Tequed Labs
Invent - Innovate - Iterate

A PROJECT REPORT ON “USED BIKE DATA” ANALYSIS

YASHASWINI S
TL23A0045

INDEX

Sr.no	Topic	Page no
1	Abstract	3
2	Introduction	4
3	Analyzing dataset	5-33
4	Future enhancement	34
5	Conclusion	35

ABSTRACT

The market for used bikes has witnessed significant growth in recent years, with consumers increasingly looking for cost-effective and sustainable transportation options. Understanding the factors influencing the pricing of used bikes is crucial for both buyers and sellers. In this study, we present an analysis of a comprehensive dataset containing information on various used bikes, with a focus on predicting their prices based on key attributes, namely, bike brand, number of kilometers driven, power, and age.

Our dataset, collected from various sources, includes a wide range of used bikes, providing a rich and diverse set of data points for analysis. We employ machine learning techniques to build predictive models that can estimate the price of a used bike based on the aforementioned attributes. This predictive model has the potential to serve as a valuable tool for both buyers and sellers in the used bike market.

INTRODUCTION

In today's fast-paced world, the demand for affordable and sustainable transportation options has led to a significant rise in the sale and purchase of used bikes. Whether it's for daily commuting, leisure rides, or simply as an eco-friendly choice, used bikes have become a popular choice for many. With this growing market, the need for understanding the factors that influence used bike prices has become paramount.

This dataset provides a comprehensive repository of information related to used bikes, encompassing various attributes such as bike brand, number of kilometers driven, engine power, and age. By leveraging the data contained within this dataset, we aim to develop a robust predictive model that can forecast the price of a used bike based on these key features.

The primary objective of this project is to harness suitable machine learning algorithms to create an accurate price prediction model. Such a model can prove invaluable not only to individuals looking to buy or sell used bikes but also to dealerships and businesses involved in the second-hand bike market. By understanding the relationship between these attributes and the final price, we can make informed decisions about bike valuation, leading to better market transparency and fair pricing.

This dataset and the predictive model hold the potential to reshape the used bike industry by providing a data-driven approach to determining bike prices. It enables stakeholders to plan and strategize effectively for future transactions, enhances the buying and selling experience, and promotes sustainability by encouraging the use of pre-owned vehicles. The insights gained from this analysis can serve as a valuable resource for both consumers and businesses, facilitating a smoother, more informed, and cost-effective experience in the used bike market.

STEPS FOR ANALYSING DATASET IS AS FOLLOWS

1. DATA ACQUISITION
2. DATA CLEANING
3. DATA VISUALISATION
4. DATA MODELING
5. ACCURACY CHECK
6. TESTING DATA

STEP 1: DATA ACQUISITION

SELECTION OF DATASET

The first step is selection of the database, I choose used bike dataset as I have keen interest in automobiles and the data was so beginner friendly to analyze as it has only 8 columns

PROBLEM STATEMENT

The main aim is to predict the future bike prices by using the linear regression model and to check the model accuracy.

DETAILS ABOUT DATASET

Dataset source: Used Bikes Prices in India | Kaggle
The dataset has 8 columns and 32649 rows

Columns are as follows

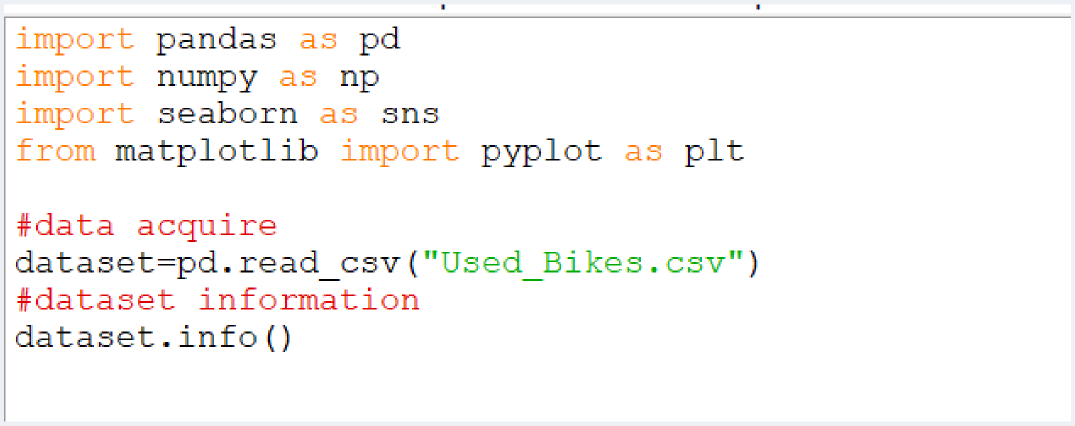
1. Bike name
2. Price of the bike
3. City in which the bike is available
4. Kilometer driven
5. How many owner it have
6. Age of the vehicle
7. Power of the vehicle
8. Brand of the vehicle

DATA ACQUSITION

In this step we mainly study the columns in dataset and try to understand their relations so that it will be easier to analyze the data and to choose the ML algorithm which is used for prediction.

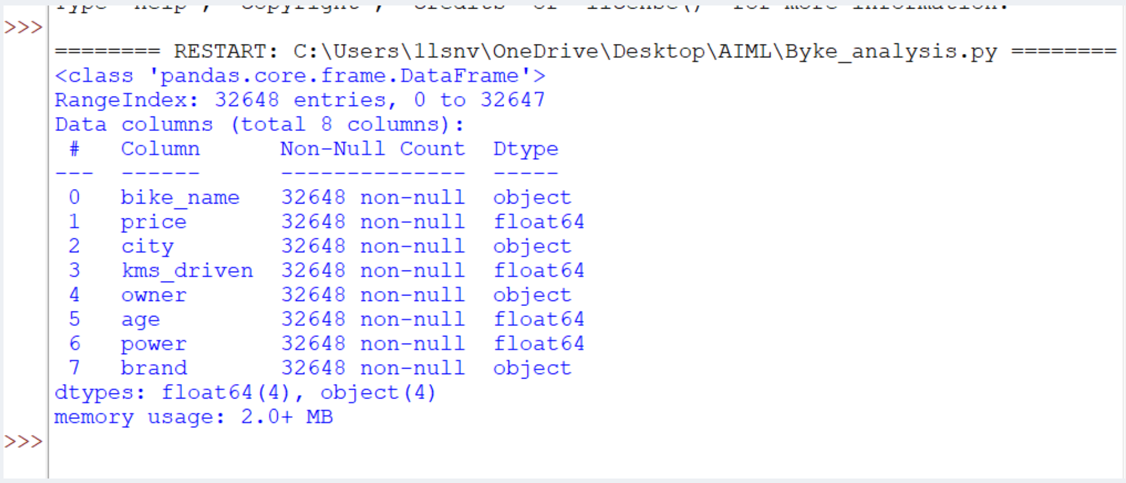
The dataset mainly has 8 rows as mentioned above we mainly use the pandas library to acquire the dataset.
let us see the first 10 instances of the dataset
By making use of **pandas.read_csv()** function to read the csv file then we use **info()** function to know the details about the instances and their datatype **iloc[]** to get the respective data instance and we use **print()** function to print those data instances

