Laptop price prediction

USING RANDOM FOREST ALGORITHM

Guided BY :
Deepak Gurav sir

External name:

Project by :
Ashwini Patil

Contents

DATA set for laptop prediction

Most of the columns in a dataset are noisy and contain lost of information . But with feature engineering you do , you get more good results . The only problem is we are having less data but we will obtain a good accuracy it . The only good thing is it is better to have a large data .

Data set

A1

	A	B	C	D	E	F	G	H	I	J	K	L
1		Company	TypeName	Inches	ScreenRes	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
2	0	Apple	Ultrabook	13.3	IPS Panel F	Intel Core	8GB	128GB SSD	Intel Iris Pl	macOS	1.37kg	71378.68
3	1	Apple	Ultrabook	13.3	1440x900	Intel Core	8GB	128GB Flas	Intel HD Gr	macOS	1.34kg	47895.52
4	2	HP	Notebook	15.6	Full HD 19:	Intel Core	8GB	256GB SSD	Intel HD Gr	No OS	1.86kg	30636
5	3	Apple	Ultrabook	15.4	IPS Panel F	Intel Core	16GB	512GB SSD	AMD Rade	macOS	1.83kg	135195.3
6	4	Apple	Ultrabook	13.3	IPS Panel F	Intel Core	8GB	256GB SSD	Intel Iris Pl	macOS	1.37kg	96095.81
7	5	Acer	Notebook	15.6	1366x768	AMD A9-Se	4GB	500GB HD	AMD Rade	Windows 1	2.1kg	21312
8	6	Apple	Ultrabook	15.4	IPS Panel F	Intel Core	16GB	256GB Flas	Intel Iris Pr	Mac OS X	2.04kg	114017.6
9	7	Apple	Ultrabook	13.3	1440x900	Intel Core	8GB	256GB Flas	Intel HD Gr	macOS	1.34kg	61735.54
10	8	Asus	Ultrabook	14	Full HD 19:	Intel Core	16GB	512GB SSD	Nvidia GeF	Windows 1	1.3kg	79653.6
11	9	Acer	Ultrabook	14	IPS Panel F	Intel Core	8GB	256GB SSD	Intel UHD	Windows 1	1.6kg	41025.6
12	10	HP	Notebook	15.6	1366x768	Intel Core	4GB	500GB HD	Intel HD Gr	No OS	1.86kg	20986.99
13	11	HP	Notebook	15.6	Full HD 19:	Intel Core	4GB	500GB HD	Intel HD Gr	No OS	1.86kg	18381.07
14	12	Apple	Ultrabook	15.4	IPS Panel F	Intel Core	16GB	256GB SSD	AMD Rade	macOS	1.83kg	130001.6
15	13	Dell	Notebook	15.6	Full HD 19:	Intel Core	4GB	256GB SSD	AMD Rade	Windows 1	2.2kg	26581.39
16	14	Apple	Ultrabook	12	IPS Panel F	Intel Core	8GB	256GB SSD	Intel HD Gr	macOS	0.92kg	67260.67
17	15	Apple	Ultrabook	13.3	IPS Panel F	Intel Core	8GB	256GB SSD	Intel Iris Pl	macOS	1.37kg	80908.34
18	16	Dell	Notebook	15.6	Full HD 19:	Intel Core	8GB	256GB SSD	AMD Rade	Windows 1	2.2kg	39693.6
19	17	Apple	Ultrabook	15.4	IPS Panel F	Intel Core	16GB	512GB SSD	AMD Rade	macOS	1.83kg	152274.2
20	18	Lenovo	Notebook	15.6	Full HD 19:	Intel Core	8GB	1TB HDD	Nvidia GeF	No OS	2.2kg	26586.72
21	19	Dell	Ultrabook	13.3	IPS Panel F	Intel Core	8GB	128GB SSD	Intel UHD	Windows 1	1.22kg	52161.12
22	20	Asus	Netbook	11.6	1366x768	Intel Atom	2GB	32GB Flash	Intel HD Gr	Windows 1	0.98kg	10224.43
23	21	Lenovo	Gaming	15.6	IPS Panel F	Intel Core	8GB	128GB SSD	Nvidia GeF	Windows 1	2.5kg	53226.72
24	22	HP	Notebook	15.6	1366x768	AMD E-Ser	4GB	500GB HD	AMD Rade	No OS	1.86kg	13746.24
25	23	Dell	2 in 1 Conv	13.3	Full HD / T	Intel Core	8GB	256GB SSD	Intel UHD	Windows 1	1.62kg	43636.32
26	24	HP	Ultrabook	15.6	Full HD 19:	Intel Core	8GB	256GB SSD	Intel HD Gr	Windows 1	1.91kg	35111.52

laptop data

Basic understanding of Laptop price prediction Data

Naw let us start working on a dataset in our jupyter Notebook . The first step is to import the libraries and load data. After that we will take a basic understanding of data like its shape ,sample, is there are any NULL values present in the dataset . Understanding the data is an important step for prediction or any machine learning project.

