Estimation of laptop final price - Data Science Project:

Import Libraries

All libraries are used for specific tasks including data preprocessing, visualization, transformation, modeling and evaluation

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression,Lasso
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
import warnings
warnings.filterwarnings("ignore")
```

Import Data

Read Training Data

```
import pandas as pd

laptopDataTrain = pd.read_csv('/content/laptops.csv')

laptopDataTrain.head()
```

	Laptop	Status	Brand	Model	CPU	RAM	Storage	Storage type	GPU	S
0	ASUS ExpertBook B1 B1502CBA-EJ0436X Intel Core...	New	Asus	ExpertBook	Intel Core i5	8	512	SSD	NaN	
1	Alurin Go Start Intel Celeron N4020/8GB/256GB ...	New	Alurin	Go	Intel Celeron	8	256	SSD	NaN	

Shape Function

```
laptopDataTrain.shape
```

```
(2160, 12)
```

Checking for null values in each column and displaying the sum of all null values in each column

```
laptopDataTrain.isnull().sum()
```

```
Laptop            0
Status            0
Brand             0
Model             0
CPU               0
RAM               0
Storage           0
Storage type     42
GPU            1371
Screen            4
Touch             0
Final Price       0
dtype: int64
```

Removing the rows with empty values since the number of empty rows is small and the dataset is huge. This is the best approach compared to replacing with mean or random values for this case study

```
laptopDataTrain = laptopDataTrain.dropna()
```

Checking if null values are eliminated

```
laptopDataTrain.isnull().sum()
```

```
Laptop          0
Status          0
Brand           0
Model           0
CPU             0
RAM             0
Storage         0
Storage type    0
GPU             0
Screen          0
Touch           0
Final Price     0
dtype: int64
```

```
laptopDataTrain.shape
```

```
(781, 12)
```

Checking the data types to see if all the data is in correct format. All the data seems to be in their required format.

```
laptopDataTrain.dtypes
```

```
Laptop           object
Status           object
Brand            object
Model            object
CPU              object
RAM               int64
Storage           int64
Storage type     object
GPU              object
Screen          float64
Touch            object
Final Price     float64
dtype: object
```

Checking the correlation between the numerical features and target

EDA (Exploratory Data Analysis)

Visualizations are used to understand the relationship between the target variable and the features, in addition to the quantatitave metrics such as correlation coefficient and p-value. The visuals include regression plot,boxplot etc.

```
laptopDataTrain.describe()
```

	RAM	Storage	Screen	Final Price
count	781.000000	781.000000	781.000000	781.000000
mean	22.822023	888.010243	15.935980	1960.431690
std	11.557896	376.492149	0.871735	1012.765083
min	8.000000	256.000000	13.400000	477.590000
25%	16.000000	512.000000	15.600000	1199.000000
50%	16.000000	1000.000000	15.600000	1699.900000
75%	32.000000	1000.000000	16.100000	2461.210000
max	128.000000	4000.000000	18.000000	7150.470000

The above descriptive analysis code displays all numerical data. It also helps to check incorrect entries and anomalies. As it is display all the data is pure and high quality.

NOTE :

In this Case, there is no Outlier. After describing the data train it is clear that all data is real and correct entries.

