Department Of Applied Mathematics and Statistics

Institute Of Technology Of Cambodia

Team05 : Project Report

Laptop Price Prediction using Statistical Learning

Programming for Data Science
2022-2023

Professor:

Mr. Sophal Chan (Course and TP)

Team Member:

1. Chea Rotha (e20200702)
2. Chou Vandy (e20200664)
3. Chorn Seyhak (e20201099)
4. Ek Vong Panharith (e20200877)

Contents

1 Abstract

Laptop prices in Cambodia have been steadily increasing in recent years, due to a number of factors, including the rising cost of imported goods, the increasing demand for laptops, and the depreciation of the Cambodian currency.

This project will use machine learning to predict laptop prices in Cambodia. The data set will be collected from the Khmer24 website, which is a popular online news portal in Cambodia. The data set will include information on the price, specs, brand, and retailer of each laptop.

The machine learning model will be trained on this data set to learn the factors that affect laptop prices in Cambodia. Once the model is trained, it will be able to predict the price of a laptop based on its specs, brand, and retailer.

The predicted price of a laptop can be used by consumers to make informed decisions about which laptop to purchase. For example, if a consumer is looking for a laptop with a specific set of specs, they can use the predicted price to compare different laptops and find the one that is the best value for their money.

Businesses can also use the predicted price of a laptop to set prices that are competitive and profitable. For example, a business can use the predicted price to determine how much to markup a laptop in order to make a profit.

This project has the potential to benefit both consumers and businesses in Cambodia. By using machine learning to predict laptop prices, consumers can make informed decisions about which laptop to purchase, and businesses can set prices that are competitive and profitable.

The following are some of the challenges that may be encountered in this project:

- The data set may be incomplete or inaccurate.
- The machine learning model may not be able to learn the factors that affect laptop prices in Cambodia.
- The predicted price of a laptop may not be accurate.

Despite these challenges, this project has the potential to be a valuable tool for consumers and businesses in Cambodia. By using machine learning to predict laptop prices, this project can help people make informed decisions about which laptop to purchase and help businesses set prices that are competitive and profitable.

2 Introduction

Laptops are becoming increasingly popular in Cambodia, as more and more people are using them for work, school, and entertainment. However, the prices of laptops can vary widely, depending on the brand, model, and specs. This can make it difficult for consumers to know how much to pay for a laptop.

This project aims to use machine learning to predict the price of laptops in Cambodia. This will help consumers to make informed decisions about which laptop to purchase, and it will also help businesses to set prices that are competitive and profitable.

3 Data collection

Web scraping is a process of extracting data from websites using automated means. It can be done by using a variety of tools and techniques, but the basic idea is to send a request to a website, parse the HTML response and extract the desired data. Some powerful websites can not be scrape by normal means, you need to have API in order to access to data and extract them.

To collect data using web scraping, you will need to:

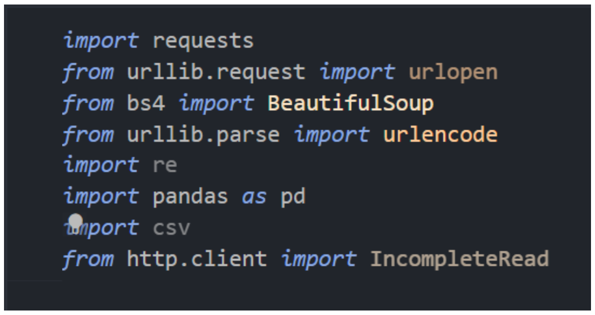
- Choose libraries to use.
- Install the library.
- Write code to extract the data from the website.

- Run the code.

For our case the website that we will scraping data from is called Khmer24 in the laptop section. In the below figure we will show our full python code of web scraping on Khmer24 website.

First of all we will import our crucial libraries that we use them to scrape our website. Those libraries are :

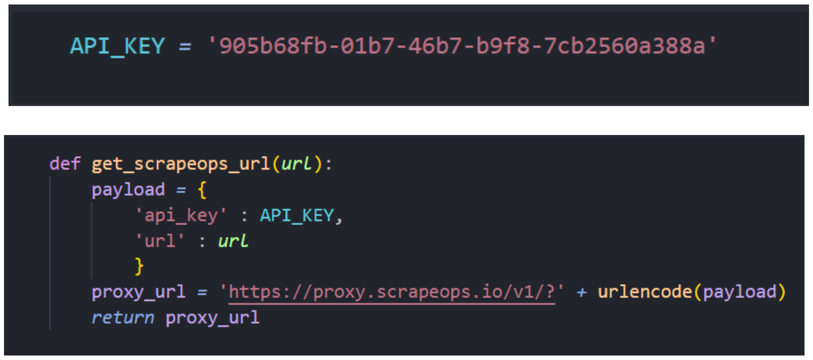
- Beautiful Soup.
- Requests.
- urlopen.
- urlencode.
- ImcompleteRead.
- RegularExpression.



```python
import requests
from urllib.request import urlopen
from bs4 import BeautifulSoup
from urllib.parse import urlencode
import re
import pandas as pd
import csv
from http.client import IncompleteRead
```

Figure 1: Important libraries for our web scraping

Next we will request permission to scrape this Khmer24 website by using an API and the requested url. For instance,this function below will return a new proxy url after we request for permission and we will use it to scrape our overall data.

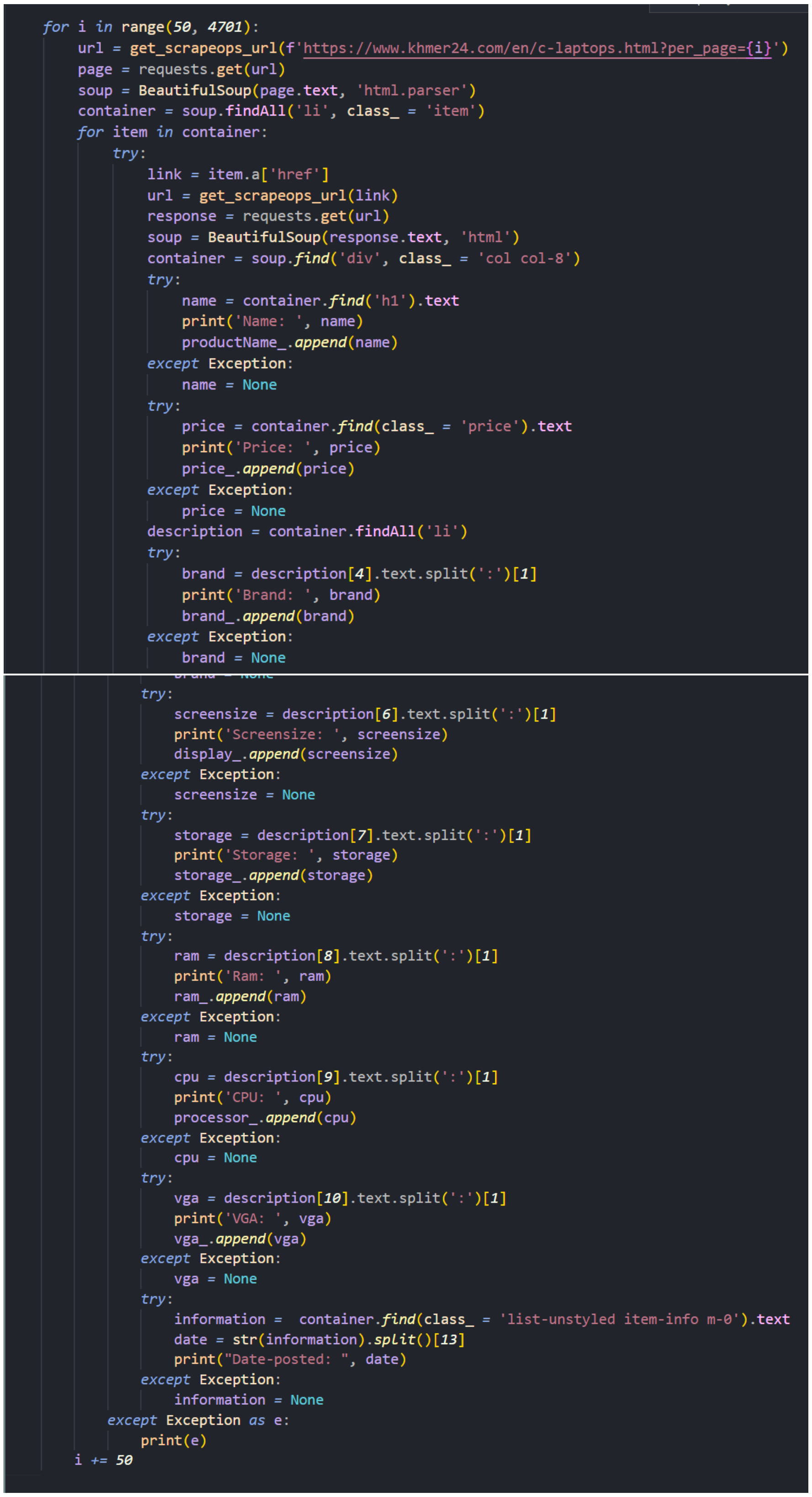


```python
API_KEY = '905b68fb-01b7-46b7-b9f8-7cb2560a388a'
```

```python
def get_scrapeops_url(url):
    payload = {
        'api_key' : API_KEY,
        'url' : url
        }
    proxy_url = 'https://proxy.scrapeops.io/v1/?' + urlencode(payload)
    return proxy_url
```

Figure 2: Requesting permission to access the website by using API key.

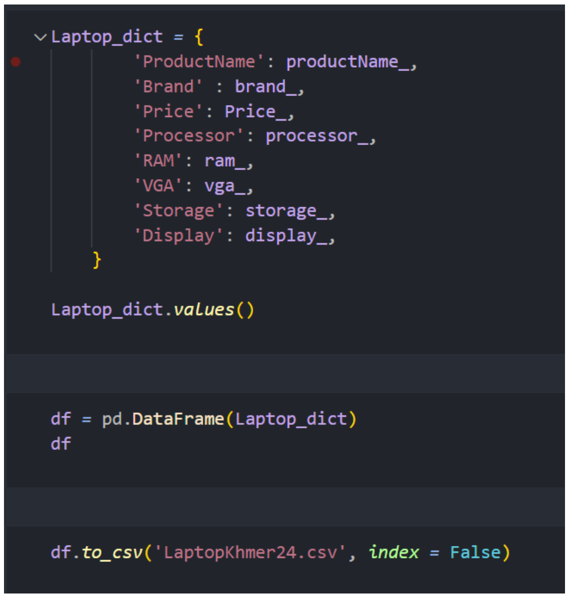
For our next step, after requesting the website we will scrape in all crucial formations of our laptop. There are a total of 4701 pages in our website and we have to loop through each page and scrape the important data from them.



```python
for i in range(50, 4701):
    url = get_scrapeops_url(f'https://www.khmer24.com/en/c-laptops.html?per_page={i}')
    page = requests.get(url)
    soup = BeautifulSoup(page.text, 'html.parser')
    container = soup.findAll('li', class_ = 'item')
    for item in container:
        try:
            link = item.a['href']
            url = get_scrapeops_url(link)
            response = requests.get(url)
            soup = BeautifulSoup(response.text, 'html')
            container = soup.find('div', class_ = 'col col-8')
            try:
                name = container.find('h1').text
                print('Name: ', name)
                productName_.append(name)
            except Exception:
                name = None
            try:
                price = container.find(class_ = 'price').text
                print('Price: ', price)
                price_.append(price)
            except Exception:
                price = None
            description = container.findAll('li')
            try:
                brand = description[4].text.split(':')[1]
                print('Brand: ', brand)
                brand_.append(brand)
            except Exception:
                brand = None
            try:
                screensize = description[6].text.split(':')[1]
                print('Screensize: ', screensize)
                display_.append(screensize)
            except Exception:
                screensize = None
            try:
                storage = description[7].text.split(':')[1]
                print('Storage: ', storage)
                storage_.append(storage)
            except Exception:
                storage = None
            try:
                ram = description[8].text.split(':')[1]
                print('Ram: ', ram)
                ram_.append(ram)
            except Exception:
                ram = None
            try:
                cpu = description[9].text.split(':')[1]
                print('CPU: ', cpu)
                processor_.append(cpu)
            except Exception:
                cpu = None
            try:
                vga = description[10].text.split(':')[1]
                print('VGA: ', vga)
                vga_.append(vga)
            except Exception:
                vga = None
            try:
                information =  container.find(class_ = 'list-unstyled item-info m-0').text
                date = str(information).split()[13]
                print("Date-posted: ", date)
            except Exception:
                information = None
        except Exception as e:
            print(e)
    i += 50
```

Figure 3: Looping through all pages and scrape the important information.