DATASET INFORMATION CODE



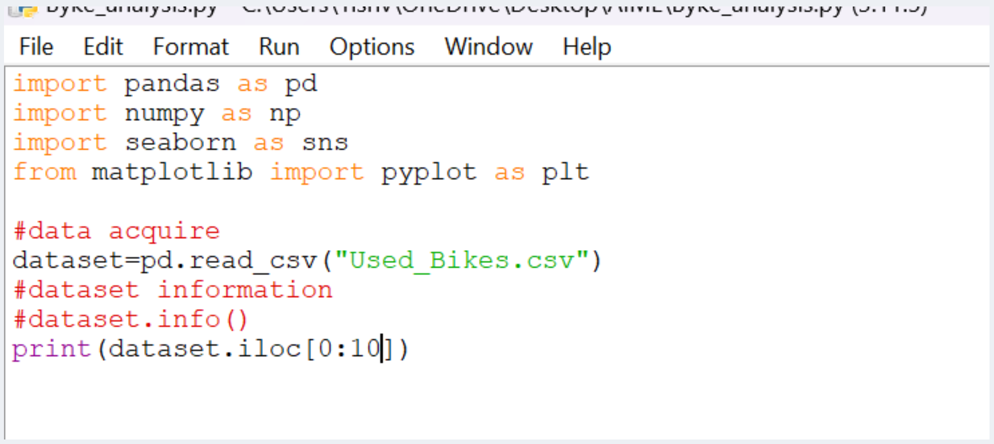
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

#data acquire
dataset=pd.read_csv("Used_Bikes.csv")
#dataset information
dataset.info()
```

OUT PUT



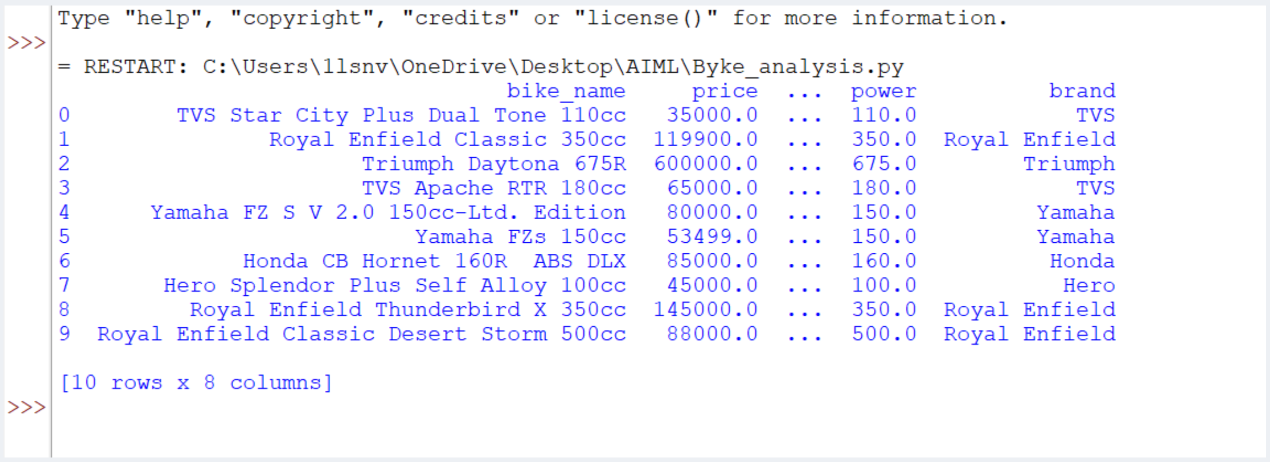
```
>>>
    ======== RESTART: C:\Users\11snv\OneDrive\Desktop\AIML\Byke_analysis.py ========
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 32648 entries, 0 to 32647
    Data columns (total 8 columns):
     #   Column      Non-Null Count  Dtype
    ---  ------      --------------  -----
     0   bike_name   32648 non-null  object
     1   price       32648 non-null  float64
     2   city        32648 non-null  object
     3   kms_driven  32648 non-null  float64
     4   owner       32648 non-null  object
     5   age         32648 non-null  float64
     6   power       32648 non-null  float64
     7   brand       32648 non-null  object
    dtypes: float64(4), object(4)
    memory usage: 2.0+ MB
>>>
```

DATASET ACQUISITION CODE



```python
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

#data acquire
dataset=pd.read_csv("Used_Bikes.csv")
#dataset information
#dataset.info()
print(dataset.iloc[0:10])
```

OUTPUT



```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\11snv\OneDrive\Desktop\AIML\Byke_analysis.py
                                  bike_name     price  ...  power           brand
0        TVS Star City Plus Dual Tone 110cc   35000.0  ...  110.0             TVS
1                Royal Enfield Classic 350cc  119900.0  ...  350.0   Royal Enfield
2                       Triumph Daytona 675R  600000.0  ...  675.0         Triumph
3                       TVS Apache RTR 180cc   65000.0  ...  180.0             TVS
4      Yamaha FZ S V 2.0 150cc-Ltd. Edition   80000.0  ...  150.0          Yamaha
5                           Yamaha FZs 150cc   53499.0  ...  150.0          Yamaha
6                Honda CB Hornet 160R  ABS DLX   85000.0  ...  160.0           Honda
7          Hero Splendor Plus Self Alloy 100cc   45000.0  ...  100.0            Hero
8            Royal Enfield Thunderbird X 350cc  145000.0  ...  350.0   Royal Enfield
9   Royal Enfield Classic Desert Storm 500cc   88000.0  ...  500.0   Royal Enfield

