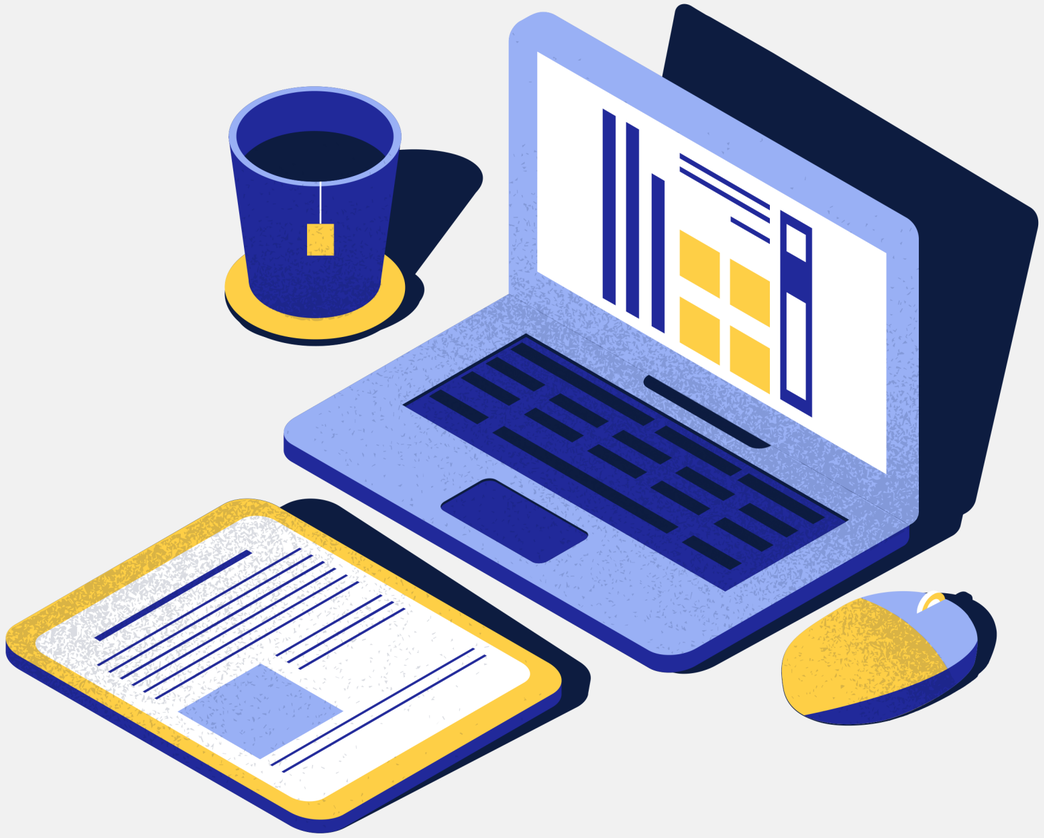
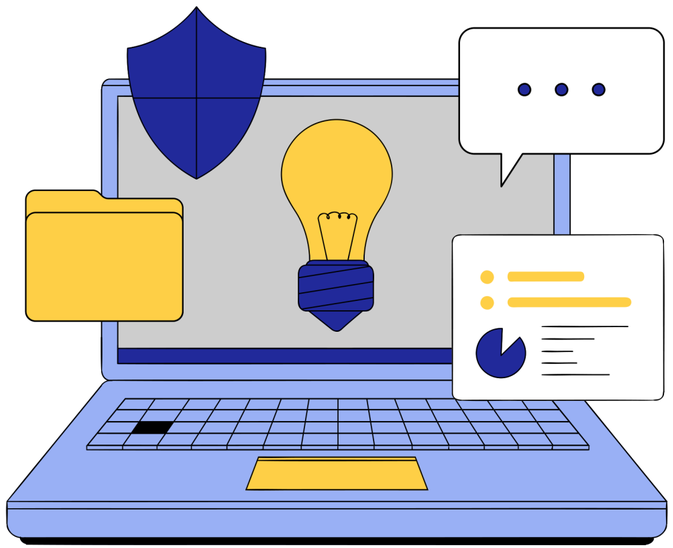


SC1015 Mini Project

Laptop Price Prediction

ECDS1 Group 5 - Gabriella, Li Song, Jian Yuan

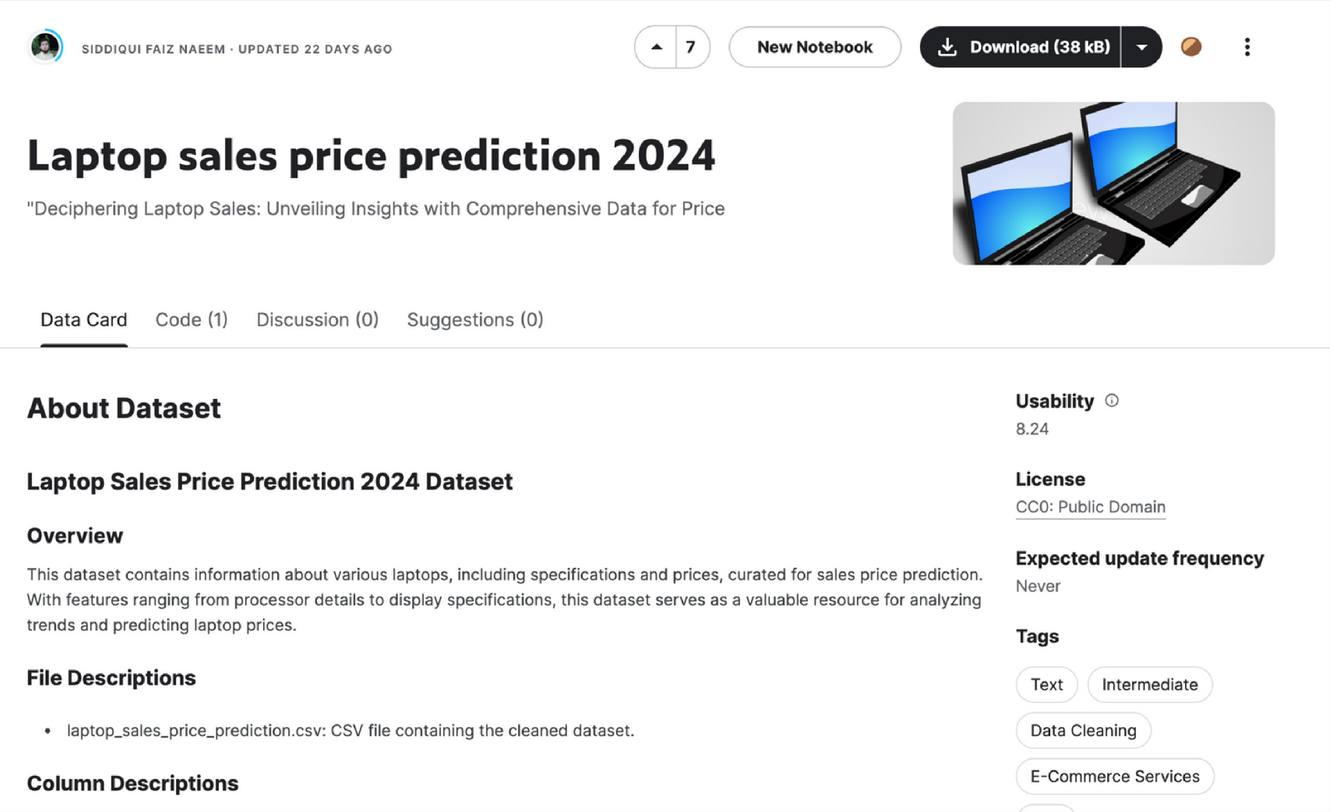
Outline



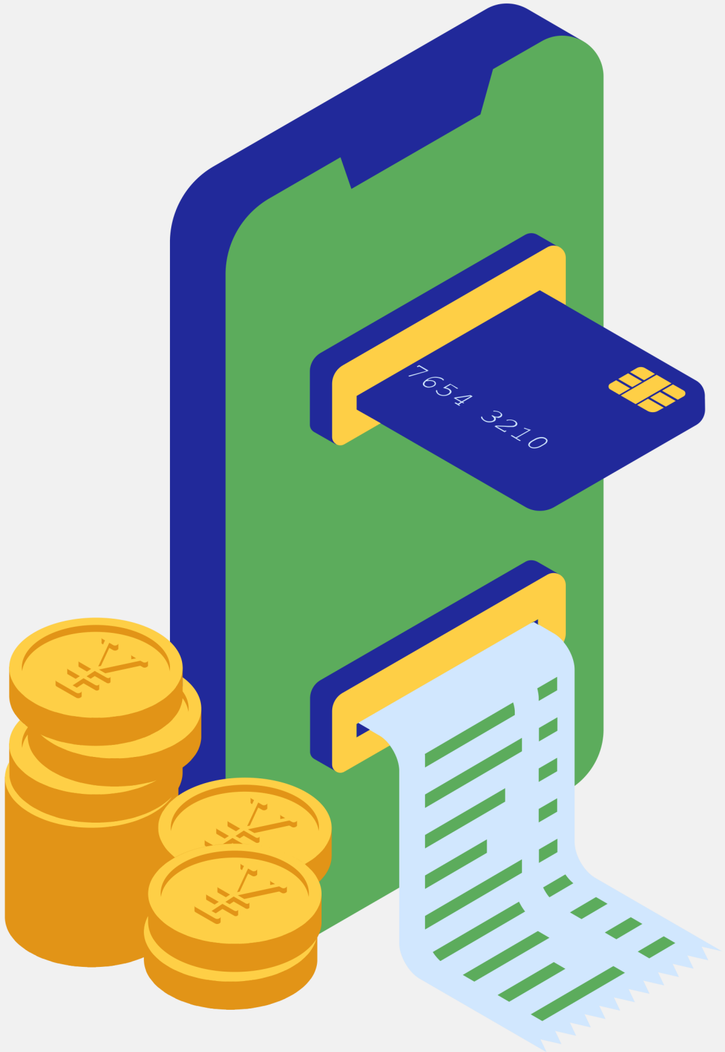
✓	Step 1	Our Motivation
✓	Step 2	Exploratory Data Analysis
✓	Step 3	Machine Learning
✓	Step 4	Conclusion

Dataset

Our dataset is a dataset from Kaggle: "Laptop sales price prediction 2024" by Siddiqui Faiz Naeem



OUR MOTIVATION



Problem Definition

- How do different factors affect the price of laptop?
- Which model would be the best to predict laptop price?