Finally, after we done scraping all the information we will store them inside and dictionary with keys value corresponding each feature then we will save those data inside a csv file.



```
Laptop_dict = {
            'ProductName': productName_,
            'Brand' : brand_,
            'Price': Price_,
            'Processor': processor_,
            'RAM': ram_,
            'VGA': vga_,
            'Storage': storage_,
            'Display': display_,
        }

Laptop_dict.values()
```

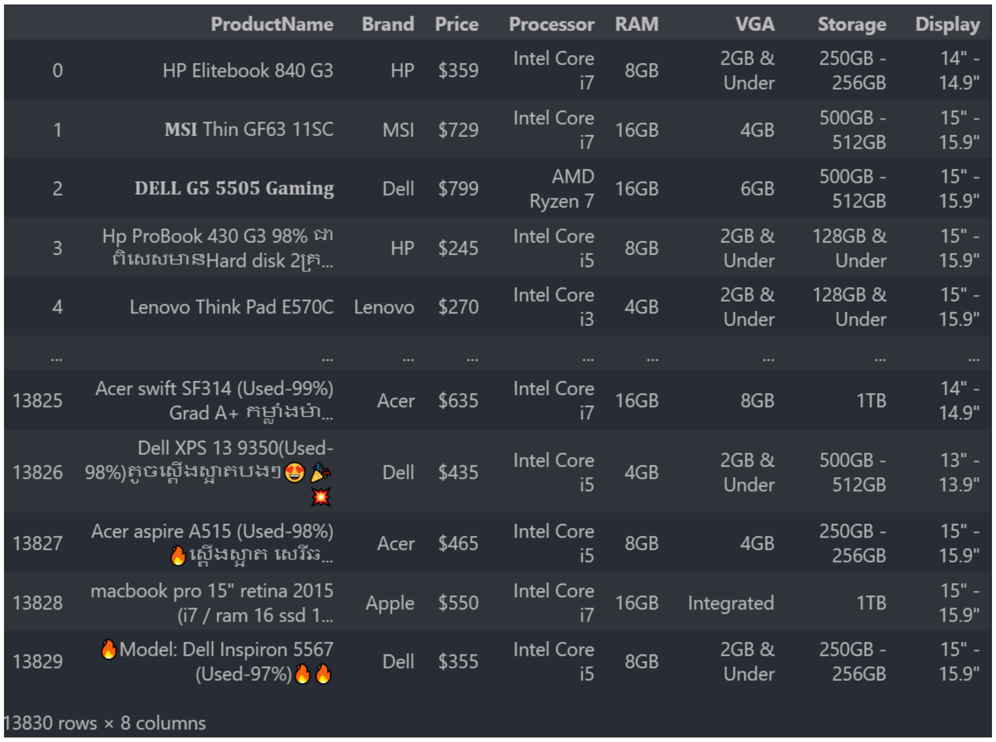
```
df = pd.DataFrame(Laptop_dict)
df
```

```
df.to_csv('LaptopKhmer24.csv', index = False)
```

4 Data description and preparation

4.1 Data description

This data set contains information about laptops, including their name, brand, ram, storage, screen size, processor, graphic and finally price. This data set was collected especially from Khmer24 website and it contains 13830 data points and 8 important features.



	ProductName	Brand	Price	Processor	RAM	VGA	Storage	Display
0	HP Elitebook 840 G3	HP	$359	Intel Core i7	8GB	2GB & Under	250GB - 256GB	14" - 14.9"
1	**MSI** Thin GF63 11SC	MSI	$729	Intel Core i7	16GB	4GB	500GB - 512GB	15" - 15.9"
2	**DELL G5 5505 Gaming**	Dell	$799	AMD Ryzen 7	16GB	6GB	500GB - 512GB	15" - 15.9"
3	Hp ProBook 430 G3 98% ជា ពិសេសមានHard disk 2គ្រ...	HP	$245	Intel Core i5	8GB	2GB & Under	128GB & Under	15" - 15.9"
4	Lenovo Think Pad E570C	Lenovo	$270	Intel Core i3	4GB	2GB & Under	128GB & Under	15" - 15.9"
...	...	...	...	...	...	...	...	...
13825	Acer swift SF314 (Used-99%) Grad A+ កម្លាំងម៉ា...	Acer	$635	Intel Core i7	16GB	8GB	1TB	14" - 14.9"
13826	Dell XPS 13 9350(Used-98%)តូចស្តើងស្អាតបងៗ😍🎉💥	Dell	$435	Intel Core i5	4GB	2GB & Under	500GB - 512GB	13" - 13.9"
13827	Acer aspire A515 (Used-98%) 🔥ស្តើងស្អាត សេរីឆ...	Acer	$465	Intel Core i5	8GB	4GB	250GB - 256GB	15" - 15.9"
13828	macbook pro 15" retina 2015 (i7 / ram 16 ssd 1...	Apple	$550	Intel Core i7	16GB	Integrated	1TB	15" - 15.9"
13829	🔥Model: Dell Inspiron 5567 (Used-97%)🔥🔥	Dell	$355	Intel Core i5	8GB	2GB & Under	250GB - 256GB	15" - 15.9"

13830 rows × 8 columns

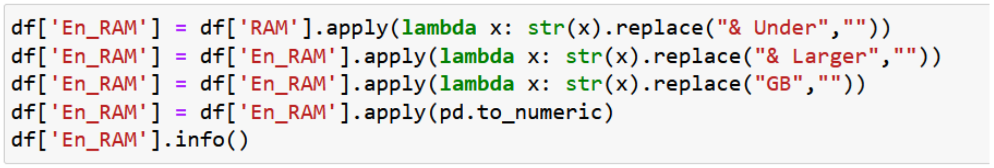
Features :

- Price : It is the most important variable of our overall project
- Laptop's name : The title of the laptop for sale.
- Brand : The model of the laptop.
- Screen size : Laptop's display or size of the screen.
- Storage : The type and capacity of laptop's storage.
- Processor : The name and type of laptop's central unit processing.
- Graphic : The name and type of laptop's graphic unit processing.
- Ram : The amount of RAM installed inside the laptop.

4.2 Data preparation

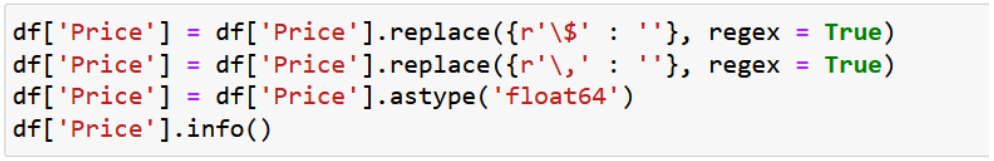
We will Modify Values in Columns of all Our Features

Modify values in column 'RAM'



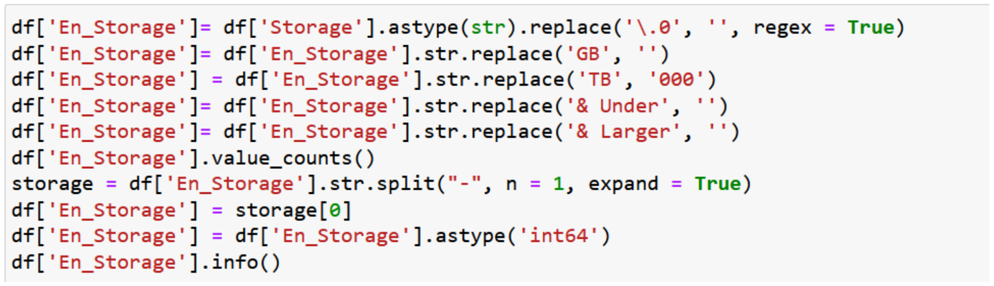
```
df['En_RAM'] = df['RAM'].apply(lambda x: str(x).replace("& Under",""))
df['En_RAM'] = df['En_RAM'].apply(lambda x: str(x).replace("& Larger",""))
df['En_RAM'] = df['En_RAM'].apply(lambda x: str(x).replace("GB",""))
df['En_RAM'] = df['En_RAM'].apply(pd.to_numeric)
df['En_RAM'].info()
```

Modify Values in Column 'Price'



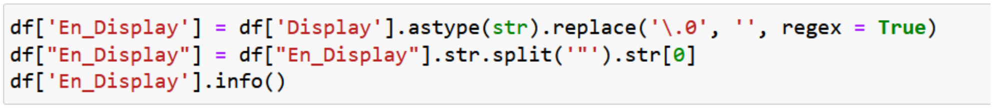
```
df['Price'] = df['Price'].replace({r'\$' : ''}, regex = True)
df['Price'] = df['Price'].replace({r'\,' : ''}, regex = True)
df['Price'] = df['Price'].astype('float64')
df['Price'].info()
```

Modify values in Column 'Storage'



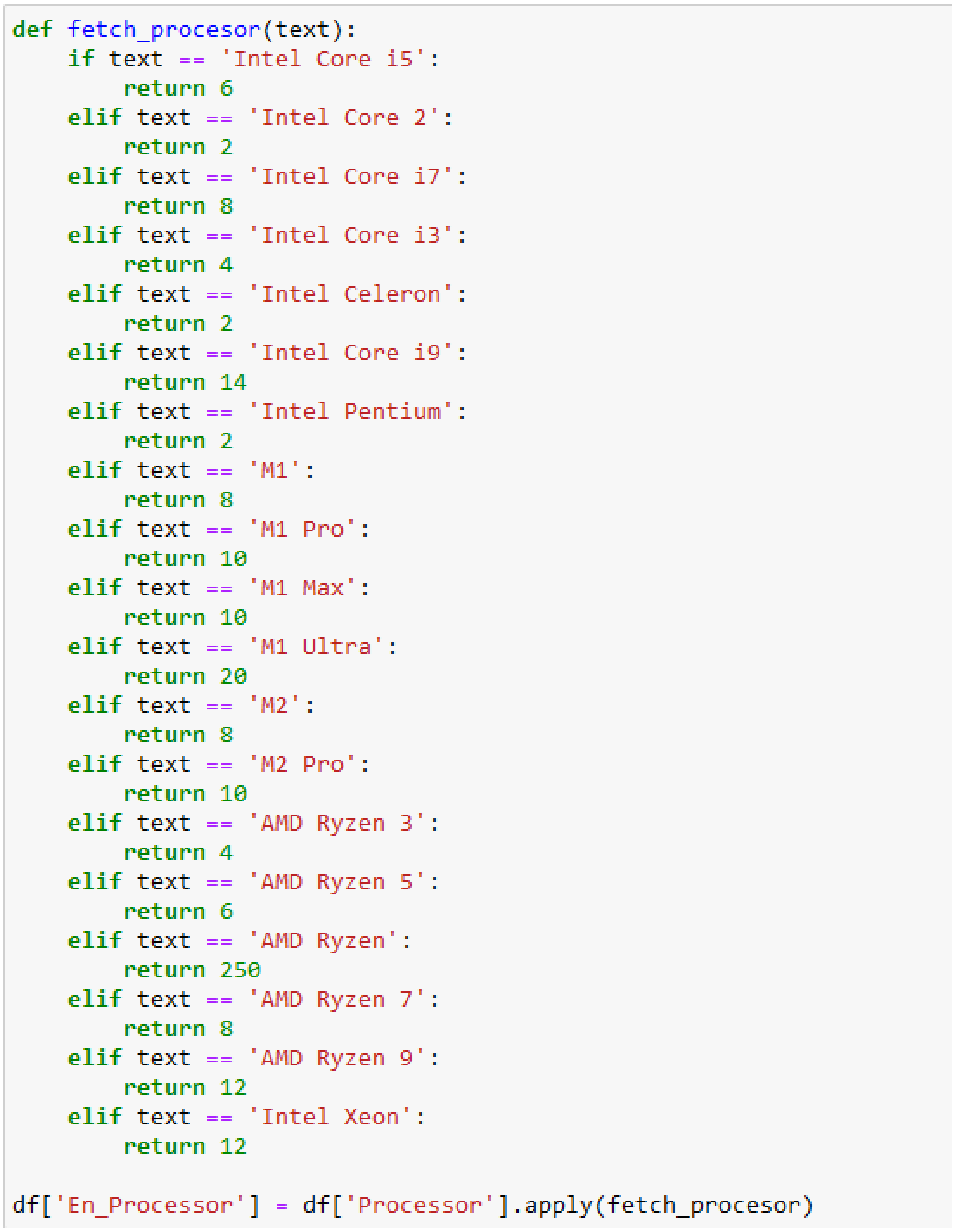
```
df['En_Storage']= df['Storage'].astype(str).replace('\.0', '', regex = True)
df['En_Storage']= df['En_Storage'].str.replace('GB', '')
df['En_Storage'] = df['En_Storage'].str.replace('TB', '000')
df['En_Storage']= df['En_Storage'].str.replace('& Under', '')
df['En_Storage']= df['En_Storage'].str.replace('& Larger', '')
df['En_Storage'].value_counts()
storage = df['En_Storage'].str.split("-", n = 1, expand = True)
df['En_Storage'] = storage[0]
df['En_Storage'] = df['En_Storage'].astype('int64')
df['En_Storage'].info()
```

