

PROJECT-LAPTOP PRICE PREDICTION  
(DATA ANALYSIS AND MACHINE  
LEARNING PIPELINE)  
ABHISHEK JADHAV

# PROJECT *WORKFLOW*

HAVE A LOOK BELOW FOR ANY KEY POINTS AND INFORMATION  
REGARDING THIS PROJECT.

## **DATA LOADING AND EDA**

IMPORTING NECESSARY  
LIBRARIES LIKE PANDAS  
FOR LOADING DATA INTO  
DATAFRAME. CHECKING  
MISSING VALUES AND  
PERFORMING EDA.

## **DATA VISUALIZATION & SPLITTING**

IMPORTING MATPLOTLIB FOR  
PLOTING GRAPHS. SPLITTING THE  
DATA INTO TRAINING AND  
TESTING SETS.

## **FEATURE SCALING, MODEL TRAINING AND PRICE PREDICTION**

APPLYING ONE-HOT ENCODE,  
INITIALIZING SCALER, TRAINING  
MODEL ON LINEAR REGRESSION  
AND USING MODEL FOR PRICE  
PREDICTION

## **HYPER-PARAMETER TUNING, VALIDATION, ESTIMATION.**

HYPERPARAMETER TUNING FOR  
FOREST REGRESSOR. MODEL  
PERFORMANCE. MEAN  
ABSOLUTE ERROR

## **MORE ACTIONS AHEAD**

# Table of Contents

<b>1.Introduction.....</b>	<b>4</b>
<b>2.Client.....</b>	<b>4</b>
<b>3.Dataset.....</b>	<b>4</b>
<b>4.Data Loading and Preprocessing.....</b>	<b>5</b>
<b>5.Exploratory Data Analysis.....</b>	<b>7</b>
<b>6.Data Visualization.....</b>	<b>8</b>
a. Distribution of laptop prices.....	8
b. Relationship between weight vs price.....	9
<b>7.Data Splitting.....</b>	<b>10</b>
<b>8.Feature Scaling.....</b>	<b>10</b>
<b>9.Model Training.....</b>	<b>12</b>
<b>10.Price Prediction.....</b>	<b>13</b>
<b>11.Model Evaluation.....</b>	<b>14</b>
<b>12.Feature Engineering.....</b>	<b>14</b>
<b>13.Model Selection.....</b>	<b>16</b>
<b>14.Hyperparameter Tuning.....</b>	<b>17</b>
<b>15.Model Interpretation.....</b>	<b>19</b>
<b>16.Residual Analysis.....</b>	<b>20</b>
<b>17.Validation.....</b>	<b>21</b>
<b>18.Uncertainty Estimation.....</b>	<b>22</b>
<b>19.Conclusion.....</b>	<b>23</b>
<b>20.Further Analysis.....</b>	<b>24</b>
<b>21.Strategic Recommendations.....</b>	<b>24</b>
<b>22.Figures.....</b>	
a. Figure 4.1.....	5
b. Figure 4.2.....	6
c. Figure 5.1.....	7
d. Figure 6.1.....	8
e. Figure 6.2.....	9
f. Figure 8.1.....	11
g. Figure 10.1.....	13
h. Figure 12.1.....	15
i. Figure 15.1.....	19
j. Figure 16.1.....	20

# 1.INTRODUCTION

The Laptop Price Prediction dataset is a collection of laptop products. The dataset contains 1300 rows and 12 columns, including information about the laptop manufacturer, brand and model, type, screen size, screen resolution, CPU, RAM, memory, GPU, operating system, weight, and price.

The dataset can be used to explore trends, patterns, and correlations among different laptop specifications. It can also be used to build predictive models to estimate laptop prices, assess performance benchmarks, or predict user preferences based on specific features.

## 2.Client

A laptop manufacturer could use the dataset to identify the features that are most important to consumers when they are making a laptop purchase. This information could then be used to design laptops that are more likely to appeal to consumers.

An online retailer could use the dataset to identify the laptops that are most popular with consumers. This information could then be used to highlight these laptops on the retailer's website and to promote them to consumers.

## 3. Dataset

This dataset provides a comprehensive collection of laptop attributes and features, Containing information on thousands of laptops, the dataset encompasses a wide range of brands, models, and configurations. It includes both entry-level and high-end laptops, catering to diverse user needs and preferences. Each laptop entry within the dataset offers a plethora of attributes which are listed below.

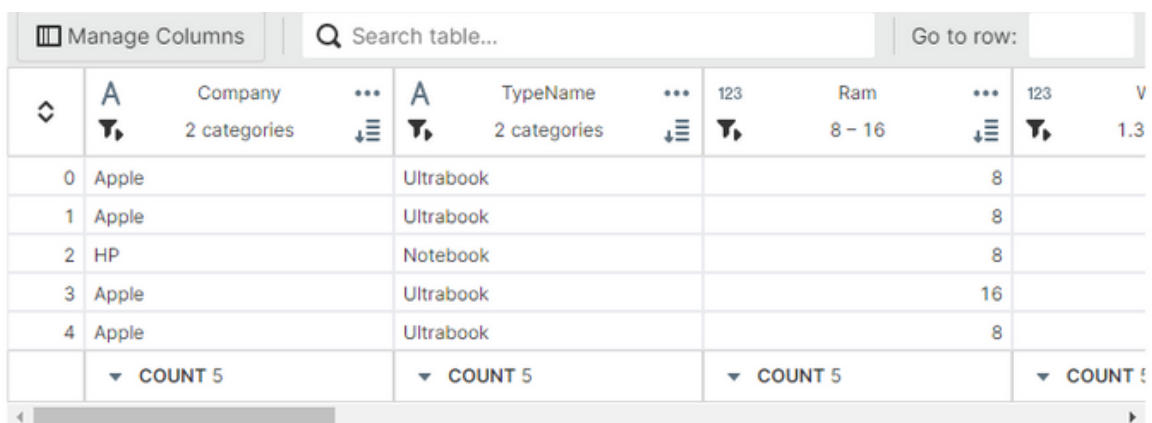
- Manufacturer: The name of the laptop manufacturer.
- Model: The model number of the laptop.
- Type: The type of laptop, such as a notebook, ultrabook, or gaming laptop.
- Screen size: The size of the laptop's screen, in inches.
- Screen resolution: The resolution of the laptop's screen, in pixels.
- CPU: The type of processor that the laptop uses.