Set Up

Importing dataset and essential libraries

```
import numpy as np                    # a librar
import pandas as pd                   # a librar
import seaborn as sb                  # a librar
import matplotlib.pyplot as plt       # a librar
sb.set()                                # this i
```

```
laptopData = pd.read_csv('laptop_cleaned2.csv')
laptopData.head()
```

	Unnamed: 0	Name	Brand	Price	Rating	Processor_brand	Processor_name	Processor_variant	Processor_ge
0	0	HP Victus 15-fb0157AX Gaming Laptop (AMD Ryzen...	HP	50399	4.30	AMD	AMD Ryzen 5	5600H	5.
1	1	Lenovo V15 G4 82YU00W7IN Laptop (AMD Ryzen 3 ...	Lenovo	26690	4.45	AMD	AMD Ryzen 3	7320U	7.
2	2	HP 15s-fq5007TU Laptop (12th Gen Core i3/ 8GB/...	HP	37012	4.65	Intel	Intel Core i3	1215U	12.
3	3	Samsung Galaxy Book2 Pro 13 Laptop (12th Gen C...	Samsung	69990	4.75	Intel	Intel Core i5	1240P	12.
4	4	Tecno Megabook T1 Laptop (11th Gen Core i3/ 8G...	Tecno	23990	4.25	Intel	Intel Core i3	1115G4	11.

5 rows × 29 columns

Overview of Dataset

```
[ ] laptopData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1020 entries, 0 to 1019
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Unnamed: 0              1020 non-null   int64  
 1   Name                    1020 non-null   object 
 2   Brand                   1020 non-null   object 
 3   Price                   1020 non-null   int64  
 4   Rating                  1020 non-null   float64
 5   Processor_brand         1020 non-null   object 
 6   Processor_name          1020 non-null   object 
 7   Processor_variant       996 non-null    object 
 8   Processor_gen           891 non-null    float64
 9   Core_per_processor      1008 non-null   float64
 10  Total_processor         573 non-null    float64
 11  Execution_units         573 non-null    float64
 12  Low_Power_Cores         1020 non-null   float64
 13  Energy_Efficient_Units  1020 non-null   int64  
 14  Threads                 972 non-null    float64
 15  RAM_GB                  1020 non-null   int64  
 16  RAM_type                998 non-null    object 
 17  Storage_capacity_GB     1020 non-null   int64  
 18  Storage_type            1020 non-null   object 
 19  Graphics_name           1018 non-null   object 
 20  Graphics_brand          1018 non-null   object 
 21  Graphics_GB             368 non-null    float64
 22  Graphics_integreted     1018 non-null   object 
 23  Display_size_inches     1020 non-null   float64
 24  Horizontal_pixel        1020 non-null   int64  
 25  Vertical_pixel          1020 non-null   int64  
 26  ppi                     1020 non-null   float64
 27  Touch_screen            1020 non-null   bool   
 28  Operating_system        1020 non-null   object 
dtypes: bool(1), float64(10), int64(7), object(11)
memory usage: 224.2+ KB
```

Data Preparation and Cleaning

First up, we want to ensure all the datas are unique as we do not want duplicated data.

```
print("Total number of rows:\t\t", len(laptopData))
print("Total number of unique Name:\t", len(laptopData["Name"].unique()))
```

```
Total number of rows:           1020
Total number of unique Name:    1020
```

We then remove ‘Unnamed’ and ‘Name’

```
[ ]  laptopData = laptopData.drop(['Unnamed: 0','Name'], axis=1)
     laptopData
```

Checking for null values

```
laptopData.isnull().sum()
```

```
Brand                      0
Price                      0
Rating                     0
Processor_brand            0
Processor_name             0
Processor_variant         24
Processor_gen            129
Core_per_processor        12
Total_processor          447
Execution_units          447
Low_Power_Cores            0
Energy_Efficient_Units     0
Threads                   48
RAM_GB                     0
RAM_type                  22
Storage_capacity_GB        0
Storage_type               0
Graphics_name              2
Graphics_brand             2
Graphics_GB              652
Graphics_integreted        2
Display_size_inches        0
Horizontal_pixel           0
Vertical_pixel             0
ppi                        0
Touch_screen               0
Operating_system           0
dtype: int64
```

Correlation with Core_per_processor

```
correlation_matrix = laptopData[['Price','Core_per_processor', 'Total_processor', 'Execution_units','Graphics_GB']].corr()
print(correlation_matrix)
```

```
                      Price  Core_per_processor  Total_processor  \
Price              1.000000            0.713484         0.624849
Core_per_processor 0.713484            1.000000         0.747524
Total_processor    0.624849            0.747524         1.000000
Execution_units    0.533587            0.864156         0.336390
Graphics_GB        0.818311            0.702463         0.661515

                   Execution_units  Graphics_GB
Price                     0.533587     0.818311
Core_per_processor        0.864156     0.702463
Total_processor           0.336390     0.661515
Execution_units           1.000000     0.617562
Graphics_GB               0.617562     1.000000
```

Dropping columns

```
laptopData = laptopData.drop(['Total_processor','Execution_units','Graphics_GB'], axis=1)
laptopData.isnull().sum()
```

Six variables related to processors

- Processor_brand
- Processor_name
- Processor_variant
- Processor_gen
- Core_per_processor
- Low_Power_Cores

Drop processor name and gen

```
laptopData = laptopData.drop(['Processor_name','Processor_gen'
laptopData.isnull().sum()
```

Removing rows with null variables

```
laptopData_clean.drop(laptopData_clean.index[laptopData_clean["Processor_variant"].isnull() | laptopData_cl
                | laptopData_clean["RAM_type"].isnull() | laptopData_clean["Graphics_name"].isnull() | laptopl
                | laptopData_clean["Graphics_integreted"].isnull()], inplace=True)
laptopData_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 960 entries, 0 to 1019
Data columns (total 22 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Brand                  960 non-null    object
 1   Price                  960 non-null    int64
 2   Rating                 960 non-null    float64
 3   Processor_brand        960 non-null    object
 4   Processor_variant      960 non-null    object
 5   Core_per_processor     960 non-null    float64
 6   Low_Power_Cores        960 non-null    float64
 7   Energy_Efficient_Units 960 non-null    int64
 8   Threads                960 non-null    float64
 9   RAM_GB                 960 non-null    int64
 10  RAM_type               960 non-null    object
 11  Storage_capacity_GB    960 non-null    int64
 12  Storage_type           960 non-null    object
 13  Graphics_name          960 non-null    object
 14  Graphics_brand         960 non-null    object
 15  Graphics_integreted    960 non-null    object
 16  Display_size_inches    960 non-null    float64
 17  Horizontal_pixel       960 non-null    int64
 18  Vertical_pixel         960 non-null    int64
 19  ppi                    960 non-null    float64
 20  Touch_screen           960 non-null    bool
 21  Operating_system       960 non-null    object
dtypes: bool(1), float64(6), int64(6), object(9)
memory usage: 165.9+ KB
```

Changing currency & final cleaned data

```
[ ] laptopData_clean['Price'] = laptopData_clean['Price']/61.24
```