In [129]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [130]:
```
df=pd.read_csv('laptop_data.csv')
df
```

Out[130]:

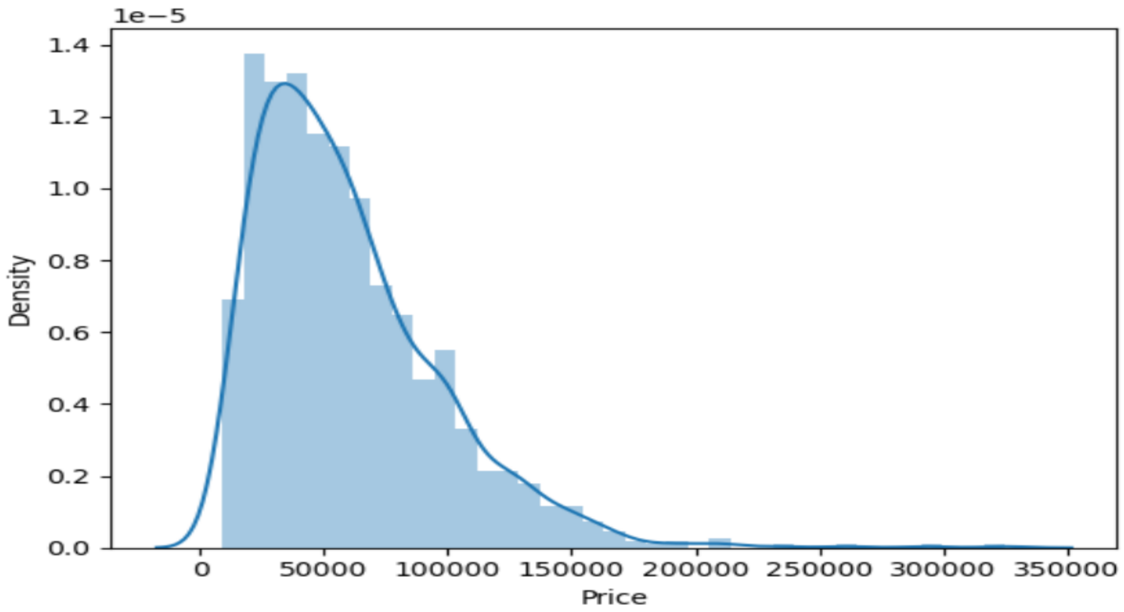
	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	P
0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.
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.
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.

Distribution of target columns

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

```
In [142]:  1  sns.distplot(df['Price'])
```

```
Out[142]: <Axes: xlabel='Price', ylabel='Density'>
```



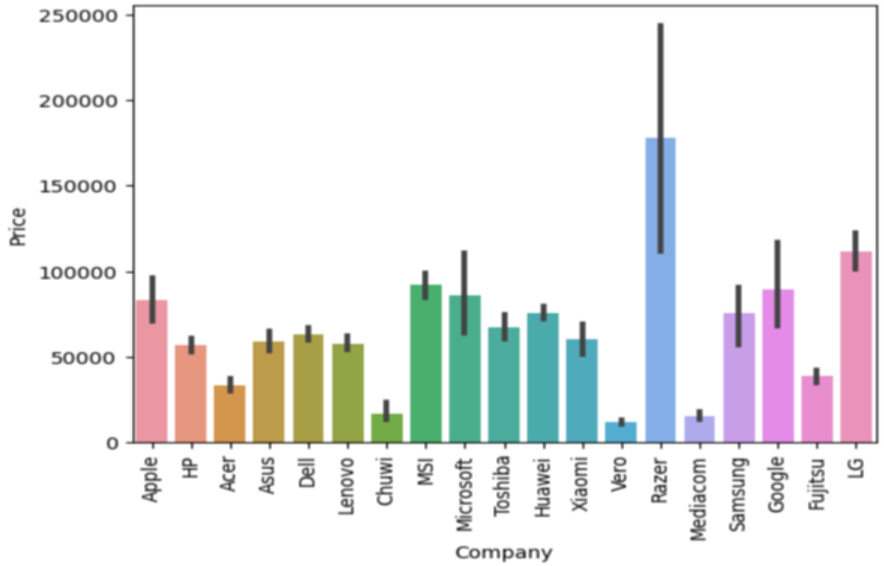
Company column

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

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

In [144]:

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



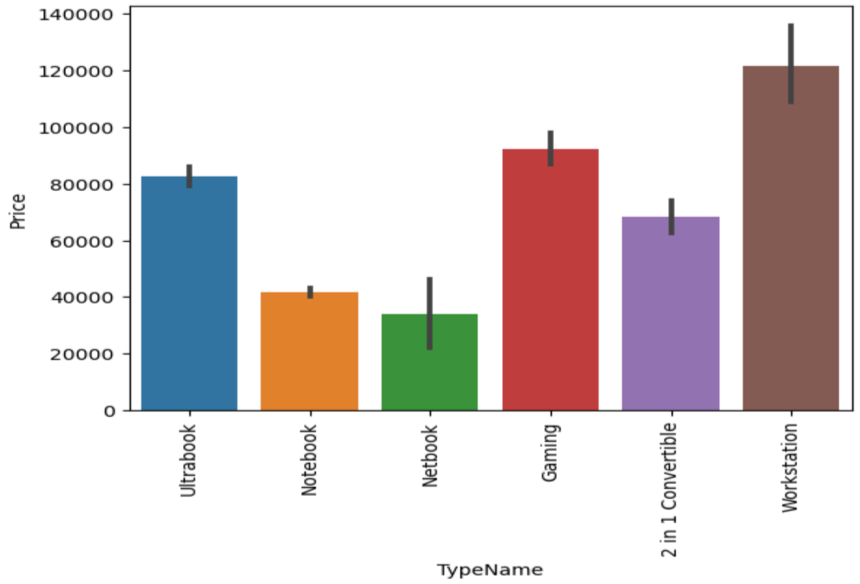
Razer, Apple, LG, Microsoft, Google, MSI laptops are expensive, and others are in the budget range.

Company column

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

In [146]:

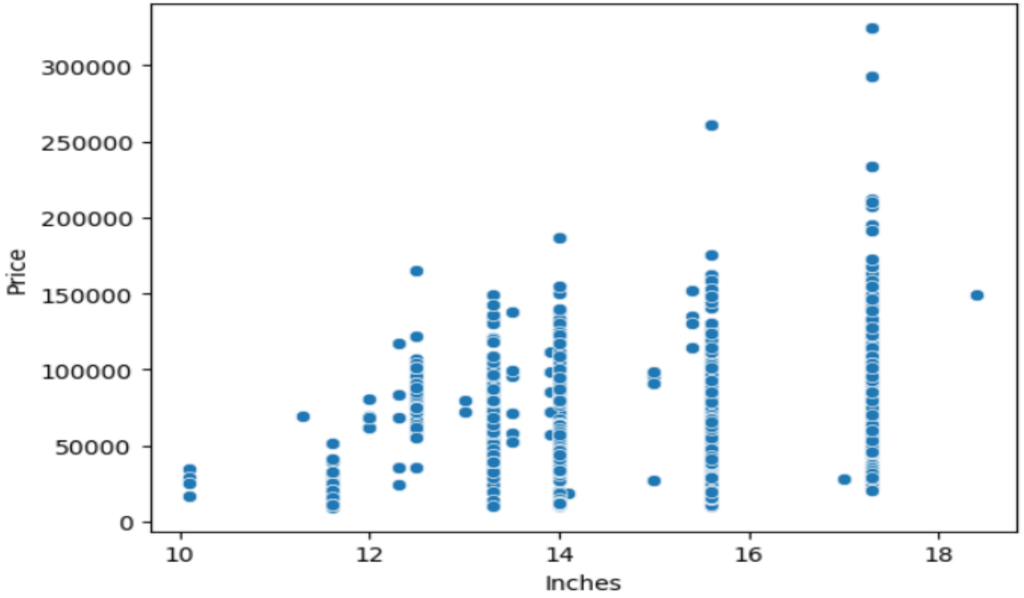
```
sns.barplot(x=df['TypeName'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



Dose the price vary with laptop size in inches ?

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