- RAM: The amount of RAM that the laptop has.
- Memory: The type of memory that the laptop has, such as DDR3 or DDR4.
- GPU: The type of graphics card that the laptop has.
- Operating system: The operating system that the laptop runs.
- Weight: The weight of the laptop, in pounds.
- Price: The price of the laptop, in US dollars.

## 4. Data Loading and Preprocessing

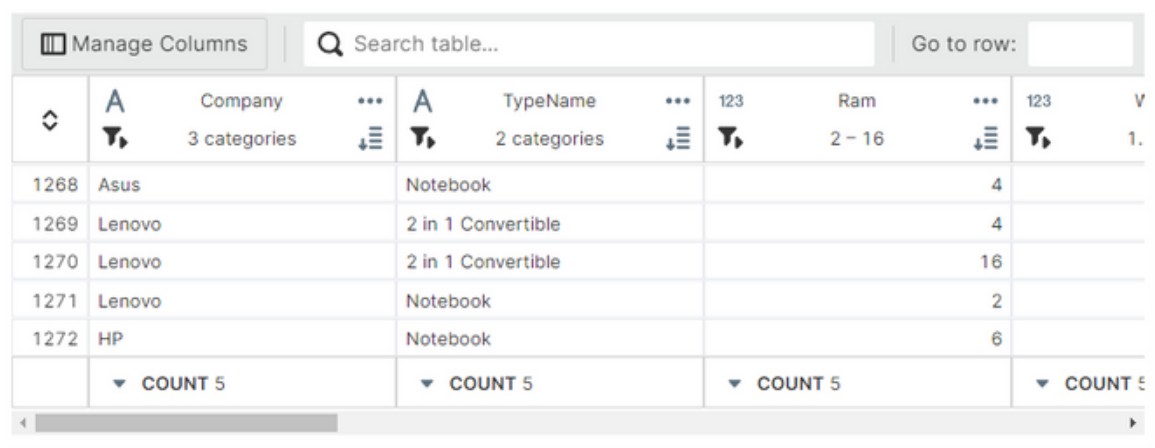
Data loading refers to reading data from a data source (eg. `laptop_data = pd.read_csv(file name)`). Using pandas library.

Once data is loaded then it is important that we analyze the data which we are going to work with like display first few rows of data.



	Company 2 categories	TypeName 2 categories	Price 123	Ram 8 - 16	Weight 123
0	Apple	Ultrabook		8	
1	Apple	Ultrabook		8	
2	HP	Notebook		8	
3	Apple	Ultrabook		16	
4	Apple	Ultrabook		8	
	▼ COUNT 5	▼ COUNT 5	▼ COUNT 5		▼ COUNT 5

We can also display last few rows of data.



	Company 3 categories	TypeName 2 categories	Price 123	Ram 2 - 16	Weight 123
1268	Asus	Notebook		4	
1269	Lenovo	2 in 1 Convertible		4	
1270	Lenovo	2 in 1 Convertible		16	
1271	Lenovo	Notebook		2	
1272	HP	Notebook		6	
	▼ COUNT 5	▼ COUNT 5	▼ COUNT 5		▼ COUNT 5

