Annexure I

A Report

on

Laptop Price Prediction: Data Science & ML Project

SUBMITTED TO SCHOOL OF CSE, LPU

For the partial fulfilment of the

Summer Training

Under the guidance of
Pranay Sharma, Allsoft Solutions

GitHub
https://github.com/abhikritimoti/-Laptop_Price_Prediction.git

Submitted by
Name – Abhikriti Moti
Reg Num – 12007073

Annexure II

STUDENT'S DECLARATION

I hereby certify that the work which is being done in the project entitled "**LAPTOP PRICE PREDICTION**" by "**ABHIKRITI MOTI,** as per course requirement of **SUMMER TRAINING** submitted to SCHOOL OF COMPUTER SCIENCE - **LPU,** under **UNIVERSITY** is record of our own work carried out under the supervision of "Pranay Sharma" ALLSOFT SOLUTIONS – IBM CE.

Abhikriti Moti

Name of the Student

20th JULY 2022

DATE

Annexure III

Certification by the ALLSOFT SOLUTIONS – IBM CE for completion of project.

ACHIEVEMENT

Allsoft Solutions
An ISO 9001 : 2008 Certified Company

PROJECT COMPLETION CERTIFICATE

In recognition of the commitment to achieve professional excellence this is to certify that Ms./Mr.

Abhikriti Moti

has successfully completed an Industry-oriented project.

Project Name	Laptop Price Prediction
Technologies Used	seaborn, pickle, streamlit, numpy, pandas, matplotlib, Decision Tree, Linear Reg, Ridge Reg, KNN
Reference No.	AIP/CEP2021/IN/13149
Training Date	June 2022 – July 2022
Training Duration	6 Weeks
Training Location	Online Live Mode

Program Co-ordinator
Industry/Academic Alliance

Business Partner IBM

Director
Training and Development
Allsoft Solutions and Services

BIG DATA - ANALYTICS | IoT | ORACLE | J2EE | PHP | CLOUD COMPUTING

Verify at https://coursera.org/verify/professional-cert/D6YB6XXCWSKJ

This additional course was done in continuation with the summer training project.

[Click to verify](#)

[Click to verify](#)

Table of contents

Contents

Problem Statement

I'm going to make a project for Laptop price prediction. The problem statement is that if any user wants to buy a laptop, then the application should be compatible to provide a tentative price of laptop according to the user configurations. While this looks like a simple project or just developing a model, the dataset we have is noisy and needs lots of feature engineering, and pre-processing that will drive your interest in developing this project.

Dataset used for the project

I have downloaded the dataset from Kaggle: https://www.kaggle.com/code/danielbethell/laptop-prices-prediction/data

Most of the columns in a dataset are noisy and contain lots of information. But with feature engineering you do, you will get more good results.

The only problem is we are having less data, but we will obtain a good accuracy over it. The only good thing is it is better to have a large data. we will develop a website that could predict a tentative price of a laptop based on user configuration.

Basic Understanding of Laptop Price Prediction Data

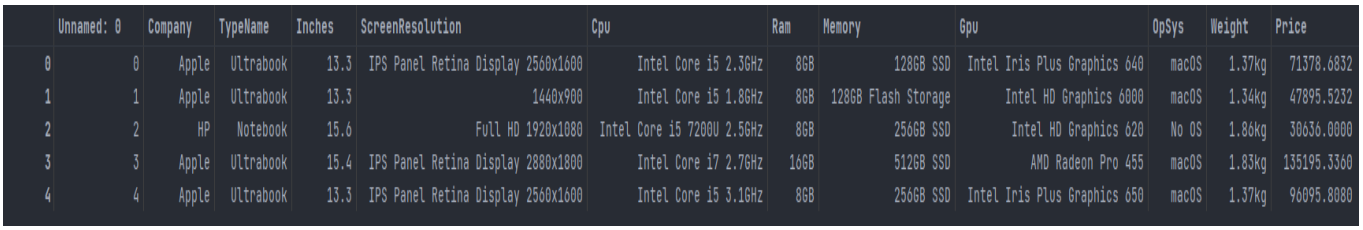
Now let us start working on a dataset in our Jupyter Notebook.

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.

```
import numpy as np
import pandas as pd

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

df.head()
```



	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0000
3	3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.3360
4	4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.8080

```
df.shape
```

```
(1303, 12)
```

It is good that there are no NULL values. And we need little changes in weight and Ram column to convert them to numeric by removing the unit written after value. So, we will perform data cleaning here to get the correct types of columns.

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

```
Table: 5 rows × 11 columns
```

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

```
df.head()
```

```
Table: 5 rows × 11 columns
```

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

EDA of Laptop Price Prediction Dataset

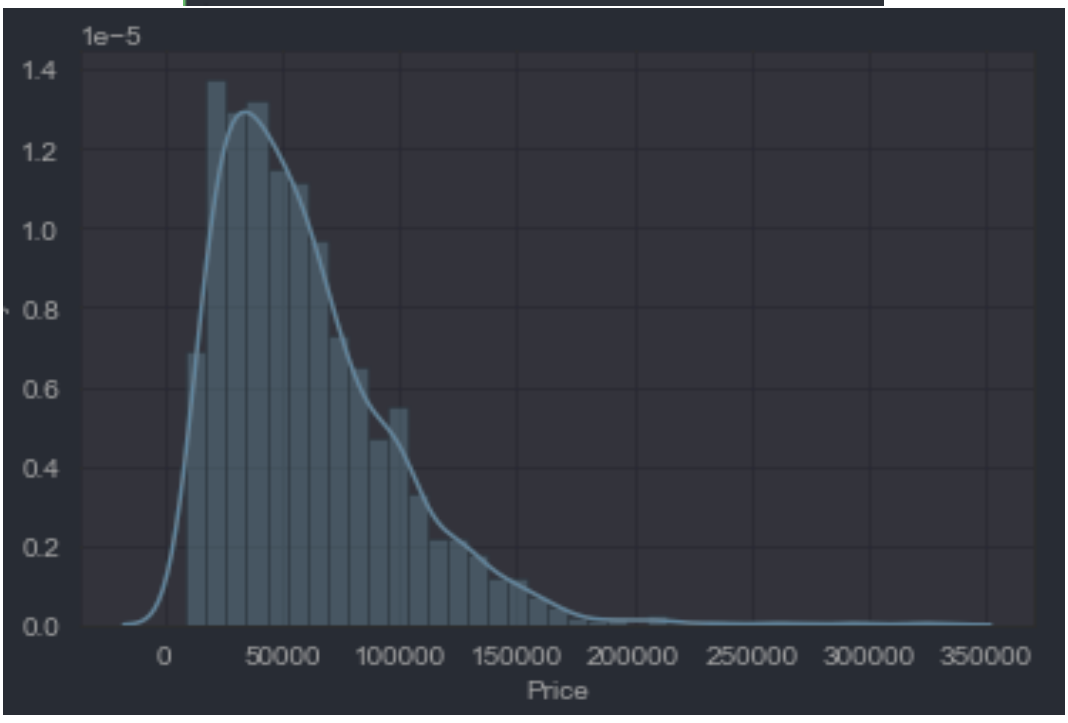
Exploratory analysis is a process to explore and understand the data and data relationship in a complete depth so that it makes feature engineering and machine learning modeling steps smooth and streamlined for prediction. EDA involves Univariate, Bivariate, or Multivariate analysis. EDA helps to prove our assumptions true or false. In other words, it helps to perform hypothesis testing.

