implement Pipeline for training and testing

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

# Add an imputation step to handle NaNs before the KNeighborsRegressor
step2 = SimpleImputer(strategy='mean')  # Replace NaNs with the mean of the column

step3 = RandomForestRegressor(n_estimators=100,
                              random_state=3,
                              max_samples=0.5,
                              max_features=0.75,
                              max_depth=15)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2),
    ('step3',step3)
])

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 0.8765144459399266
MAE 0.1641282888254933
```

LINEAR REGRESSION

```
# ... (Your existing code for train_test_split and X_train)



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

# Add an imputation step to handle NaNs
step2 = SimpleImputer(strategy='mean')  # Replace NaNs with the mean of the column

step3 = LinearRegression()

pipe = Pipeline([
    ('step1',step1),
    ('step2', step2),  # Impute missing values before fitting the model
    ('step3',step3)
])

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 0.8276505512108473
MAE 0.19421244887656317
```

DECISION TREE

```
# ... (Your existing code for train_test_split and X_train)

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

# Add an imputation step to handle NaNs before the KNeighborsRegressor
step2 = SimpleImputer(strategy='mean')  # Replace NaNs with the mean of the column

step3 = DecisionTreeRegressor(max_depth=8)

pipe = Pipeline([
    ('step1',step1),
    ('step2', step2),  # Impute missing values
    ('step3',step3)
])

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 0.8470529703979255
MAE 0.1747368226634528
```

STACKING

What is stacking in machine learning?

Stacking is one of the most popular ensemble machine learning techniques.

used to predict multiple nodes to build a new model and improve models performance.

```
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import r2_score, mean_absolute_error

# Add an imputation step to your pipeline
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first', handle_unknown = 'ignore'),[0,1,7,10,11])
],remainder='passthrough')

step2 = SimpleImputer(strategy='mean') # Impute missing values with the mean

step3 = RandomForestRegressor(n_estimators=100,
                              random_state=3,
                              max_samples=0.5,
                              max_features=0.75,
                              max_depth=15)

pipe = Pipeline([
    ('step1',step1),
    ('step2', step2), # Add imputation step
    ('step3',step3)
])

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 0.8765144459399266
MAE 0.1641282888254933
```

CONCLUSION

The development of a robust machine learning model for predicting laptop *prices has yielded valuable insights into the dynamic landscape of the tech market for SmartTech Co.* Through meticulous data exploration, preprocessing, and model development, we've identified key features influencing[(Company) (TypeName) (Ram) (Weight) (IPS) (Touchscreen) (ppi) (Cpu brand) (SSD) (HDD) (Gpu brand) (os)] while demonstrating the model's efficacy in accurately predicting prices across various brands and specifications. Leveraging real-time prediction capabilities, SmartTech Co. can strategically position its laptops in the market, optimize pricing strategies, and swiftly respond to market shifts, ensuring competitiveness and customer satisfaction in an ever-evolving industry.

Questions to Explore:

1). Which features have the most significant impact on laptop prices?

 -> The most significant features on price is 'Ram','Gpu brand','os' and'PPi' due to it have strong correlation with price.

2). Does the brand of the laptop significantly influence its price?

-> Only applies to apple and gaming laptop. But for remaining it depends on the configuration.

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

 -> Model doesn't differentiate and performs reasonably well on both ends of specifications.

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

certain brands, configurations etc.

 -> Major Limitation is lack of sufficient data of most laptop companies which will adversely effect in predictingcertain brands,configurations etc

Create GUI of Laptop Price Prediction Mode

!pip install gradio==4.29.0