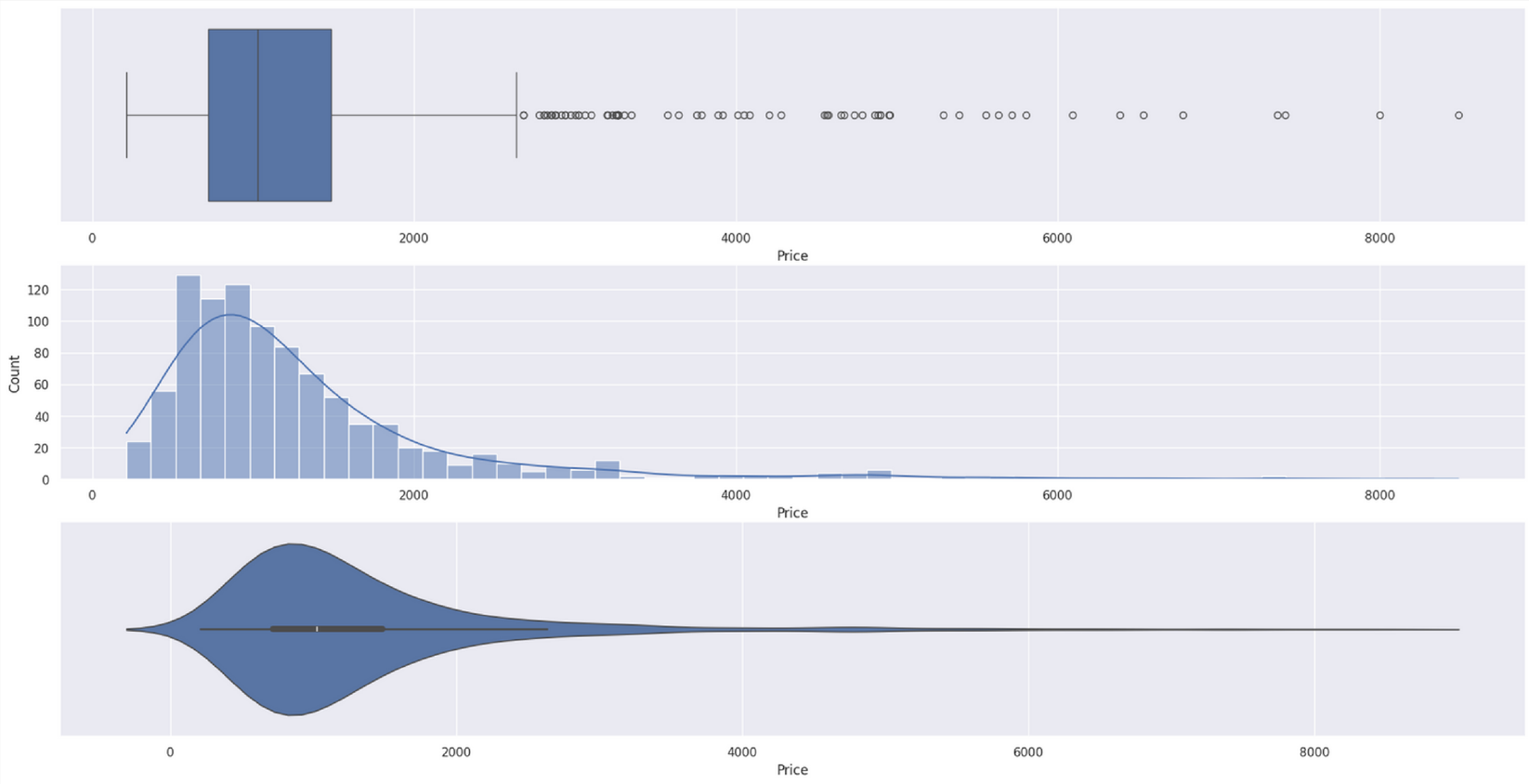
```
laptopData_clean.head()
```

	Brand	Price	Rating	Processor_brand	Processor_variant	Core_per_processor	Low_Power_Cores	Energy_Eff:
0	HP	822.975180	4.30	AMD	5600H	6.0	0.0	
1	Lenovo	435.826257	4.45	AMD	7320U	4.0	0.0	
2	HP	604.376225	4.65	Intel	1215U	6.0	0.0	
3	Samsung	1142.880470	4.75	Intel	1240P	12.0	0.0	
4	Tecno	391.737427	4.25	Intel	1115G4	2.0	0.0	

5 rows × 22 columns

EXPLORATORY DATA ANALYSIS

Exploring Response Variable: Price

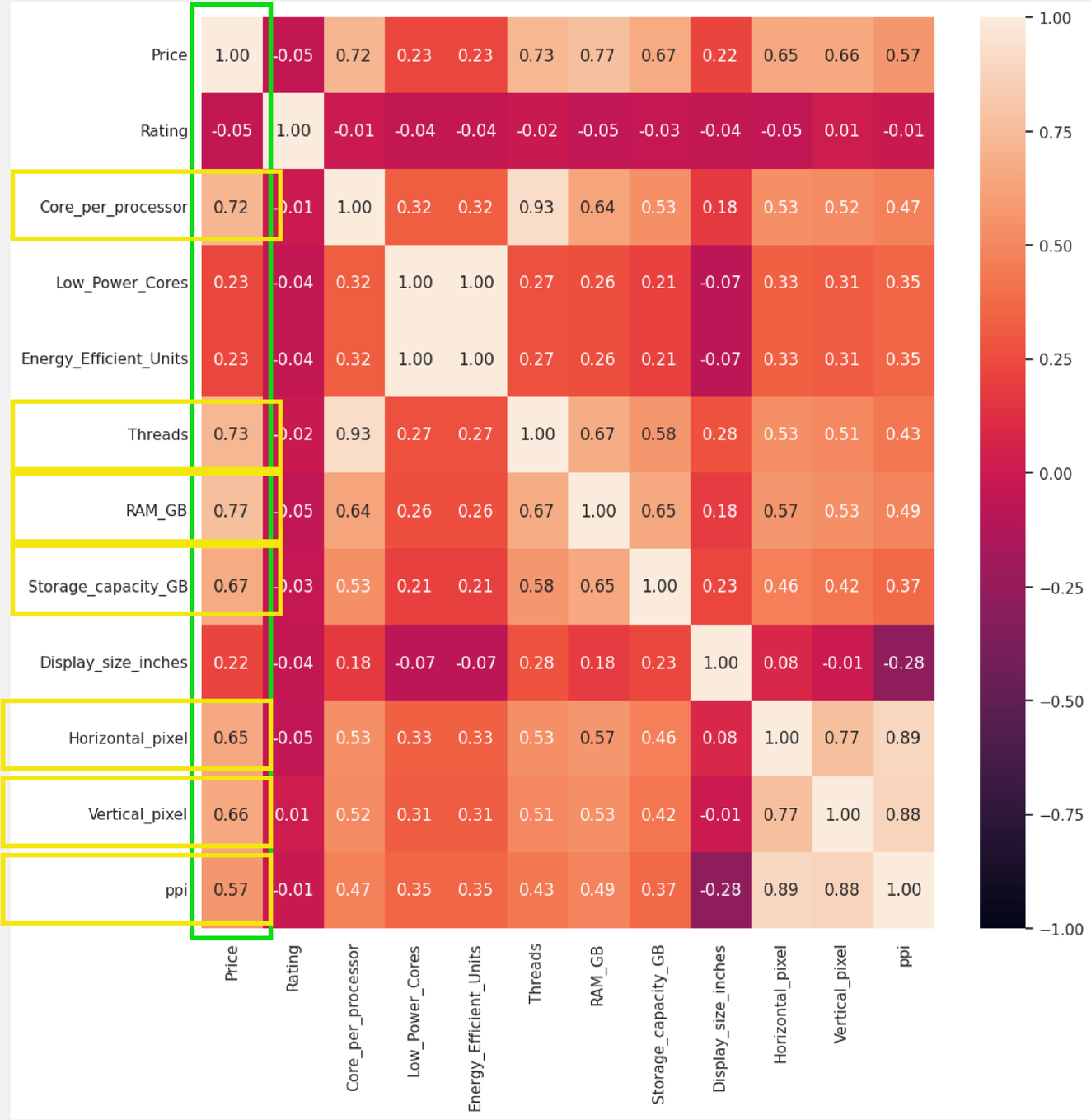


	Price
count	960.000000
mean	1309.988604
std	1020.610742
min	212.116264
25%	718.321359
50%	1026.959504
75%	1485.793599
max	8491.018942

Numerical Variables

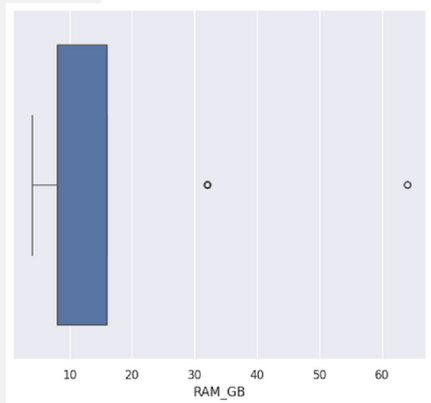
Initially, we have10 numeric variables.

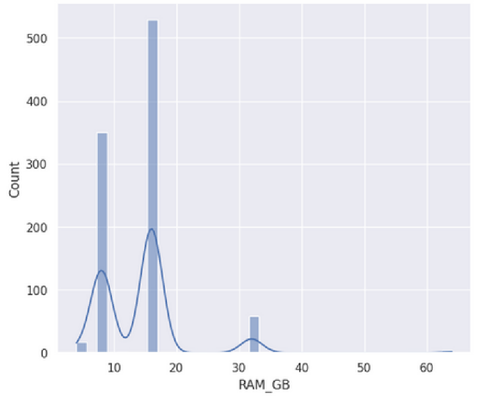
Highest correlation with Price?

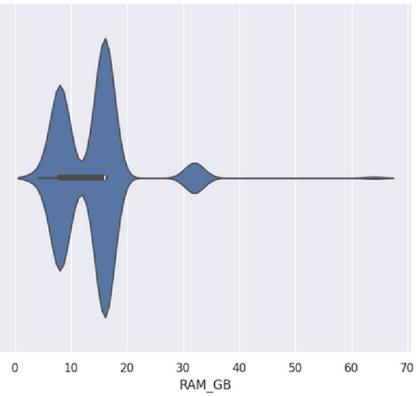


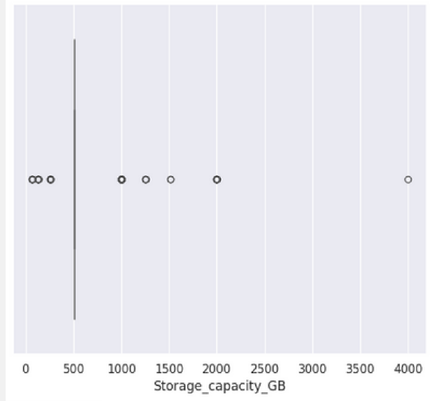
Numerical Variables

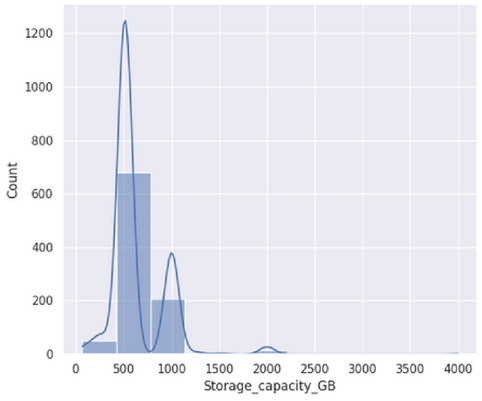
We focus on 3 numerical variables which are **ram**, **storage capacity** and **vertical pixel**.