[10 rows x 8 columns]
>>>
```

STEP 2: DATA CLEANING

Data cleaning, also known as data preprocessing or data cleansing, is a crucial step in the machine learning (ML) pipeline. It involves the process of identifying and correcting errors, inconsistencies, and inaccuracies in your dataset to ensure that the data is of high quality and suitable for training machine learning models. Data cleaning is essential because the performance of your ML model is heavily dependent on the quality of the input data.

Data Inspection: Examine the dataset to get an initial understanding of its structure, size, and content. Check for missing values, outliers, and anomalies.

Handling Missing Data: Missing data can significantly impact model performance.
You can handle missing data by:

- Removing rows or columns with too many missing values.
- Imputing missing values with statistical measures like mean, median, or mode.
- Using advanced imputation methods like regression, K-nearest neighbors, or predictive modeling.

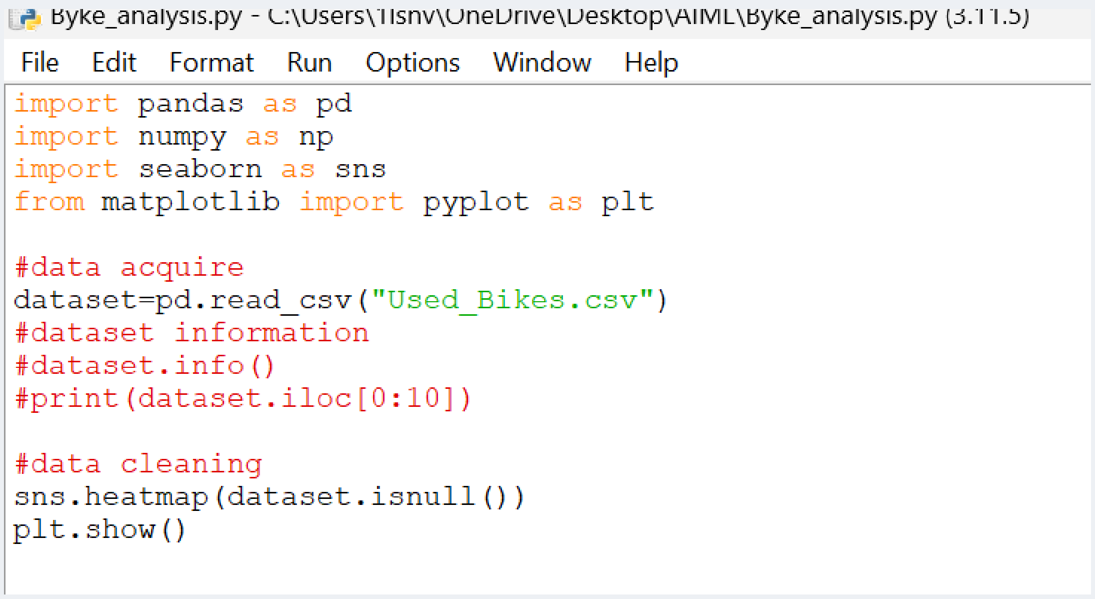
Dealing with Outliers: Outliers are data points that deviate significantly from the rest of the data.
You can handle outliers by:

- Detecting and removing extreme values that may be errors.
- Transforming the data using techniques like log transformation or winsorization to mitigate the impact of outliers.

Data Formatting: Ensure that data types are consistent and appropriate for modeling. Convert categorical variables to numerical representations through techniques like one-hot encoding or label encoding.

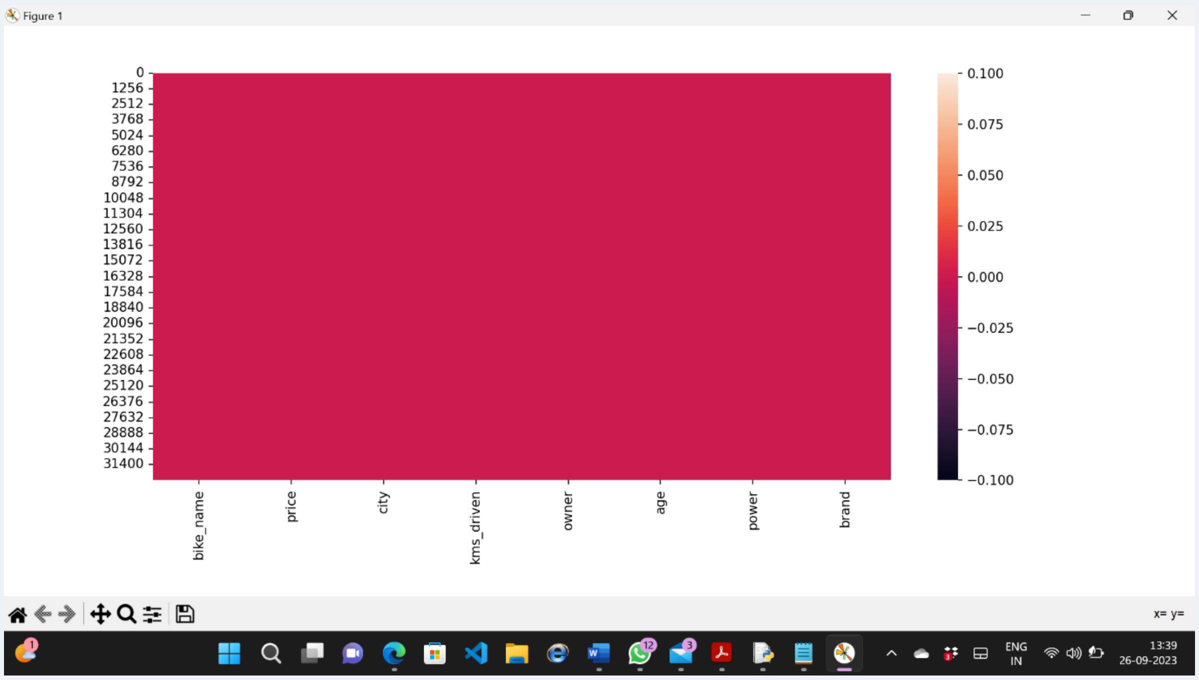
Handling Duplicate Data: Identify and remove duplicate records from the dataset to prevent bias and overfitting.

CODE FOR CHECKING MISSING VALUES


Byke_analysis.py - C:\Users\Tlsnv\OneDrive\Desktop\AIML\Byke_analysis.py (3.11.5)
File Edit Format Run Options Window Help

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

#data acquire
dataset=pd.read_csv("Used_Bikes.csv")
#dataset information
#dataset.info()
#print(dataset.iloc[0:10])

#data cleaning
sns.heatmap(dataset.isnull())
plt.show()
```

