Sudhanshu Narsude 2019130044
Omkar Padir 2019130046
TE COMPS
Batch C
AIML Lab

Mini Project

Problem Statement: Laptop price Prediction using Machine Learning.

Theory:

A random forest regressor:

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Implementation:

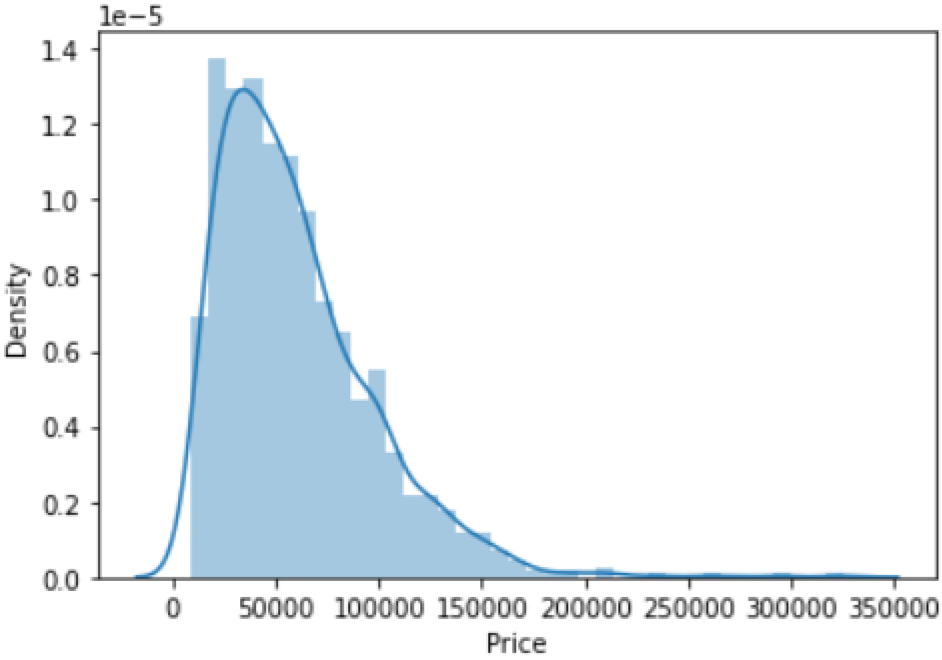
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read_csv("laptop_data.csv")

data.shape

data.isnull().sum()
```

```
data.head()
```

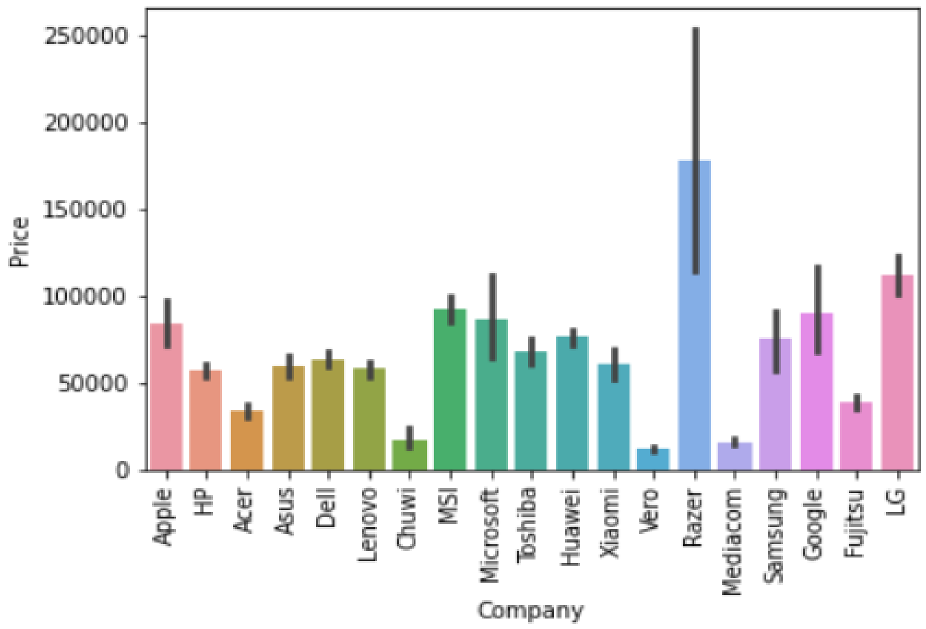
```
data.drop(columns=['Unnamed: 0'],inplace=True)
```