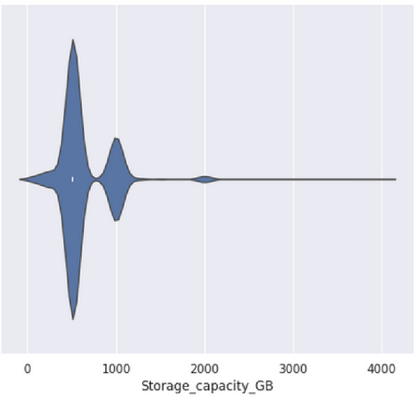


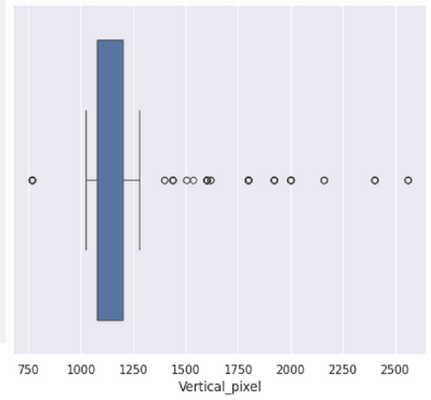


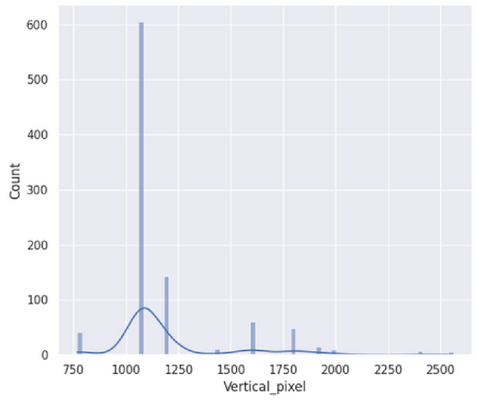


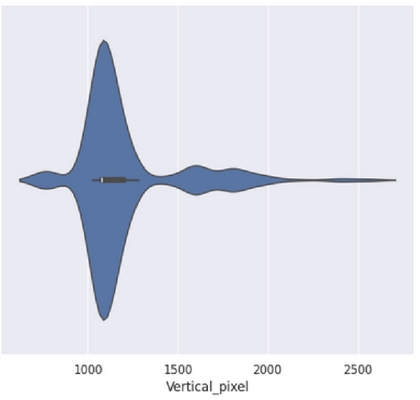












Skewness and Outliers

Our data is highly skewed.

```
[ ]  for i in laptopData_numeric:
         print("Skewness of ", i, "\t: ", laptopData_numeric[i].skew())
```

```
Skewness of  Price      :  2.934335036903041
Skewness of  RAM_GB     :  2.4075665523167578
Skewness of  Storage_capacity_GB        :  3.0391823433398386
Skewness of  Vertical_pixel     :  2.0071510231492997
```

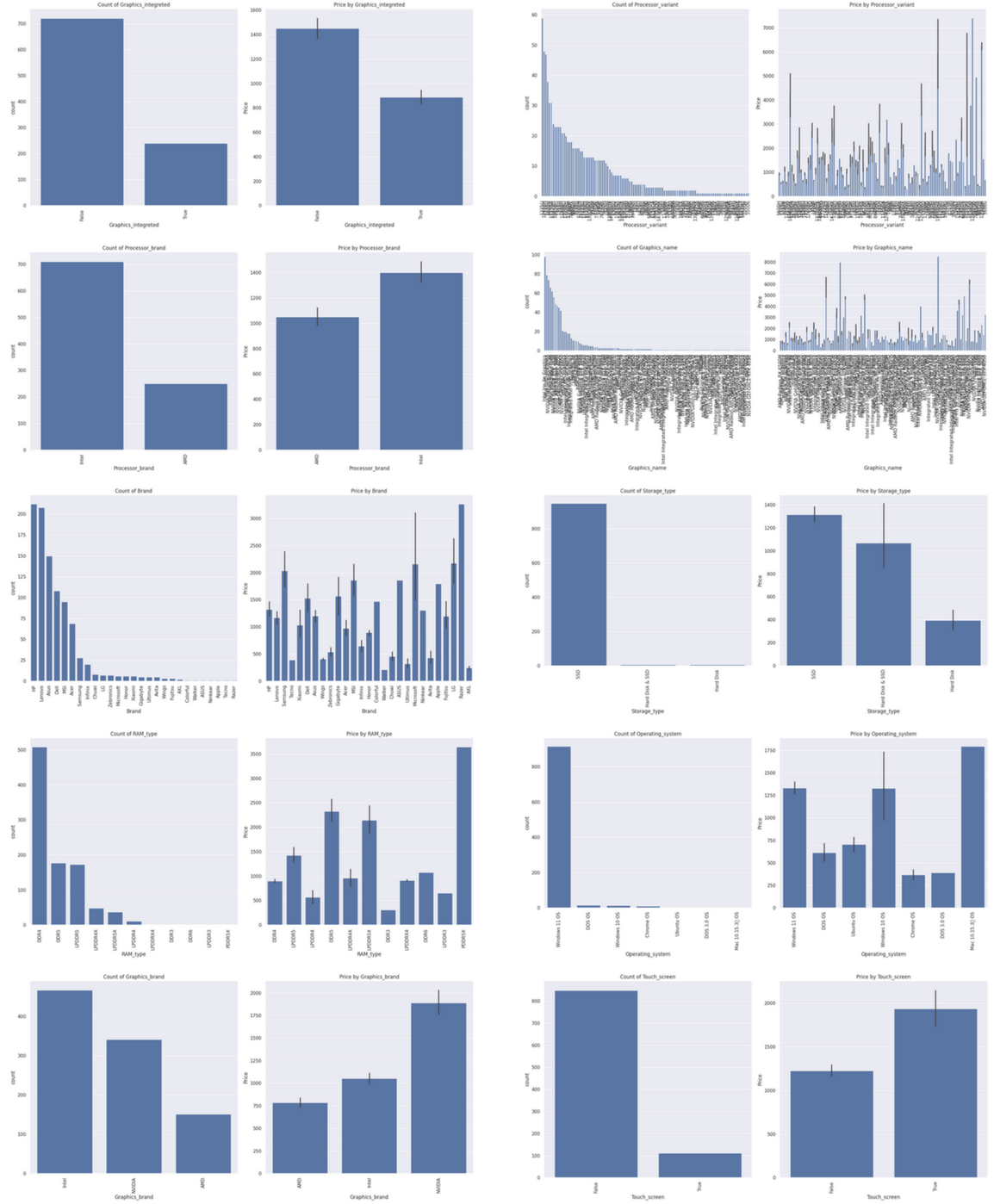
Presence of outliers

```
q1 = laptopData_numeric.quantile(0.25)
q3 = laptopData_numeric.quantile(0.75)
iqr = q3-q1
outliers = ((laptopData_numeric < q1-1.5*iqr) | (laptopData_numeric > q3+1.5*iqr)).sum()
outliers
```

```
Price                  71
RAM_GB                 62
Storage_capacity_GB    280
Vertical_pixel         208
dtype: int64
```

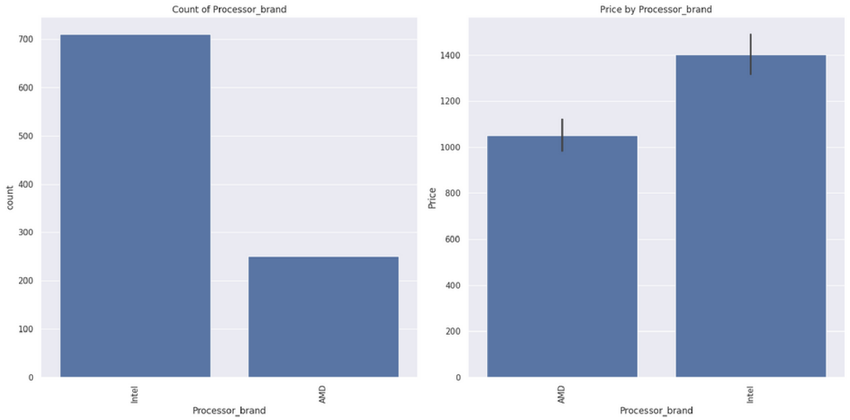
Categorical Variables

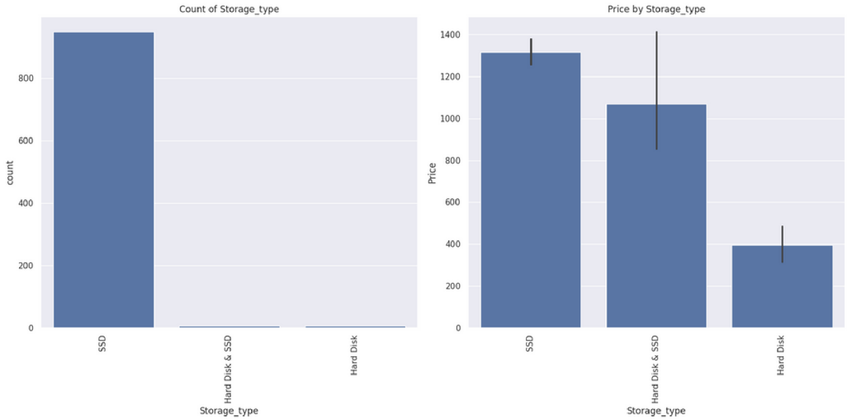
Initially, we have 10 categorical variables