```
laptopDataTrain.describe(include = 'object')
```

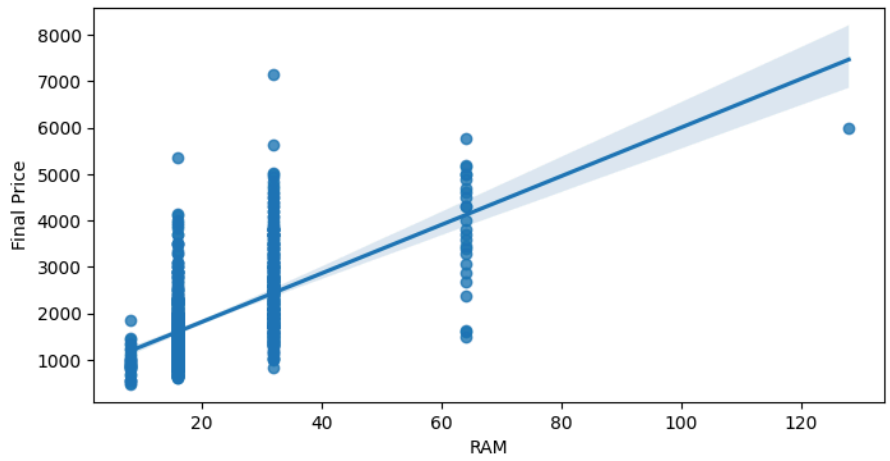
	Laptop	Status	Brand	Model	CPU	Storage type	GPU	Touch
count	781	781	781	781	781	781	781	781
unique	781	2	16	65	10	1	44	2
top	MSI Katana GF66 12UC-082XES Intel Core i7-1270...	New	MSI	ROG	Intel Core i7	SSD	RTX 3050	No

Regression plot

This regression plot shows the correlation between **RAM** and **Final price**. A moderate positive correlation is observed which shows that Final price is being affected by the change in RAM value.

```
import seaborn as sns
plt.figure(figsize=(8,4))
sns.regplot(x="RAM", y="Final Price", data=laptopDataTrain)
```

```
<Axes: xlabel='RAM', ylabel='Final Price'>
```



As observed in the plot above, **a positive correlation** is observed

```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(laptopDataTrain['RAM'], laptopDataTrain['Final Price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

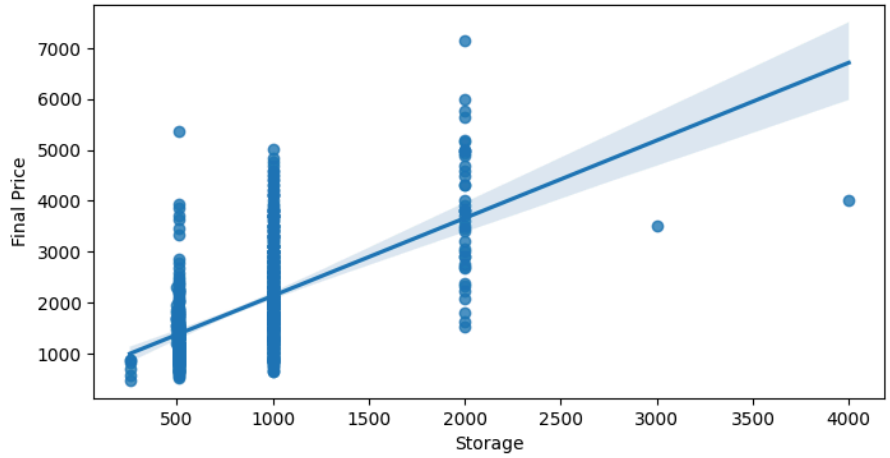
```
The Pearson Correlation Coefficient is 0.5980815389073553  with a P-value of P = 6.1915179553114954e-77
```

-- Here, the Pearson Correlation Coefficient is approximately 0.598, which is closer to 1 . This indicates a moderate positive linear relationship between the two variables being correlated.