We will start from the first column and explore each column and understand what impact it creates on the target column. At the required step, we will also perform pre-processing 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. so, let's get started with analysing and preparing the dataset for prediction.

1) Distribution of target column

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

```
import seaborn as sns
sns.distplot(df['Price'])
```



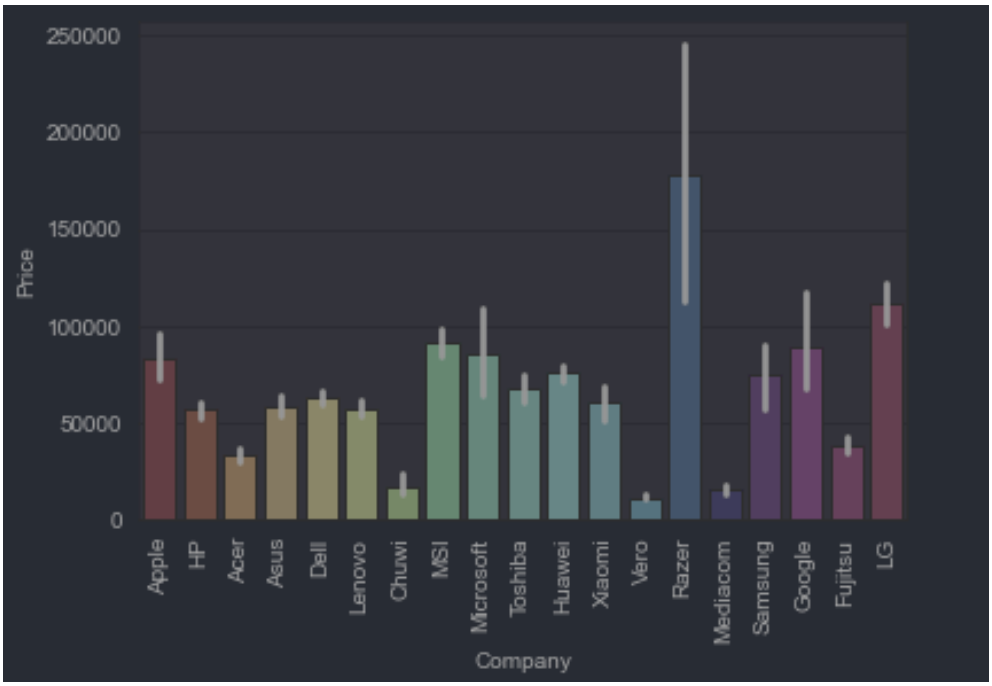
The distribution of the target variable is skewed, and it is obvious that commodities with low prices are sold and purchased more than the branded ones.

2) 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.

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

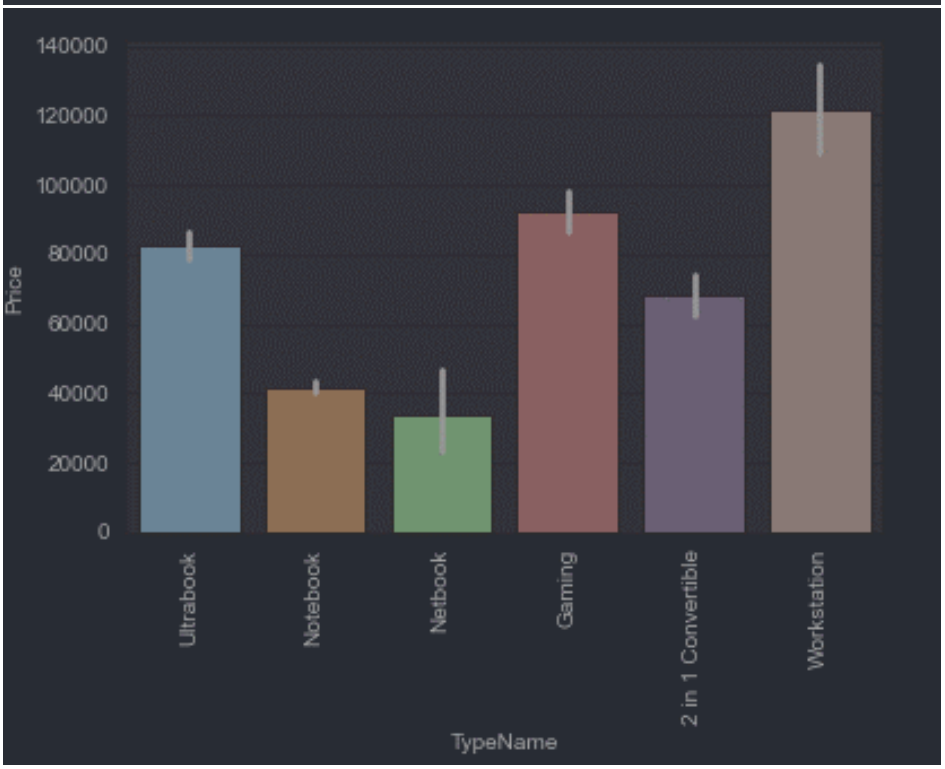


Razer, Apple, LG, Microsoft, Google, MSI laptops are expensive, and others are in the budget range.

3) Type of laptop

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.