Modify Values in Column 'Display'



```
df['En_Display'] = df['Display'].astype(str).replace('\.0', '', regex = True)
df["En_Display"] = df["En_Display"].str.split('"').str[0]
df['En_Display'].info()
```

Modify Values in Column in 'Processor'

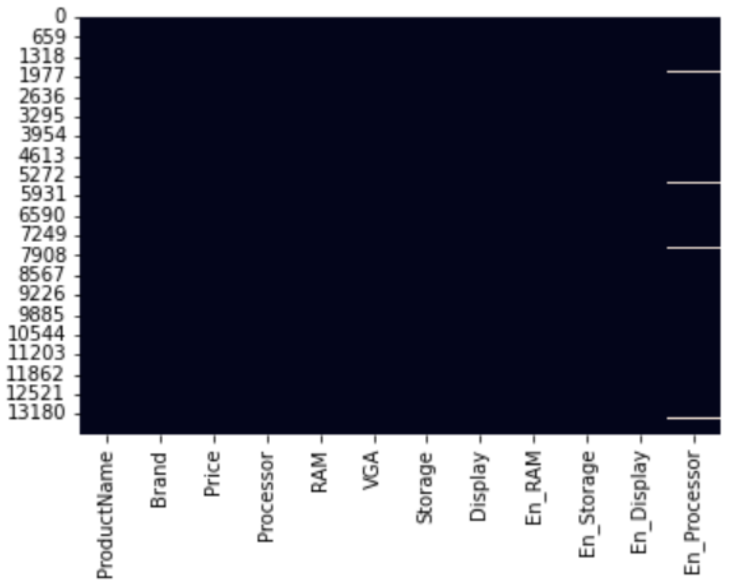


```python
def fetch_procesor(text):
    if text == 'Intel Core i5':
        return 6
    elif text == 'Intel Core 2':
        return 2
    elif text == 'Intel Core i7':
        return 8
    elif text == 'Intel Core i3':
        return 4
    elif text == 'Intel Celeron':
        return 2
    elif text == 'Intel Core i9':
        return 14
    elif text == 'Intel Pentium':
        return 2
    elif text == 'M1':
        return 8
    elif text == 'M1 Pro':
        return 10
    elif text == 'M1 Max':
        return 10
    elif text == 'M1 Ultra':
        return 20
    elif text == 'M2':
        return 8
    elif text == 'M2 Pro':
        return 10
    elif text == 'AMD Ryzen 3':
        return 4
    elif text == 'AMD Ryzen 5':
        return 6
    elif text == 'AMD Ryzen':
        return 250
    elif text == 'AMD Ryzen 7':
        return 8
    elif text == 'AMD Ryzen 9':
        return 12
    elif text == 'Intel Xeon':
        return 12

df['En_Processor'] = df['Processor'].apply(fetch_procesor)
```

- Understanding Data

Check for percentages of missing values inside each feature

```python
sns.heatmap(df.isnull(), cbar=False)
plt.show()
```



Checking Duplicate Values

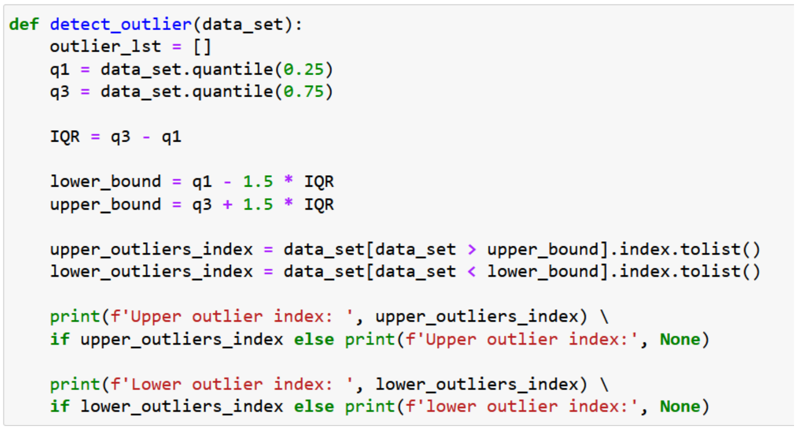
```python
data_with_nan = [feature for feature in df.columns if df[feature].isnull().sum]

for feature in data_with_nan:
    print(feature, np.round(df[feature].isnull().mean(), 4), '% missing values.')
```

```
ProductName 0.0 % missing values.
Brand 0.0 % missing values.
Price 0.0 % missing values.
Processor 0.0 % missing values.
RAM 0.0 % missing values.
VGA 0.0 % missing values.
Storage 0.0 % missing values.
Display 0.0 % missing values.
En_RAM 0.0 % missing values.
En_Storage 0.0 % missing values.
En_Display 0.0 % missing values.
En_Processor 0.0187 % missing values.
```

- This show that there are some missing values in one of our column in our data set. Therefore our data set is cleaned, because we replace them with its mean value.
- There are a tons of duplicated values in our data set. Therefore it can be explained that some specs of our laptops might be the same.

Checking for outliers

```python
def plot_boxplot(data, feature):
    sns.boxplot(x = data[feature], showmeans = True, data = df)
    plt.show()

for feature in df.select_dtypes(include = ['float64', 'int64']):
    plot_boxplot(df, feature)
```



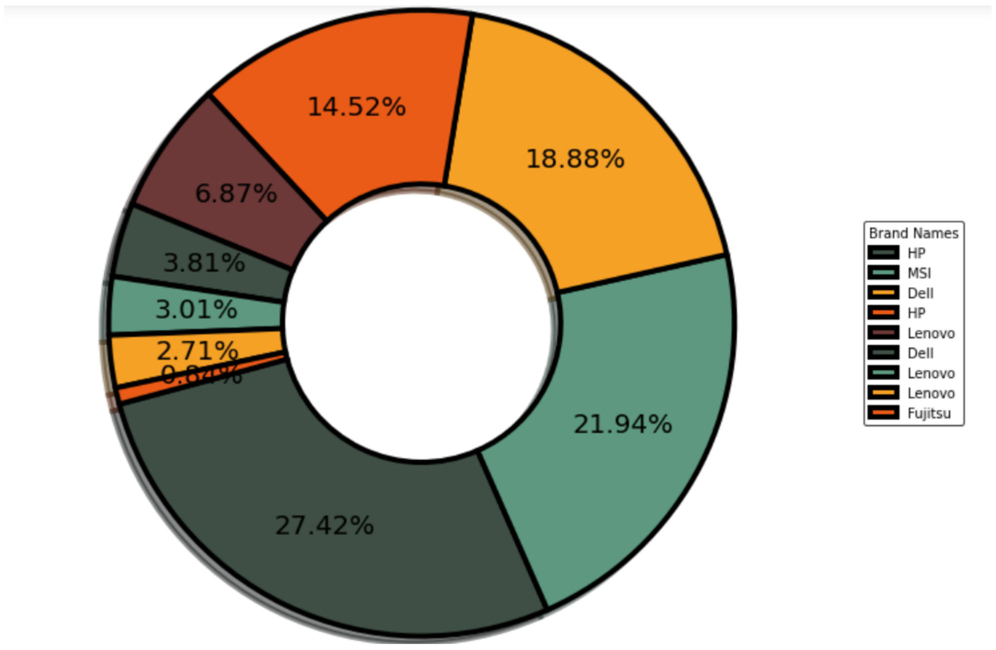
```python
def detect_outlier(data_set):
    outlier_lst = []
    q1 = data_set.quantile(0.25)
    q3 = data_set.quantile(0.75)

    IQR = q3 - q1

    lower_bound = q1 - 1.5 * IQR
    upper_bound = q3 + 1.5 * IQR

    upper_outliers_index = data_set[data_set > upper_bound].index.tolist()
    lower_outliers_index = data_set[data_set < lower_bound].index.tolist()

    print(f'Upper outlier index: ', upper_outliers_index) \
    if upper_outliers_index else print(f'Upper outlier index:', None)

    print(f'Lower outlier index: ', lower_outliers_index) \
    if lower_outliers_index else print(f'lower outlier index:', None)
```

5 Exploratory Data Analysis

We will do Data Exploration on each features in Data set in order to understand more information

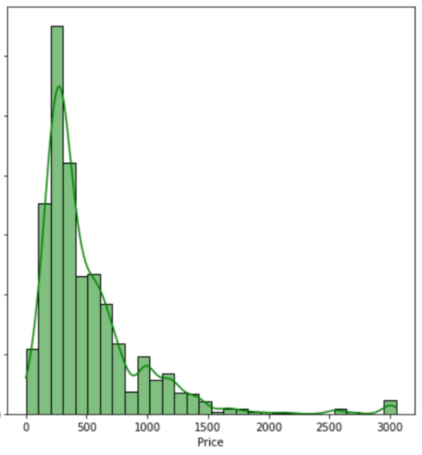
5.1 Performing Uni-variate Analysis

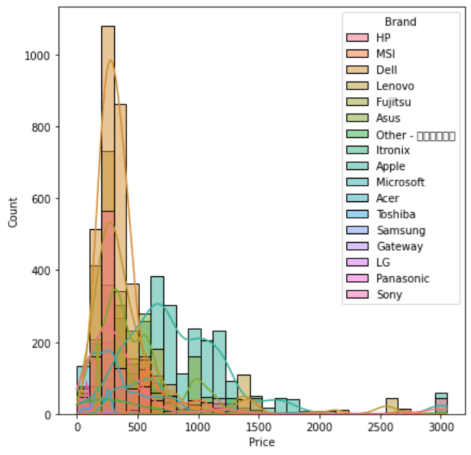
- Pie Chart

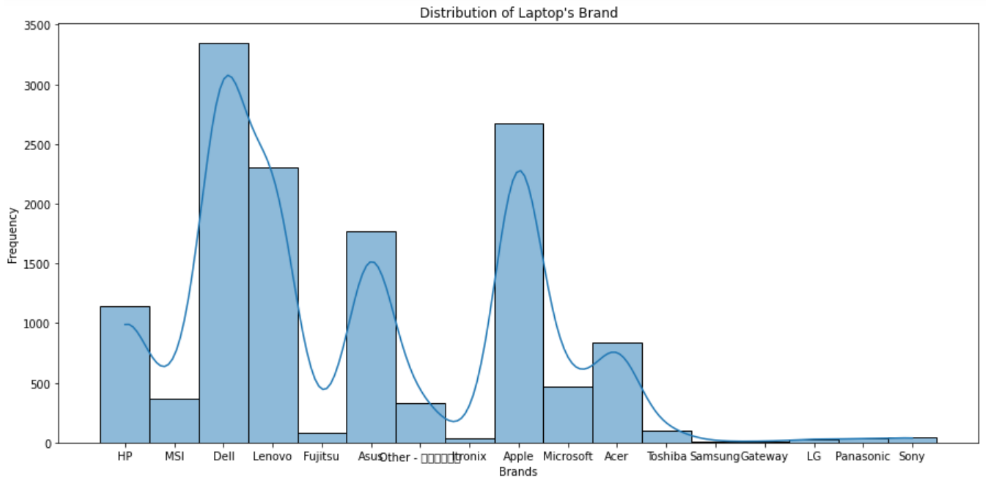


Explanation : Keep in mind that for features with many categories, it's not recommended to use pie chart, because it can be hard to distinguish the small differences of the proportions by just eye balling. So we will use other visualization techniques such as histograms, box plots, or density plots for continuous variables and bar charts for categorical variables.

- Distribution Plot







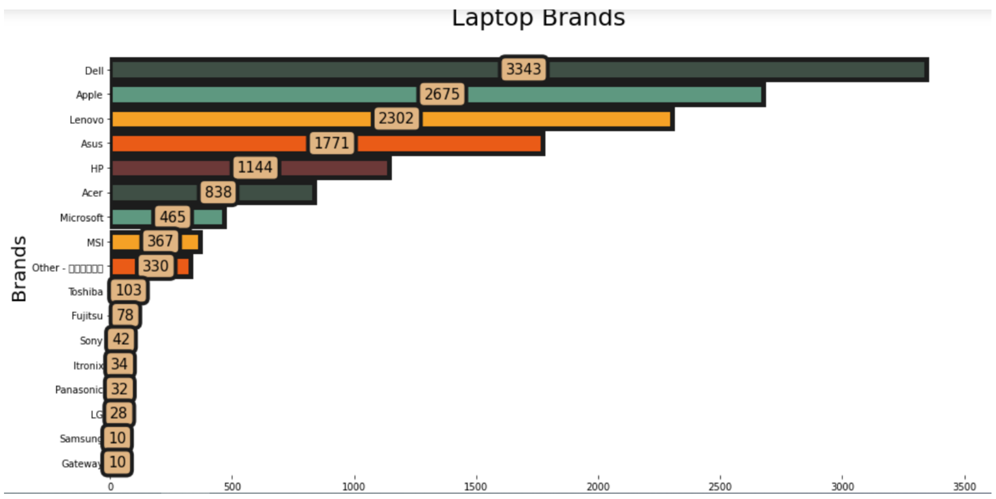
Explanation:

- From the above distribution plot of Price data is likely to be positively skewed. This is because there are a few laptops are quite expensive from the rest of laptops, which will cause the tail of the histogram to extend to the right.