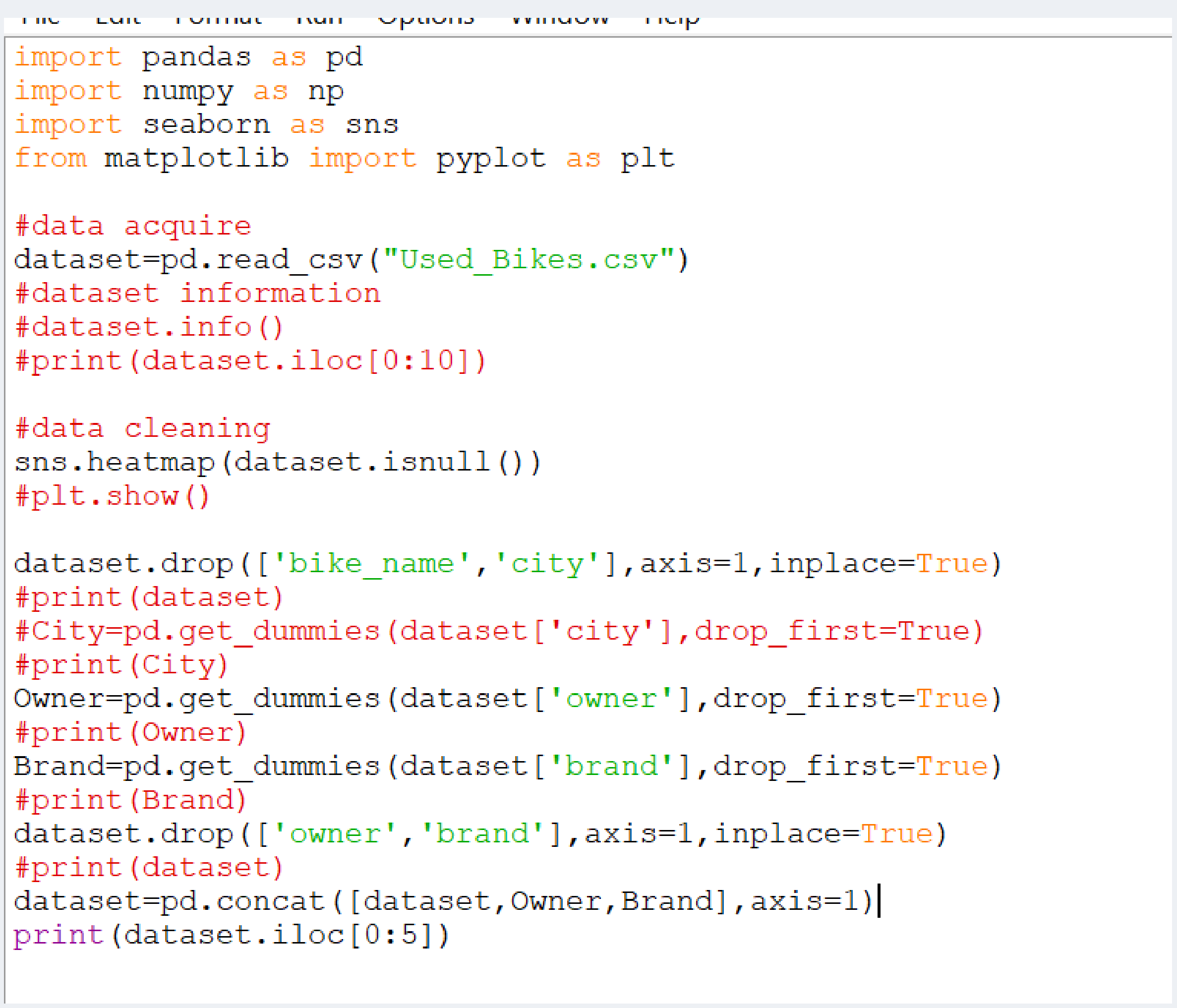
OUTPUT



We main make use of **seaborn** library and **matplotlib** library for plotting the heat maps
Since the dataset has is no null values hence the heatmap is uniform.

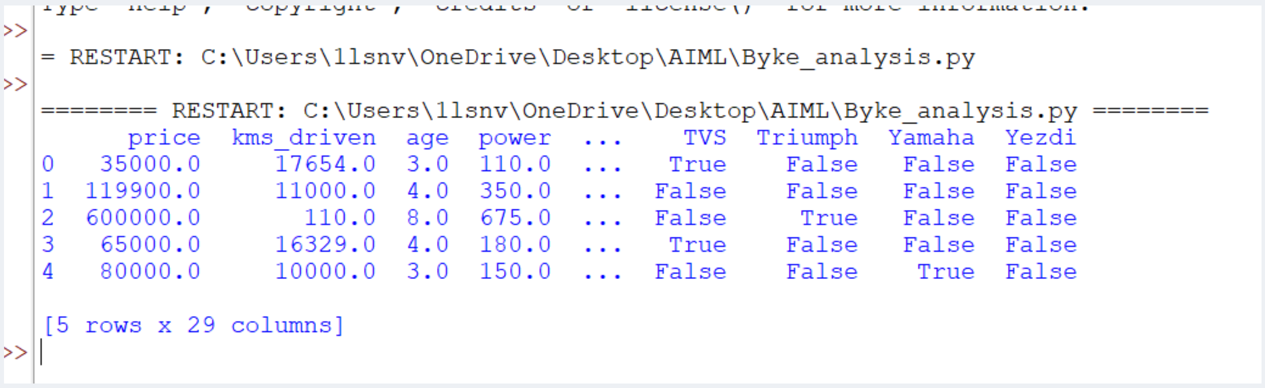
But the dataset has the outliers and different format data instances like owner, brand which are of string datatype these need to be converted.
Here the bike_name and city are the outliers so we drop the bike_name and city using **drop()** function
We make use of **get_dummies()** function to convert the different format data to required data format and store them in new variables. Then we drop the previous columns and add the new columns using **concat()** function

PYTHON CODE



```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

#data acquire
dataset=pd.read_csv("Used_Bikes.csv")
#dataset information
#dataset.info()
#print(dataset.iloc[0:10])

#data cleaning
sns.heatmap(dataset.isnull())
#plt.show()

dataset.drop(['bike_name','city'],axis=1,inplace=True)
#print(dataset)
#City=pd.get_dummies(dataset['city'],drop_first=True)
#print(City)
Owner=pd.get_dummies(dataset['owner'],drop_first=True)
#print(Owner)
Brand=pd.get_dummies(dataset['brand'],drop_first=True)
#print(Brand)
dataset.drop(['owner','brand'],axis=1,inplace=True)
#print(dataset)
dataset=pd.concat([dataset,Owner,Brand],axis=1)
print(dataset.iloc[0:5])
```