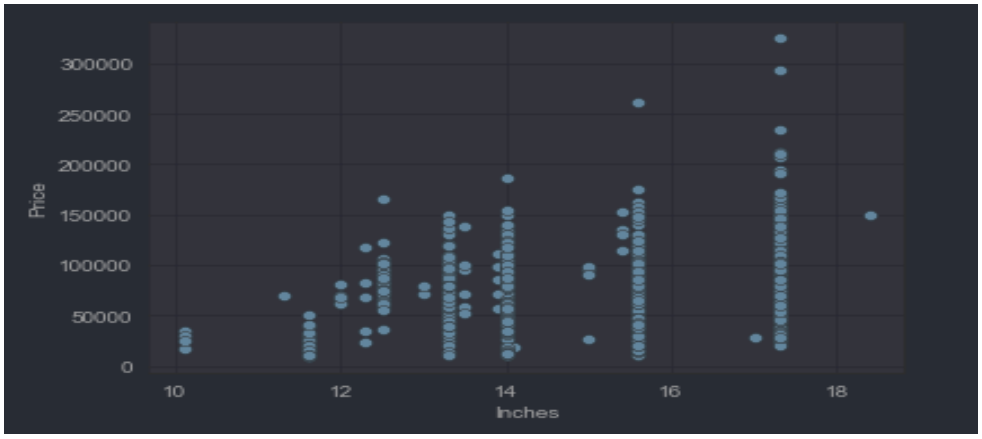
```
sns.barplot(x=df['TypeName'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



4) Does 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 strong relationship between the price and size column.

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



Feature Engineering and Pre-processing of Laptop Price Prediction Model

Feature engineering is a process to convert raw data to meaningful information. there are many methods that come under feature engineering like transformation, categorical encoding, etc.

Now the columns we have are noisy, so we need to perform some feature engineering steps.

5) Screen Resolution

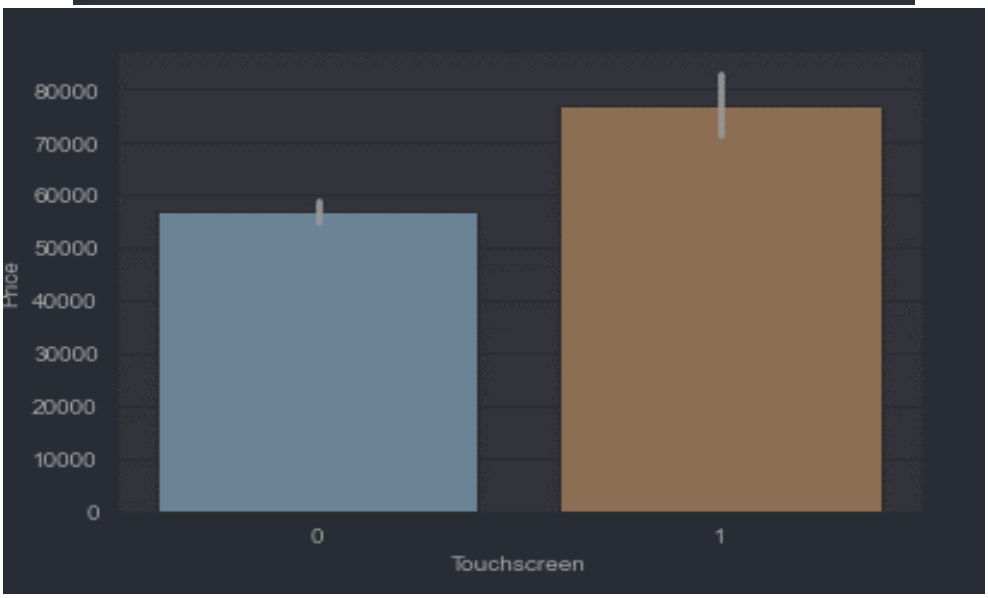
Screen resolution contains lots of information. before any analysis first, we need to perform feature engineering over it. If you observe unique values of the column then we can see that all value gives information related to the presence of an IPS panel, are a laptop touch screen or not, and the X-axis and Y-axis screen resolution. So, we will extract the column into 3 new columns in the dataset.

Extract Touch screen information

It is a binary variable so we can encode it as 0 and 1. one means the laptop is a touch screen and zero indicates not a touch screen.

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

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



Extract IPS Channel presence information

It is a binary variable, and the code is the same we used above. The laptops with IPS channel are present less in our data but by observing relationship against the price of IPS channel laptops are high.

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

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

Extract X-axis and Y-axis screen resolution dimensions

Now both the dimension is present at end of a string and separated with a cross sign. So first we will split the string with space and access the last string from the list. then split the string with a cross sign and access the zero and first index for X and Y-axis dimensions.

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

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

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

Replacing inches, X and Y resolution to PPI

If you find the correlation of columns with price using the **corr** method then we can see that inches do not have a strong correlation but X and Y-axis resolution have a very strong resolution so we can take advantage of it and convert these three columns to a single column that is known as Pixel per inches (PPI). In the end, our goal is to improve the performance by having fewer features.

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

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

Now when you will see the correlation of price then PPI is having a strong correlation.

```
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
ppi            0.473487
Name: Price, dtype: float64
```

So now we can drop the extra columns which are not of use. At this point, we have started keeping the important columns in our dataset.

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

6) CPU column

If you observe 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 pre-processors in laptops and speed.