the P-value of the Correlation Coefficient is less than 0.05, the Correlation is statistically significant. Hence this feature can be used for prediction.

```
plt.figure(figsize=(8,4))
sns.regplot(x="Storage", y="Final Price", data=laptopDataTrain)
```

```
<Axes: xlabel='Storage', ylabel='Final Price'>
```




```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(laptopDataTrain['Storage'], laptopDataTrain['Final Price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

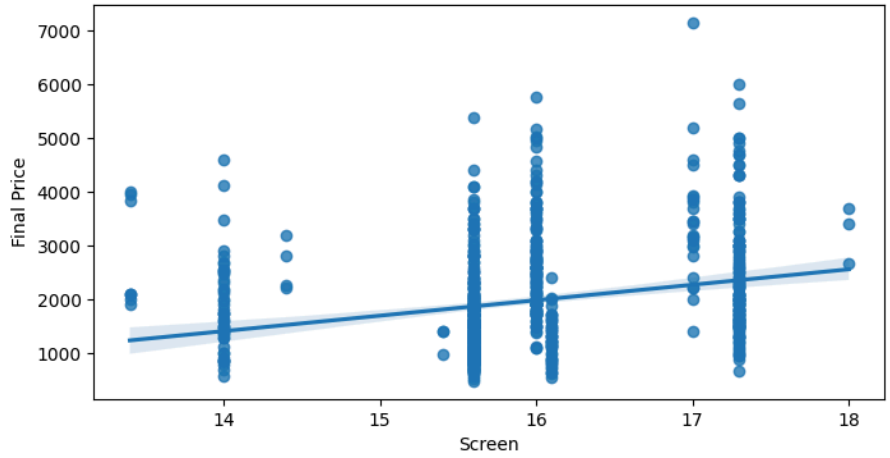
```
The Pearson Correlation Coefficient is 0.5676954082541027  with a P-value of P = 7.858284152354033e-68
```

The regression plot above shows a relationship between the storage of the laptop and the final price of the laptop. A moderate positive correlation is observed between the two variables. This shows that the price increases with increase in storage capacity of the laptop.

As observed above, a moderate positive correlation of 0.568 is calculated and the p-value less than 0.05.So, this indicates that the correlation between the variables is statistically significant. hence storage feature can be used for prediction

```
plt.figure(figsize=(8,4))
sns.regplot(x="Screen", y="Final Price", data=laptopDataTrain)
```

```
<Axes: xlabel='Screen', ylabel='Final Price'>
```




```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(laptopDataTrain['Screen'], laptopDataTrain['Final Price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.24761254362433668  with a P-value of P = 2.2447328809467415e-12
```

This regression plot shows the correlation between **Screen** and **Final price**. A weak positive correlation is observed which shows that Final price is being affected by the change in Screen size.

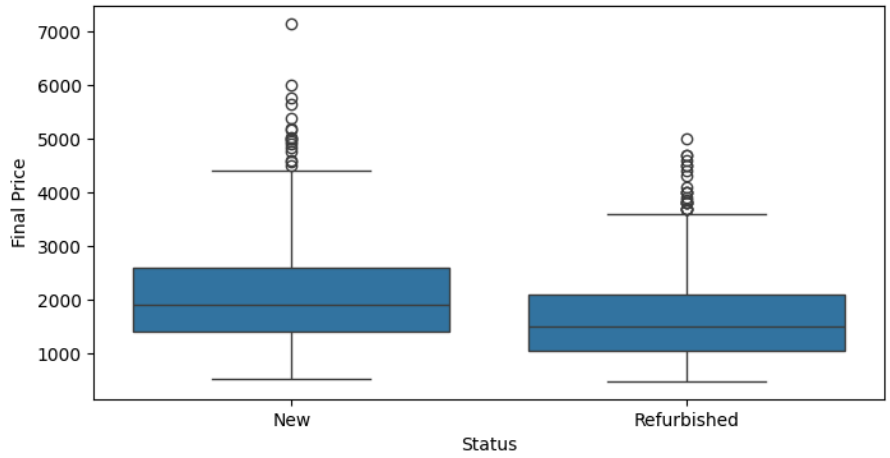
Since the correlation cofficient is 0.248, it indicates that there is a small positive linear relationship between the two veriabels.The P-value of correlation is less than 0.05.So, the correlation is statistically significant.Hence this feature can it be used for prediction.

Box Plot

These plots are used for categorical data to determine the importance of features for prediction.