```
[191]: !pip install gradio==4.29.0
Requirement already satisfied: httpx>=0.24.1 in c:\users\16reh\appdata\roaming\python\python311\site-packages (from gradio==4.29.0) (0.27.2)
Requirement already satisfied: huggingface-hub>=0.19.3 in c:\users\16reh\appdata\roaming\python\python311\site-packages (from gradio==4.29.0) (0.24.
6)
Requirement already satisfied: importlib-resources<7.0,>=1.3 in c:\users\16reh\appdata\roaming\python\python311\site-packages (from gradio==4.29.0)
(6.4.4)
Requirement already satisfied: jinja2<4.0 in c:\programdata\anaconda3\lib\site-packages (from gradio==4.29.0) (3.1.3)
Requirement already satisfied: markupsafe~=2.0 in c:\programdata\anaconda3\lib\site-packages (from gradio==4.29.0) (2.1.3)
Requirement already satisfied: matplotlib~=3.0 in c:\programdata\anaconda3\lib\site-packages (from gradio==4.29.0) (3.8.0)
Requirement already satisfied: numpy~=1.0 in c:\programdata\anaconda3\lib\site-packages (from gradio==4.29.0) (1.26.4)
Requirement already satisfied: orjson~=3.0 in c:\users\16reh\appdata\roaming\python\python311\site-packages (from gradio==4.29.0) (3.10.7)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from gradio==4.29.0) (23.1)
Requirement already satisfied: pandas<3.0,>=1.0 in c:\programdata\anaconda3\lib\site-packages (from gradio==4.29.0) (2.1.4)
Requirement already satisfied: pillow<11.0,>=8.0 in c:\programdata\anaconda3\lib\site-packages (from gradio==4.29.0) (10.2.0)
Requirement already satisfied: pydantic>=2.0 in c:\users\16reh\appdata\roaming\python\python311\site-packages (from gradio==4.29.0) (2.9.0)
Requirement already satisfied: pydub in c:\users\16reh\appdata\roaming\python\python311\site-packages (from gradio==4.29.0) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.9 in c:\users\16reh\appdata\roaming\python\python311\site-packages (from gradio==4.29.0) (0.0.9)
Requirement already satisfied: pyyaml<7.0,>=5.0 in c:\programdata\anaconda3\lib\site-packages (from gradio==4.29.0) (6.0.1)
Requirement already satisfied: ruff>=0.2.2 in c:\users\16reh\appdata\roaming\python\python311\site-packages (from gradio==4.29.0) (0.6.4)
```

import gradio as grimport pandas as pdimport pickle

df = pd.read_csv('laptop.csv’)

pipe = pickle.load(open('pipe.pkl','rb'))

```python
import gradio as gr
import pandas as pd
import pickle

df = pd.read_csv('laptop.csv')
pipe = pickle.load(open('pipe.pkl','rb'))
```

```python
def predict_price(Company,TypeName,Ram,Weight,Touchscreen,IPS,ppi,Cpu_brand,HDD,SSD,Gpu_brand,os):
    input_df = pd.DataFrame([[Company,TypeName,Ram,Weight,Touchscreen,IPS,ppi,Cpu_brand,HDD,SSD,Gpu_brand,os]],columns=['Company','TypeName','Ram','Weight

    # Preprocess the input
    Company = input_df['Company'].map({'Apple':0,'Razer':1,'Mediacom':2,'Samsung':3,'Toshiba':4,'MSI':5,'Vero':6,'Dell':7,'Huawei':8,'Chuwi':9,'Fujitsu':1
    input_df['Company'] = Company

    TypeName = input_df['TypeName'].map({'Ultrabook':0,'Notebook':1,'Netbook':2,'Gaming':3,'2 in 1 Convertible':4,'Workstation':5})
    input_df['TypeName'] = TypeName

    Cpu_brand = input_df['Cpu brand'].map({'Intel Core i7':0,'Intel Core i5':1,'Intel Core i3':2,'Other Intel Processor':3,'AMD Processor':4})
    input_df['Cpu brand'] = Cpu_brand

    Gpu_brand = input_df['Gpu brand'].map({'Intel':0,'Nvidia':1,'AMD':2,'ARM':3})
    input_df['Gpu brand'] = Gpu_brand

    os = input_df['os'].map({'Mac':0,'Others/No OS/Linux':1,'Windows':2})
    input_df['os'] = os