```
## remove gb and kg from Ram and weight and convert the cols to numeric
data['Ram'] = data['Ram'].str.replace("GB", "")
data['Weight'] = data['Weight'].str.replace("kg", "")
data['Ram'] = data['Ram'].astype('int32')
data['Weight'] = data['Weight'].astype('float32')
```

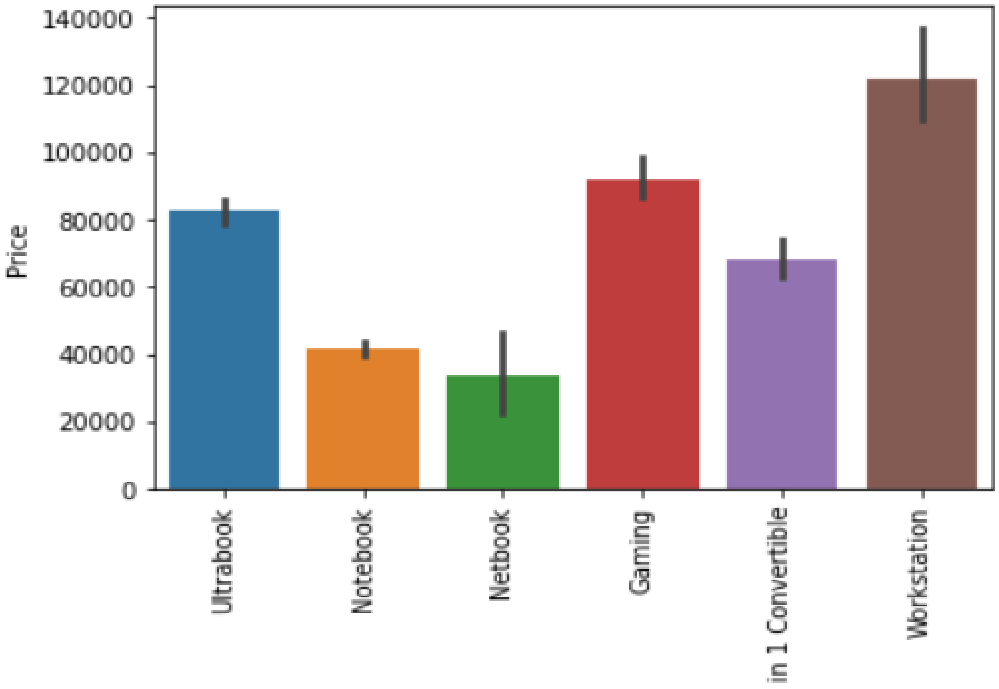
```
sns.distplot(data['Price'])
plt.show()
```



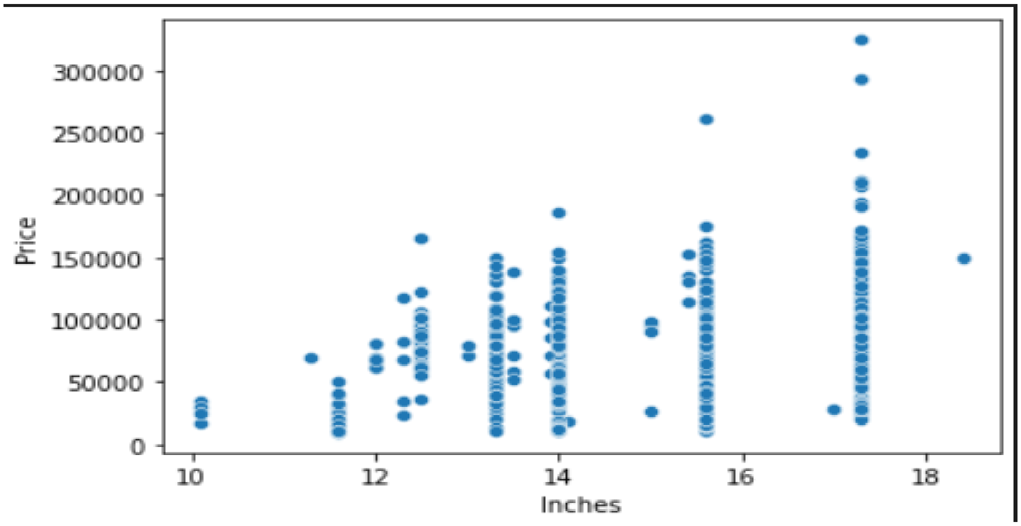
```
#what is avg price of each brand?