```
plt.figure(figsize=(8,4))
sns.boxplot(x="Status", y="Final Price", data=laptopDataTrain)
```

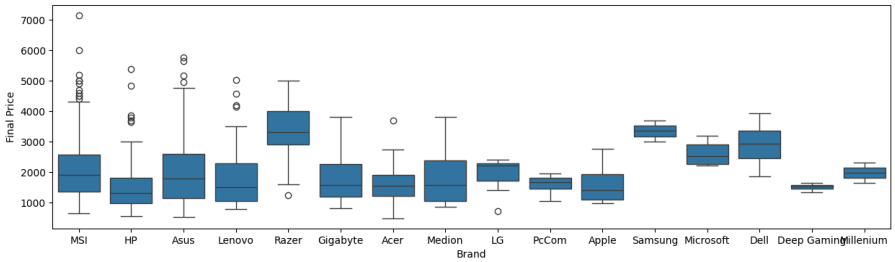
```
<Axes: xlabel='Status', ylabel='Final Price'>
```



In the given plot above, it is observed that the final price range vary for new and refurbished laptops status. This indicates the categories can vary with price hence feature status can be used for prediction

```
plt.figure(figsize=(15,4))
sns.boxplot(x="Brand", y="Final Price", data=laptopDataTrain)
```

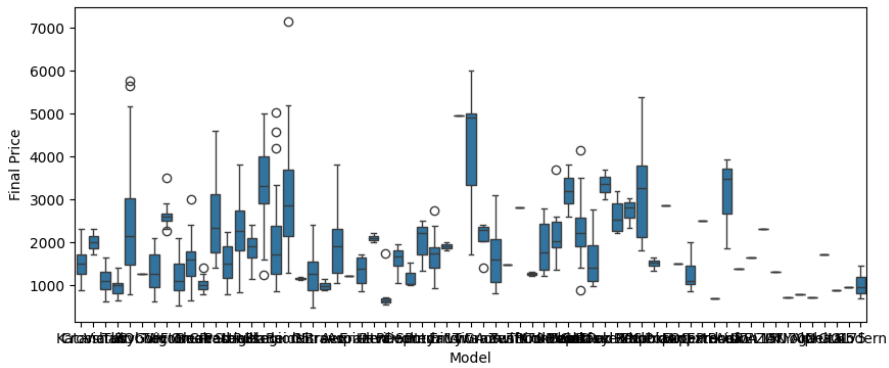
```
<Axes: xlabel='Brand', ylabel='Final Price'>
```



The above box plot shows how final prices vary based on different brands. This shows that brand can be used as a feature for price prediction.

```
plt.figure(figsize=(10,4))
sns.boxplot(x="Model", y="Final Price", data=laptopDataTrain)
```

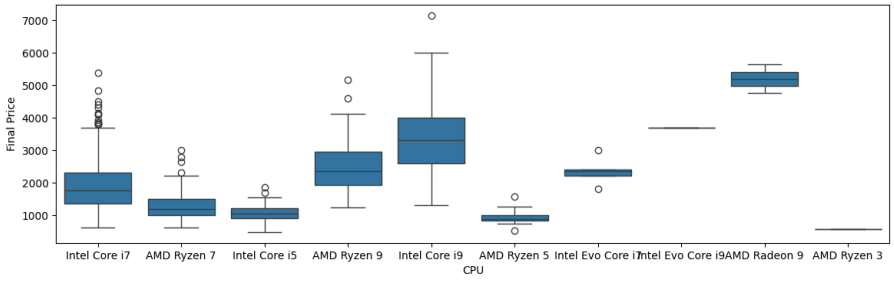
```
<Axes: xlabel='Model', ylabel='Final Price'>
```



The box plot above shows model categories with varying final prices per category hence this feature can be used for price prediction

```
plt.figure(figsize=(14,4))
sns.boxplot(x="CPU", y="Final Price", data=laptopDataTrain)
```

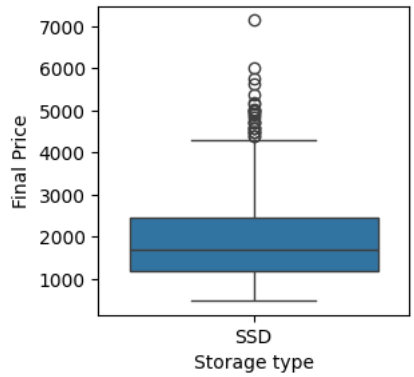
```
<Axes: xlabel='CPU', ylabel='Final Price'>
```



The CPU type shows that this categories have vary final price range which will bring differences in price when prediction is made.