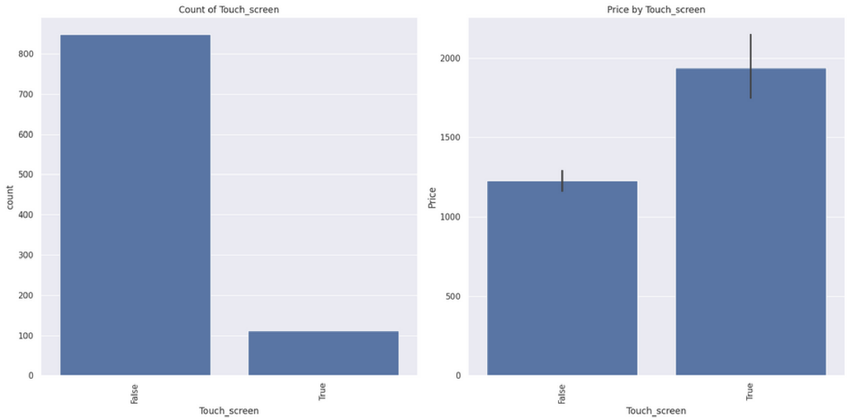


Categorical Variables

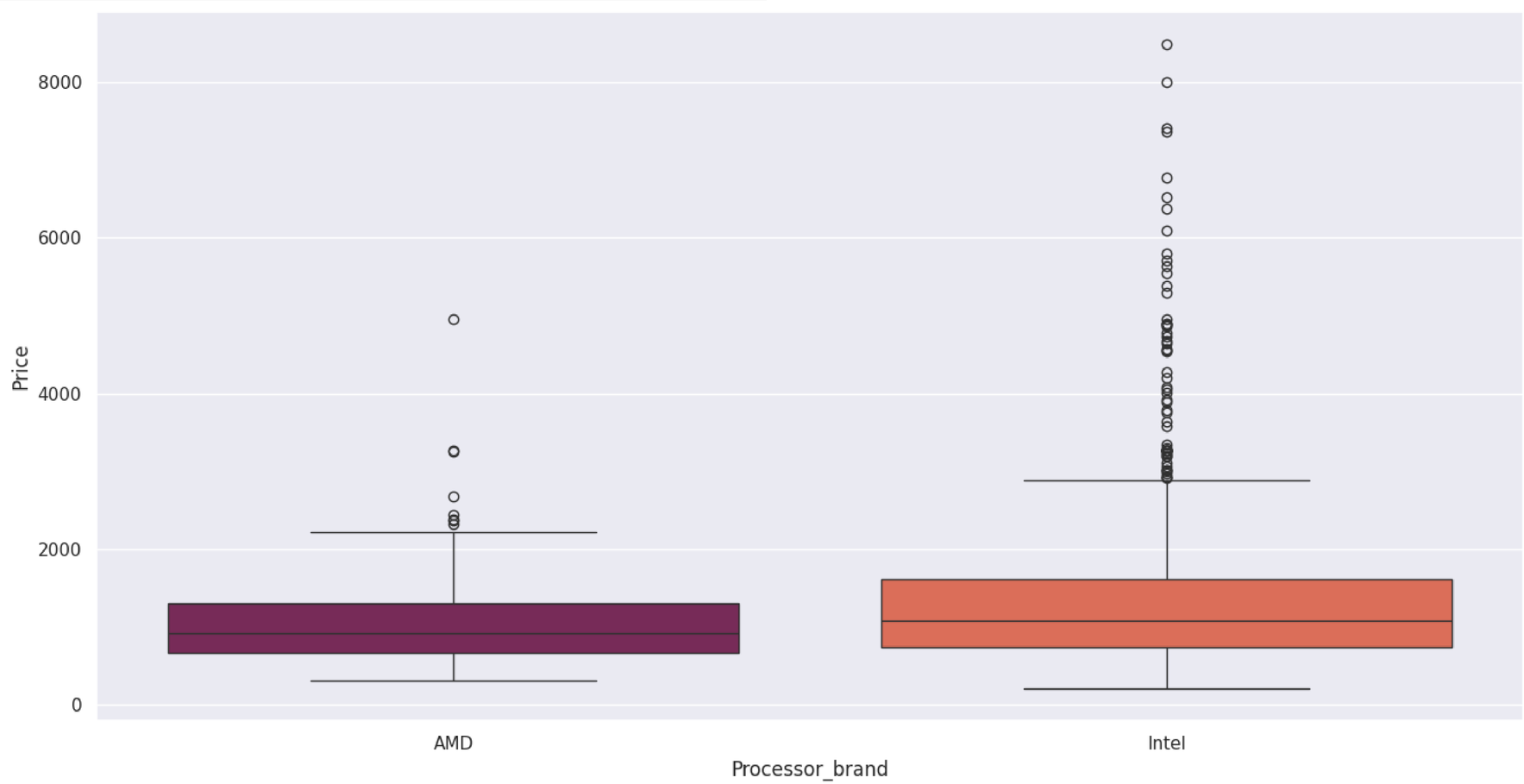
We choose Processor_brand, Storage_type, Touch_screen