sns.barplot(x=data['Company'], y=data['Price'])

plt.xticks(rotation="vertical")

plt.show()
```



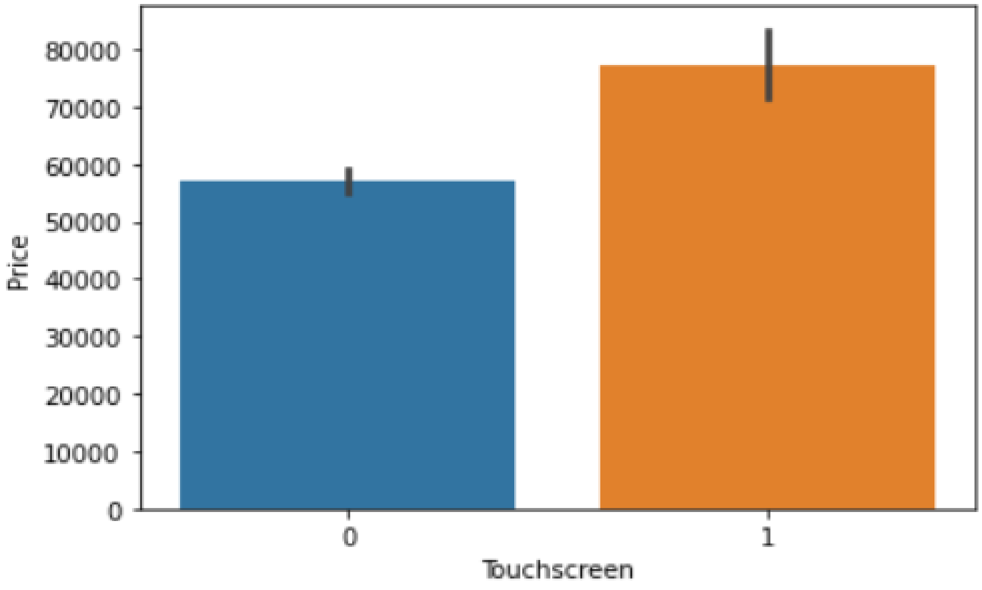
```
#data['TypeName'].value_counts().plot(kind='bar')
sns.barplot(x=data['TypeName'], y=data['Price'])
plt.xticks(rotation="vertical")
plt.show()
```



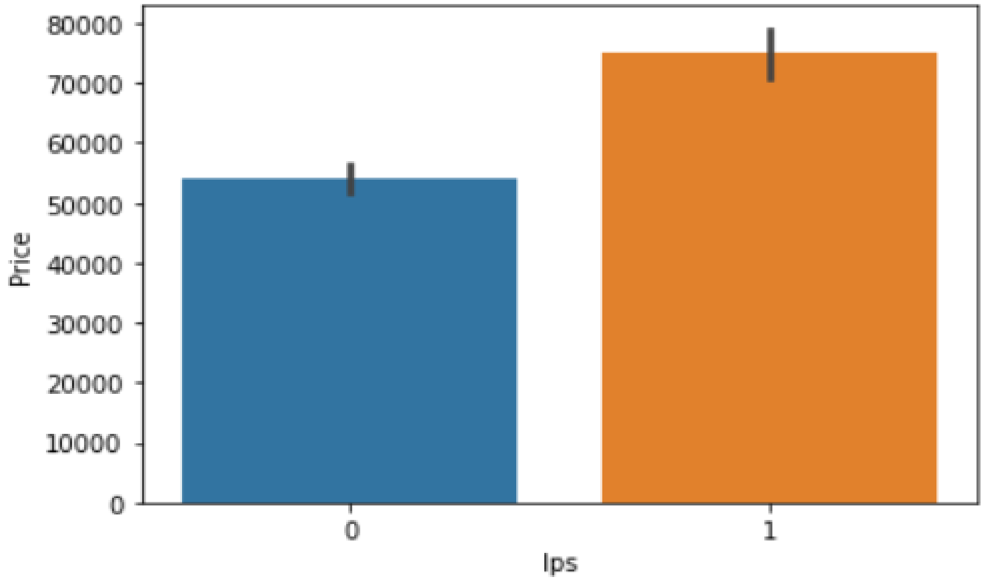
```
sns.scatterplot(x=data['Inches'],y=data['Price'])
```