```
plt.figure(figsize=(3,3))
sns.boxplot(x="Storage type", y="Final Price", data=laptopDataTrain)
```

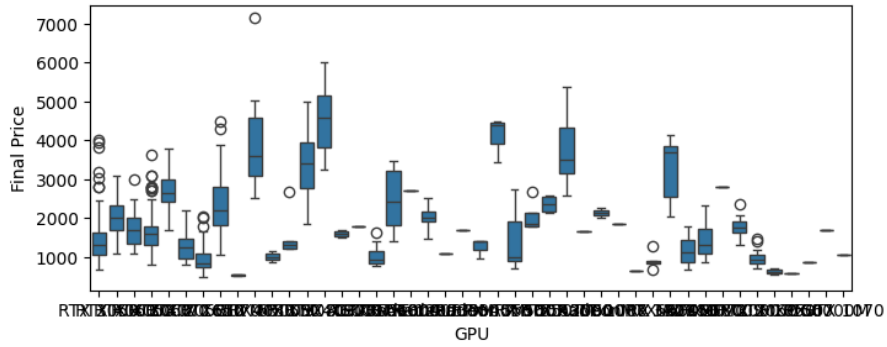
```
<Axes: xlabel='Storage type', ylabel='Final Price'>
```



No price range for one categorie, so this feature is not suitable for prediction.

```
plt.figure(figsize=(8,3))
sns.boxplot(x="GPU", y="Final Price", data=laptopDataTrain)
```

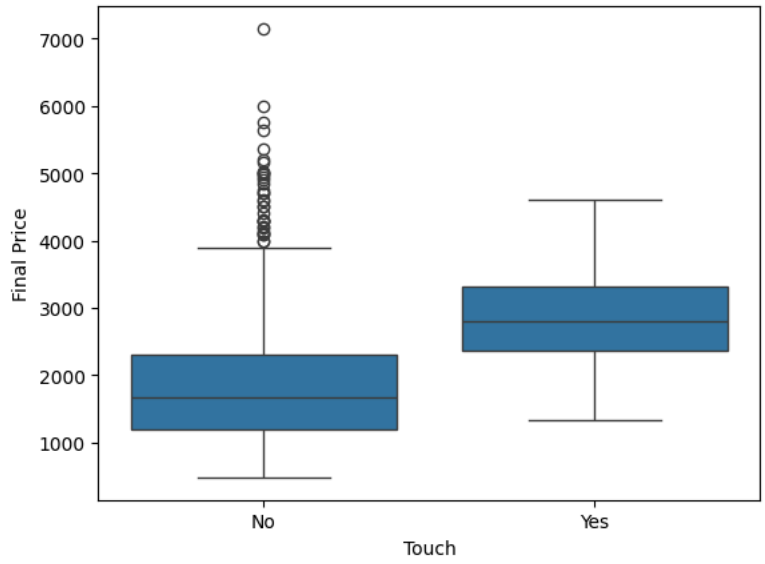
```
<Axes: xlabel='GPU', ylabel='Final Price'>
```



Has GPU feature shows a huge difference in price ranges between laptop with GPU and vice versa. This feature is very important for price prediction as the bigger the difference in range the better the feature

```
sns.boxplot(x="Touch", y="Final Price", data=laptopDataTrain)
```

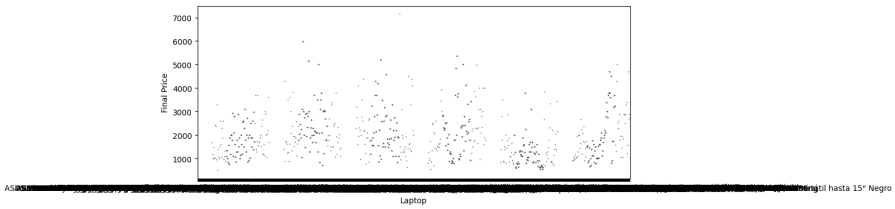
```
<Axes: xlabel='Touch', ylabel='Final Price'>
```



This feature is similar to the feature above, athese two categories have wider price ranges between one another. This feature is also crucial for price prediction.