```
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'
```

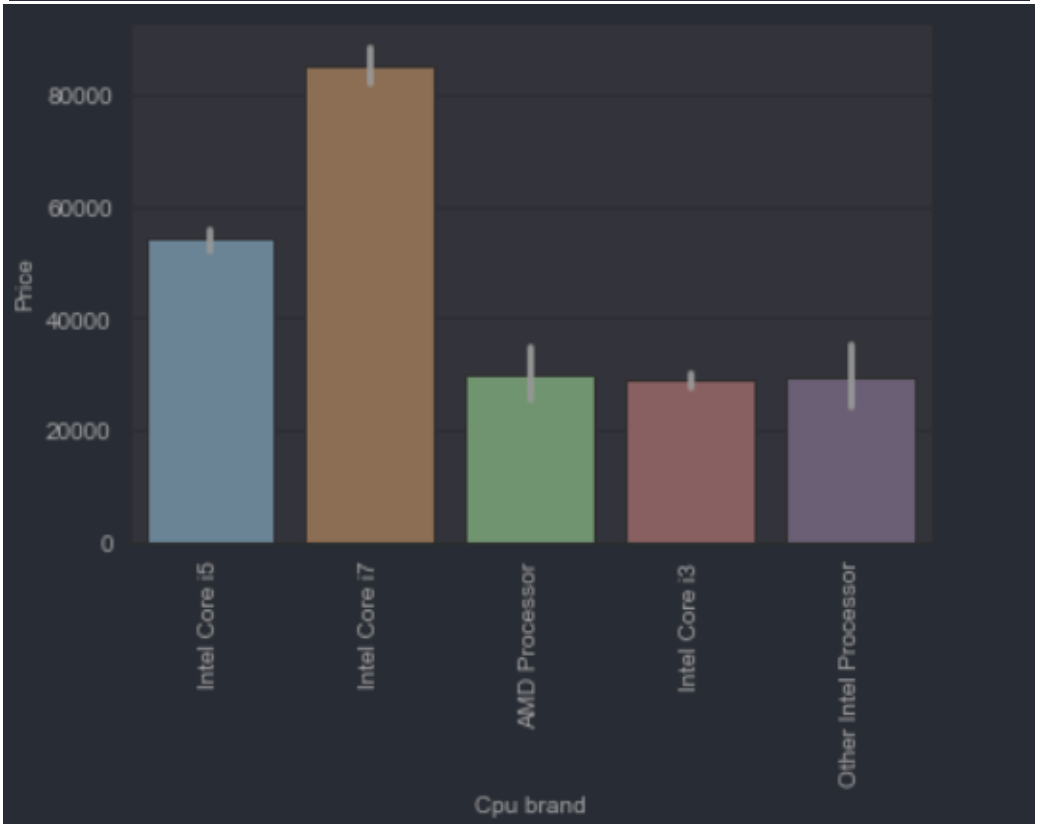
```
df['Cpu brand'] = df['Cpu Name'].apply(fetch_processor)
```

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

How does the price vary with processors?

we can again use our 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 pre-processor.

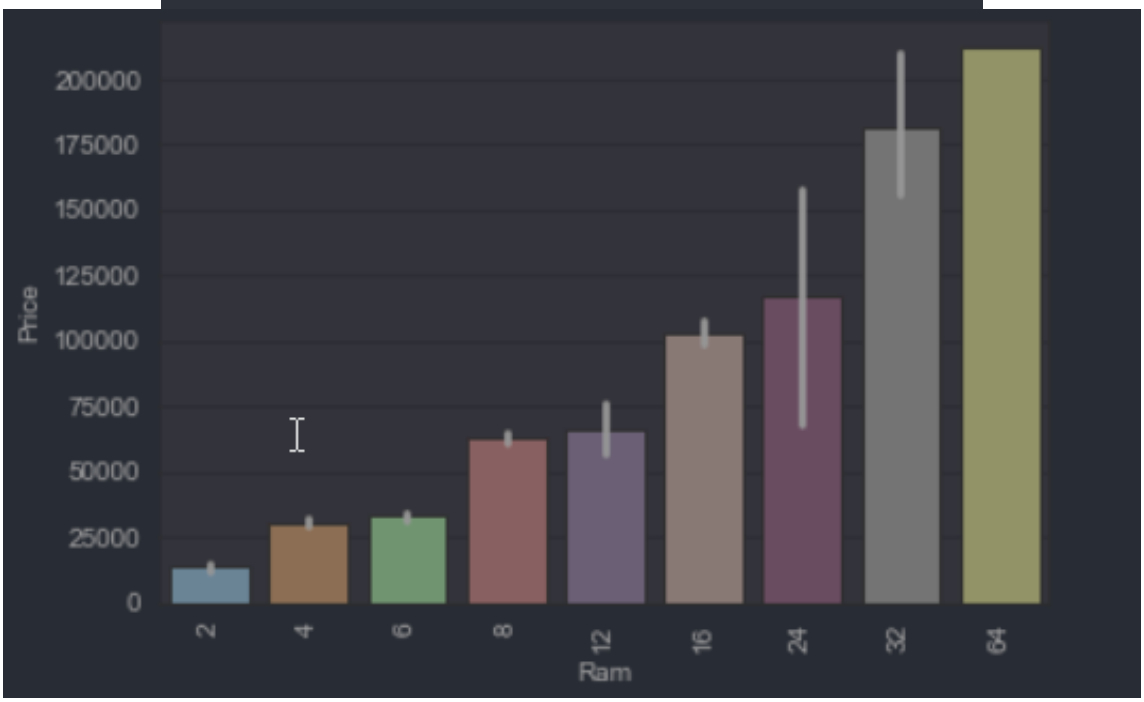
```
sns.barplot(x=df['Cpu brand'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



7) Price with Ram

Again, Bivariate analysis of price with Ram. If you observe the plot, then Price is having a very strong positive correlation with Ram, or you can say a linear relationship.

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



8) 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 in some there is an external slot present to insert after purchase. This column can disturb your analysis if not feature engineer it properly. So If you use value counts on a column then we are having 4 different categories of memory as HHD, SSD, Flash storage, and hybrid

```
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["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)
```

First, we have cleaned the memory column and then made 4 new columns which are a binary column where each column contains 1 and 0 indicate that amount four is present and which is not present. Any laptop has a single type of memory or a combination of two. so in the first column, it consists of the first memory size and if the second slot is present in the laptop then the second column contains it else we fill the null values with zero.

After that in a particular column, we have multiplied the values by their binary value. It means that if in any laptop particular memory is present then it contains binary value as one and the first value will be multiplied by it, and same with the second combination. For the laptop which does have a second slot, the value will be zero multiplied by zero is zero.

Now when we see the correlation of price then Hybrid and flash storage have very less or no correlation with a price. We will drop this column with CPU and memory which is no longer required.

```
In 59   1  df.drop(columns=['Memory'],inplace=True)
```

9) 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, 12 Gb) graphic card is present. so we will simply extract the name of the brand.

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

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.