OUTPUT


```
>>
= RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py
>>
======== RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py ========
      price  kms_driven  age  power  ...    TVS  Triumph  Yamaha  Yezdi
0   35000.0     17654.0  3.0  110.0  ...   True    False   False  False
1  119900.0     11000.0  4.0  350.0  ...  False    False   False  False
2  600000.0       110.0  8.0  675.0  ...  False     True   False  False
3   65000.0     16329.0  4.0  180.0  ...   True    False   False  False
4   80000.0     10000.0  3.0  150.0  ...  False    False    True  False

[5 rows x 29 columns]
>>
```

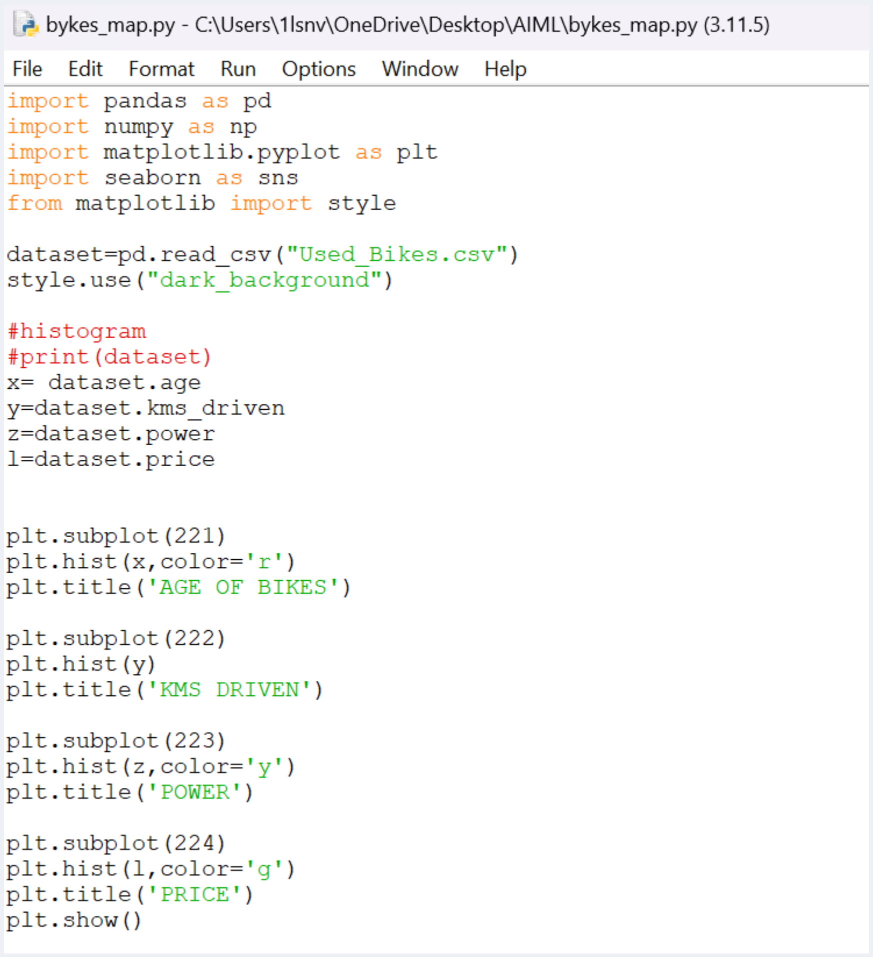
STEP 3: DATA VISUALIZATION

Data visualization in machine learning (ML) is the process of representing and displaying data in a graphical or visual format to help analysts, data scientists, and stakeholders better understand patterns, trends, and insights within the data. It is a crucial step in the ML workflow because it can provide valuable insights into the data, aid in feature selection, and assist in model evaluation and interpretation

Big data visualization often goes beyond the typical techniques used in normal visualization, such as pie charts, histograms and corporate graphs. It instead uses more complex representations, such as heat maps and fever charts. Big data visualization requires powerful computer systems to collect raw data, process it and turn it into graphical representations that humans can use to quickly draw insights.

Popular tools for data visualization in ML include Python libraries like Matplotlib, Seaborn, Plotly, and libraries integrated into ML platforms like TensorFlow and scikit-learn. These tools make it easier to create a wide range of visualizations to support different stages of the machine learning pipeline

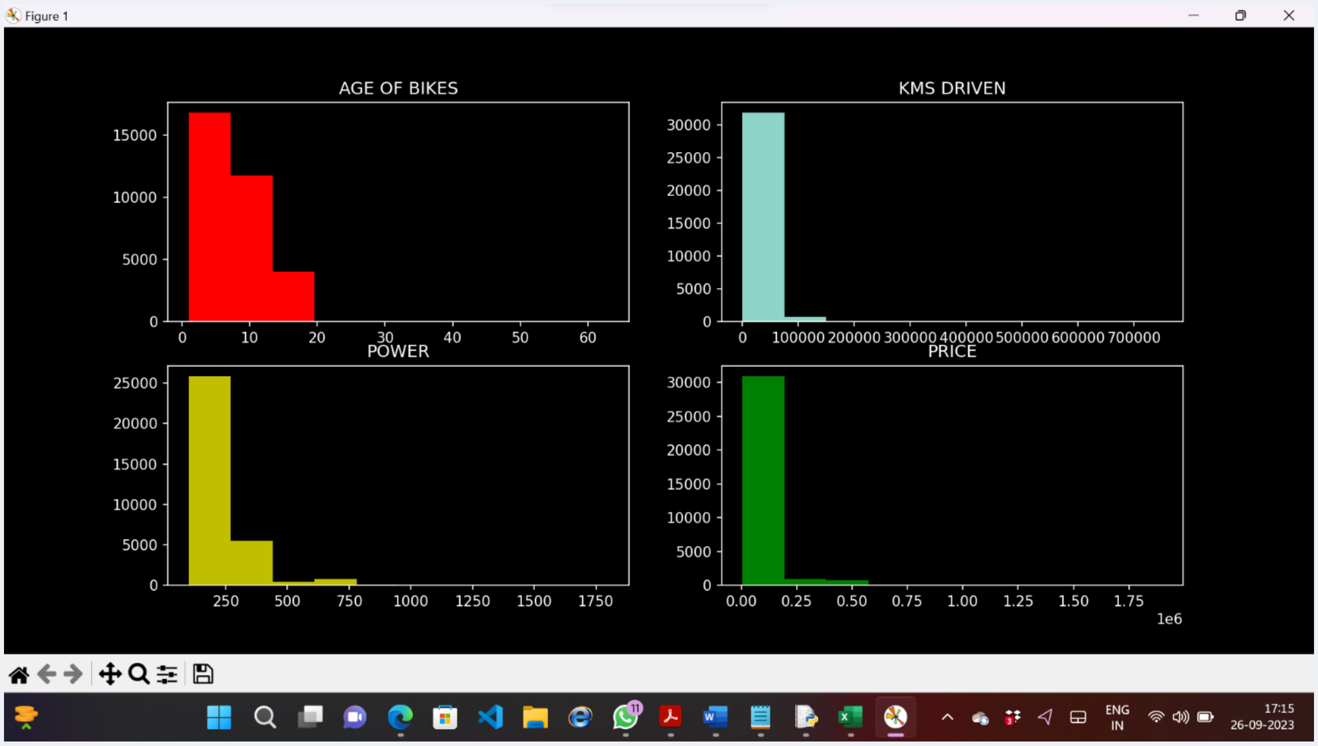
CODE 1:


bykes_map.py - C:\Users\1lsnv\OneDrive\Desktop\AIML\bykes_map.py (3.11.5)
File Edit Format Run Options Window Help

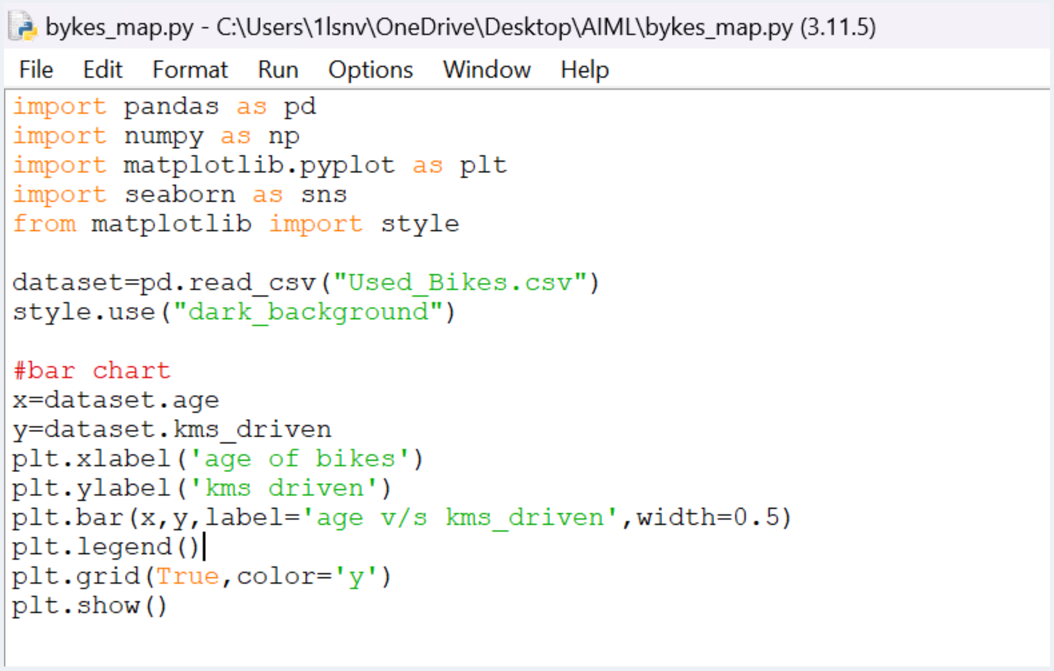
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style

dataset=pd.read_csv("Used_Bikes.csv")
style.use("dark_background")

#histogram
#print(dataset)
x= dataset.age
y=dataset.kms_driven
z=dataset.power
l=dataset.price


plt.subplot(221)
plt.hist(x,color='r')
plt.title('AGE OF BIKES')

plt.subplot(222)
plt.hist(y)
plt.title('KMS DRIVEN')

plt.subplot(223)
plt.hist(z,color='y')
plt.title('POWER')