```
data['Touchscreen'] = data['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in x else 0)
#how many laptops in data are touchscreen
sns.countplot(data['Touchscreen'])
#Plot against price
sns.barplot(x=data['Touchscreen'],y=data['Price'])
```

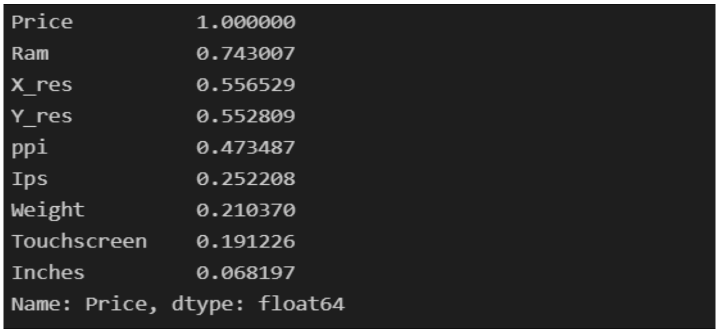


```
#extract IPS column
data['Ips'] = data['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)
sns.barplot(x=data['Ips'],y=data['Price'])
```



```
def findXresolution(s):
 return s.split()[-1].split("x")[0]
def findYresolution(s):
 return s.split()[-1].split("x")[1]
#finding the x_res and y_res from screen resolution
data['X_res'] = data['ScreenResolution'].apply(lambda x: findXresolution(x))
data['Y_res'] = data['ScreenResolution'].apply(lambda y: findYresolution(y))
#convert to numeric
data['X_res'] = data['X_res'].astype('int')
data['Y_res'] = data['Y_res'].astype('int')
```

```
data['ppi'] = (((data['X_res']**2) + (data['Y_res']**2))**0.5/data['Inches']).astype('float')
data.corr()['Price'].sort_values(ascending=False)
```



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

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

```
#first we will extract Name of CPU which is first 3 words from Cpu column and then we will check which processor it is
def fetch_processor(x):
  cpu_name = " ".join(x.split()[0:3])
  if cpu_name == 'Intel Core i7' or cpu_name == 'Intel Core i5' or cpu_name == 'Intel Core i3':
    return cpu_name
  elif cpu_name.split()[0] == 'Intel':
    return 'Other Intel Processor'
  else:
    return 'AMD Processor'
data['Cpu_brand'] = data['Cpu'].apply(lambda x: fetch_processor(x))
```

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

```
sns.barplot(data['Ram'], data['Price'])
plt.show()
```

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

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

data["second"]= new[1]

data["Layer1HDD"] = data["first"].apply(lambda x: 1 if "HDD" in x else 0)
data["Layer1SSD"] = data["first"].apply(lambda x: 1 if "SSD" in x else 0)
data["Layer1Hybrid"] = data["first"].apply(lambda x: 1 if "Hybrid" in x else 0)
```

```
data["Layer1Flash_Storage"] = data["first"].apply(lambda x: 1 if "Flash Storage" in x else 0)

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

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

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

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

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

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

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

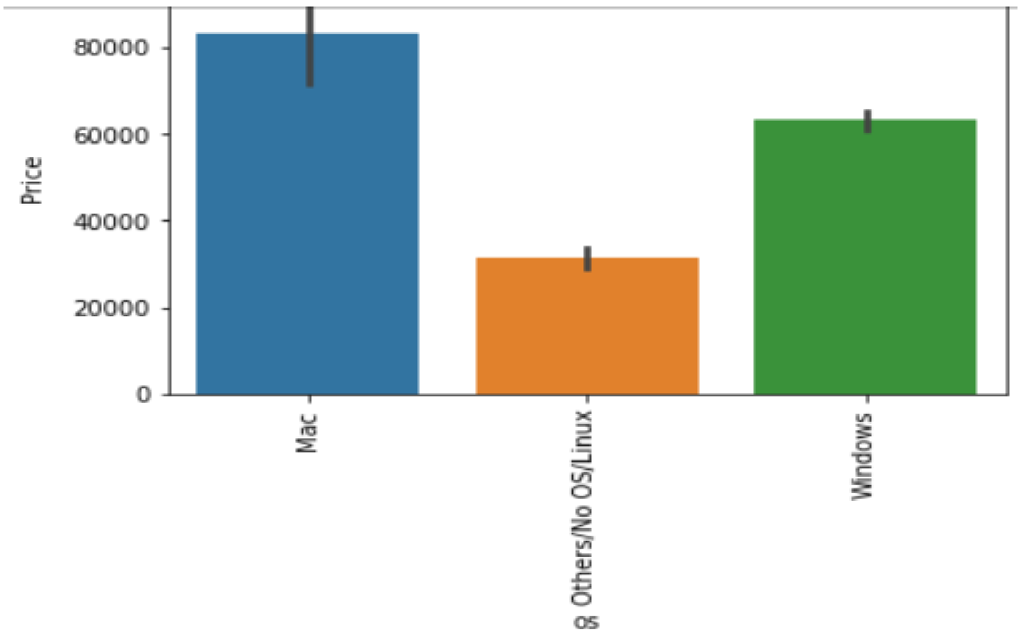
```
data.drop(columns=['Hybrid','Flash_Storage','Memory','Cpu'],inplace=True)
```