- A positive skewed histogram can also be caused by a non-normal distribution. A non-normal distribution is a distribution that is not symmetrical .

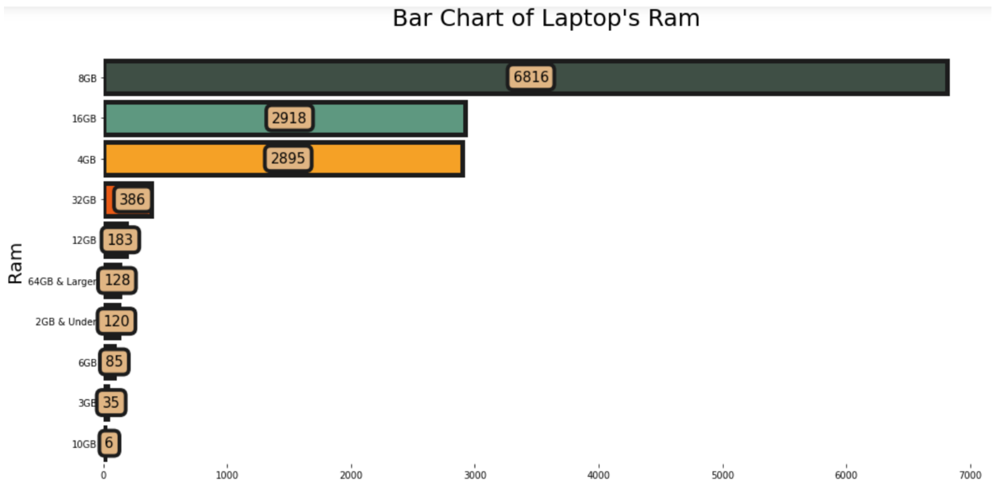
5.2 Performing Bi-variate Analysis

- Bar Chart of Laptops Brand



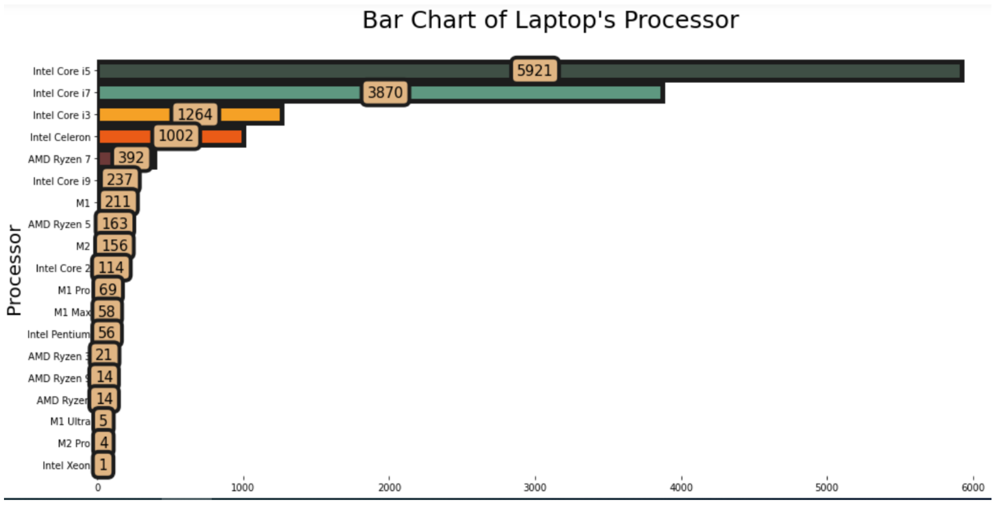
Explanation: When we looking at the trend of our data, we can see that Dell is the dominance brand following by Apple, Lenovo and so on. Brand Gateway is a least brand sold on the Khmer24.

- Bar Chart of Laptops Ram



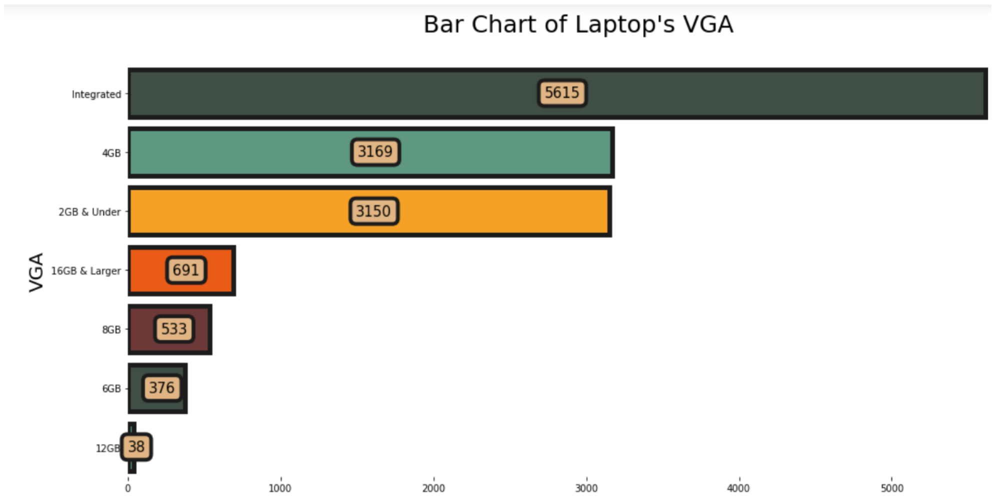
Explanation : When we looking at the trend of our data, we can see that 8GB is the dominance RAM following by 16GB, 4GB and so on. RAM 10GB is a least RAM sold on the Khmer24.

- Bar Chart of Laptops Processor



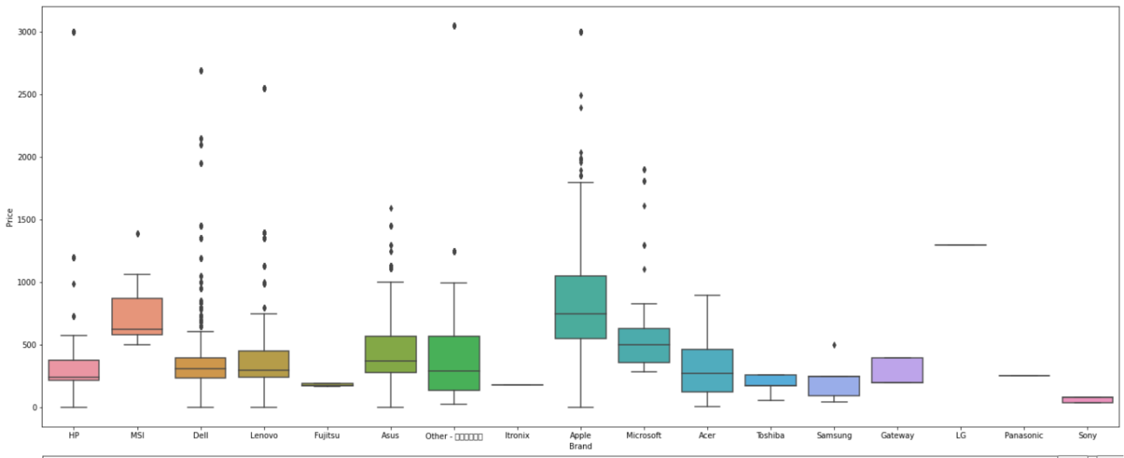
Explanation : When we looking at the trend of our data, we can see that intel core i5 is the dominance Processor following by intel core i7 , intel core i3 and so on. Intel Xeon is a least Processor sold on the Khmer24.

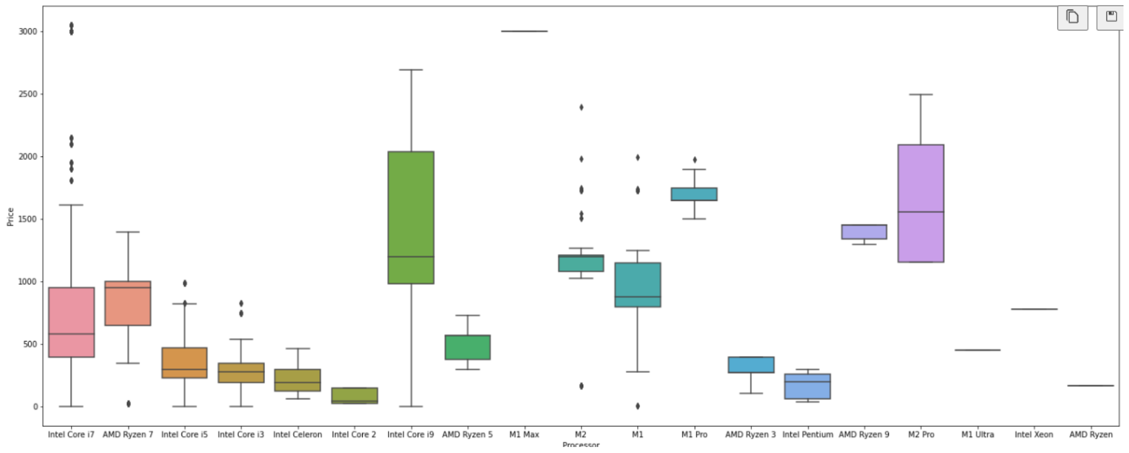
- Bar Chart of Laptops VGA

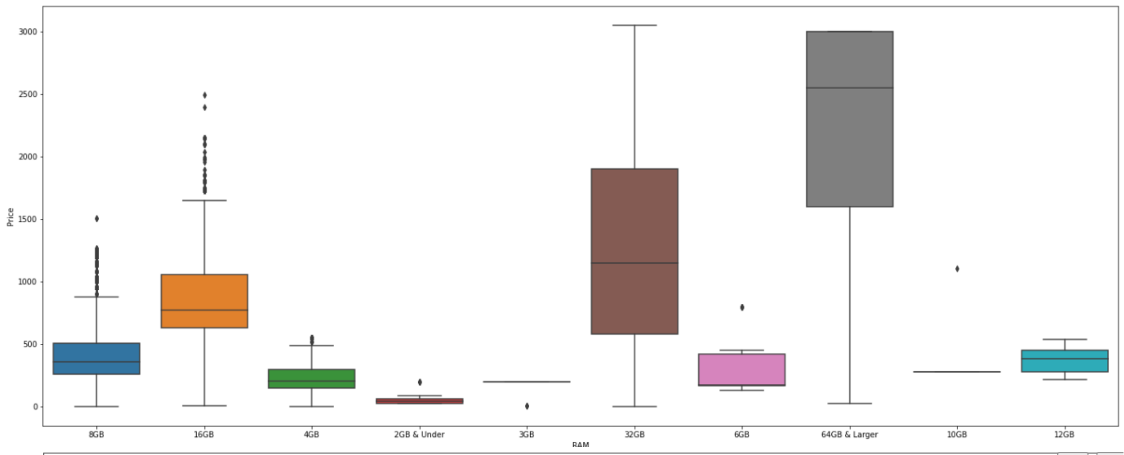


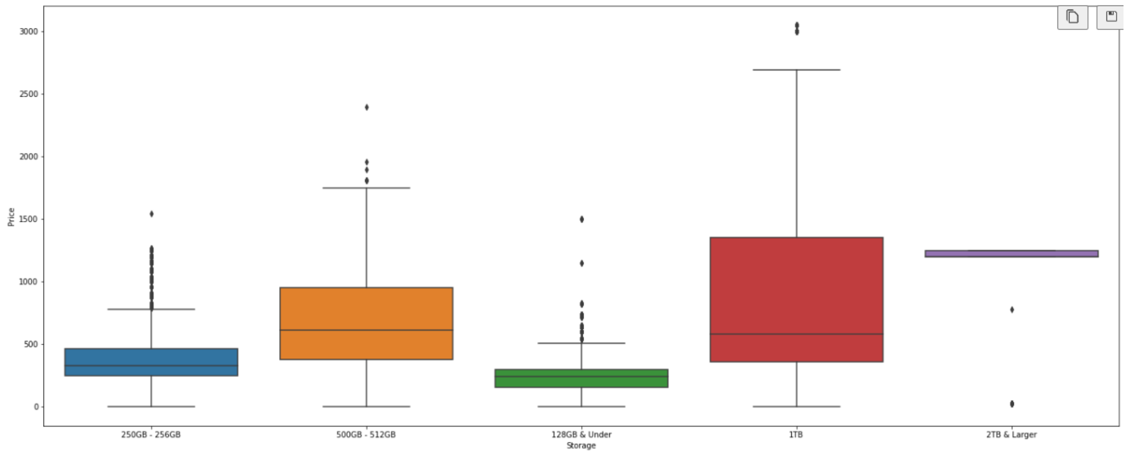
Explanation : When we looking at the trend of our data, we can see that Integrated is the dominance VGA following by 4GB , 2GB and Under and so on. 12GB is a least VGA sold on the Khmer24.