```
plt.figure(figsize=(10,4))
sns.boxplot(x="Laptop", y="Final Price", data=laptopDataTrain)
```

```
<Axes: xlabel='Laptop', ylabel='Final Price'>
```



This plot above shows that the laptop name is not that important when selling a laptop. The variety of price ranges for all categories prove that the feature is insignificant for price prediction. However, there are a few cases, where there is significant variation found, like say second last laptop name and also in a real scenario, this feature has huge importance.

Using Exploratory data analysis, one feature can be dropped because it had no impact on the price prediction

```
laptopDataTrain.drop(['Storage type'], axis = 1, inplace = True)
```

```
laptopDataTrain.shape
```

```
(781, 11)
```

Data Transformation

Label encoding of categorical features in the training set. Label encoding is converting categorical data into numerical data since the model cant understand textual data.

```
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()
laptopDataTrain.Laptop = labelencoder.fit_transform(laptopDataTrain.Laptop)
laptopDataTrain.Status = labelencoder.fit_transform(laptopDataTrain.Status)
laptopDataTrain.Brand = labelencoder.fit_transform(laptopDataTrain.Brand)
laptopDataTrain.Model = labelencoder.fit_transform(laptopDataTrain.Model)
laptopDataTrain.CPU = labelencoder.fit_transform(laptopDataTrain.CPU)
laptopDataTrain.GPU = labelencoder.fit_transform(laptopDataTrain.GPU)
laptopDataTrain.Touch = labelencoder.fit_transform(laptopDataTrain.Touch)
```

Checking on the remaining features and if label encoding is applied to all categorical features

```
laptopDataTrain.head(10)
```

	Laptop	Status	Brand	Model	CPU	RAM	Storage	GPU	Screen	Touch	Final Price
3	556	0	9	30	6	16	1000	17	15.6	0	1199.0
5	496	0	9	15	6	32	1000	22	17.3	0	1699.0
9	350	0	6	59	6	16	512	17	16.1	0	1149.0
11	689	0	9	53	6	16	1000	21	15.6	0	1399.0
12	18	0	2	45	3	16	512	17	15.6	0	1199.0
19	510	0	9	53	6	16	512	17	15.6	0	999.0
20	497	0	9	16	5	16	1000	22	15.6	0	1249.0
23	85	0	2	52	5	16	512	17	15.6	0	1099.0
25	90	0	2	52	6	16	512	17	15.6	0	1179.0
27	328	0	6	59	5	16	512	17	15.6	0	999.0

--Data Transfornation (normalization) ---- z-score used for scaling down the features between the range of -1 and 1.

```
import scipy.stats as stats
laptopDataTrain = stats.zscore(laptopDataTrain)
```

```
laptopDataTrain
```

	Laptop	Status	Brand	Model	CPU	RAM	Storage	GPU	
3	0.736289	-0.853157	0.678504	-0.481652	0.391744	-0.590626	0.297646	-0.230477	-
5	0.470160	-0.853157	0.678504	-1.428388	0.391744	0.794596	0.297646	0.500646	
9	-0.177419	-0.853157	-0.115966	1.348706	0.391744	-0.590626	-0.999360	-0.230477	
11	1.326206	-0.853157	0.678504	0.970011	0.391744	-0.590626	0.297646	0.354421	-
12	-1.649996	-0.853157	-1.175260	0.465085	-1.885813	-0.590626	-0.999360	-0.230477	-
...	...	...	...	...	...	...	...	...	
2153	1.641125	1.172118	2.002621	-1.680851	0.391744	-0.590626	-0.999360	0.061972	-
2154	1.681044	1.172118	2.002621	-1.680851	1.150929	0.794596	0.297646	0.208197	
2155	1.685480	1.172118	2.002621	-1.680851	0.391744	-0.590626	0.297646	-0.084252	
2156	1.689915	1.172118	2.002621	-1.680851	0.391744	-0.590626	0.297646	0.061972	
2157	1.694351	1.172118	2.002621	-1.680851	0.391744	0.794596	0.297646	0.208197	

Dividing the data for training and testing accordingly. X takes the all features while Y takes the target variable

We have 11 actual columns [0-10 index]; 10 are predictor variables and 1 is the target variable