Figure 4.1

It is important to take a look at the data types the dataset has as well as if the dataset contains null values or not. Generally the cleaned dataset does not contain any null values but still its a good practise.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1273 entries, 0 to 1272
Data columns (total 13 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Company         1273 non-null   object
1   TypeName         1273 non-null   object
2   Ram              1273 non-null   int64
3   Weight          1273 non-null   float64
4   Price           1273 non-null   float64
5   TouchScreen     1273 non-null   int64
6   Ips             1273 non-null   int64
7   Ppi             1273 non-null   float64
8   Cpu_brand       1273 non-null   object
9   HDD             1273 non-null   int64
10  SSD             1273 non-null   int64
11  Gpu_brand       1273 non-null   object
12  Os              1273 non-null   object
dtypes: float64(3), int64(5), object(5)
memory usage: 129.4+ KB
```

The dataset does not contain any missing values.

**Figure 4.2**

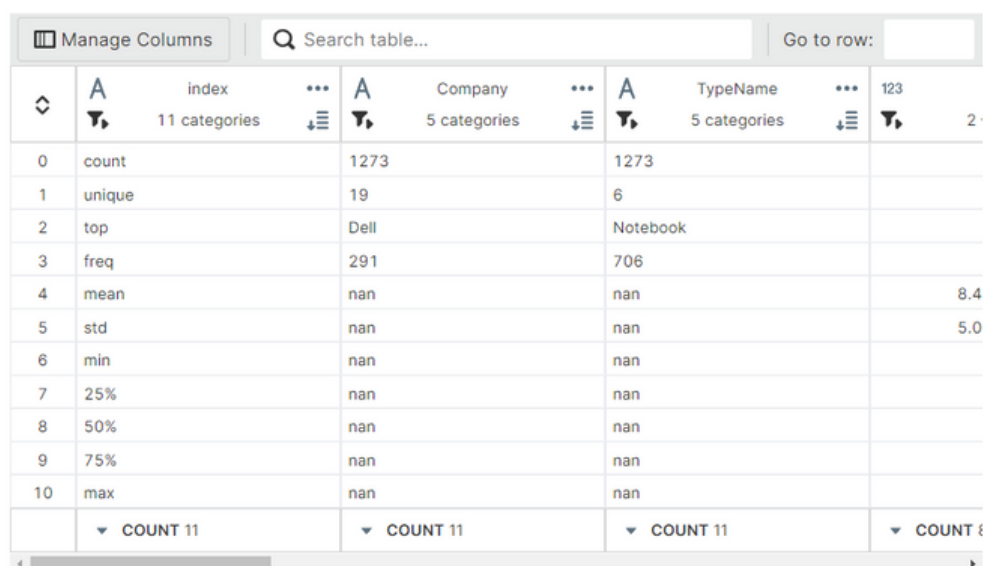
The shape of a DataFrame is a tuple of two numbers, representing the number of rows and the number of columns in the DataFrame.

In this case, the shape of data is (1273, 13). This means that there are 1273 rows and 13 columns in the DataFrame. The 1273 rows represents the 1273 laptops in the dataset and the 13 columns represent the 13 features of each laptop.

Here shape of the data DataFrame is used to get an overview of the data, used it here to determine the number of rows and columns in the dataframe.

## 5. Exploratory Data Analysis

Exploratory data analysis (EDA) is a critical process of performing initial studies on data so as to discover patterns and understand the data in depth. This understanding is essential for informed decisions about data.



The screenshot shows a data table interface with a search bar and a 'Go to row' field. The table displays summary statistics for three columns: 'index' (11 categories), 'Company' (5 categories), and 'TypeName' (5 categories). The statistics include count, unique values, top values, frequency, mean, standard deviation, minimum, and 25%, 50%, 75%, and maximum quantiles. The 'index' column has a count of 1273, 19 unique values, 'Dell' as the top value, a frequency of 291, and a mean of nan. The 'Company' column has a count of 1273, 6 unique values, 'Notebook' as the top value, a frequency of 706, and a mean of nan. The 'TypeName' column has a count of 123, 2 unique values, and a mean of 8.4. The table also shows the standard deviation for 'index' as 5.0 and the minimum for 'Company' as nan. The 25%, 50%, 75%, and maximum quantiles for all three columns are nan.

	index 11 categories	Company 5 categories	TypeName 5 categories	
0	count	1273	1273	
1	unique	19	6	
2	top	Dell	Notebook	
3	freq	291	706	
4	mean	nan	nan	8.4
5	std	nan	nan	5.0
6	min	nan	nan	
7	25%	nan	nan	
8	50%	nan	nan	
9	75%	nan	nan	
10	max	nan	nan	
	▼ COUNT 11	▼ COUNT 11	▼ COUNT 11	▼ COUNT 11

Figure 5.1

- Count: The number of rows in the DataFrame.
- Mean: The average value of the column.
- Standard deviation: The amount of variation in the column.
- Minimum: The smallest value in the column.
- Maximum: The largest value in the column.
- 25% quantile: The value below which 25% of the values in the column fall.
- 50% quantile: The value below which 50% of the values in the column fall.
- 75% quantile: The value below which 75% of the values in the column fall.

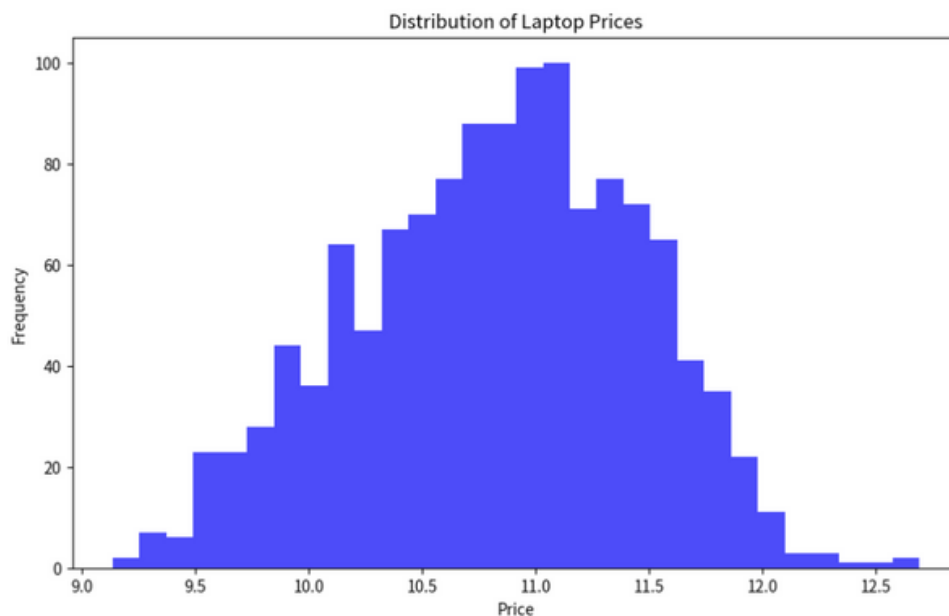
The `include='all'` in the code snippet tells `describe()` function to include all the columns in the DataFrame, even if they are not numeric. This is useful as it can help to identify any categorical columns in the DataFrame.

## 6.Data Visualization

Data Visualization is the first step in data analysis and typically involves summarizing the main characteristics of a dataset. It is commonly conducted using visual analytics tools. Data Visualization is the best way to explore the data because it allows users to quickly and simply view most of the relevant features of the dataset. Displaying the data graphically using bar charts, pie chart, scatter plots by this users can identify variables that they might be helpful for further in-depth analysis.

I used matplotlib python library for my visualization. Here i have plotted two graphs for two different parameters from the dataset.

a. Distribution of laptop prices.