- Box plot of Model, Ram , Storage and Processor


Price
0
500
1000
1500
2000
2500
3000
HP
MSI
Dell
Lenovo
Fujitsu
Asus
Other
Itronix
Apple
Microsoft
Acer
Toshiba
Samsung
Gateway
LG
Panasonic
Sony
Brand


Price
0
500
1000
1500
2000
2500
3000
Intel Core i7
AMD Ryzen 7
Intel Core i5
Intel Core i3
Intel Celeron
Intel Core 2
Intel Core i9
AMD Ryzen 5
M1 Max
M2
M1
M1 Pro
AMD Ryzen 3
Intel Pentium
AMD Ryzen 9
M2 Pro
M1 Ultra
Intel Xeon
AMD Ryzen
Processor


Price
0
500
1000
1500
2000
2500
3000
8GB
16GB
4GB
2GB & Under
3GB
32GB
6GB
64GB & Larger
10GB
12GB
RAM


Price
0
500
1000
1500
2000
2500
3000
250GB - 256GB
500GB - 512GB
128GB & Under
1TB
2TB & Larger
Storage

Explanation :

1. As expected, Apple brand is the most expensive laptop in the market.
2. For Ram and Storage the more powerful they are, the more expensive they become.
3. For Processor Intel core i9 and M2 pro is the most expensive processor due to their powerful performance.

6 Feature Engineering

6.1 Encoding categorical variable

```
from sklearn.preprocessing import LabelEncoder

label_encoder_gpu = LabelEncoder()

df['En_VGA'] = label_encoder_gpu.fit_transform(df['VGA'])
```

Setting dependent and response variables for feature engineering

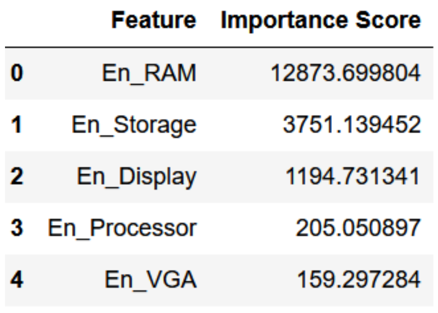
```
new_df = df[['Price', 'En_RAM', 'En_Storage', 'En_Display', 'En_Processor', 'En_VGA']].copy()
X = new_df.drop(['Price'], axis = 1)
y = new_df['Price']
```

6.2 Feature Importance

Checking Importance Score with each Features

```
from sklearn.feature_selection import f_regression

def feature_importance(X, y):
    features = list(X.columns)
    importance_scores = f_regression(X[features], y)
    importance_df = pd.DataFrame({'Feature': features, 'Importance Score': importance_scores[0]})
    importance_df = importance_df.sort_values('Importance Score', ascending = False)

    return importance_df
```

Result



	Feature	Importance Score
0	En_RAM	12873.699804
1	En_Storage	3751.139452
2	En_Display	1194.731341
3	En_Processor	205.050897
4	En_VGA	159.297284

6.3 Testing for linearity

6.3.1 Assumptions of Linear Regression

Here are the assumptions that we must not violated in order to build a perfect linear regression model, those are:

- Linearity: The relationship between the independent and dependent variables is linear. This means that the change in the dependent variable is proportional to the change in the independent variable.
- Homoscedasticity: The variance of the residuals is constant across all values of the independent variable. This means that the errors are evenly spread out around the regression line.
- Normality: The residuals are normally distributed. This means that the errors are bell-shaped and symmetrical around the mean.
- Independence: The residuals are independent of each other. This means that the errors are not correlated with each other.
- Multicollinearity: There is no multicollinearity among the independent variables. This means that the independent variables are not highly correlated with each other.

6.3.2 Correlation Matrix and Pairplot

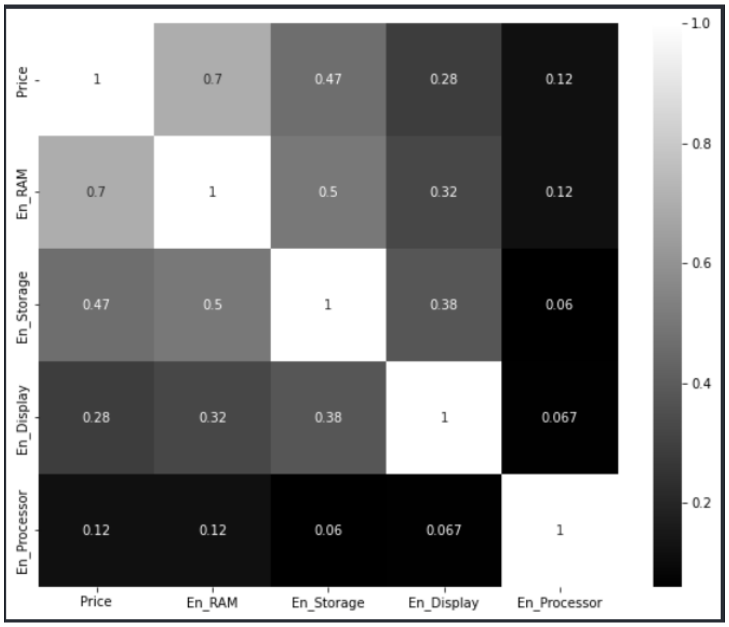


Figure 4: Correlation matrix : Show how much each variables in our data set correlated or what relationship that have with each other.

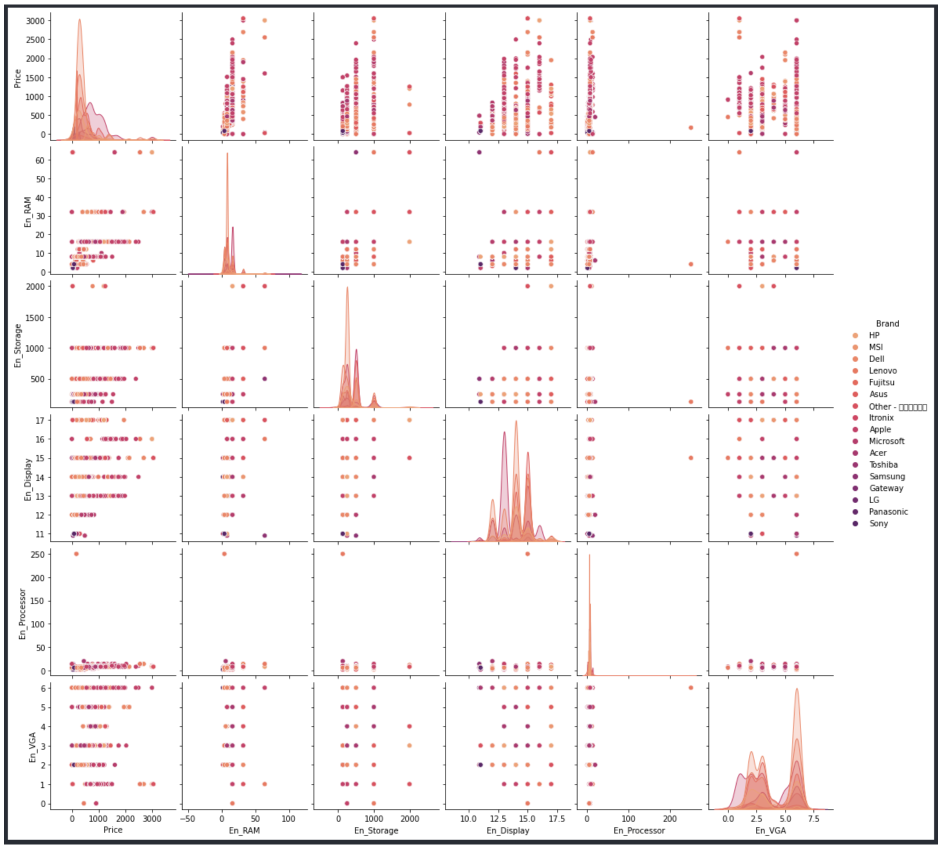


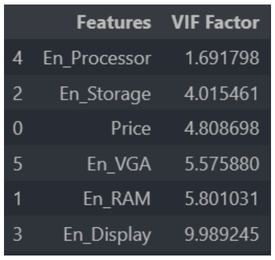
Figure 5: Pairplot : Show scatter plot between all variables and the distribution plot of each variable

6.3.3 Variance Inflation Factor (VIF)

Variance inflation factor (VIF) is a measure of the amount of multicollinearity in a multiple regression model Multicollinearity occurs when two or more independent variables in a regression model are highly correlated. This can cause the standard errors of the regression coefficients to increase, making the coefficients less reliable. The reason for this is that when two independent variables are highly correlated, they are essentially measuring the same thing. This means that the model is not able to distinguish between the two variables, and the standard errors of the coefficients will increase.

Here are some guidelines for interpreting VIF values:

- VIF is lesser than 1.5 : No multicollinearity
- VIF is equal to 1.5 to 5: Moderate multicollinearity
- VIF is greater than 5: Severe multicollinearity



	Features	VIF Factor
4	En_Processor	1.691798
2	En_Storage	4.015461
0	Price	4.808698
5	En_VGA	5.575880
1	En_RAM	5.801031
3	En_Display	9.989245

Figure 6: VIF table score of each feature

Explanation:

- As you can see from the above data frame we can see that all of our features have moderate and severe multicollinearity since theirs VIF factor score is higher than 1.5 and 5.
- Hence, to fix this multicollinearity problem we have choose dimensionality reduction method by using PCA.

6.3.4 QQ-plot : Testing for normality

A QQ plot, short for quantile-quantile plot, is a graphical method for comparing a data distribution to a theoretical distribution. In a QQ plot, the quantiles of the two distributions are plotted against each other. If the data follows the theoretical distribution closely, the points on the plot will form a straight line. Deviations from the straight line indicate differences between the two distributions. A QQ plot can be used to visually assess whether the data follows a particular distribution, such as the normal distribution.

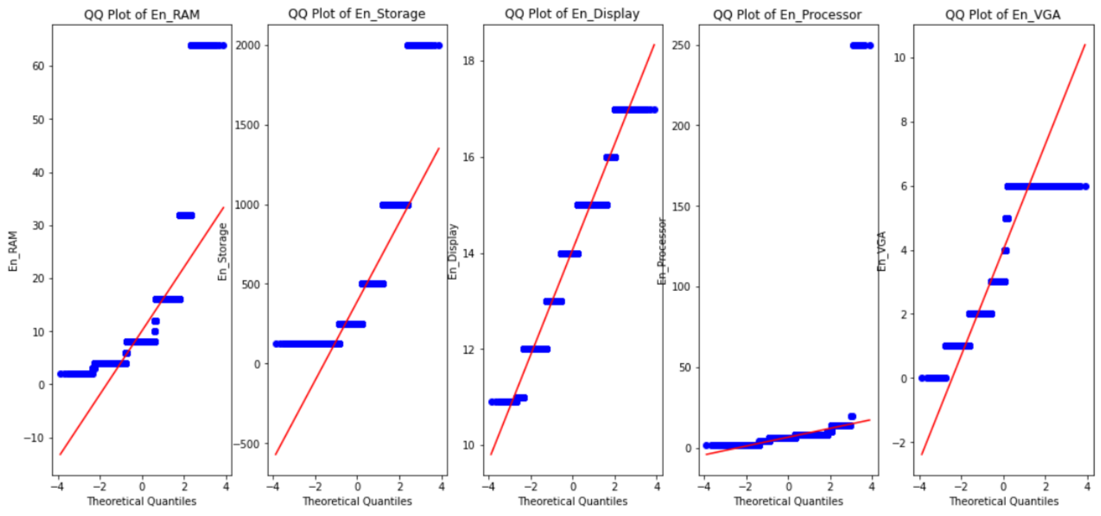


Figure 7: QQ-plot of our data set.

Explanation: Based on the QQ-plots above indicate that our features are violated with normality assumption.

6.3.5 Homoscedasticity

A homoscedasticity plot is a scatter plot of the residuals against the predicted values in a linear regression model. If the residuals are evenly spread out around the horizontal line, then homoscedasticity is met. However, if the residuals are not evenly spread out, then homoscedasticity is not met.