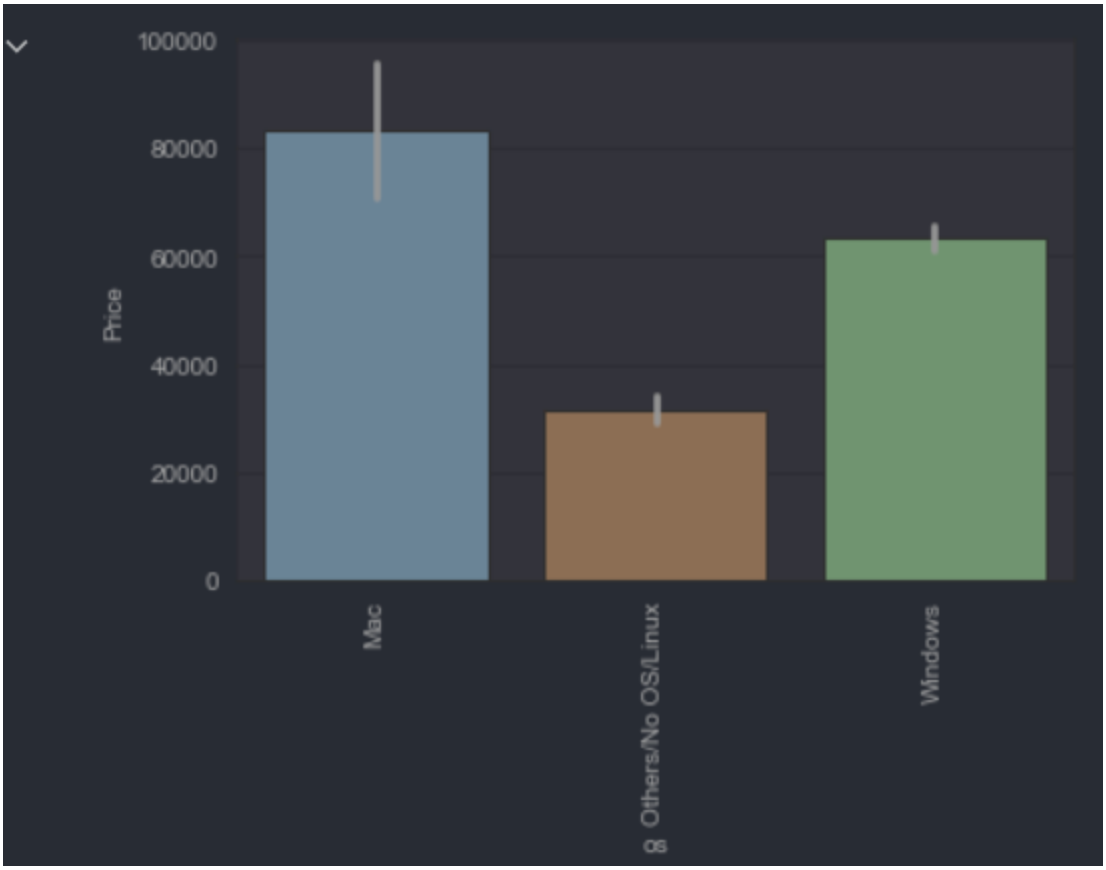
10) Operating System Column

There are many categories of operating systems. we will keep all windows categories in one, Mac 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.

```
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)
```

when you plot price against operating system then as usual Mac is most expensive.

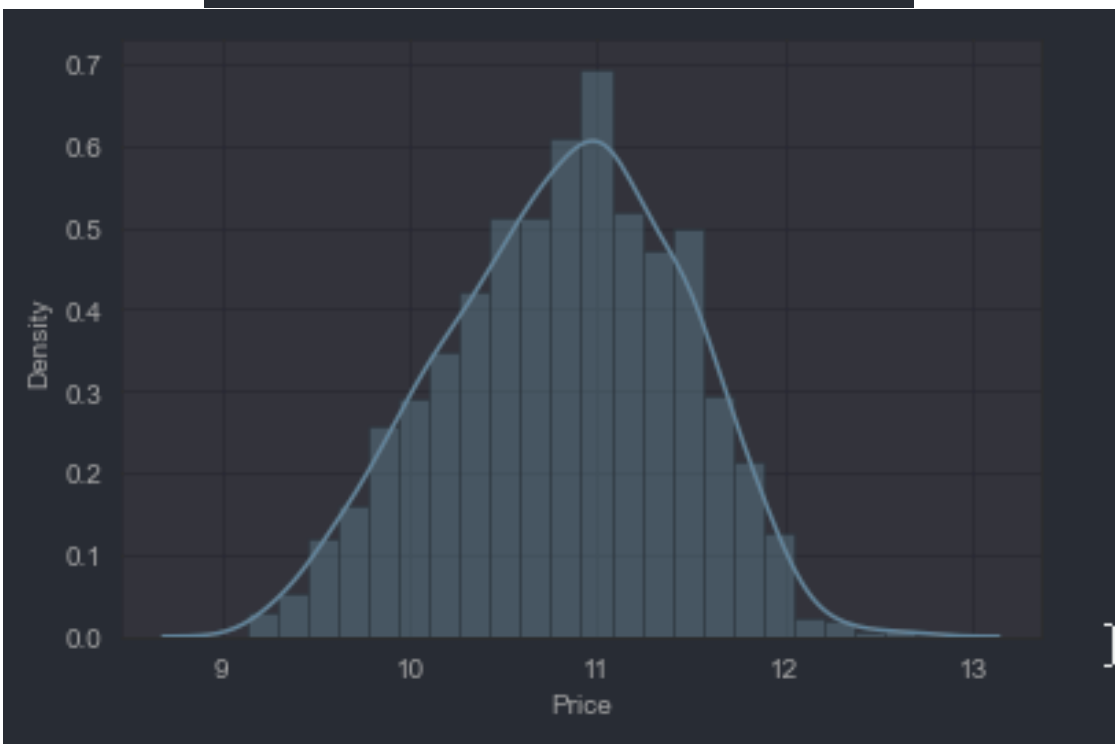



```
In 78   1  sns.barplot(x=df['os'],y=df['Price'])
        2  plt.xticks(rotation='vertical')
        3  plt.show()
```

Log-Normal Transformation

we saw the distribution of the target variable above which was right-skewed. By transforming it to normal distribution performance of the algorithm will increase. we take the log of values that transform to the normal distribution which you can observe below. So while separating dependent and independent variables we will take a log of price, and in displaying the result perform exponent of it.