**Figure 6.1**

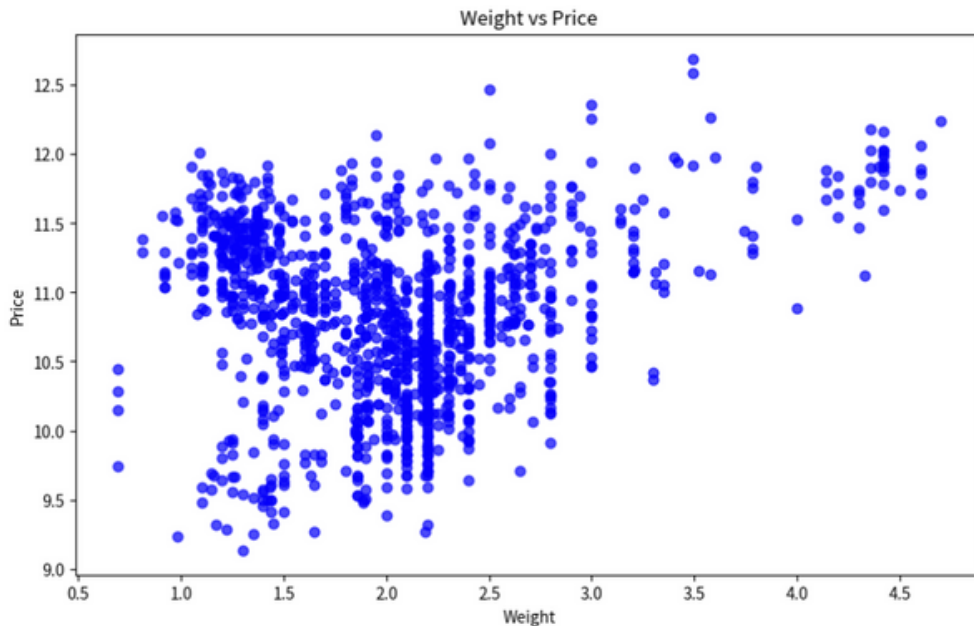
This graph shows the map of laptop prices each bar represents that how many laptops fall into a certain price range and the height of the bar tells us how many laptops are in that price range.

- The tallest bars are where most laptops are priced. This is the most common price range for laptops.
- The width of the graph from left to right shows us the cheapest to the most expensive laptops.



- If the bars are mostly on the left (lower prices) with a few bars stretching out to the right (higher prices), it means there are a few very expensive laptops. If it's the other way around, it means there are a few very cheap laptops.
- If there are any bars far away from the others, these are unusually cheap or expensive laptops.

b. Relationship between weight vs price.



**Figure 6.2**

In this graph, each dot on this plot represents a laptop. The position of the dot shows the weight of the laptop (on the x-axis) and its price (on the y-axis).

- If the dots seem to go up as you move to the right, it means heavier laptops tend to be more expensive. If they go down, it means heavier laptops are cheaper. If there's no clear direction, it means the weight doesn't really affect the price.
- If the dots are close to forming a line, it means the weight strongly affects the price. If they're all over the place, it means the weight doesn't have much effect on the price.
- If there are any dots far away from the others, these are unusual laptops. For example, a very light laptop that's very expensive, or a very heavy laptop that's very cheap.

## 7.Data Splitting

Splitting the data into training and testing sets is called a fundamental practise in machine learning and below are steps why we do it:

- **Model Training:** The training data is used to train the machine learning model where model learns from this data by adjusting its parameters to minimize the difference between its predictions and actual values.
- **Model Evaluation:** The testing data is used to evaluate the model's performance. It's data that the model hasn't seen during training, so it's a good measure of how the model will perform on new, unseen data in the future.
- **Prevent Overfitting:** If we used all our data for training, the model might perform very well on that data but poorly on new, unseen data. This is because the model might overfit to the training data, learning not only the general patterns but also the noise and outliers specific to the training data. By holding out a test set, we can ensure that the model also performs well on unseen data, which is ultimately what we care about.
- **Model Selection:** If we're comparing different models or tuning hyperparameters, the test set provides an unbiased way to select the best model or hyperparameters.

So with my dataset the data was successfully split into training and testing .The training set contains 1018 samples, and the test set contains 255 samples.

## 8.Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

In your dataset, feature scaling is performed using the StandardScaler from the sklearn.preprocessing module. This is a type of feature scaling called standardization and rest process are listed below.

- **Combining the Training and Test Sets:** The training and test sets are combined into one dataset. This is done because the next steps (encoding and scaling) need to be applied consistently across the entire dataset to ensure that the model can understand the data correctly.
- **One-Hot Encoding:** The combined dataset is then one-hot encoded. One-hot encoding is a process that turns categorical data into a format that can be used by machine learning algorithms. It creates new binary columns for each category of each original categorical column.
- **Splitting the Encoded Data:** The encoded data is then split back into training and test sets. This is done so that the model can be trained on one set of data (training set) and then tested on unseen data (test set) to evaluate its performance.
- **Scaling the Data:** The data is then scaled using a method called Standard Scaling. This process transforms the data to have a mean of 0 and a standard deviation of 1. This is done because many machine learning algorithms perform better when numerical input variables are scaled to a standard range.

In summary, these steps are performed to transform the raw data into a format that can be effectively used by a machine learning model.