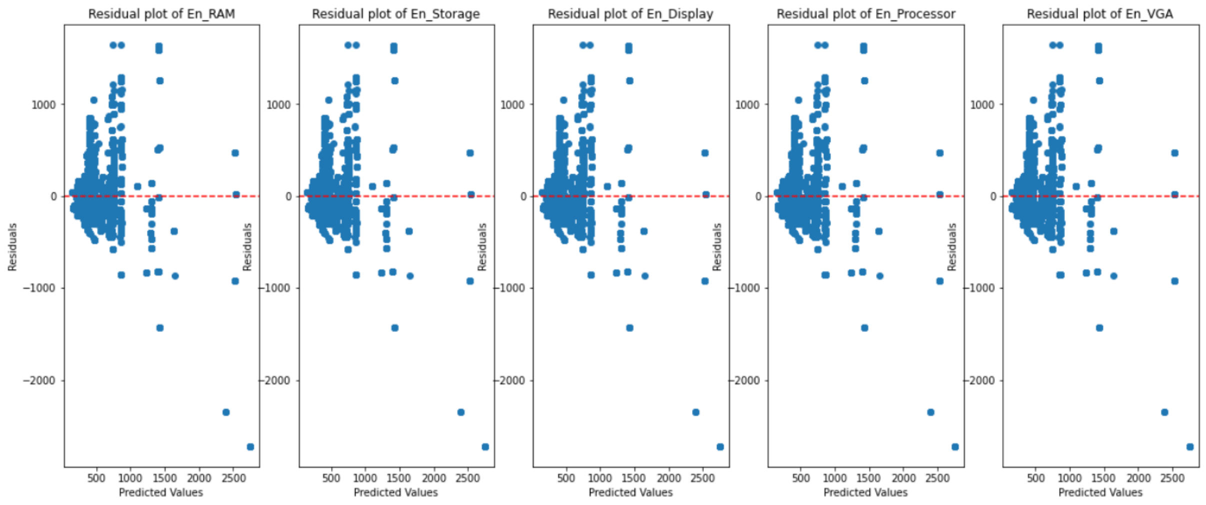


Figure 8: Residual plot of each feature.

Base on the graphs above show that homoscedasticity is violated when the variance of the residuals is not constant across all values of the independent variable. This can happen for a number of reasons, including:

- Non-linear relationships: If the relationship between the independent and dependent variables is non-linear, then the variance of the residuals may not be constant.
- Outliers: Outliers can also cause the variance of the residuals to be non-constant.
- Heterogeneity of variance: Heterogeneity of variance refers to the situation where the variance of the dependent variable is different across different groups of observations. This can also cause the variance of the residuals to be non-constant.

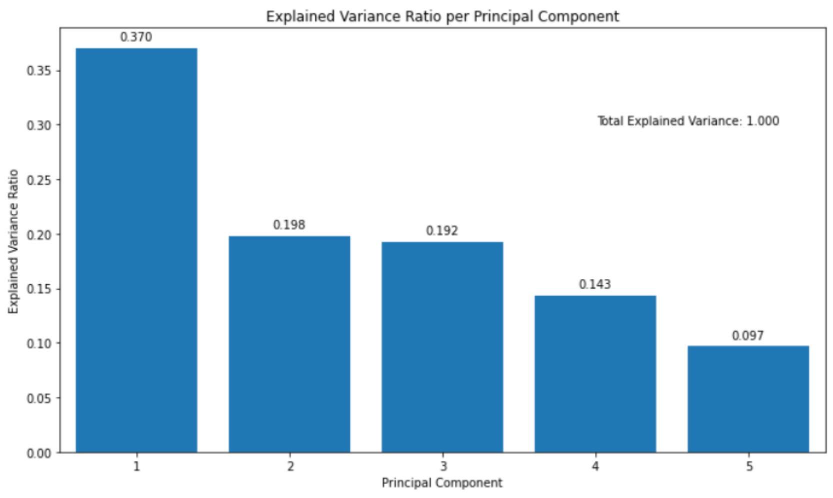
6.4 Dimensionality reduction

The dimensionality reduction is a technique used to reduce the number of features or columns in a data set while preserving as much information as possible. This can be useful for a variety of purposes, such as simplifying data visualization, improving the performance of machine learning algorithms, and making data more interpretable.

There are a number of different dimensionality reduction techniques available, each with its own strengths and weaknesses but for this project we will specifically use Principle Component Analysis also known as PCA to reduce the dimension in our data set before use them to train and build our model.

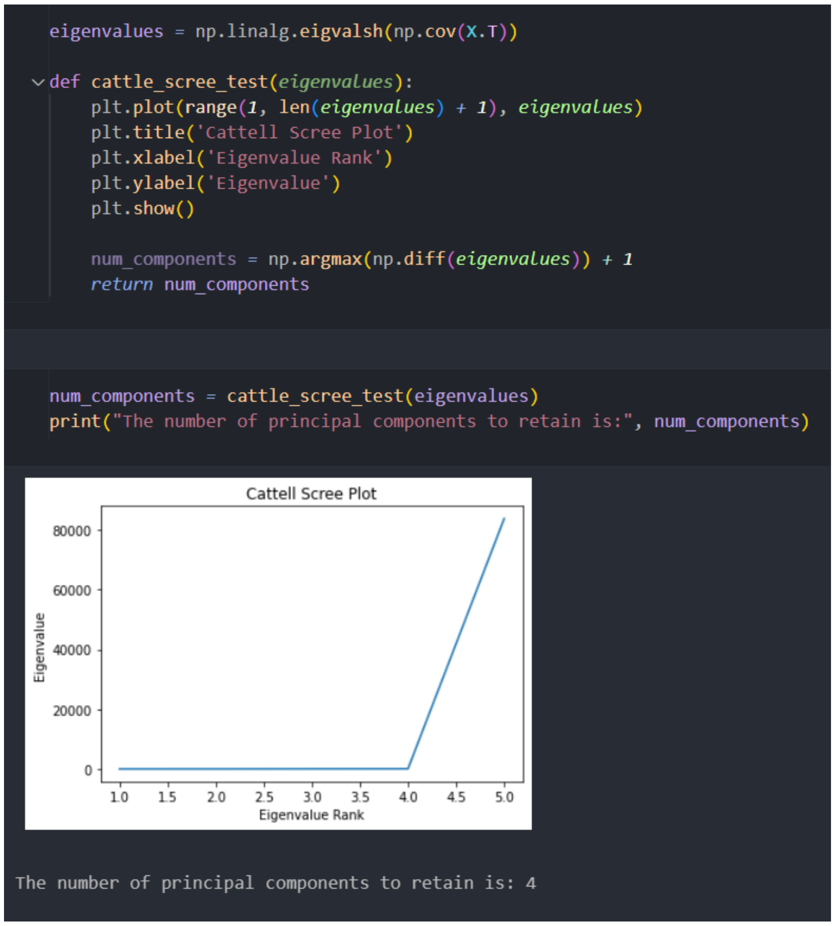
6.4.1 Bar Graph of Variance Ratio

The variance ratio is a measure of how much variation there is between the different features of the data set. A bar graph of variance ratio can be used to visualize the distribution of the variance ratio and to identify any features that have a high variance ratio. In our case, we will use bar graph of variance ratio to measure the the variance(Eigenvalue) of each component of our data set after performing dimensionality reduction on them. And base on the variance of each component we will be selecting a few components by using Cattell scree test.



6.4.2 Cattell Scree test

The Cattell scree test is a graphical method for identifying the number of principal components to retain. The scree test plots the eigenvalues of the covariance matrix against their rank, and the number of principal components to retain is determined by the point at which the scree plot starts to level off.



```
eigenvalues = np.linalg.eigvalsh(np.cov(X.T))

def cattle_scree_test(eigenvalues):
    plt.plot(range(1, len(eigenvalues) + 1), eigenvalues)
    plt.title('Cattell Scree Plot')
    plt.xlabel('Eigenvalue Rank')
    plt.ylabel('Eigenvalue')
    plt.show()

    num_components = np.argmax(np.diff(eigenvalues)) + 1
    return num_components
```

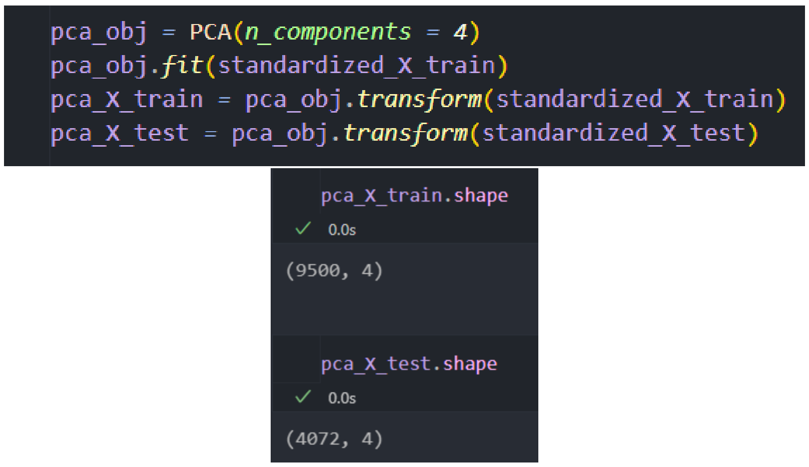
```
num_components = cattle_scree_test(eigenvalues)
print("The number of principal components to retain is:", num_components)
```

```
The number of principal components to retain is: 4
```

Figure 9: Cattle Scree Test of choosing the number of optimal components of PCA

Explanation : From the above plot, we will select only 4 components of our data set after performing dimensionality reduction (PCA) with a total variance of 0.903.

6.4.3 Applied Principle Component Analysis



```
pca_obj = PCA(n_components = 4)
pca_obj.fit(standardized_X_train)
pca_X_train = pca_obj.transform(standardized_X_train)
pca_X_test = pca_obj.transform(standardized_X_test)
```

```
pca_X_train.shape
✓ 0.0s
(9500, 4)
```

```
pca_X_test.shape
✓ 0.0s
(4072, 4)
```

Figure 10: Applied Principle Analysis on both our training and testing data set by choosing 4 components which is recommended by the above Cattell Scree Test.

7 Model Building and Evaluation

7.1 Model selection

Model selection is the process of choosing machine learning models that best fit the data set and that achieves the best performance. There are a variety of factors to consider when selecting a model, including the type of data, the desired accuracy and the computational resources available.

The model selection process typically involves the following steps:

1. Identify the type of data: The first step is to identify the type of data that is available. The type of data will determine the type of model that can be used.
2. Define the desired accuracy: The next step is to define the desired accuracy. The desired accuracy will determine the complexity of the model that needs to be used.
3. Consider the computational resources: The computational resources available will also affect the choice of model. More complex models require more computational resources, so it is important to choose a model that is feasible to train and deploy.
4. Evaluate the different models: Once the different models have been identified, they need to be evaluated. The evaluation process typically involves using a held-out data set to test the accuracy of the models.
5. Choose the best model: The best model is the one that achieves the best accuracy on the held-out data set.

Here are some important machine learning algorithms that we will be using and testing in our project:

- Multiple Linear Regression
- Lasso and Ridge Regression
- Polynomial Regression
- Random Forest

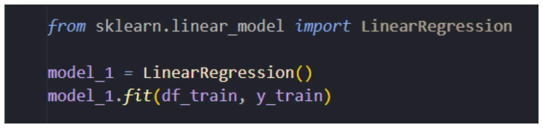
- Support Vector Regressor
- Neural Networks

7.2 Model Training and Evaluation

7.2.1 Multiple Linear Regression

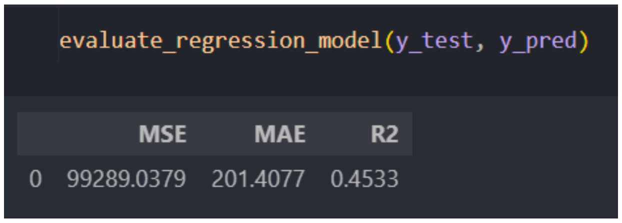
Multiple linear regression is a statistical method that uses multiple independent variables to predict a single dependent variable. The independent variables are typically continuous variables, but they can also be categorical variables. The dependent variable is typically a continuous variable.