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



Machine Learning Modelling

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

Import Libraries

We have imported libraries to split data, and algorithms you can try. At a time, we do not know which is best so you can try all the imported algorithms.

```
In 91  1  from sklearn.compose import ColumnTransformer
       2  from sklearn.pipeline import Pipeline
       3  from sklearn.preprocessing import OneHotEncoder
       4  from sklearn.metrics import r2_score,mean_absolute_error
```

```
In 92  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
       from xgboost import XGBRegressor
```

Split in train and test

As discussed, we have taken the log of the dependent variables. And the training data looks something below the dataframe.

```
X = df.drop(columns=['Price'])
y = np.log(df['Price'])
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.15,random_state=2)
```

```
X_train
```

	Company	TypeName	Ram	Weight	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD
183	Toshiba	Notebook	8	2.00	0	0	100.454670	Intel Core i5	0	1
1141	MSI	Gaming	8	2.40	0	0	141.211998	Intel Core i7	1000	1
1049	Asus	Netbook	4	1.20	0	0	135.094211	Other Intel Processor	0	
1020	Dell	2 in 1 Convertible	4	2.08	1	1	141.211998	Intel Core i3	1000	
878	Dell	Notebook	4	2.18	0	0	141.211998	Intel Core i5	1000	1
1066	Asus	Gaming	64	3.58	0	1	127.335675	Intel Core i7	0	1(
315	Dell	Notebook	8	2.33	0	0	141.211998	Intel Core i5	1000	1
1003	HP	Notebook	4	1.64	0	0	111.935204	Intel Core i5	500	

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.

Linear Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:

```
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 0.8073277448418656
MAE 0.21017827976428838
```

Ridge Regression

Ridge regression is one of the types of linear regression in which a small amount of bias is introduced so that we can get better long-term predictions.

Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as L2 regularization.

In this technique, the cost function is altered by adding the penalty term to it. The amount of bias added to the model is called Ridge Regression penalty. We can calculate it by multiplying with the lambda to the squared weight of each individual feature.

```
In 95
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = Ridge(alpha=10)

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 0.8127331031311809
MAE 0.20926802242582965
```

Lasso Regression

Lasso regression stands for Least Absolute Shrinkage and Selection Operator. It adds penalty term to the cost function. This term is the absolute sum of the coefficients. As the value of coefficients increases from 0 this term penalizes, cause model, to decrease the value of coefficients in order to reduce loss. The difference between ridge and lasso regression is that it tends to make coefficients to absolute zero as compared to Ridge which never sets the value of coefficient to absolute zero.

```
In 96
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = Lasso(alpha=0.001)

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 0.8071853945317105
MAE 0.21114361613472565
```

K Neighbors Regressor - KNN

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

```
In 97
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 0.803148868705085
MAE 0.19264883332948868
```

Decision Tree

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

```
In 98
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 0.8398192951898331
MAE 0.1833331863869568
```

Random forest

Random Forest is a popular machine learning algorithm that belongs to the

supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

```
In 99
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
```

Gradient Boost

Gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As we know that the errors in machine learning algorithms are broadly classified into two categories i.e. Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model.

Unlike, Adaboosting algorithm, the base estimator in the gradient boosting algorithm cannot be mentioned by us. The base estimator for the Gradient Boost algorithm is fixed and i.e. Decision Stump. Like, AdaBoost, we can tune the n_estimator of the gradient boosting algorithm. However, if we do not mention the value of n_estimator, the default value of n_estimator for this algorithm is 100.

Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is Mean Square Error (MSE) and when it is used as a classifier then the cost function is Log loss.

```
In 100
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 0.8821257337313104
MAE 0.15930208416538152
```

In the first step for categorical encoding, we passed the index of columns to encode, and pass-through means pass the other numeric columns as it is. The best accuracy I got is with all-time favourite Random Forest. But you can use this code again by changing the algorithm and its parameters. I am showing Linear Regression.

Exporting the Model

Now we have done with modelling. we will save the pipeline object for the development of the project website. we will also export the data frame which will be required to create dropdowns in the website.

```
import pickle

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

Create Web Application for Deployment

Now we will use streamlit to create a web app to predict laptop prices. In a web application, we need to implement a form that takes all the inputs from users that we have used in a dataset, and by using the dumped model we predict the output and display it to a user.

Streamlit

Streamlit is an open-source web framework written in Python. It is the fastest way to create data apps and it is widely used by data science practitioners to deploy machine learning models. To work with this it is not important to have any knowledge of frontend languages.

Streamlit contains a wide variety of functionalities, and an in-built function to meet your requirement. It provides you with a plot map, flowcharts, slider, selection box, input field, the concept of caching, etc. install streamlit using the below pip command.

```
> pip install streamlit
```

create a file named app.py in the same working directory where we will write code for streamlit.

```python
import streamlit as st
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# import the model
pipe = pickle.load(open('pipe.pkl','rb'))
df = pickle.load(open('df.pkl','rb'))

st.image("image.jpg")
st.title("Laptop Price Prediction :chart_with_upwards_trend:")
```

```
st.markdown("""###
#### This model predicts laptop price according to the user input configuration """

st.sidebar.subheader("Correlation b/w Laptop configurations")
hm = pd.read_csv('clean.csv')
fig = plt.figure(figsize=(5, 3)) #facecolor= "#262730"
sns.heatmap(hm.corr())
sns.set_style("white")
st.sidebar.pyplot(fig)

st.sidebar.title("Source Code")
st.sidebar.image("git.png")
st.sidebar.write("[Click here](https://github.com/abhikritimoti/-Laptop_Price_Predi
st.sidebar.markdown("""### <br>
## :copyright: Abhikriti Moti <br> Email: <a href="abhikriti.12007073@lpu.in">abhik


company = st.selectbox('Brand',df['Company'].unique())

type = st.selectbox('Type',df['TypeName'].unique())

ram = st.selectbox('RAM (in GB)',[2,4,6,8,12,16,24,32,64])

weight = st.number_input('Weight of the Laptop')

touchscreen = st.selectbox('Touchscreen',['No','Yes'])

ips = st.selectbox('IPS',['No','Yes'])

screen_size = st.number_input('Screen Size')

resolution = st.selectbox('Screen Resolution',['1920x1080','1366x768','1600x900','3840x2