```
(array([[ -0.09515399,  0.40042557, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136],
       [ -0.09515399, -1.39675382, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136],
       [ -0.86700066, -0.43825814,  2.42453673, ..., -0.13416408,
        -0.36382592,  0.39434136],
       ...,
       [ -0.09515399,  0.01103671, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136],
       [ -0.09515399,  0.40042557, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136],
       [ -0.86700066, -2.01079011,  2.42453673, ..., -0.13416408,
        -0.36382592,  0.39434136]]),
 array([[ -0.86700066,  0.25066062, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136],
       [ -0.09515399, -0.49816412, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136],
       [ 1.44853936,  1.14925032, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136],
       ...,
       [ 1.44853936,  3.24595961, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136],
       [ 1.44853936, -1.21703588,  2.42453673, ..., -0.13416408,
        -0.36382592,  0.39434136],
       [ 1.44853936,  0.69995547, -0.41244993, ..., -0.13416408,
        -0.36382592,  0.39434136]]))
```

**Figure 8.1**

## 9. Model Training

In this process a Linear Regression model is being trained. I preferred to use the Linear Regression model because the task is a regression task- predicting a continuous outcome of laptop prices from a set of features which are Laptop specifications and below are some reasons why Linear Regression should be used.

- **Simplicity:** Linear Regression is a simple model that can be trained and interpreted easily. The coefficients of the model give direct insights into how each feature affects the prediction.
- **Efficiency:** Linear Regression is computationally efficient compared to some other algorithms, which can be an advantage when dealing with large datasets.
- **Baseline Model:** Due to its simplicity and efficiency, Linear Regression is often used as a baseline in regression tasks. The performance of the Linear Regression model can be used as a benchmark, and you can then try more complex models to see if they give a significant improvement.
- **Assumptions:** Linear Regression assumes a linear relationship between the features and the target variable. If this assumption holds true (at least approximately), Linear Regression can give good results.

Below are the steps how it works with my dataset.

- **Import the Model:** The first line imports the LinearRegression model from the `sklearn.linear_model` module. Linear Regression is a basic and commonly used type of predictive analysis which is used when the target (or dependent variable) is continuous. It establishes a relationship between dependent variable (Y) and one or more independent variables (X).
- **Initialize the Model:** The second line creates an instance of the LinearRegression model.
- **Train the Model:** The third line trains the model on the training data. The fit method is called on the model and passed the scaled training data (`X_train_scaled`) and training labels (`y_train`).

# 10.Price Prediction

The model is now successfully trained. The Linear Regression model is now ready to make predictions on the test data and that is what i am going to perform in this section.

This step is crucial in the machine learning process because it allows us to see how well our model performs on unseen data. After making these predictions, we would typically compare them to the actual prices (the `y_test` values) to evaluate the model's performance. This could be done using various metrics such as Mean Absolute Error, Mean Squared Error, or R-squared, depending on the specific requirements of your task which i have in the further part.

Breakdown of what's happening:

- **Predictions:** The `predict` method is called on the trained model (model) and passed the scaled test data (`X_test_scaled`). The model uses the patterns it learned during training to predict the laptop prices for the test data. These predictions are stored in the `y_pred` variable.
- **Output Predictions:** The predicted prices (`y_pred`) are then printed out. These are the prices that the model has predicted for the laptops in the test set, based on their specifications. These predictions are in the form of array.