```
# Which brand GPU is in laptop
data['Gpu_brand'] = data['Gpu'].apply(lambda x:x.split()[0])
#there is only 1 row of ARM GPU so remove it
data = data[data['Gpu_brand'] != 'ARM']
```

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

```
#Get which OP sys
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'
data['os'] = data['OpSys'].apply(cat_os)
data.drop(columns=['OpSys'],inplace=True)
```

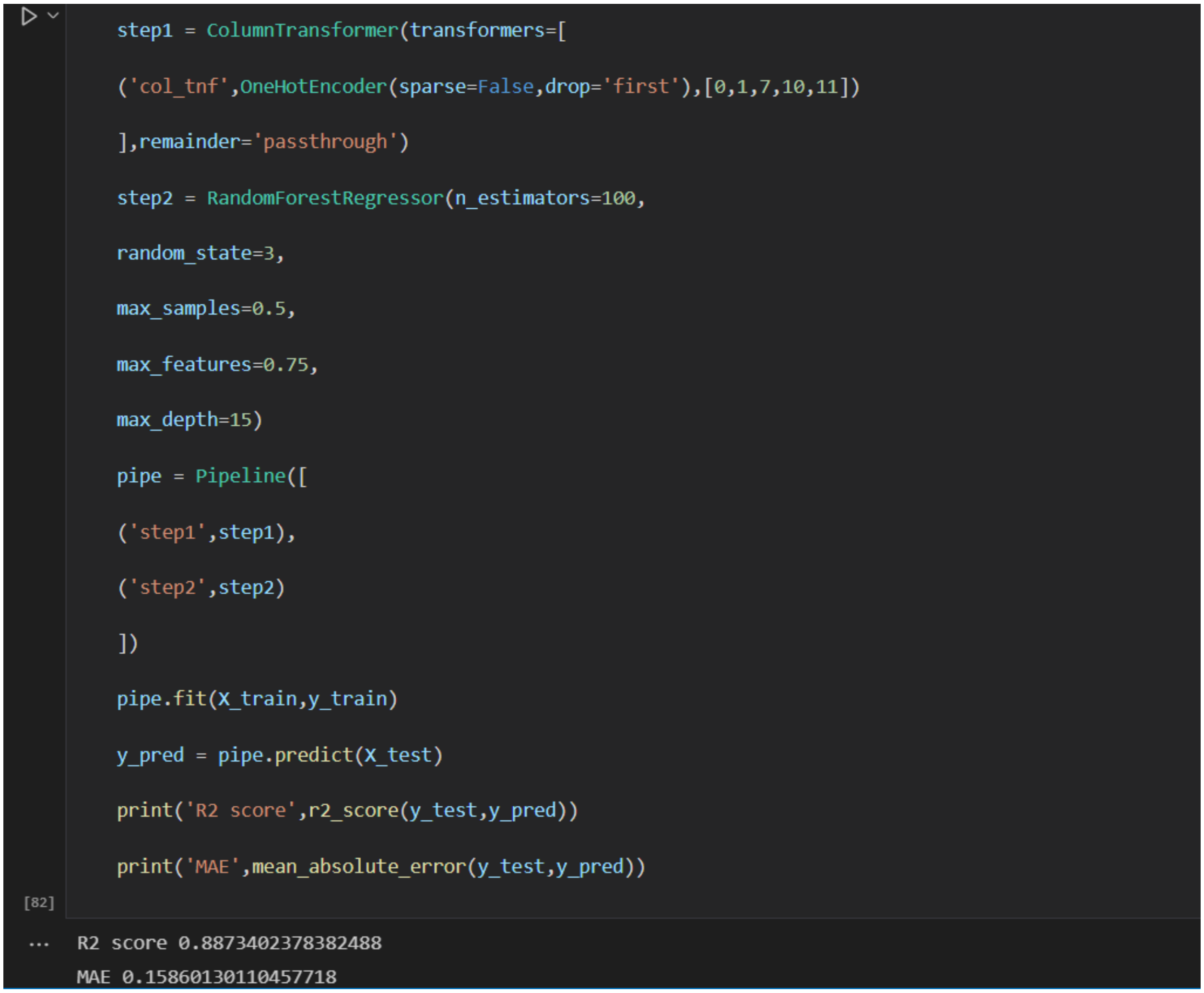
```
sns.barplot(x=data['os'],y=data['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



```
sns.distplot(np.log(data['Price']))
plt.show()
```

```
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score,mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
```

```
X = data.drop(columns=['Price'])

y = np.log(data['Price'])

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.15,random_state=2)
```



```
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))
```