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())
```

```python
if st.button('Predict Price'):
    # query
    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, type, ram, weight, touchscreen, ips, ppi, cpu, hdd, ssd, gpu,

    query = query.reshape(1,12)
    st.markdown("## The predicted price of this configuration is ")
    st.markdown("# ₹ " + str(int(np.exp(pipe.predict(query)[0]))))

st.markdown(""" ## <br> """, True)
st.markdown("""---""")
st.markdown(""" ## Dataset Used :mag_right: <br>
####  [kaggle link](https://www.kaggle.com/code/danielbethell/laptop-prices-prediction/data)
data = pd.read_csv('laptop_data.csv')
st.dataframe(data)
```

Explanation – First we load the data frame and model that we have saved. After that, we create an HTML form of each field based on training data columns to take input from users. In categorical columns, we provide the first parameter as input field name and second as select options which is nothing but the unique categories in the dataset. In the numerical field, we provide users with an increase or decrease in the value.

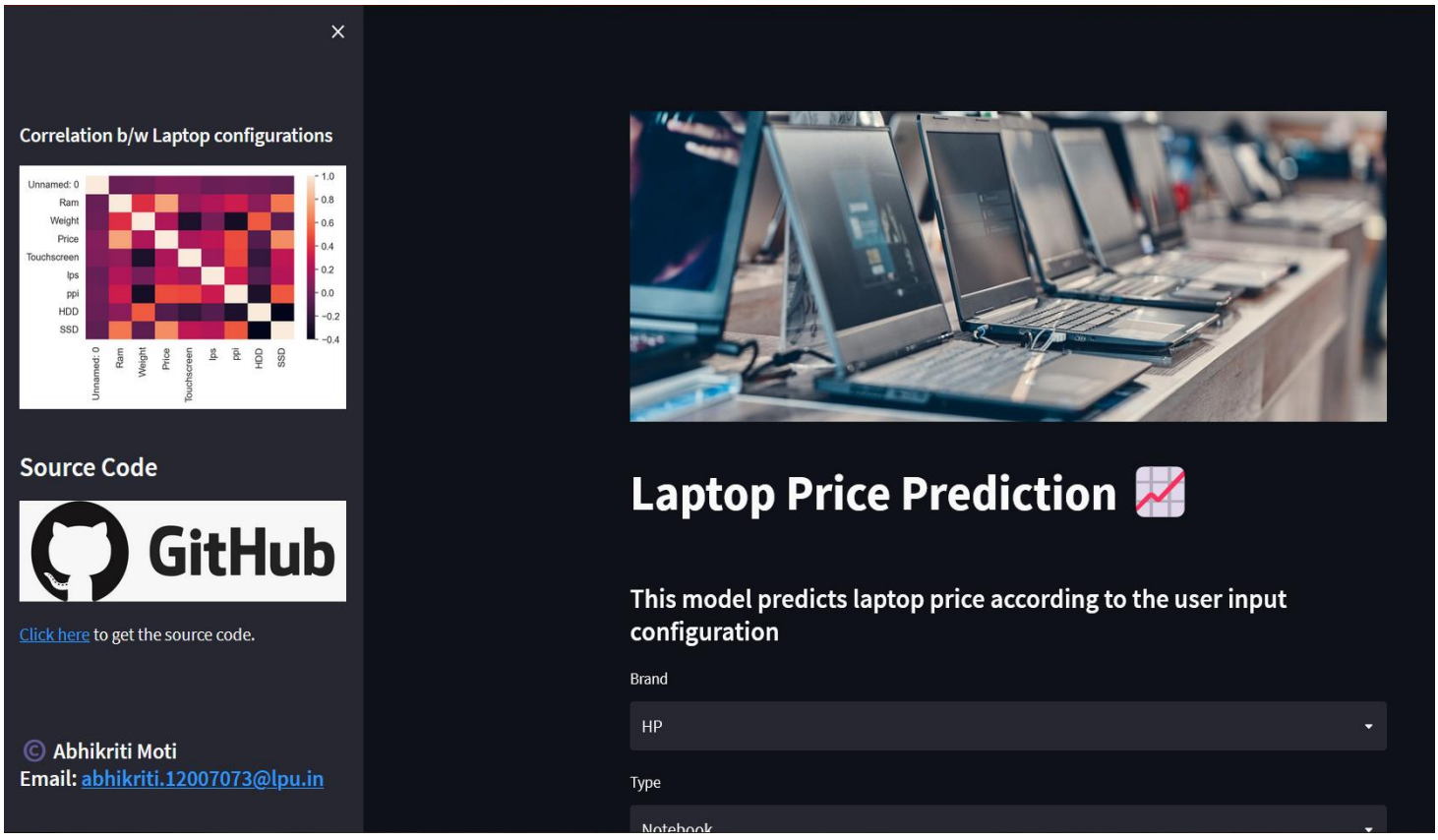
After that, we created the prediction button, and whenever it is triggered, it will encode some variable and prepare a two-dimension list of inputs and pass it to the model to get the prediction that we display on the screen. Take the exponential of predicted output because we have done a log of the output variable.