```
array([10.64949934, 11.17116587, 11.04490702, 11.42626039, 11.12639372,
       11.16795198,  9.93721144, 10.04369776, 10.99946348, 11.91976778,
       10.80402796, 10.83155958, 11.04169981, 10.43536798, 11.34690992,
       11.31614916, 11.27254907,  9.99725668, 11.34872286, 11.26789705,
        9.94811766, 10.81297342, 10.77122251, 11.32626097, 10.09285776,
       11.32742159, 10.39855615, 10.78473036, 10.89244406, 10.91643659,
       11.0803198 , 11.10132353, 10.24745409, 10.44309275, 11.1911873 ,
       10.58307306, 10.73668043, 10.86405603, 11.47962801, 10.12960855,
       10.19458238, 11.49939577, 11.15379183, 11.23041002, 10.33935968,
       11.1437277 , 11.28896944, 10.22468225, 10.9505171 , 10.79536192,
       10.55766146, 11.20388643, 11.12686674, 10.02708952, 10.96347467,
       10.75838129, 10.98777525, 10.35925142, 10.98503725, 12.8575577 ,
       11.81586496, 10.88114874, 10.75314085,  9.92990248, 11.33706133,
       10.57400934, 10.07301752, 10.57321874, 10.66113798, 10.86251776,
       11.63468973, 10.19132654, 11.59814875, 10.89485018, 10.15996782,
       11.4671921 , 10.86740629, 11.39226381, 10.18812219, 10.93637697,
       10.98869078, 10.65909235, 10.13804285,  9.98833029, 10.51730387,
       11.42312471, 10.62006895, 10.67252867, 10.45433466, 10.7469372 ,
       11.33421175, 10.98100702, 11.12573854, 10.88925878, 10.73493425,
```

Figure 10.1

Note- The output has more values and is not limited as shown in this image checkout jupyter notebook for exact output

## 11. Model Evaluation

In model evaluation, I have compare the model's predictions on the test set to the actual values. There are various metrics For a regression task like this one, but here i am going with Mean Absolute Error(MAE).

Mean Absolute Error (MAE): This is the average of the absolute differences between the predicted and actual values. It gives an idea of how wrong the predictions were. Here's breakdown of what's happening.

- **Import the Metric:** The first line imports the `mean_absolute_error` function from the `sklearn.metrics` module. This function calculates the Mean Absolute Error.
- **Calculate the MAE:** The second line calculates the MAE of the model's predictions. The `mean_absolute_error` function is called and passed the actual values (`y_test`) and the predicted values (`y_pred`). The MAE is the average of the absolute differences between the predicted and actual values.
- **Print the MAE:** The third line prints the calculated MAE. The output of this cell is 0.2151762562177748, which means that on average, the model's predictions are off by about 0.215 (assuming the prices are scaled).

Output: The mean absolute error (MAE) of the model on the test data is approximately 0.215. This means that on average, the model's predictions are about 0.215 units away from the actual prices.

## 12. Feature Engineering

Feature engineering is the process of creating new features or modifying existing features to improve the performance of a machine learning model. Here I have created a new feature by name 'Ram\_per\_weight' which is the ratio of ram to weight then split the data and perform the LinearRegression model again on that new data.

Then next I have split the data into a training set and a test set. This new feature represents the amount of Ram per unit weight for each laptop.

Feature engineering like this can often improve the performance of a machine learning model.




	123 	Ram_per_Weight 2.3 - 8.42	... 
490		3.4782608695652177	
405		7.2727272727272725	
156		2.2988505747126435	
650		8.421052631578947	
770		6.9565217391304355	
	▼ COUNT 5		

Figure 12.1

Successfully created a new feature 'Ram\_per\_Weight', which is the ratio of Ram to Weight. Here are the first few values of this new feature in the training set. Next is the process I have performed on it.

- Initialize the Model: This is my machine learning model that will be trained.
- Train the Model: In the training process, the model learns patterns in the training data that map the features to the target variable.
- Make Predictions: The model uses the patterns it learned during training to predict the laptop prices for the test data. These predictions are stored in the 'y\_pred' variable.
- Calculate the MAE: This calculates the Mean Absolute Error (MAE) of the model's predictions.
- Print the MAE: This prints the score which is '0.214'.

The mean absolute error (MAE) of the model on the test data after feature engineering is approximately 0.214. This is slightly lower than the MAE before feature engineering (0.215), suggesting that the new feature 'Ram\_per\_Weight' may have slightly improved the model's performance.

## 13. Model Selection

Model selection is the task of selecting a statistical model from a set of candidate models. Given a set of data, you want to select the best model that makes the best predictions on unseen data. Below is the breakdown of the process performed in the cell.

- **Importing the Models:** The first two lines import the RandomForestRegressor and SVR (Support Vector Regressor) models from the sklearn.ensemble and sklearn.svm modules, respectively.
- **Initialize the Models:** The next three lines create instances of the LinearRegression, RandomForestRegressor, and SVR models. These are the machine learning models that will be trained.
- **Train the Models:** The next three lines train the models on the training data. The fit method is called on each model and passed the encoded training data (X\_train\_encoded) and training labels (y\_train). During the training process, each model learns patterns in the training data that map the features (laptop specifications and the 'Ram\_per\_Weight' feature) to the target variable (laptop price).
- **Make Predictions:** The next three lines use the trained models to make predictions on the test data. The predict method is called on each model and passed the encoded test data (X\_test\_encoded). Each model uses the patterns it learned during training to predict the laptop prices for the test data.
- **Calculate the MAE:** The next three lines calculate the Mean Absolute Error (MAE) of each model's predictions. The mean\_absolute\_error function is called and passed the actual values (y\_test) and the predicted values (y\_pred1, y\_pred2, y\_pred3).
- **Print the MAE:** The last line prints the calculated MAE for each model. The output of this cell is (0.2136121865339871, 0.1638017287072435, 0.2916094453377771), which are the MAEs for the Linear Regression, Random Forest, and SVR models, respectively.



The mean absolute error (MAE) of each model on the test data is as follows:

- Linear Regression: 0.214
- Random Forest Regressor: 0.164
- Support Vector Regressor: 0.292

The Random Forest Regressor has the lowest MAE, suggesting that it is the best performing model among the three.

## 14. Hyperparameter Tuning

Hyperparameter tuning is the process of optimizing the settings of a machine learning model to improve its performance. Hyperparameters are the parameters of the model that are not learned from the data, but are set before training. They control aspects of the model such as the complexity, the learning rate, or the regularization strength.

Hyperparameter tuning can be done using various methods, such as grid search, random search, or more advanced methods like Bayesian optimization. The goal is to find the combination of hyperparameters that gives the best performance on a validation set or via cross-validation. Below is the breakdown of what's happening.

- **Import GridSearchCV:** The first line imports the GridSearchCV function from the sklearn.model\_selection module. GridSearchCV is a function that performs grid search cross-validation, which is a method of hyperparameter tuning.