[82]

```
R2 score 0.8873402378382488
MAE 0.15860130110457718
```

```
import pickle
```

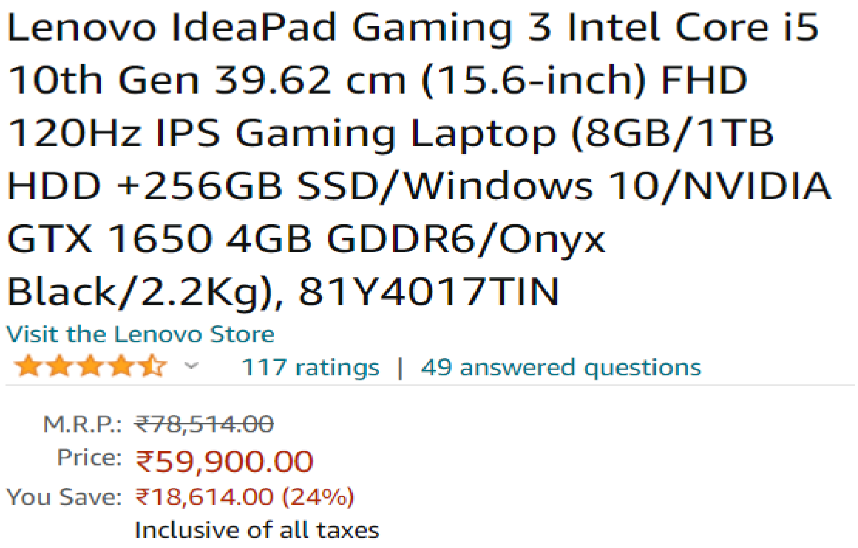
```
data.to_csv("df.csv", index=False)
pickle.dump(pipe,open('pipe.pkl','wb'))
```

App.py :

```
import streamlit as st
import pickle
import numpy as np
import pandas as pd
#load the model and dataframe
df = pd.read_csv("df.csv")
pipe = pickle.load(open("pipe.pkl", "rb"))
st.title("Laptop Price Predictor")
#Now we will take user input one by one as per our dataframe
#Brand
#company = st.selectbox('Brand', df['Company'].unique())
company = st.selectbox('Brand', df['Company'].unique())
#Type of laptop
lap_type = st.selectbox("Type", df['TypeName'].unique())
#Ram
ram = st.selectbox("Ram(in GB)", [2,4,6,8,12,16,24,32,64])
#weight
weight = st.number_input("Weight of the Laptop")
#Touch screen
touchscreen = st.selectbox("TouchScreen", ['No', 'Yes'])
#IPS
ips = st.selectbox("IPS", ['No', 'Yes'])
#screen size
screen_size = st.number_input('Screen Size')
# resolution
resolution = st.selectbox('Screen Resolution',['1920x1080','1366x768','1600x900','3840x2160','3200x1800','2880x1800','2560x1600','2560x1440','2304x1440'])
#cpu
cpu = st.selectbox('CPU',df['Cpu_brand'].unique())
```

```
hdd = st.selectbox('HDD(in GB)',[0,128,256,512,1024,2048])
ssd = st.selectbox('SSD(in GB)',[0,8,128,256,512,1024])
gpu = st.selectbox('GPU',df['Gpu_brand'].unique())
os = st.selectbox('OS',df['os'].unique())
#Prediction
if st.button('Predict Price'):
    ppi = None
    if touchscreen == "Yes":
        touchscreen = 1
    else:
        touchscreen = 0
    if ips == "Yes":
        ips = 1
    else:
        ips = 0
    X_res = int(resolution.split('x')[0])
    Y_res = int(resolution.split('x')[1])
    ppi = ((X_res ** 2) + (Y_res**2)) ** 0.5 / screen_size
    query = np.array([company,lap_type,ram,weight,touchscreen,ips,ppi,cpu,hdd,ssd,gpu,os])
    query = query.reshape(1, 12)
    prediction = str(int(np.exp(pipe.predict(query)[0])))
    st.title("The predicted price of this configuration is " + prediction)
```