```
In [148]:  1  sns.scatterplot(x=df['Inches'],y=df['Price'])
Out[148]: <Axes: xlabel='Inches', ylabel='Price'>
```



CPU column

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

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

In [183]:

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

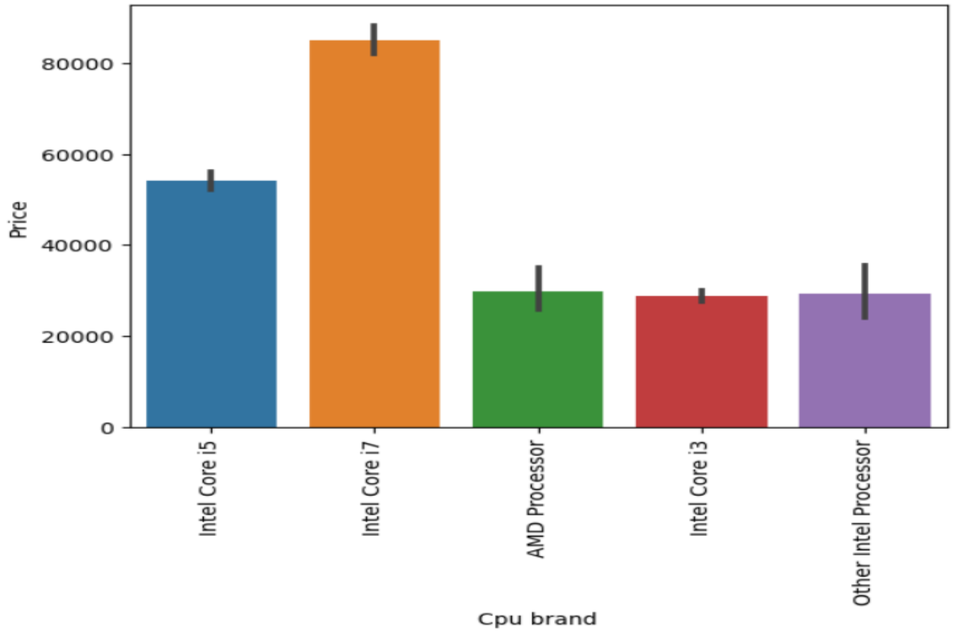
In [184]:

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

How does the price vary with processor?

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

In [188]:
```
sns.barplot(x=df['Cpu brand'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



Memory column

Memory column is again a noisy column that gives an understanding of hard drives . Many laptops came with HHD and SSD both , as well om some there is an external slot present to insert after purchase. So if you use value counts on a column then we are having 4 different categories of memory of memory as HHD , SSD , storage , flash and hybrid .

In [194]:

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

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

df["second"]= new[1]

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

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

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

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

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

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

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

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

GPU variable

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

*If you use the value counts function then there is a row with GPU of ARM so we have removed that row and after extracting the brand GPU column is no longer needed .

In [202]:
```
df['Gpu brand'] = df['Gpu'].apply(lambda x:x.split()[0])
```

In [203]:
```
df.head()
```

Out[203]:

	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	0	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	0	220.534624	Intel Core i7	0	512	AMD
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	0	226.983005	Intel Core i5	0	256	Intel

In [204]:
```
df['Gpu brand'].value_counts()
```

Out[204]:
```
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: Gpu brand, dtype: int64
```

In [205]:
```
df = df[df['Gpu brand'] != 'ARM']
```

In [206]:
```
df['Gpu brand'].value_counts()
```

Out[206]:
```
Intel     722
Nvidia    400
AMD       180
Name: Gpu brand, dtype: int64
```

In [207]:
```
sns.barplot(x=df['Gpu brand'],y=df['Price'],estimator=np.median)
plt.xticks(rotation='vertical')
plt.show()
```