- **Define the Hyperparameters:** The next few lines define the hyperparameters to be tuned and their potential values. The hyperparameters for the RandomForestRegressor model are 'n\_estimators' (the number of trees in the forest), 'max\_depth' (the maximum depth of the trees), and 'min\_samples\_split' (the minimum number of samples required to split a node).
- **Initialize GridSearchCV:** The next line initializes the GridSearchCV function. It's passed the RandomForestRegressor model, the hyperparameters, the number of folds for cross-validation (cv=5), and the scoring metric ('neg\_mean\_absolute\_error').

- **Perform Grid Search:** The last line calls the fit method on the GridSearchCV object to perform the grid search. It's passed the encoded training data (X\_train\_encoded) and training labels (y\_train). GridSearchCV will train a RandomForestRegressor model for each combination of hyperparameter values, perform cross-validation, and calculate the mean absolute error for each model.

The grid search has completed successfully. The best hyperparameters for the Random Forest Regressor were found and the model was retrained with these hyperparameters.

The best model found by the grid search is being used to make predictions on the test data, and the Mean Absolute Error (MAE) of these predictions is being calculated. Here's a process of what's happening:

- **Make Predictions:** The first line uses the best model found by the grid search to make predictions on the test data. The predict method is called on the best estimator and passed the encoded test data. The best estimator is the model with the hyperparameters that performed best during the grid search. The model uses the patterns it learned during training to predict the laptop prices for the test data. These predictions are stored in the y\_pred variable.
- **Calculate the MAE:** The second line calculates the Mean Absolute Error (MAE) of the model's predictions. The mean\_absolute\_error function is called and passed the actual values and the predicted values. The MAE is the average of the absolute differences between the predicted and actual values. It gives an idea of how wrong the predictions were, on average.
- **Print the MAE:** The last line prints the calculated MAE. The output of this cell is 0.16453728088498643, which means that on average, the model's predictions are off by about 0.165.

The mean absolute error (MAE) of the tuned model on the test data is approximately 0.165. This is higher than the MAE of the untuned model (0.164), suggesting that the hyperparameter tuning process has not improved the model's performance.

# 15. Model Interpretation

Model interpretation is about understanding how a machine learning model makes its predictions. The purpose of this cell is to understand which features are most influential in predicting laptop prices according to the RandomForestRegressor model. This can provide valuable insights into the model's decision-making process and help us understand which features are most important. Following is the breakdown of the actual process.

- **Get the Feature Importances:** The first line gets the feature importances from the best estimator. The feature importances are a measure of how much each feature contributes to the predictions of the model. For a Random Forest model, the feature importances are calculated based on the average reduction in the impurity of the target variable brought about by each feature.
- **Create a DataFrame of the Features and Their Importances:** The next two lines create a DataFrame that contains each feature and its corresponding importance. The DataFrame has two columns: 'Feature' and 'Importance'.
- **Sort the DataFrame by Importance:** The next line sorts the DataFrame by the 'Importance' column in descending order. This makes it easy to see which features are most important.
- **Print the Feature Importances:** The last line prints the sorted DataFrame. The output shows the 10 most important features and their importances.

Manage Columns		Search table...	Go to row:
	Feature 20+ categories	Importance 0 - 0.56	
0	Ram	0.5560596312696292	
29	TypeName_Notebook	0.06461242968713304	
4	Ppi	0.06374687314882312	
1	Weight	0.054172496731687746	
36	Cpu_brand_Other Intel Processor	0.04929282611301137	
43	Ram_per_Weight	0.04044651222890126	
34	Cpu_brand_Intel Core i5	0.036953980865602795	
6	SSD	0.025667276116164452	
35	Cpu_brand_Intel Core i7	0.011264632925472471	
31	TypeName_Workstation	0.010889757307174223	
41	Os_Others	0.0074133644048243645	
42	Os_Windows	0.006850208895556304	
14	Company_HP	0.006478625614011168	
2	TouchScreen	0.006384740805414836	
5	HDD	0.006028805539122052	
COUNT 44		COUNT 44	

Figure 15.1

The 'Ram' feature is by far the most important, followed by 'TypeName\_Notebook', 'Ppi', and 'Weight'. The engineered feature 'Ram\_per\_Weight' also appears in the top 10, suggesting that it was a useful addition to the dataset.

## 16. Residual Analysis

Residual analysis is a technique used in statistics and machine learning to understand the discrepancies, or residuals, between the predicted and actual values produced by a model. Residuals are the difference between the actual value and the value predicted by the model. Here's breakdown of what's happening in the cell.

- **Calculate the Residuals:** The first line calculates the residuals, which are the differences between the actual values ( $y_{\text{test}}$ ) and the predicted values ( $y_{\text{pred}}$ ).
- **Create a Scatter Plot of the Predicted vs. Actual Values:** The next few lines create a scatter plot of the predicted values on the x-axis and the residuals on the y-axis. This kind of plot is often used in residual analysis to visualize the discrepancies between the predicted and actual values.

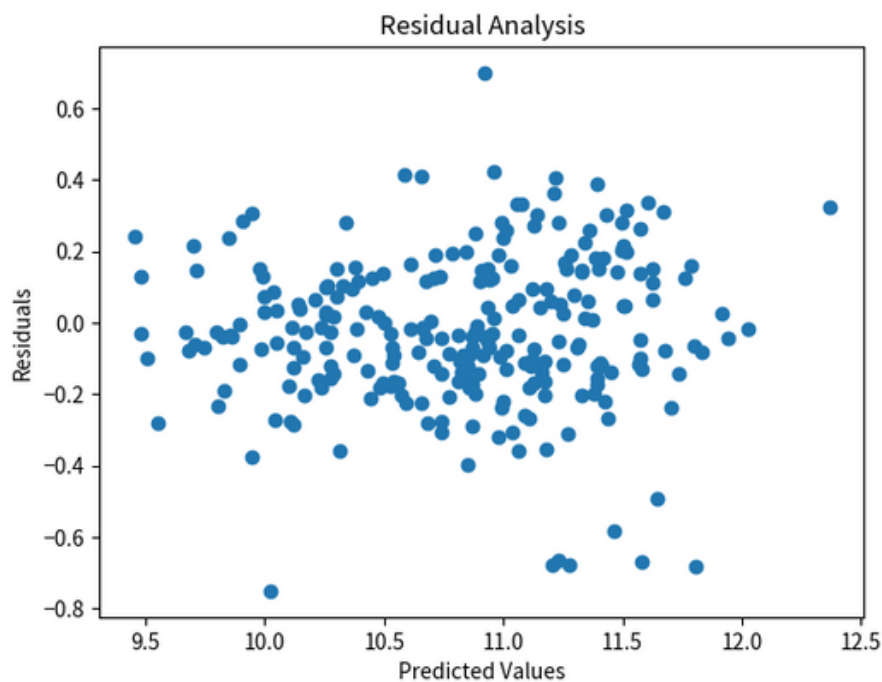


Figure 16.1

The scatter plot shows the residuals (the differences between the predicted and actual prices) plotted against the predicted prices. If the model's predictions are perfect, all points would lie on the horizontal line at  $y = 0$ . The spread of points around this line gives us an idea of the error in the model's predictions.

From the plot, we can see that the residuals are distributed around 0, but there is some structure in the residuals, suggesting that the model may not be capturing all relevant patterns in the data.

## 17.Validation

In the context of machine learning, validation is the process of evaluating a trained model's performance using a separate dataset that the model hasn't seen during training. This is typically done by splitting the original dataset into a training set and a validation set. The model is trained on the training set, and its performance is evaluated on the validation set. This helps ensure that the model can generalize well to new, unseen data. Here 5-fold cross-validation is being performed on the best estimator found by the grid search.

- **Perform 5-Fold Cross-Validation:** The first line performs 5-fold cross-validation using the `cross_val_score` function from `sklearn.model_selection`. Cross-validation is a technique for assessing how well a model will generalize to an independent dataset. It does this by dividing the data into 'folds', training the model on some of these folds, and then testing the model on the remaining fold(s). This process is repeated multiple times, with different folds used for testing each time. The 'cv' parameter specifies the number of folds.
- **Convert the Scores to Positive:** The next line converts the scores to positive. This is because `cross_val_score` returns negative values when using 'neg\_mean\_absolute\_error' as the scoring metric, so the scores are multiplied by -1 to make them positive.
- **Print the Scores:** The last line prints the scores. These are the mean absolute errors for each of the 5 folds of the cross-validation.