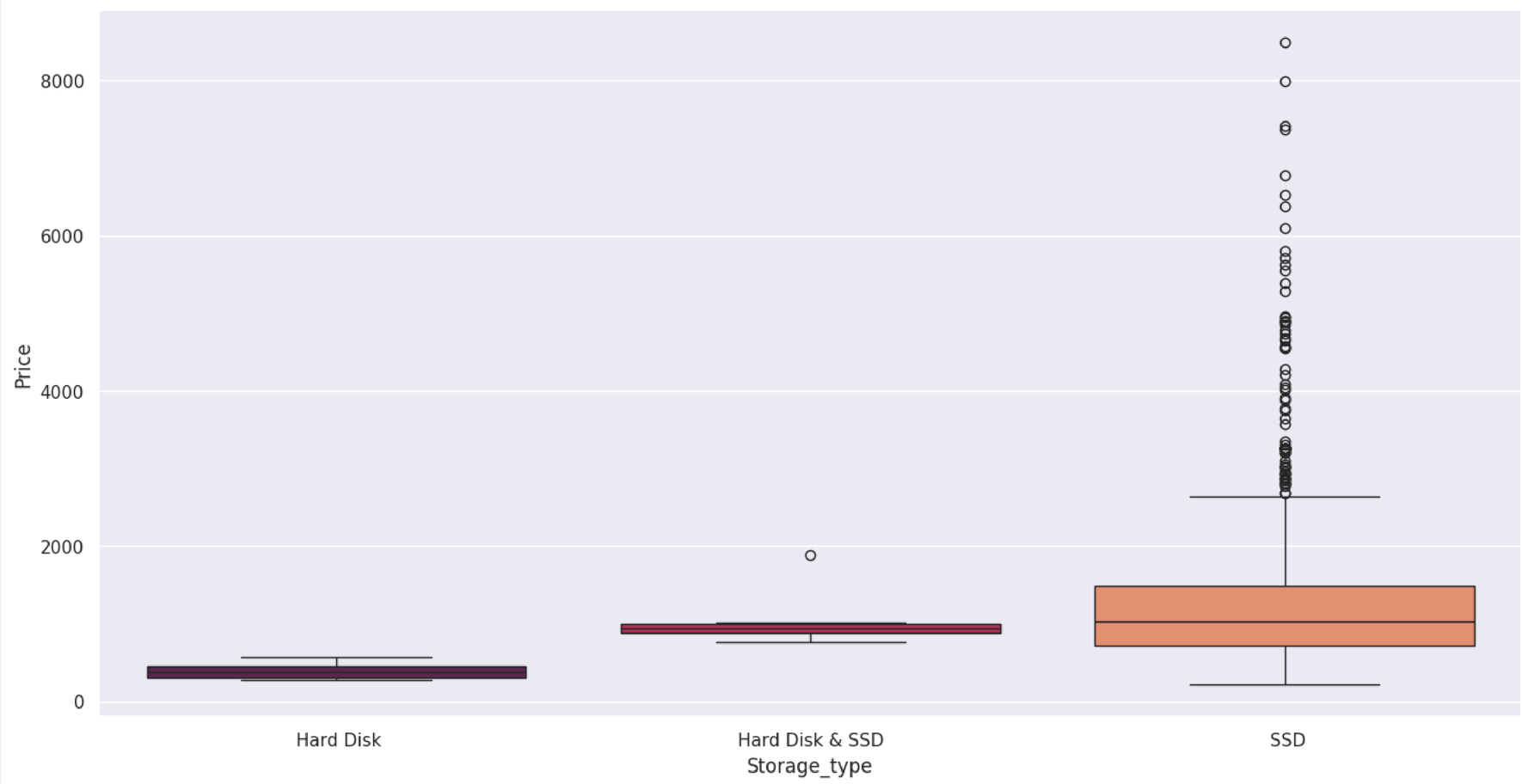




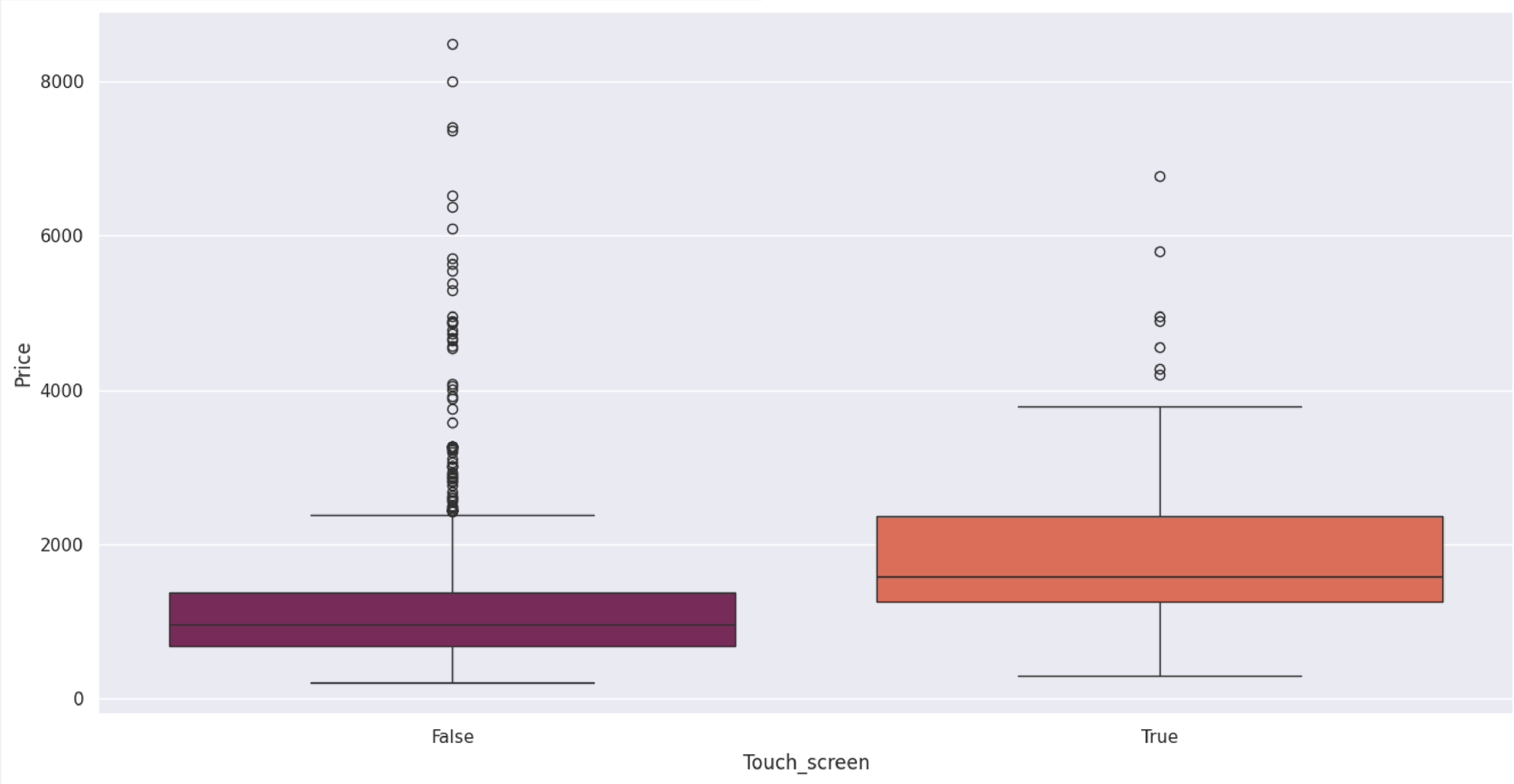
Processor_brand vs Price



Storage_type vs Price



Touch_screen vs Price



one-hot code encoding

convert categorical data into a numerical format that can be used by machine learning models

Processor_brand
AMD
AMD
Intel
Intel
Intel

Processor_brand_Intel
False
False
True
True
True

```
<class 'pandas.core.frame.DataFrame'>
Index: 960 entries, 0 to 1019
Data columns (total 8 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Price                       960 non-null    float64
 1   RAM_GB                      960 non-null    int64
 2   Storage_capacity_GB         960 non-null    int64
 3   Vertical_pixel              960 non-null    int64
 4   Touch_screen                960 non-null    bool
 5   Processor_brand_Intel       960 non-null    bool
 6   Storage_type_ SSD           960 non-null    bool
 7   Storage_type_Hard Disk & SSD  960 non-null  bool
dtypes: bool(4), float64(1), int64(3)
memory usage: 41.2 KB
```

Machine Learning

Linear Regression

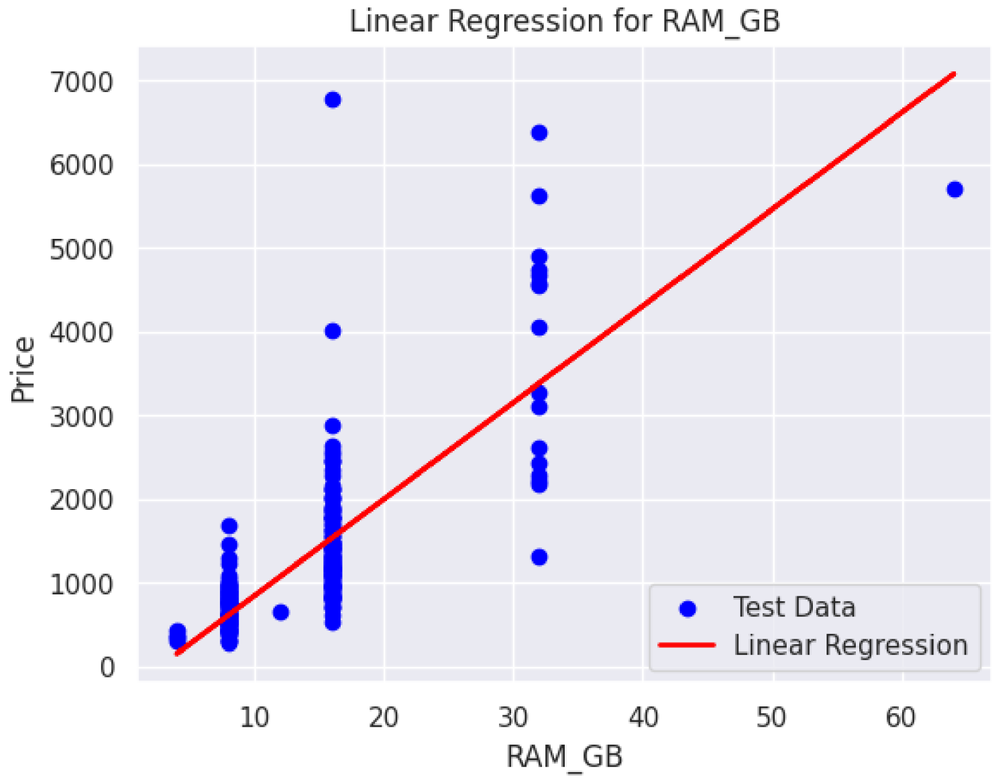
Uni-variate Linear Regression

- Response Variable : `Price`
- Predictors (numeric) : `RAM_GB`, `Storage_capacity_GB`, `Vertical_pixel`

> Regression Model 1 : Price = a × RAM_GB + b
> Regression Model 2 : Price = a × Storage_capacity_GB + b
> Regression Model 3 : Price = a × Vertical_pixel + b

Uni-variate Linear Regression

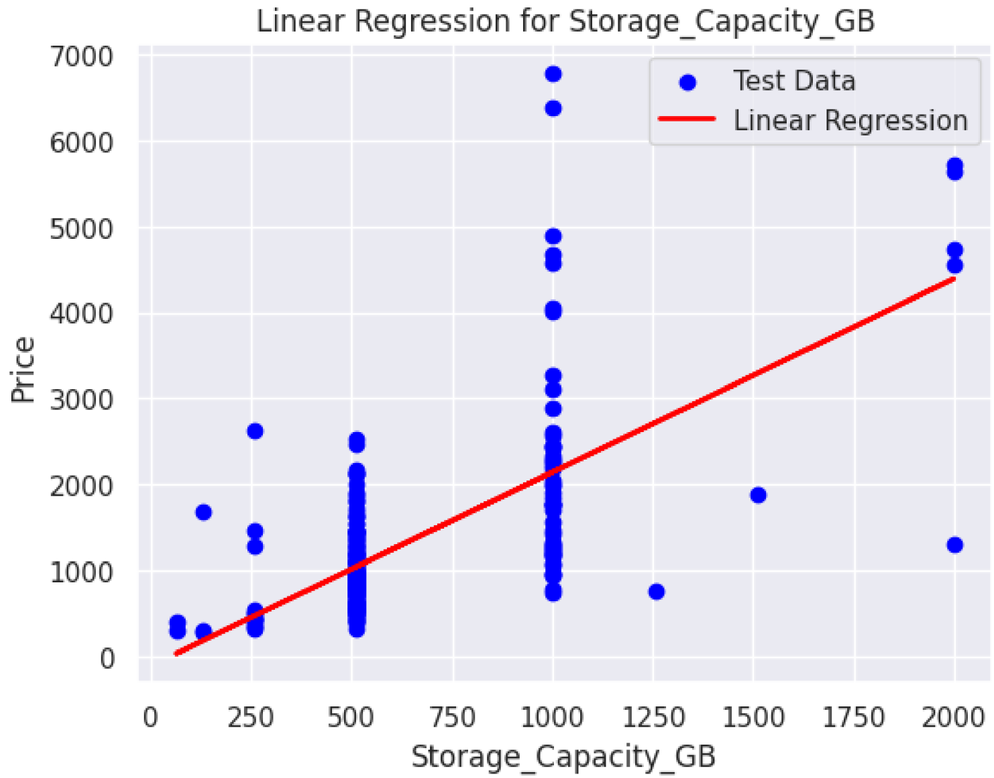
Ram_GB vs Price




```
Results for RAM_GB
Intercept of Regression          : b = [-312.52383191]
Coefficient of Regression        : a = [115.47861]
Goodness of Fit (R^2) for Train Data    : 0.5937892314432227
Goodness of Fit (R^2) for Test Data     : 0.5795842853445092
Mean Squared Error (MSE) for Train Data          : 420381.88105452555
Mean Squared Error (MSE) for Test Data  : 444526.0072530679
```

Uni-variate Linear Regression

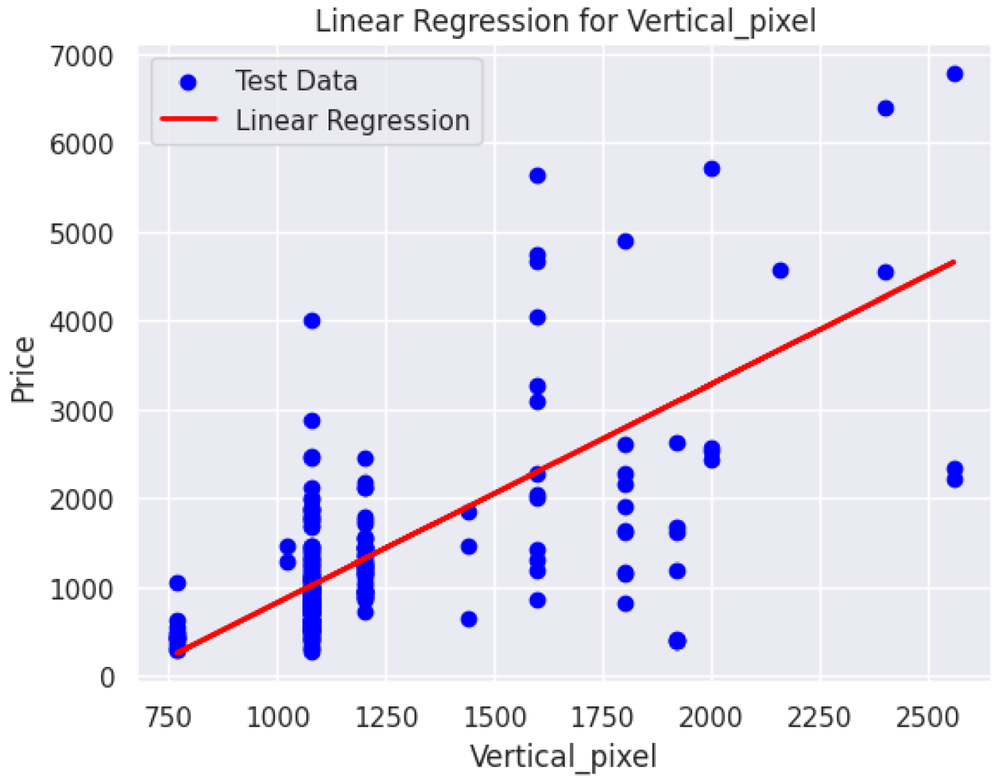
Storage_capacity_GB vs Price




```
Results for Storage_Capacity_GB
Intercept of Regression          : b = [-110.47146941]
Coefficient of Regression        : a = [2.24954682]
Goodness of Fit (R^2) for Train Data    : 0.454455272233384
Goodness of Fit (R^2) for Test Data     : 0.4081975420444347
Mean Squared Error (MSE) for Train Data          : 564576.659729428
Mean Squared Error (MSE) for Test Data  : 625741.5566235739
```