```
x_laptopDataTrain=laptopDataTrain.iloc[:,0:9]
y_laptopDataTrain=laptopDataTrain.iloc[:,10]
```

```
x_laptopDataTrain.head()
```

	Laptop	Status	Brand	Model	CPU	RAM	Storage	GPU	S
3	0.736289	-0.853157	0.678504	-0.481652	0.391744	-0.590626	0.297646	-0.230477	-0.3
5	0.470160	-0.853157	0.678504	-1.428388	0.391744	0.794596	0.297646	0.500646	1.5
9	-0.177419	-0.853157	-0.115966	1.348706	0.391744	-0.590626	-0.999360	-0.230477	0.1
11	1.326206	-0.853157	0.678504	0.970011	0.391744	-0.590626	0.297646	0.354421	-0.3
12	-1.649996	-0.853157	-1.175260	0.465085	-1.885813	-0.590626	-0.999360	-0.230477	-0.3

```
y_laptopDataTrain.head()
```

```
3    -0.752316
5    -0.258302
9    -0.801718
11   -0.554711
12   -0.752316
Name: Final Price, dtype: float64
```

```
# importing train_test_split from sklearn
from sklearn.model_selection import train_test_split
# splitting the data  # 30% for testing is used
X_laptopDataTrain, X_test, Y_laptopDataTrain, Y_test = train_test_split(x_laptopDataTrain, y_laptopDataTrain, test_size = 0.3, random_state
```

```
X_laptopDataTrain.shape
```

```
(546, 9)
```

The shape function show that the training data of featuers variables is 70%

```
Y_laptopDataTrain.shape
```

```
(546,)
```

Tha shape function show that the training data of target variable is 70%

```
X_test.shape
```

```
(235, 9)
```

The shape function show that the testing data of featuers variables is 30%

Fit Model

Multiple Linear Regression

Calling multiple linear regression model and fitting the training set

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model_mlr = model.fit(x_laptopDataTrain,y_laptopDataTrain)
```

Making price prediction using the testing set (Fit to MLR)

```
Y_pred_MLR = model_mlr.predict(X_test)
```

MLR Evaluation

Calculating the Mean Square Error for MLR model

```
mse_MLR = mean_squared_error(Y_test, Y_pred_MLR)
print('The mean square error for Multiple Linear Regression: ', mse_MLR)
```

```
The mean square error for Multiple Linear Regression:  0.4569748519236977
```

Calculating the Mean Absolute Error for MLR model

```
mae_MLR= mean_absolute_error(Y_test, Y_pred_MLR)
print('The mean absolute error for Multiple Linear Regression: ', mae_MLR)
```

```
The mean absolute error for Multiple Linear Regression:  0.4990383588346339
```

Random Forest Regressor (checking other Models)

Calling the random forest model and fitting the training data

```
rfModel = RandomForestRegressor()
model_rf = rfModel.fit(x_laptopDataTrain,y_laptopDataTrain)
```

Prediction of laptop final prices using the testing data

```
Y_pred_RF = model_rf.predict(X_test)
```

Random Forest Evaluation

Calculating the Mean Square Error for Random Forest Model

```
mse_RF = mean_squared_error(Y_test, Y_pred_RF)
print('The mean square error of price and predicted value is: ', mse_RF)
```

```
The mean square error of price and predicted value is:  0.026394289521654124
```