plt.subplot(224)
plt.hist(l,color='g')
plt.title('PRICE')
plt.show()
```

This code is used to implement the histogram to begin with, we import the libraries like **pandas, matplotlib.**Within matplotlib we have imported the library called **style** and **pyplot** for plotting the instances. Initially we acquire the dataset using **read_csv()** method then we have set the background color to black using **style()** function then we initialized the variables with the dataset columns and then we used the plotter function called **subplot()** which is mainly used to plot different graph on one screen by making compartments .The input given to this function mainly specifies i.e 221 the first 2 represent number of rows, next 2 represents number of column, then 1 represent the position where the graph need to be displayed. Then we have plotted the histogram using the **hist()** function from pyplot library and assigned the title to each graph using the **title()** function from pyplot and to display the plot we used the **show()** function from pyplot library.

OUTPUT



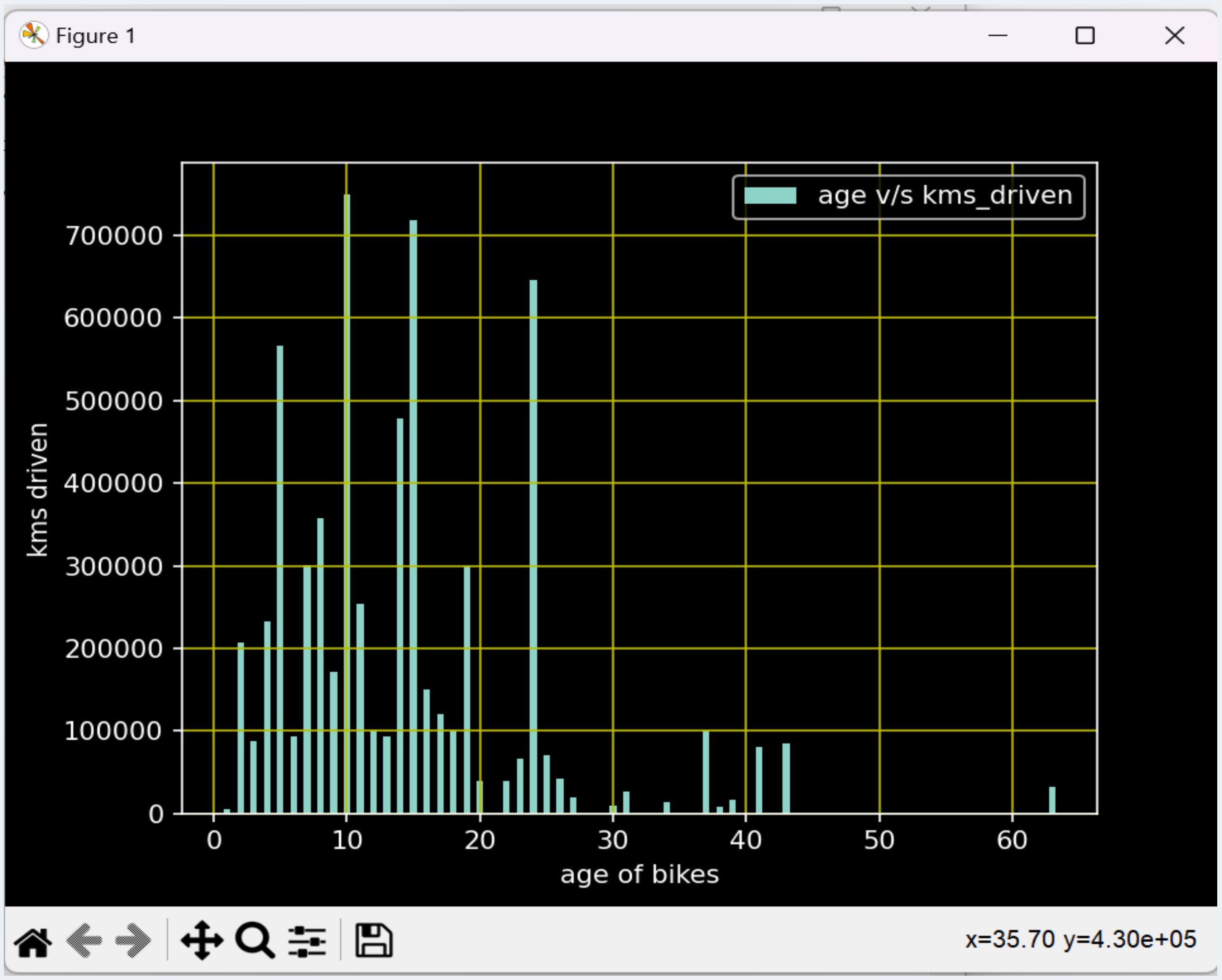
CODE 2:


bykes_map.py - C:\Users\1lsnv\OneDrive\Desktop\AIML\bykes_map.py (3.11.5)
File Edit Format Run Options Window Help

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style

dataset=pd.read_csv("Used_Bikes.csv")
style.use("dark_background")

#bar chart
x=dataset.age
y=dataset.kms_driven
plt.xlabel('age of bikes')
plt.ylabel('kms driven')
plt.bar(x,y,label='age v/s kms_driven',width=0.5)
plt.legend()
plt.grid(True,color='y')
plt.show()
```

This code mainly represents the bar graph age of bikes verses the kilometers driven by the bikes to plot this we imported the pandas and matplotlib libraries Using pandas library we acquire the dataset the we initialize the x coordinate and y coordinate values as the dataset.age and dataset.kms_driven.
Then we set the labels of x and y coordinates using the **xlabel()** and **ylabel()** functions then we plot the bar graph using **bar()** function where we input the x and y values to plot the graph then we used the **show()** function to display the plot.The **grid()** function is used to display the gridlines of the graph as visible in the output.

OUTPUT

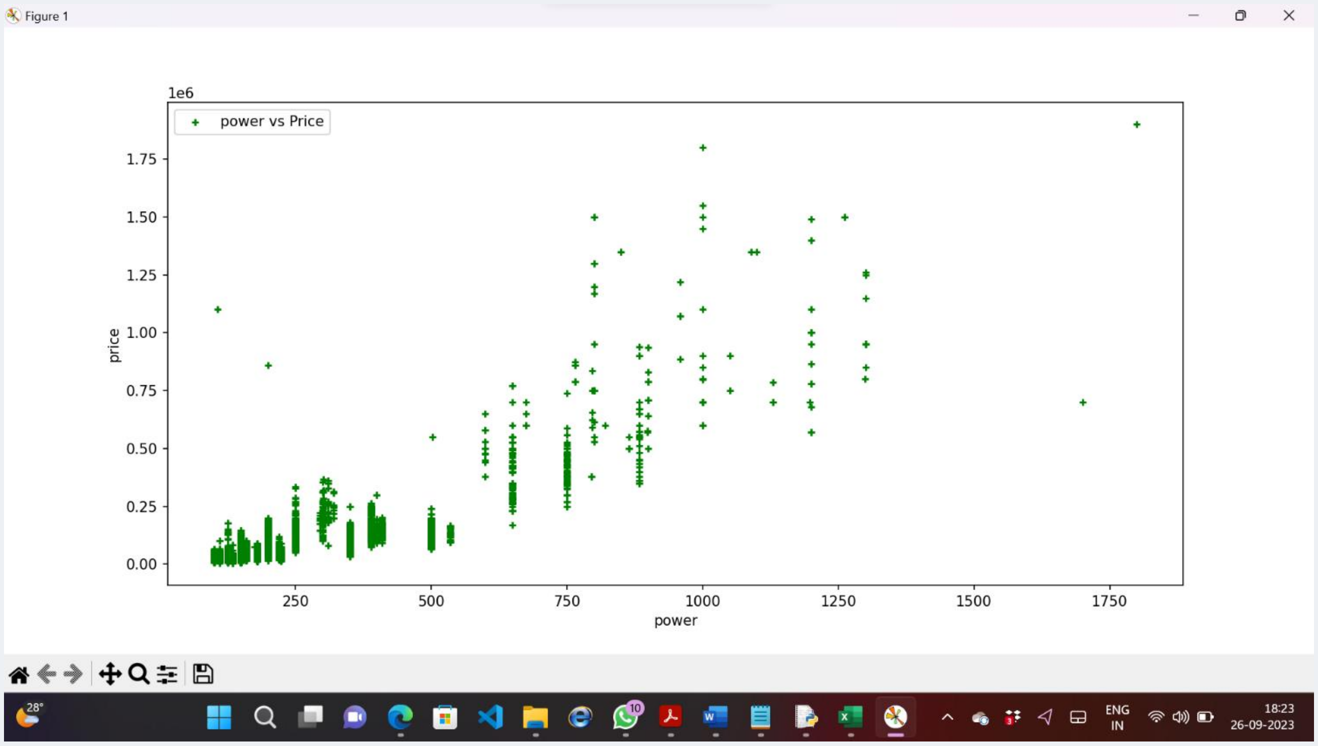


Code 3:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style

dataset=pd.read_csv("Used_Bikes.csv")