    prediction = np.exp(pipe.predict(input_df))
    return prediction[0]
```

Define the input components for the interface

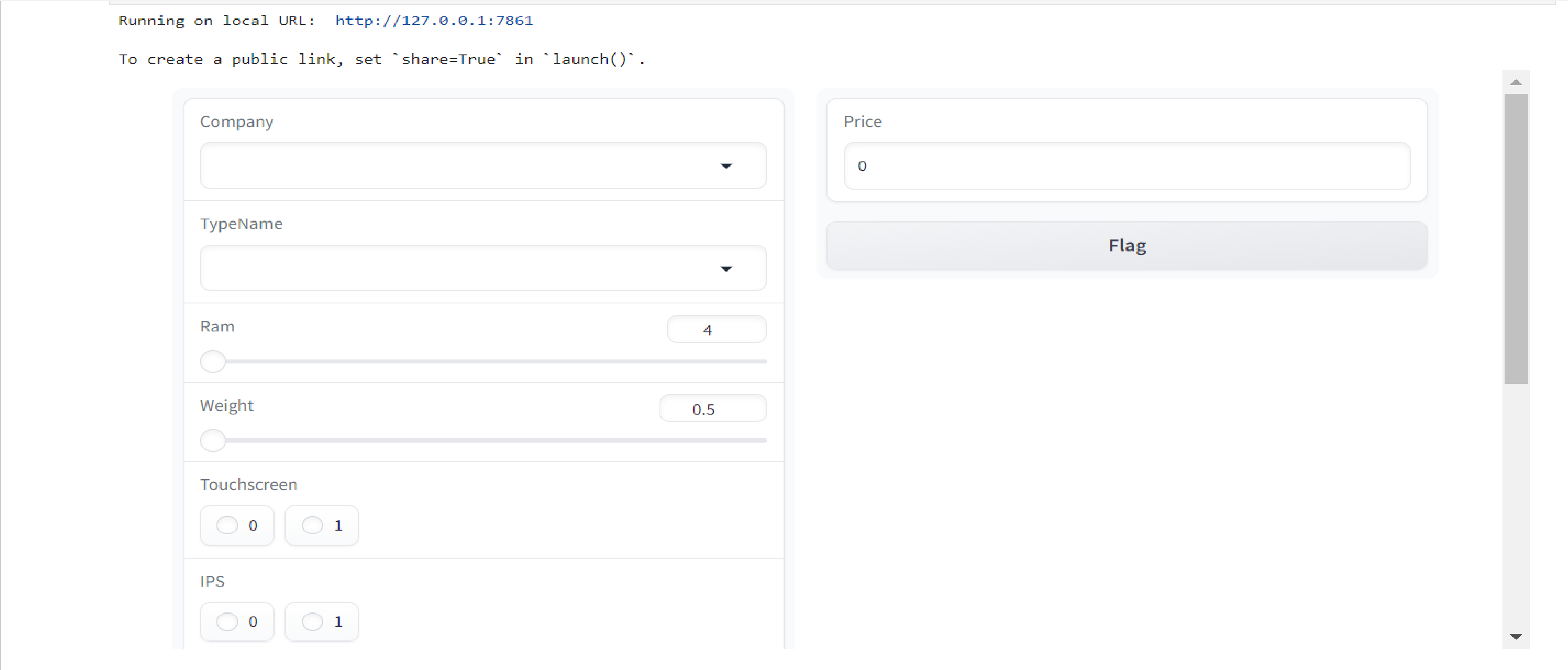
```
# Define the input components for the interface
new_var = [
    gr.Dropdown(choices=list(df['Company'].unique()), label="Company"),
    gr.Dropdown(choices=list(df['TypeName'].unique()), label="TypeName"),
    gr.Slider(4, 64, step=4, label="Ram"),
    gr.Slider(0.5, 5.0, step=0.1, label="Weight"),
    gr.Radio(choices=[0, 1], label="Touchscreen"),
    gr.Radio(choices=[0, 1], label="IPS"),
    gr.Slider(50, 500, step=1, label="ppi"),
    gr.Slider(0, 2000, step=1, label="HDD"),
    gr.Slider(0, 2000, step=1, label="SSD"),
    gr.Dropdown(choices=list(df['OpSys'].unique()), label="os") # Changed 'os' to 'OpSys'
]
inputs = new_var

# Define the output component for the interface
outputs = gr.Number(label="Price")

# Create the Gradio interface
iface = gr.Interface(fn=predict_price, inputs=inputs, outputs=outputs)

# Launch the interface
iface.launch()
share=True
```

Laptop Prediction GUI



THANK YOU

REHAN ASLAM KHAN

+91 8968832194

16rehan687@gmail.com

https://github.com/REHAN-S8882 Github.com

Contoso