Output:



Lenovo IdeaPad Gaming 3 Intel Core i5 10th Gen 39.62 cm (15.6-inch) FHD 120Hz IPS Gaming Laptop (8GB/1TB HDD +256GB SSD/Windows 10/NVIDIA GTX 1650 4GB GDDR6/Onyx Black/2.2Kg), 81Y4017TIN

Visit the Lenovo Store

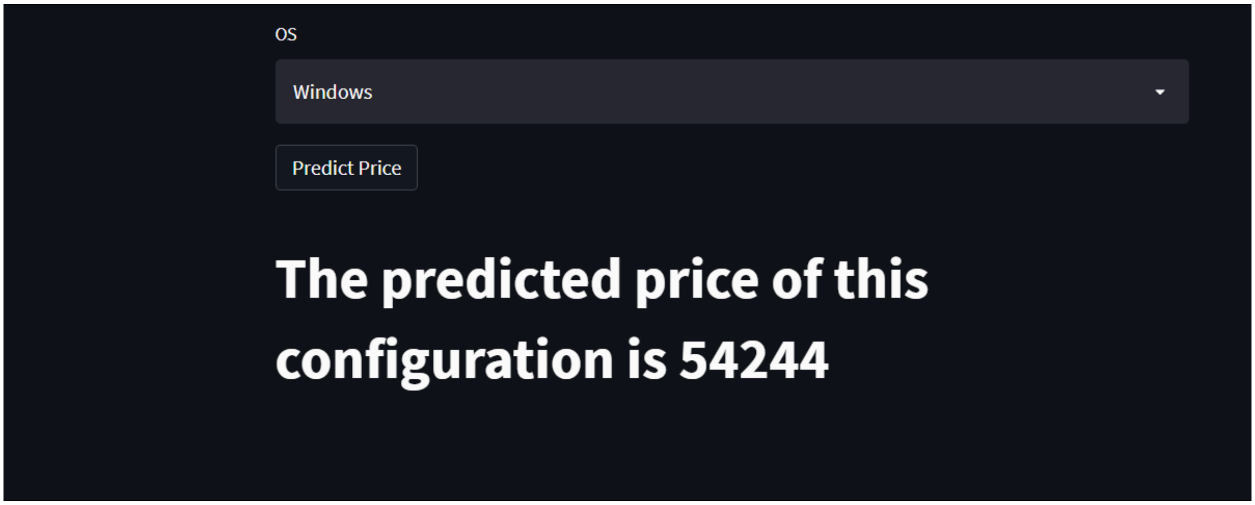
117 ratings | 49 answered questions

M.R.P.: ~~₹78,514.00~~

Price: ₹59,900.00

You Save: ₹18,614.00 (24%)

Inclusive of all taxes



OS: Windows

The predicted price of this configuration is 54244

Conclusion:

In this mini-project we have predicted the price of laptop’s using Machine Learning. The data is the latest one . To proceed 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, if there are any NULL values present in the dataset. At the required step, we will also perform preprocessing and feature engineering tasks. Our aim in performing in-depth EDA is to prepare and clean data for better machine learning modeling to achieve high performance and generalized models. Now we have prepared our data and hold a better understanding of the dataset. Then we had implemented 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. We applied a random forest regressor in our model which gives the r2 score

(Coefficient of determination) of .89 approx. Also we were able to use streamlit to create a webapp to predict the output and display it to the user. We were able to achieve 90% of accuracy in finding out the price of a laptop.