```
Output: array([0.18020028, 0.16884203, 0.15049136, 0.17329099,
0.17692554])
```

These scores gives a more robust estimate of the model's performance, as they are based on multiple splits of the training data.

## 18.Uncertainty Estimation

In the context of machine learning, uncertainty estimation is the process of quantifying the uncertainty or confidence of a machine learning model's predictions. This can be important in many applications where not only the prediction itself, but also the reliability of the prediction is important.

There are various ways to estimate uncertainty in machine learning. For some models, like Bayesian models or models that output probabilities, uncertainty can be directly estimated from the model itself. For other models, techniques like bootstrapping or ensemble methods can be used to estimate uncertainty. Here i have used bootstrap method to estimate the uncertainty of the model's mean absolute error and below is the breakdown of what's happening.

- **Perform Bootstrap Resampling:** A loop is set up to run 1000 iterations. In each iteration, a bootstrap sample is created from the training data using the resample function from sklearn.utils.
- **Train the Model and Make Predictions:** The model (a Random Forest Regressor) is trained on the bootstrap sample, and then used to make predictions on the test data.
- **Calculate the MAE:** The MAE is calculated for the model's predictions on the test data.
- **Store the MAE:** The MAE is stored in a list of bootstrap estimates.
- **Calculate the 95% Confidence Interval:** After the loop, a 95% confidence interval for the MAE is calculated using the np.percentile function. This gives the range of values that contains 95% of the bootstrap estimates.
- **Print the Confidence Interval:** The confidence interval is printed.

The purpose of this cell is to estimate the uncertainty of the model's MAE. The bootstrap method is a simple and powerful way to estimate uncertainty by simulating the process of obtaining new data samples.

Output: array([0.16743807, 0.18888478])

The 95% confidence interval for the mean absolute error (MAE) of the model, estimated using bootstrapping, is approximately [0.167, 0.189]. This means that we can be 95% confident that the true MAE of the model is within this range.

## 19. Conclusion

In this, I have performed a comprehensive data analysis and machine learning pipeline to predict laptop prices. I have started by loading and preprocessing the data, followed by exploratory data analysis and data visualization. I have then split the data into a training set and a test set, and scaled the features to prepare them for machine learning.

I have trained a linear regression model on the training data and used it to make predictions on the test data. The mean absolute error (MAE) of the model on the test data was approximately 0.215. I have then performed feature engineering by creating a new feature 'Ram\_per\_Weight', which slightly improved the model's performance (MAE = 0.214).

We also looked at the feature importances of the Random Forest model, which showed that the 'Ram' feature was the most important. We then performed residual analysis by plotting the residuals (the differences between the predicted and actual prices), which showed that the model may not be capturing all relevant patterns in the data.

Finally, I have performed validation by performing 5-fold cross-validation, and estimated the uncertainty in the model's predictions using bootstrapping. The 95% confidence interval for the MAE of the model was approximately [0.167, 0.189].

## 20. Further Analysis

While conducting my research, I felt that there could have been more information in certain areas which would help me in more accurate analysis.

- **More Detailed Specifications:** More detailed technical specifications of the laptops could help improve the accuracy of the model. This could include information like the type of processor, the speed of the processor, the type of graphics card, the resolution of the screen, etc.
- **User Ratings and Reviews:** User ratings and reviews could provide additional information that affects the price of a laptop. For example, laptops with higher user ratings might be able to command higher prices.
- **Sales Data:** Information about the number of units sold could also be useful. This could help capture the demand for different types of laptops, which could affect their price.
- **Date of Release:** The date of release of the laptop could also be a useful feature. Older models might be cheaper than newer models, all else being equal.
- **Geographical Information:** The price of a laptop could also depend on the geographical location where it's sold. If the dataset includes sales from different regions or countries, including this information could improve the model.

Here it is worth noting that, adding more features can potentially improve the accuracy of the model, but it can also make the model more complex and potentially harder to interpret.

## 21. Strategic Recommendations

- **Pricing Strategy for Sellers:** Sellers (such as laptop manufacturers or retailers) can set their prices based on some of the important features identified in the analysis, such as the type of laptop, screen size, RAM, and weight. For example, laptops with larger screens and more RAM could be priced higher.



- **Decision Making for Buyers:** Parties interested in buying a laptop should decide the price they are willing to pay based on the important features pointed out in the analysis that increase or decrease the laptop price. For example, if a larger screen size is a significant factor in increasing the price, buyers who don't need a large screen could opt for a smaller screen to save money.
- **Future Data Collection:** For future data collection, consider gathering more detailed information about the laptops, such as the brand, user ratings, and more detailed technical specifications. This could potentially improve the model's performance and provide more accurate price predictions.