```
py -m streamlit run main.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.28.9.45:8501
```

Now when you run the app file using the above command you will get two URL and it will automatically open the web application in your default browser or copy the URL and open it. the application will look something like the below figure.



Enter some data in each field and click on predict button to generate prediction. I hope you got the desired results, and the application is working fine.

Laptop Price Prediction 

This model predicts laptop price according to the user input configuration

Brand

HP

Type

Notebook

RAM (in GB)

8

Weight of the Laptop

1.00 − +

Touchscreen

Yes

IPS

No

Screen Size

1.00 − +

Screen Resolution

1920x1080

CPU

Intel Core i5

HDD(in GB)

512

SSD(in GB)

256

GPU

Intel

OS

Windows

Predict Price

The predicted price of this configuration is

₹ 72475

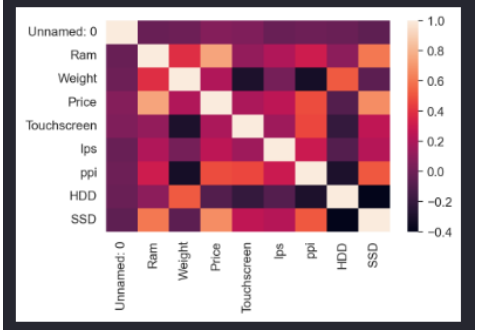
Dataset Used 🔎

[kaggle link]

	Unnar	Company	TypeName	Inches	ScreenResolution	Cpu
0	0	Apple	Ultrabook	13.3000	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz
1	1	Apple	Ultrabook	13.3000	1440x900	Intel Core i5 1.8GHz
2	2	HP	Notebook	15.6000	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz
3	3	Apple	Ultrabook	15.4000	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz
4	4	Apple	Ultrabook	13.3000	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz
5	5	Acer	Notebook	15.6000	1366x768	AMD A9-Series 9420 3GHz
6	6	Apple	Ultrabook	15.4000	IPS Panel Retina Display 2880x1800	Intel Core i7 2.2GHz
7	7	Apple	Ultrabook	13.3000	1440x900	Intel Core i5 1.8GHz
8	8	Asus	Ultrabook	14.0000	Full HD 1920x1080	Intel Core i7 8550U 1.8GHz
9	9	Acer	Ultrabook	14.0000	IPS Panel Full HD 1920x1080	Intel Core i5 8250U 1.6GHz

Made with Streamlit

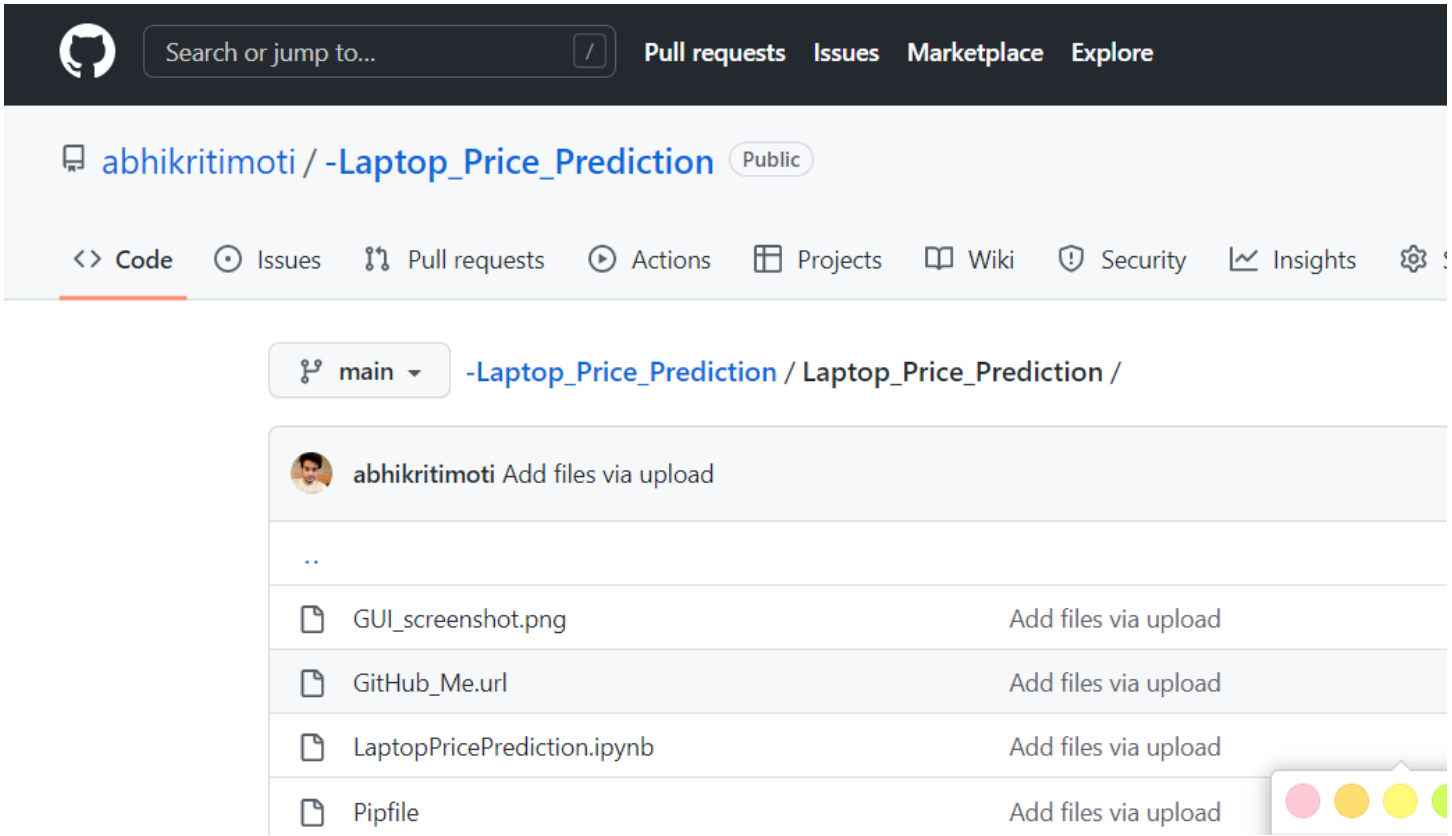
Correlation b/w Laptop configurations



Upload Code to GitHub

Log in to your GitHub account and create a new repository of the project name of your choice. Now you can either use the upload button to upload all files by selecting from a local file manager. And you can also use the GIT bash command as stated below to upload your code.

After creating a new repository copy the ssh link of a repository to connect with a repository.



Modules Used

NumPy

https://numpy.org

pandas - Python Data Analysis Library

https://pandas.pydata.org

Matplotlib — Visualization with Python

https://matplotlib.org

scikit-learn: machine learning in Python — scikit-learn 1.1.2

https://scikit-learn.org

Streamlit - The fastest way to build and share data apps

https://streamlit.io

seaborn: statistical data visualization — seaborn 0.12.0 ...

https://seaborn.pydata.org

Course Outcomes

- Develop relevant programming abilities.
- Demonstrate proficiency with statistical analysis of data.
- Develop the ability to build and assess data-based models.
- Execute statistical analyses with professional statistical software.
- Demonstrate skill in data management.
- Apply data science concepts and methods to solve problems in real-world contexts and will communicate these solutions effectively
- Investigate data and designs by loading, extracting, transforming, and analysing data from various sources.
- Implement histograms, classifiers, decision trees, sampling, linear regression, and projectiles in a scripting language.

BIBLIOGRAPHY

Introduction to Machine Learning with Python: A Guide for Data Scientists
Book by Andreas C. Müller and Sarah Guido

PYTHON DATA SCIENCE HANDBOOK: Tools for Working with Data
Book by Jake VanderPlas

Google Data Analytics Professional Certificate
GOOGLE Course by COURSERA

Allsoft Solutions
SUMMER TRAINING partner

W3Schools Online Web Tutorials
https://www.w3schools.com