Uni-variate Linear Regression

Vertical_pixel vs Price




```
Results for Vertical_pixel
Intercept of Regression          : b = [-1628.86814016]
Coefficient of Regression        : a = [2.45472406]
Goodness of Fit (R^2) for Train Data    : 0.44750696770821385
Goodness of Fit (R^2) for Test Data     : 0.37505630084195285
Mean Squared Error (MSE) for Train Data         : 571767.3635524914
Mean Squared Error (MSE) for Test Data  : 660783.4047600604
```

Multiple Linear Regression

$$\text{Price} = a + b \times \text{RAM\_GB} + c \times \text{Storage\_capacity\_GB} + d \times \text{Vertical\_pixel}$$

Multiple Linear Regression

$$\text{Price} = a + b \times \text{RAM\_GB} + c \times \text{Storage\_capacity\_GB} + d \times \text{Vertical\_pixel}$$

```
Intercept of Regression: a = [-1589.12672444]
Coefficients of Regression:
   Coefficient for RAM_GB: 65.59587919895597
   Coefficient for Storage_capacity_GB: 0.8181843703835958
   Coefficient for Vertical_pixel: 1.218591820278586

Goodness of Fit of Model on Train Dataset
Explained Variance (R^2): 0.7247379050109457
Mean Squared Error (MSE): 284864.9179972054

Goodness of Fit of Model on Test Dataset
Explained Variance (R^2): 0.6624378581238661
Mean Squared Error (MSE): 356920.8901978224
```

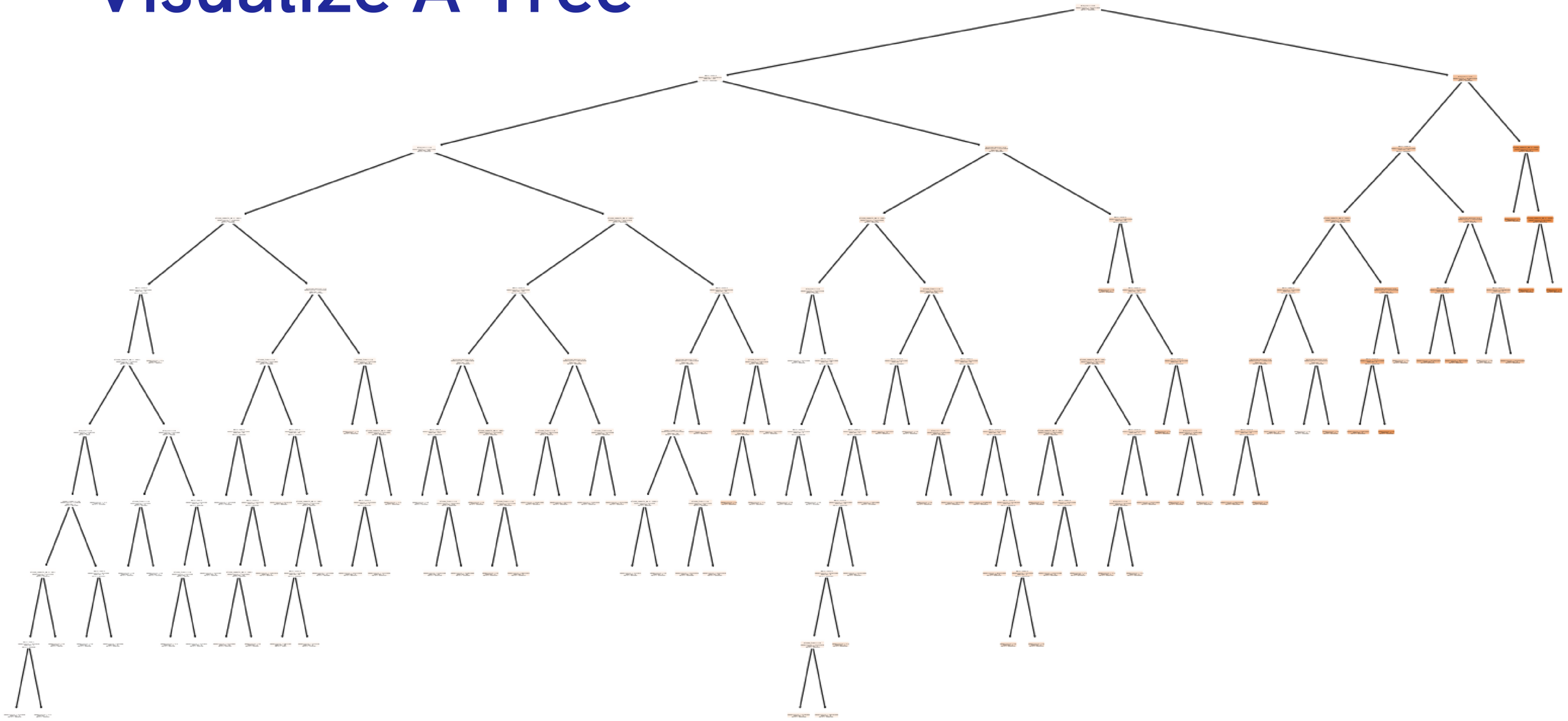
Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

```
ranforest = RandomForestRegressor(n_estimators=100, random_state=42)
ranforest.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Visualize A Tree



Visualizing actual vs predicted values

Train Data

Test Data

Model Performance

```
Goodness of Fit of Model        Train Dataset
Explained Variance (R^2)        : 0.8285298667638614
Mean Squared Error (MSE)        : 175061.87730160114

Goodness of Fit of Model        Test Dataset
Explained Variance (R^2)        : 0.6957592341740708
Mean Squared Error (MSE)        : 340374.60986594536
```

What if we remove the outliers?

Recall the outliers...

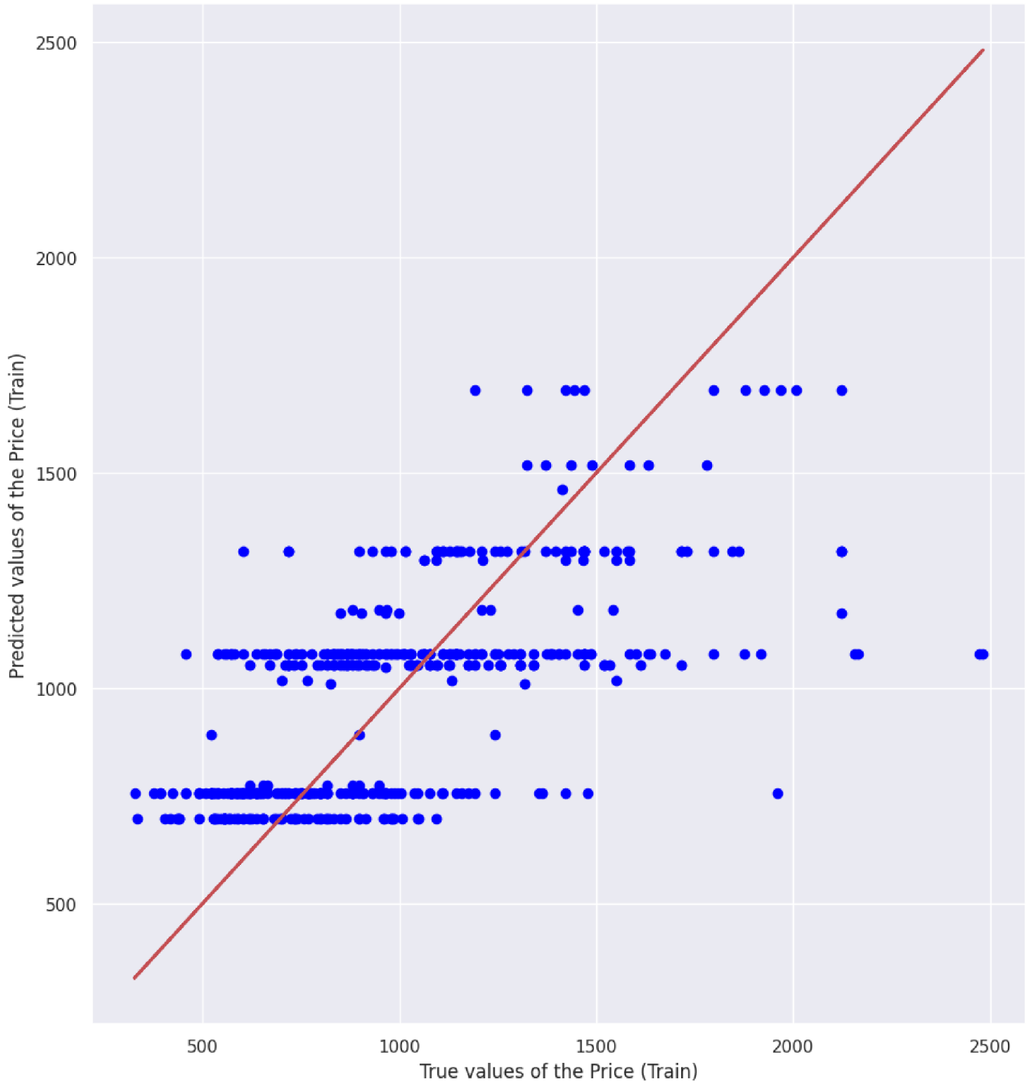
```
q1 = laptopData_numeric.quantile(0.25)
q3 = laptopData_numeric.quantile(0.75)
iqr = q3-q1
outliers = ((laptopData_numeric < q1-1.5*iqr) | (laptopData_numeric > q3+1.5*iqr)).sum()
outliers
```