Calculating the Mean Absolute Error for Random Forest Model

```
mae_RF= mean_absolute_error(Y_test, Y_pred_RF)
print('The mean absolute error of price and predicted value is: ', mae_RF)
```

```
The mean absolute error of price and predicted value is:  0.10817774748351608
```

LASSO Model

Calling the model and fitting the training data

```
LassoModel = Lasso()
model_lm = LassoModel.fit(x_laptopDataTrain,y_laptopDataTrain)
```

Price prediction uisng testing data

```
Y_pred_lasso = model_lm.predict(X_test)
```

LASSO Evaluation (checking another model)

Mean Absolute Error for LASSO Model

```
mae_lasso= mean_absolute_error(Y_test, Y_pred_lasso)
print('The mean absolute error of price and predicted value is: ', mae_lasso)
```

```
The mean absolute error of price and predicted value is:  0.7419381926639694
```

Mean Squared Error for the LASSO Model

1. List item
2. List item

```
mse_lasso = mean_squared_error(Y_test, Y_pred_lasso)
print('The mean square error of price and predicted value is: ', mse_lasso)
```

```
The mean square error of price and predicted value is:  0.9423893886495266
```

```
scores = [('MLR', mae_MLR),
          ('Random Forest', mae_RF),

mae = pd.DataFrame(data = scores, columns=['Model', 'MAE Score'])
mae
```

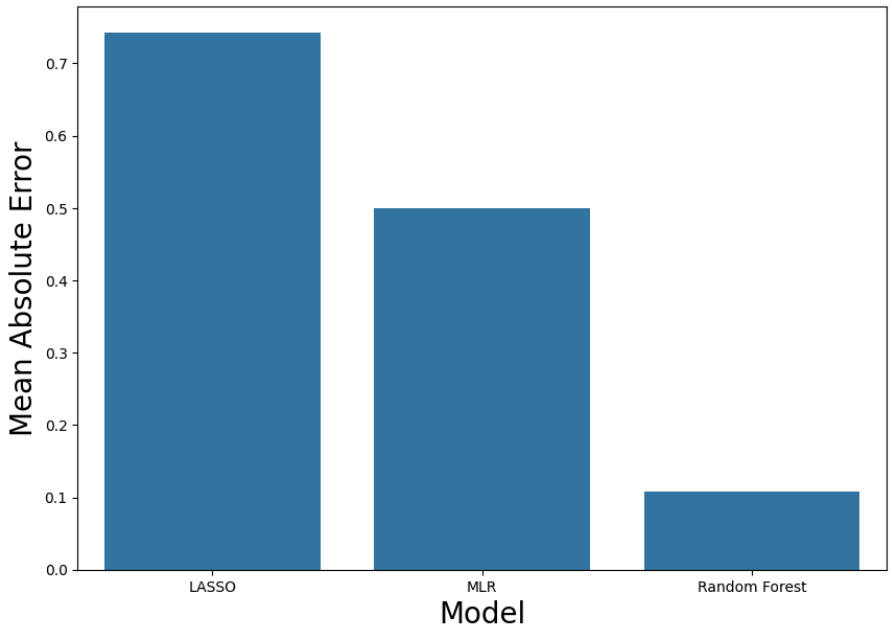
	Model	MAE Score
0	MLR	0.499038
1	Random Forest	0.108178
2	LASSO	0.741938

```
plt.figure(figsize=(4,4))
mae.sort_values(by=['MAE Score']), ascending=False, inplace=True)

f, axe = plt.subplots(1,1, figsize=(10,7))
sns.barplot(x = mae['Model'], y=mae['MAE Score'], ax = axe)
axe.set_xlabel('Model', size=20)
axe.set_ylabel('Mean Absolute Error', size=20)

plt.show()
```

```
<Figure size 400x400 with 0 Axes>
```



Based on the MAE, it is concluded that the Random Forest is the best regression model for predicting the laptop final price based on the 10 predictor variables

Done by

Mai Hammod Al-habsi ID: PG24S2274