In our case, we will first implement multiple linear regression.



```
from sklearn.linear_model import LinearRegression

model_1 = LinearRegression()
model_1.fit(df_train, y_train)
```

Figure 11: Training Multiple Linear Regression



```
evaluate_regression_model(y_test, y_pred)

           MSE       MAE      R2
0   99289.0379  201.4077  0.4533
```

Figure 12: Model1 - Multiple Linear Regression with R-square score of 0.4533

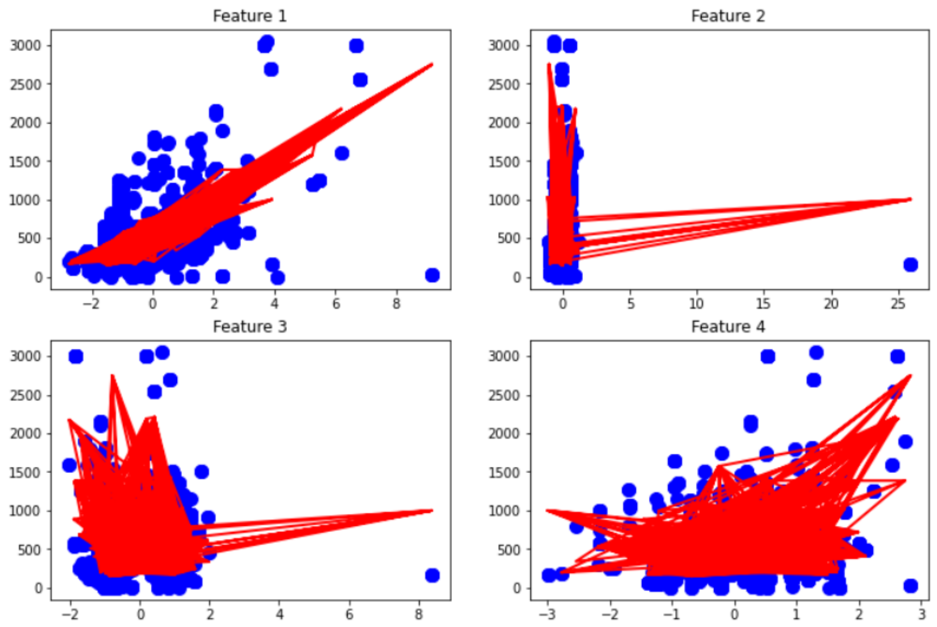


Figure 13: Model1 - Scatter plot of actual data value vs predicted value of each features corresponding to price

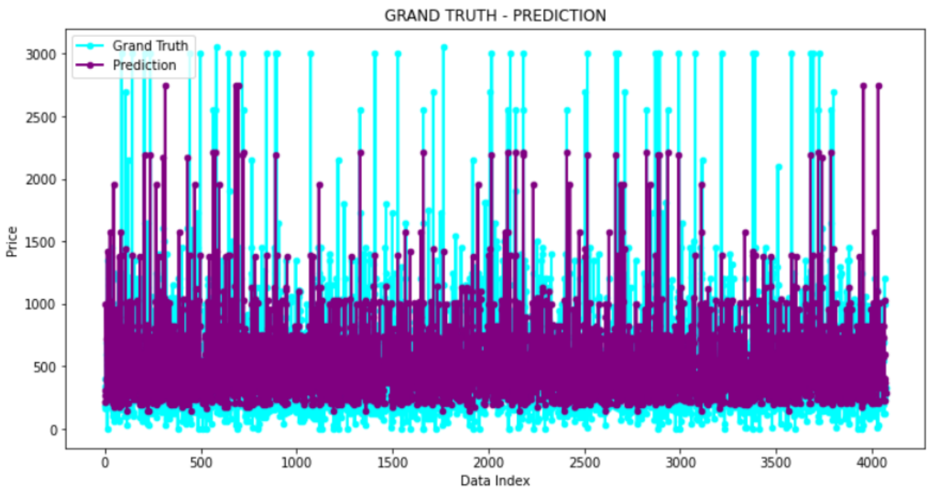


Figure 14: Model1 - Grand truth vs Prediction plot

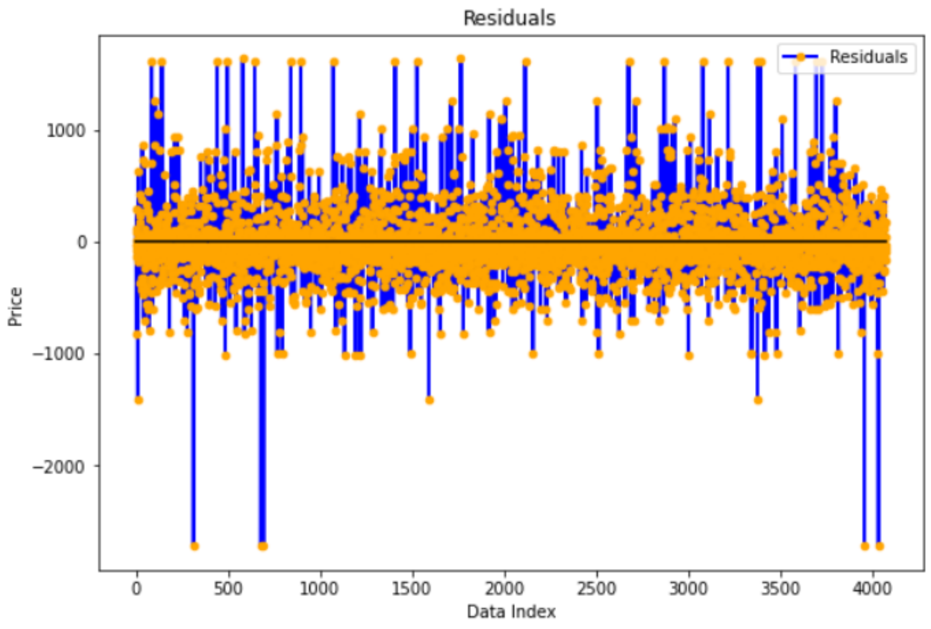


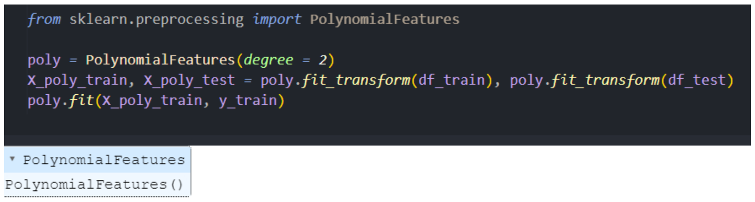
Figure 15: Model1 - Residual plot

Interpret : From the above inductions show that the linear assumptions of our data set are all violated. So it is not a surprise that Multiple Linear Regression model does not work well with our data with R-square score of only 0.4533. Thus, we will not choose this model. We will try to implement more complex model.

7.2.2 Polynomial Regression

Polynomial regression is a type of regression analysis in which the dependent variable is modeled as a polynomial function of the independent variable. The independent variable can be either continuous or categorical. The degree of the polynomial can be any positive integer.

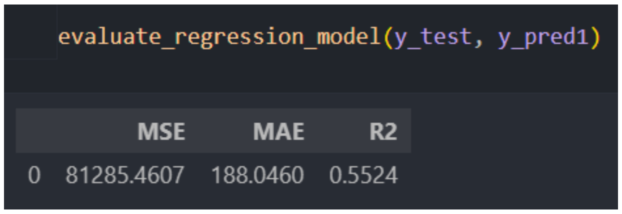
In our case, for our second model we will try to implementing Polynomial Regression.



```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 2)
X_poly_train, X_poly_test = poly.fit_transform(df_train), poly.fit_transform(df_test)
poly.fit(X_poly_train, y_train)
```

```
▾ PolynomialFeatures
PolynomialFeatures()
```

Figure 16: Training Polynomial Regression



```
evaluate_regression_model(y_test, y_pred1)
```

	MSE	MAE	R2
0	81285.4607	188.0460	0.5524

Figure 17: Model2 - Polynomial Regression with R-square score of 0.5524

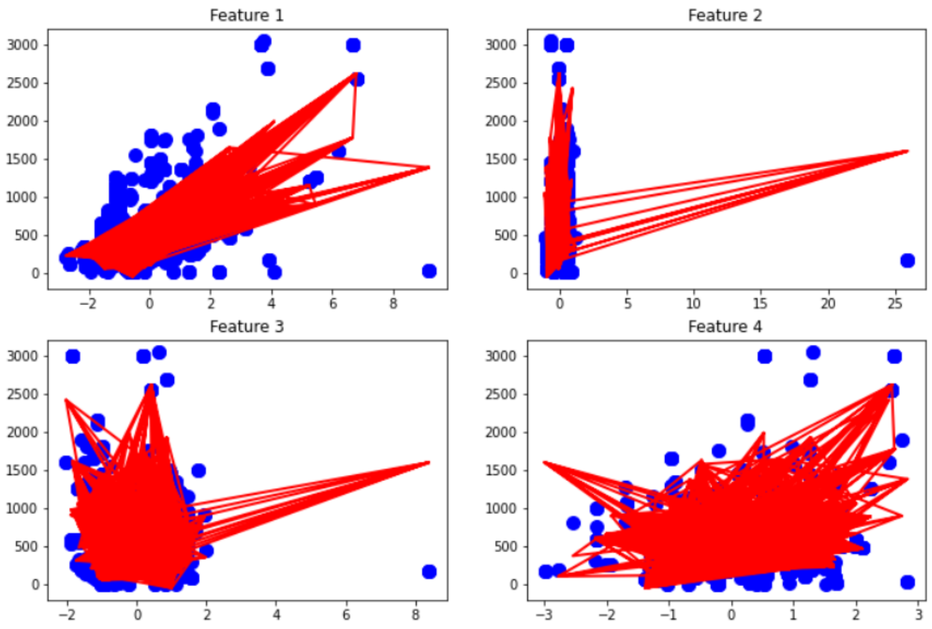


Figure 18: Model2 - Scatter plot of actual data value vs predicted value of each features corresponding to price

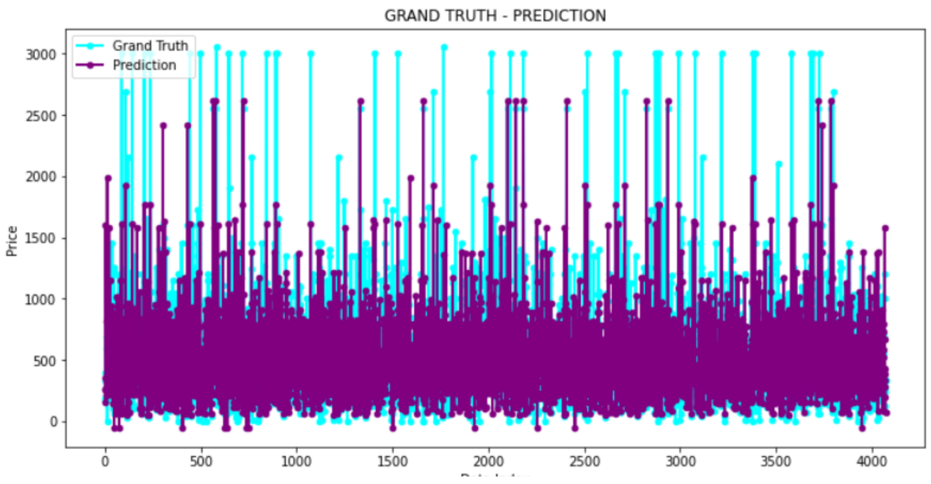


Figure 19: Model2 - Grand truth vs Prediction plot

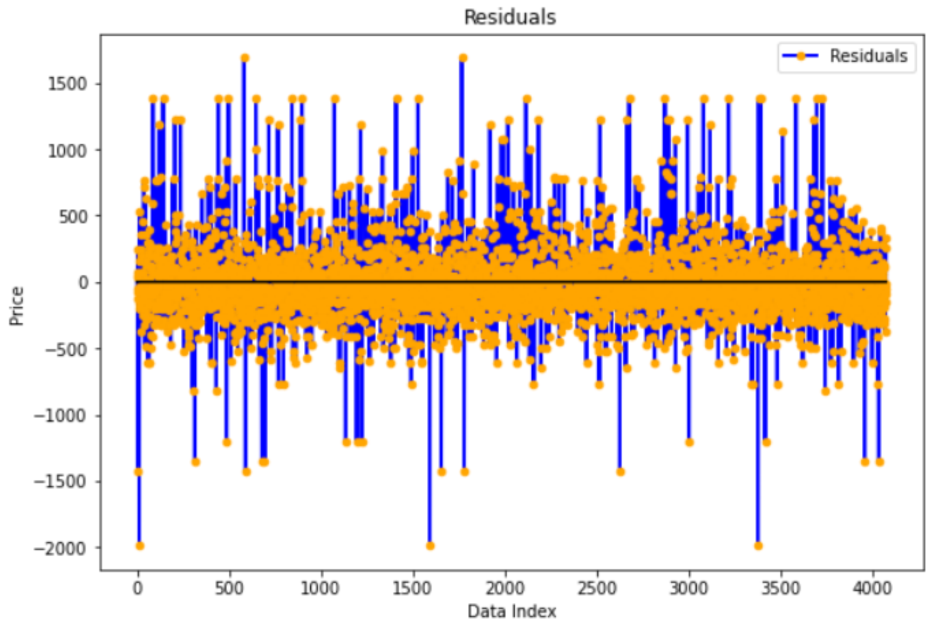


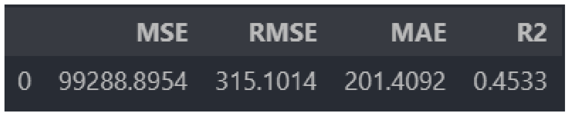
Figure 20: Model2 - Residual plot

7.2.3 Regularization Regression

Regularization is a technique used to prevent over-fitting in machine learning models. Over-fitting occurs when a model learns the training data too well and as a result, it does not generalize well to new data. Regularization helps to prevent over-fitting by adding a penalty to the model's cost function. This penalty discourages the model from learning too complex of a model, which can help to improve the model's generalization performance.