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

Operating system column

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

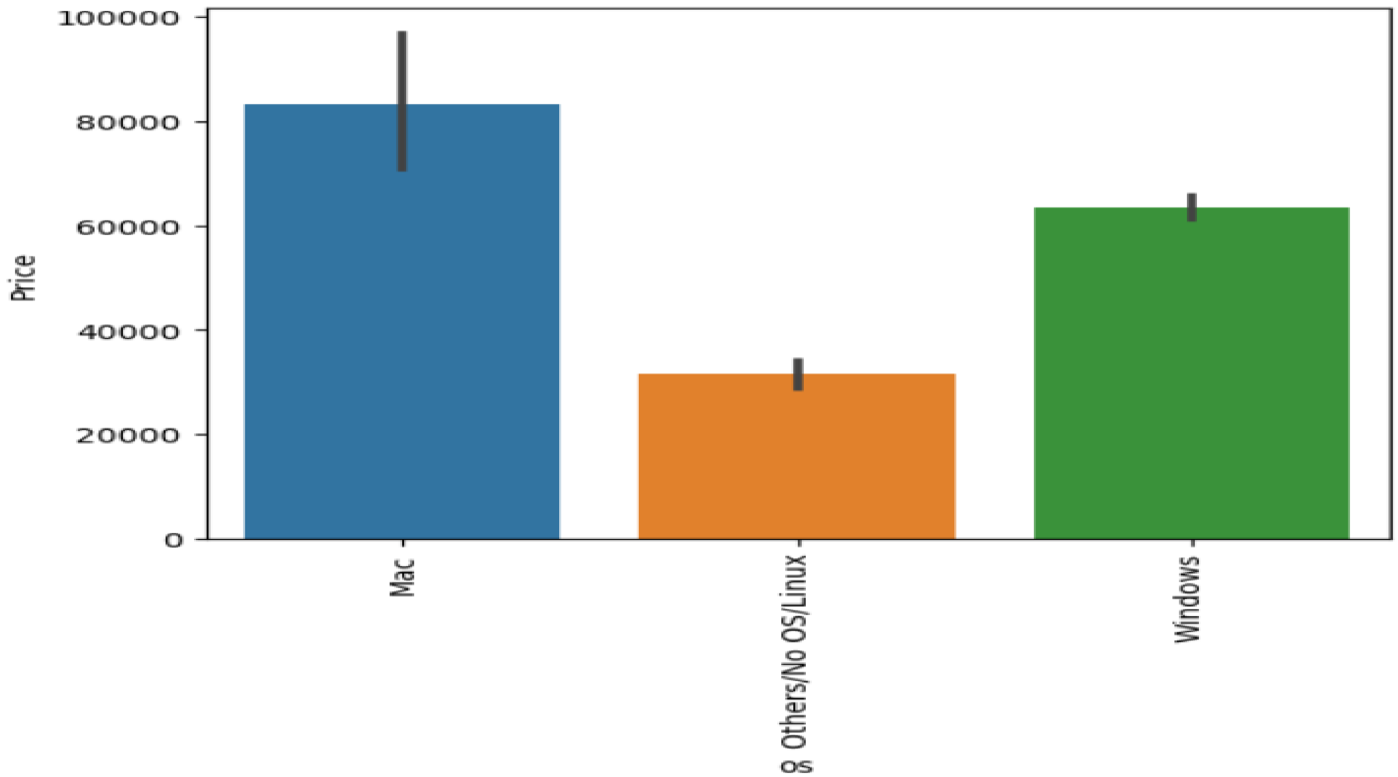
In [212]:

```
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'
```

When you plot price against operating system then as usual mac is most expensive .

In [216]:

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

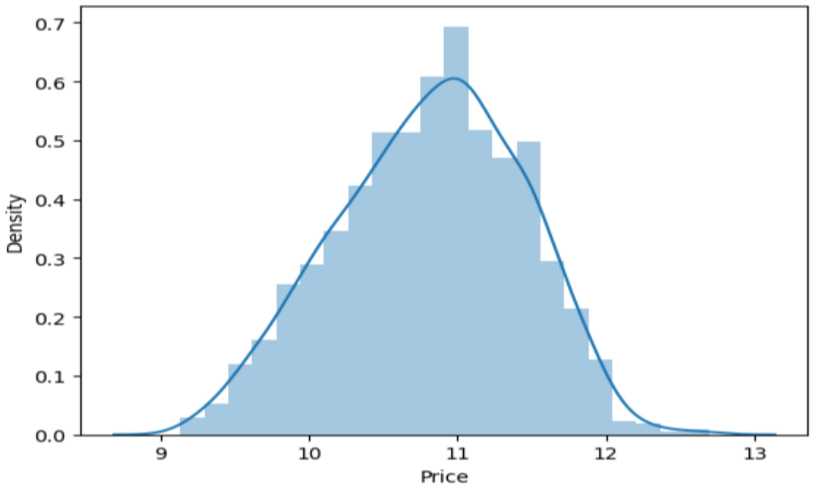


Long –normal transformation

In [95]:
```
sns.distplot(np.log(df['Price']))
```

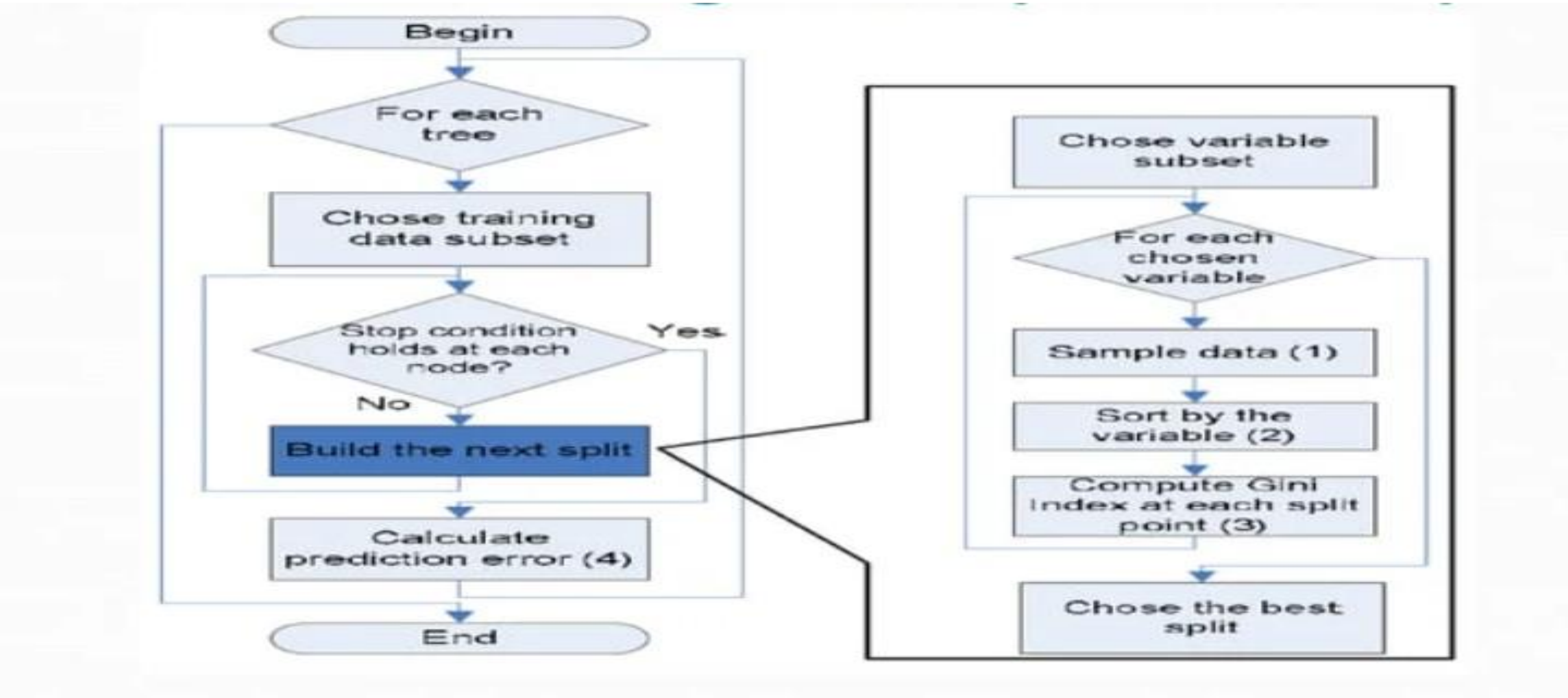
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_16676\3556049916.py:1: UserWarning:

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

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

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

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

Out[95]: <Axes: xlabel='Price', ylabel='Density'>



random forest algorithm

- an ensemble classifier using many decision tree models.
- Can be used for classification or regression .
- Accuracy and variable importance information is provided with the results .



Random forest regression:-

Random forest regression is a supervised learning algorithm and bagging technique that uses an ensemble learning method for regression in machine learning . The tress in random forests run in parallel , meaning there is no interaction between these while building the trees .

Algorithm

```
In [99]:
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)
```

```
In [101]:
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score,mean_absolute_error
```

```
In [102]:
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 SVR

```

Random Forest

```
In [103]:
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))
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: FutureWarning: `sparse` was renamed to `spar
se_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default valu
e.
  warnings.warn(
```

```
R2 score 0.8809930909806516
MAE 0.16395052625497095
```

Thank you