```
Price                 71
RAM_GB                62
Storage_capacity_GB  280
Vertical_pixel       208
dtype: int64
```

After removing outliers, we have 600 rows left. Now, we attempt to train Random Forest to predict the Price again.

Visualizing actual vs predicted values

Train Data



Test Data

Model Performance

```
Goodness of Fit of Model        Train Dataset
Explained Variance (R^2)        : 0.4072310018481684
Mean Squared Error (MSE)        : 85261.85468707916

Goodness of Fit of Model        Test Dataset
Explained Variance (R^2)        : 0.3073319572071812
Mean Squared Error (MSE)        : 96554.4666831208
```

The goodness of fit becomes worse. WHY?

The goodness of fit worsens might be due to these reasons.

1. **Bias**
2. **Non-Linear Relationship**
3. **Reduced Sample Size**

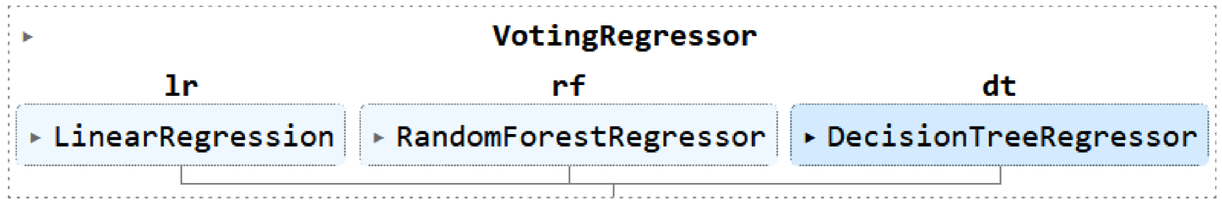
Since we have worse goodness of fit after removing the outliers, we will be using the random forest before removing the outliers.

Voting Regressor

A Voting regressor is an ensemble machine learning technique that combines multiple individual regression models to make predictions.The final prediction is typically the weighted average of these individual predictions.

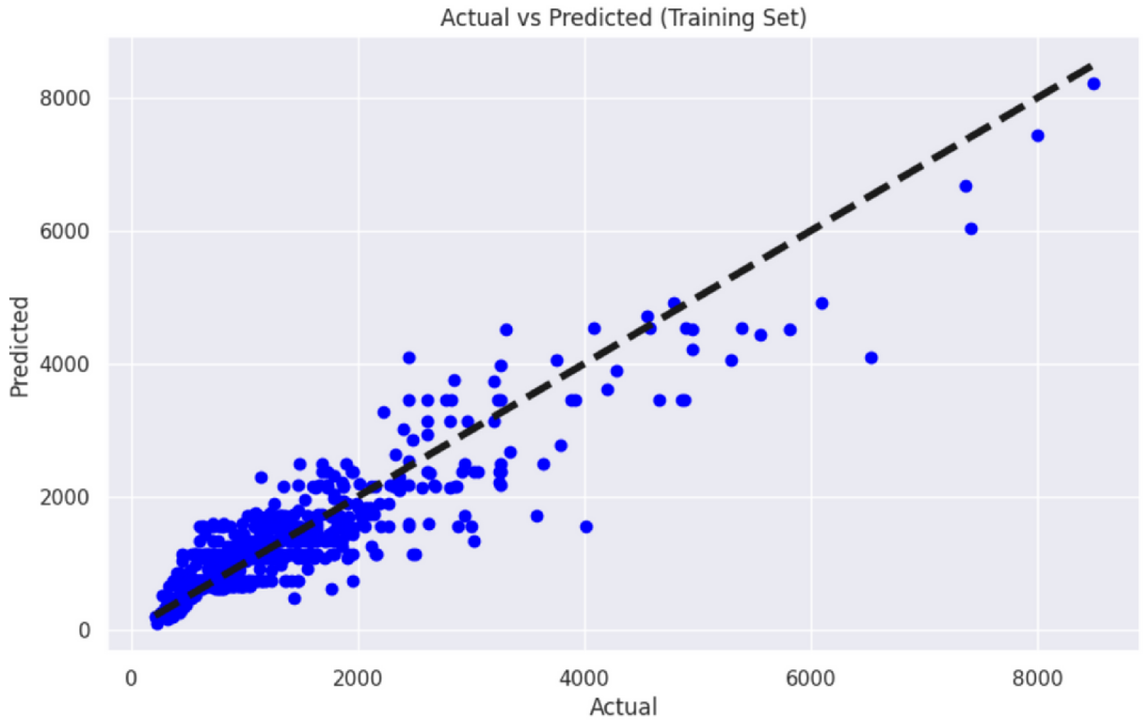
```
[ ]  voting_regressor = VotingRegressor(estimators=[('lr', linear_regressor), ('rf', rf_regressor), ('dt', dt_regressor)])
```

```
voting_regressor.fit(X_train, y_train)
```

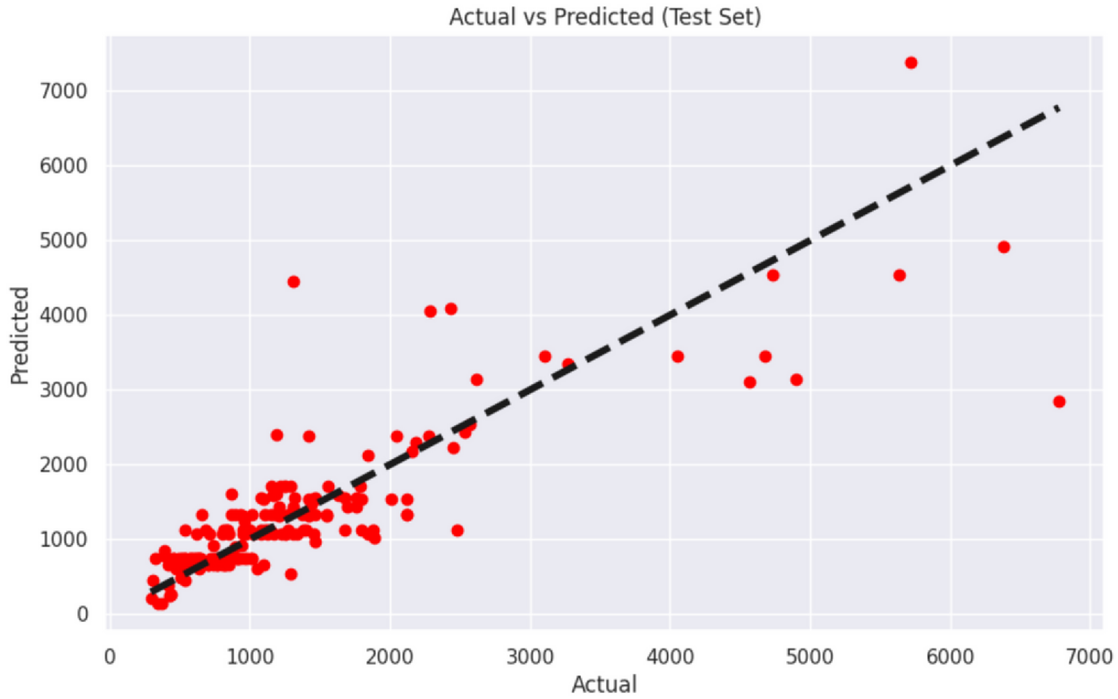


Visualizing actual vs predicted values

Train Data



Test Data



Model Performance

```
[ ]  print("Goodness of Fit of Model \tTrain Dataset")
     print("Explained Variance (R^2) \t:", voting_regressor.score(X_train, y_train))
     print("Mean Squared Error (MSE) \t:", mean_squared_error(y_train, y_train_pred_voting))
     print()

     print("Goodness of Fit of Model \tTest Dataset")
     print("Explained Variance (R^2) \t:", voting_regressor.score(X_test, y_test))
     print("Mean Squared Error (MSE) \t:", mean_squared_error(y_test, y_test_pred_voting))
     print()
```

```
Goodness of Fit of Model        Train Dataset
Explained Variance (R^2)        : 0.8217318167920208
Mean Squared Error (MSE)        : 182002.3244080463

Goodness of Fit of Model        Test Dataset
Explained Variance (R^2)        : 0.7028083943550116
Mean Squared Error (MSE)        : 332488.24020093115
```

Price Predictor

Predicting Price using RAM_Size, Vertical Resolution, Storage Capacity

```
52
53      # Predict price
54      predicted_price = predict_price(RAM_GB, Storage_capacity_GB, Vertical_pixel, model)
55
56      print("Predicted Price:", predicted_price)
57
58  if __name__ == "__main__":
59      main()
60
```

```
Training RMSE: 465.324140696853
Testing RMSE: 596.6339169151588
Enter RAM size in GB: 16
Enter storage capacity in GB: 1024
Enter vertical pixel resolution: 1080
Predicted Price: 1489.0748779982541
```

Conclusion

How do different factors affect the price of laptop?

1) Ram Size, Vertical Resolution and storage Capacity have strong positive correlation with price

2) laptops with SSD, touch screen and Intel CPU’s have higher median prices than their counterparts.

Which model would be the best to predict laptop price?

1) Random Forest

2) Highest explained variance and lowest mean squared error

3) Best prediction accuracy

Thank You