There are two main types of regularization methods:

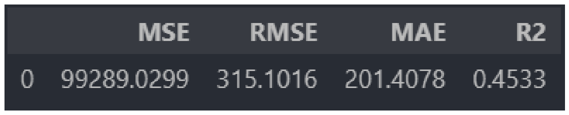
- L1 regularization (Lasso) : penalizes the sum of the absolute values of the model's coefficients. This means that L1 regularization encourages the model to have coefficients that are closer to zero. The figure below will show the evaluation metrics of our Lasso model's performance.



	MSE	RMSE	MAE	R2
0	99288.8954	315.1014	201.4092	0.4533

Figure 21: Evaluation metrics of Lasso Regression

- L2 regularization (Ridge) : penalizes the sum of the squared values of the model's coefficients. This means that L2 regularization encourages the model to have coefficients that are smaller in magnitude.
  The figure below will show the evaluation metrics of our Ridge model's performance.

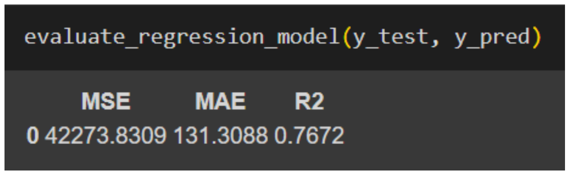


	MSE	RMSE	MAE	R2
0	99289.0299	315.1016	201.4078	0.4533

Figure 22: Evaluation metrics of Ridge Regression

7.2.4 Random Forest

Random Forest is a popular machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that combines multiple decision trees to make predictions.



```
evaluate_regression_model(y_test, y_pred)
```

	MSE	MAE	R2
0	42273.8309	131.3088	0.7672

Figure 23: Model 4 - Random Forest Regression with R-square score of 0.7672

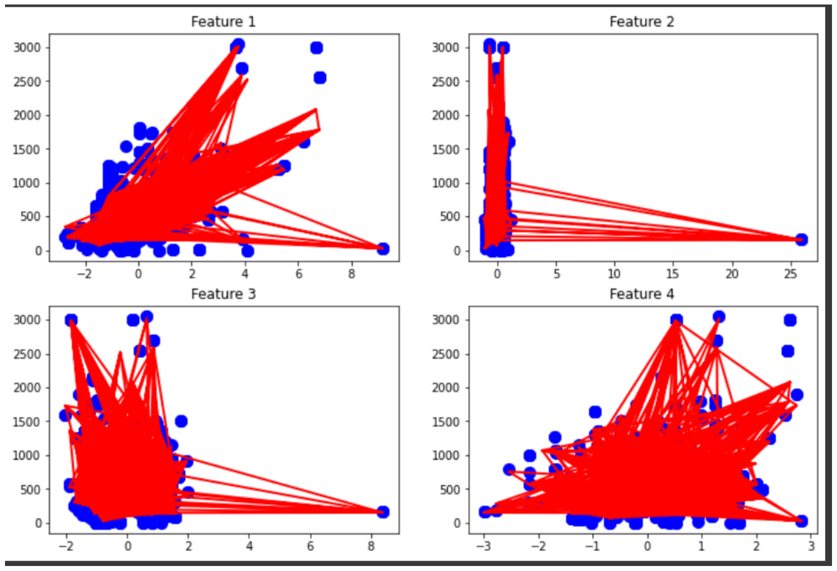


Figure 24: Model4 - Scatter plot of actual data value vs predicted value of each features corresponding to price

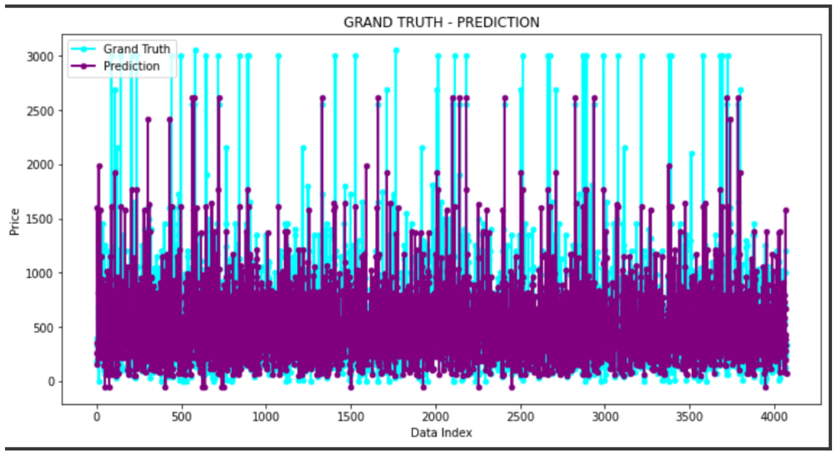


Figure 25: Model4 - Grand truth vs Prediction Plot

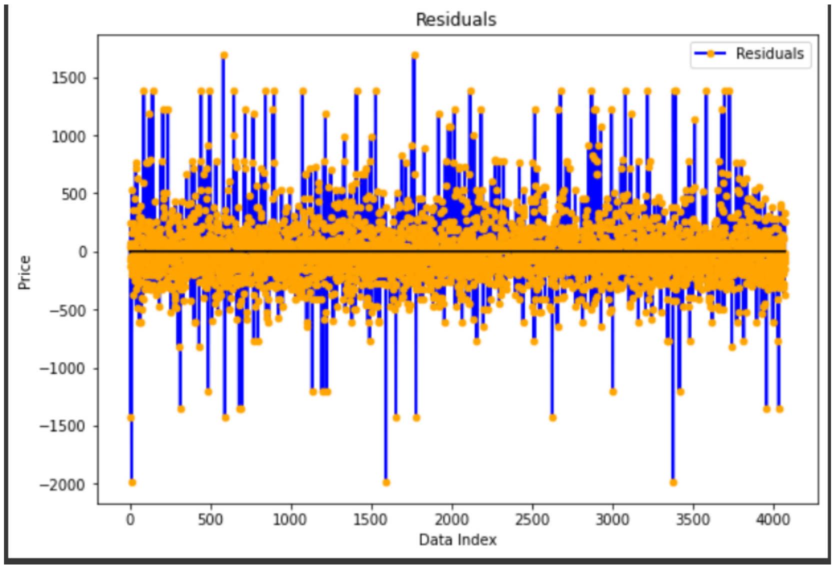
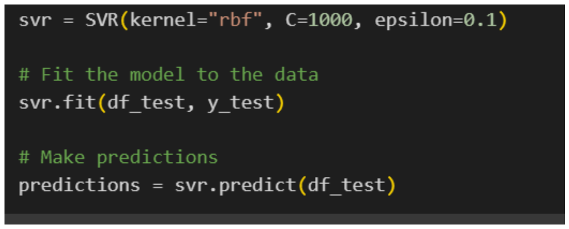


Figure 26: Model4 -Residual Plot

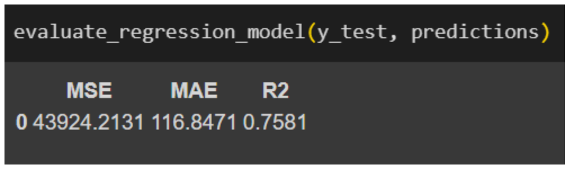
7.2.5 Support Vector Regressor

Support Vector Regressor (SVR) is a machine learning algorithm used for regression tasks. It is based on the principles of Support Vector Machines (SVM) and extends them to handle continuous target variables rather than discrete classes.



```
svr = SVR(kernel="rbf", C=1000, epsilon=0.1)

# Fit the model to the data
svr.fit(df_test, y_test)

# Make predictions
predictions = svr.predict(df_test)
```

Figure 27: Training Support Vector Regression



```
evaluate_regression_model(y_test, predictions)

         MSE       MAE     R2
0 43924.2131 116.8471 0.7581
```

Figure 28: Model 5 - Support Vector Regression with R-square score of 0.7581

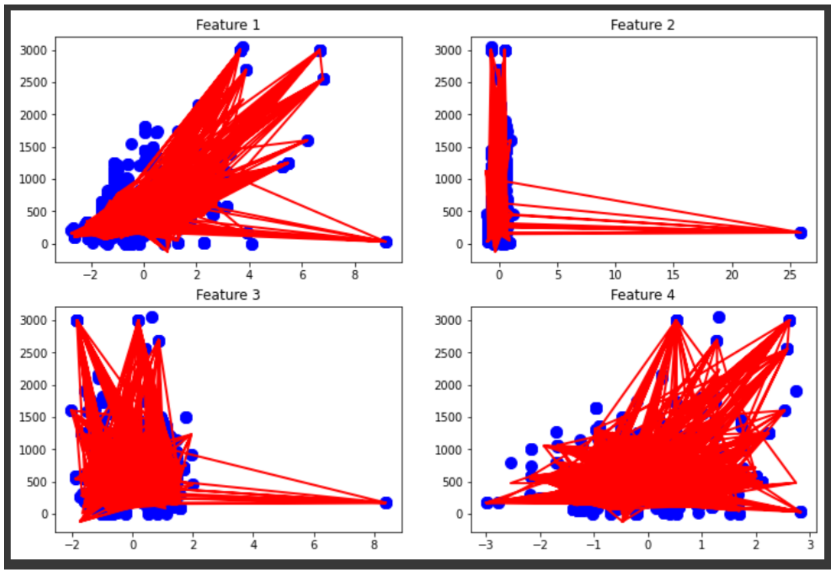


Figure 29: Model 5 - Scatter plot of actual data value vs predicted value of each features corresponding to price.

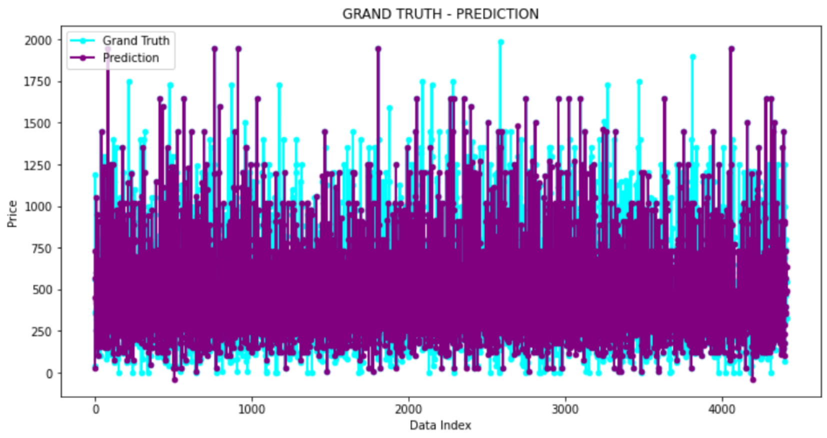


Figure 30: Model5 - Grand truth vs Prediction Plot

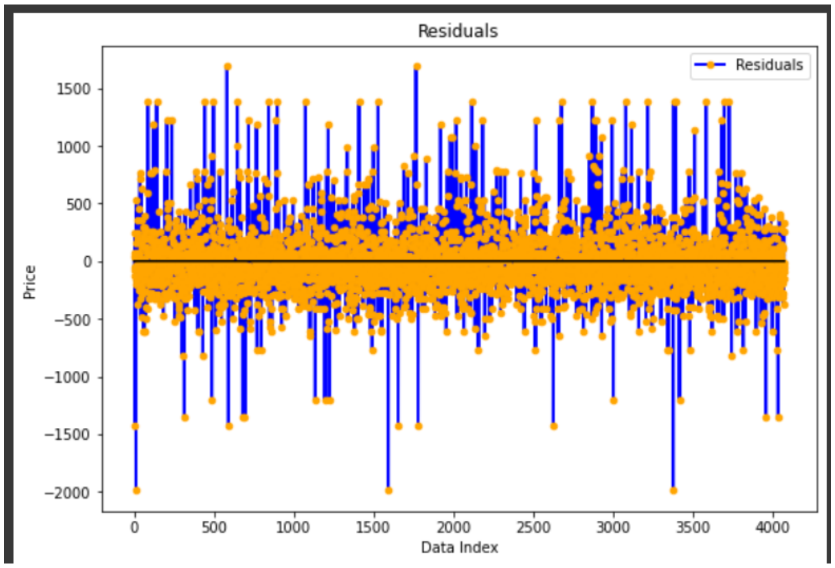


Figure 31: Model5 -Residual Plot

8 Conclusion

Laptop price prediction is a useful tool for consumers and businesses, aiding in informed decision-making. It helps consumers plan budgets, compare prices, and time their purchases. For businesses, it assists in inventory management, pricing strategies, and marketing efforts. While not infallible, continuous data collection and model refinement improve prediction accuracy. Overall, laptop price prediction provides valuable insights for navigating the market effectively. For this project, the model that we are going to use for our laptop's price prediction is Support Vector Regressor. Because this model overall shown the perfect accuracy and its prediction is close to the actual values.