#scatter chart
x=dataset.power
y=dataset.price
plt.xlabel('power')
plt.ylabel('price')
plt.scatter(x,y,c='g',label='power vs Price',s=20,marker='+')
plt.legend()
plt.show()
```

Here we have plotted the scatter graph using the scatter() function which is the power v/s price graph and we changed the marker as "+" and changed the color of marker to green.

OUTPUT

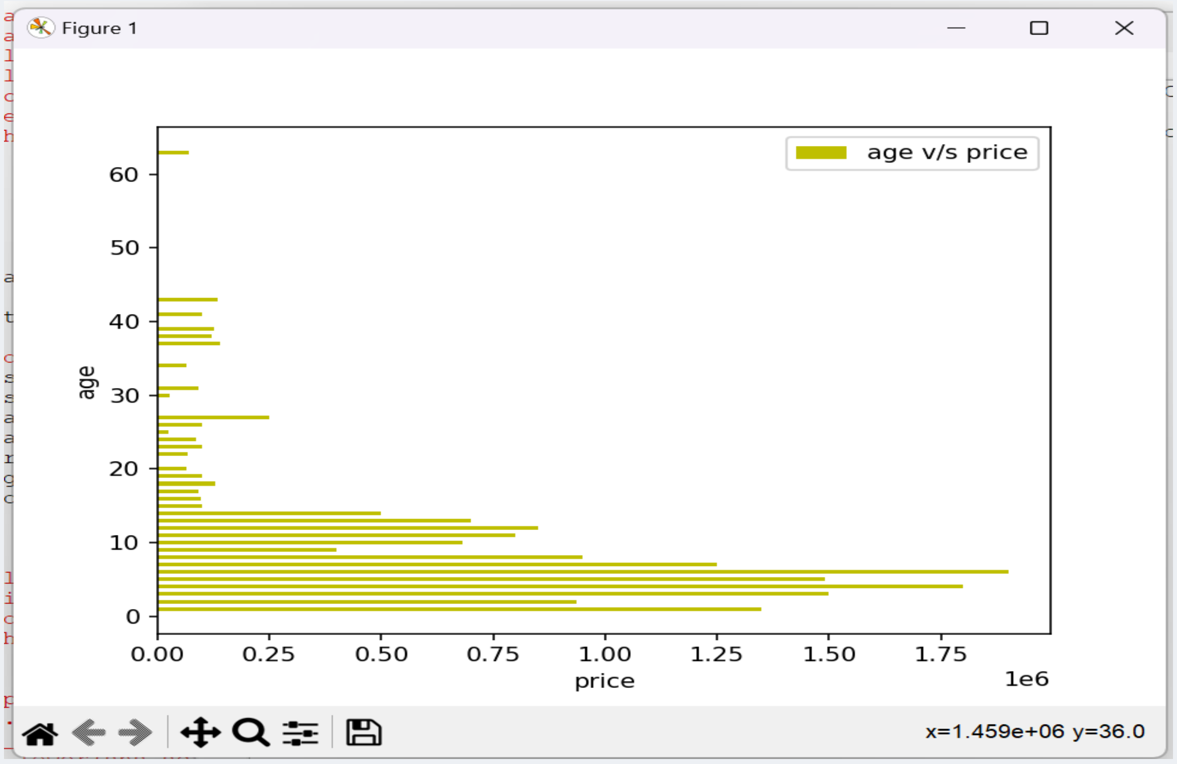


CODE 4 :

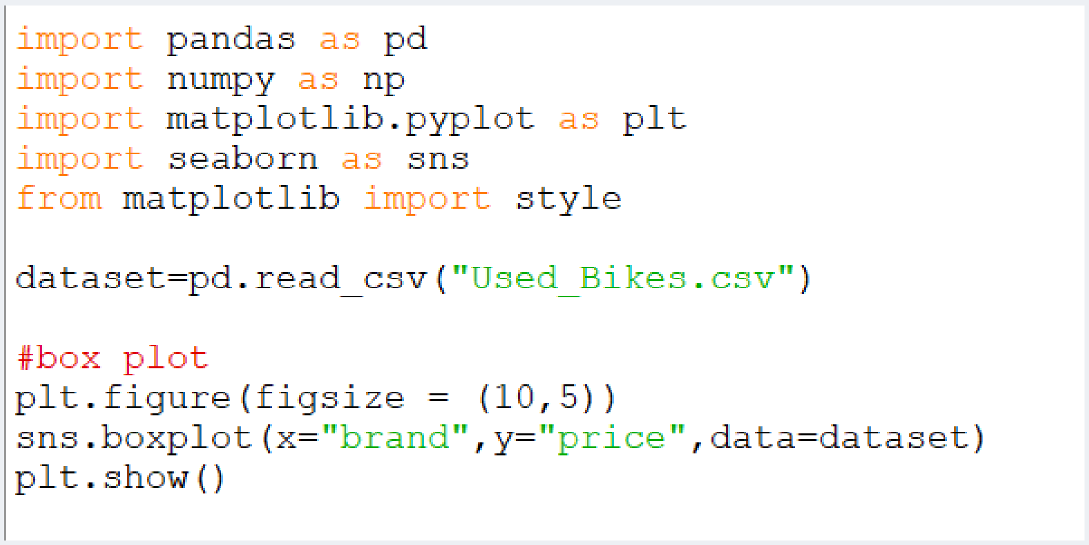
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style

dataset=pd.read_csv("Used_Bikes.csv")

#horisontal bar chart
x=dataset.age
y=dataset.price
plt.xlabel('price')
plt.ylabel('age')
plt.barh(x,y,label='age v/s price',height=0.5,color='y')
plt.legend()
plt.show()
```

This code is mainly used to plot the price v/s age graph using the horizontal bar graph

OUTPUT



CODE 5:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style

dataset=pd.read_csv("Used_Bikes.csv")

#box plot
plt.figure(figsize = (10,5))
sns.boxplot(x="brand",y="price",data=dataset)
plt.show()
```

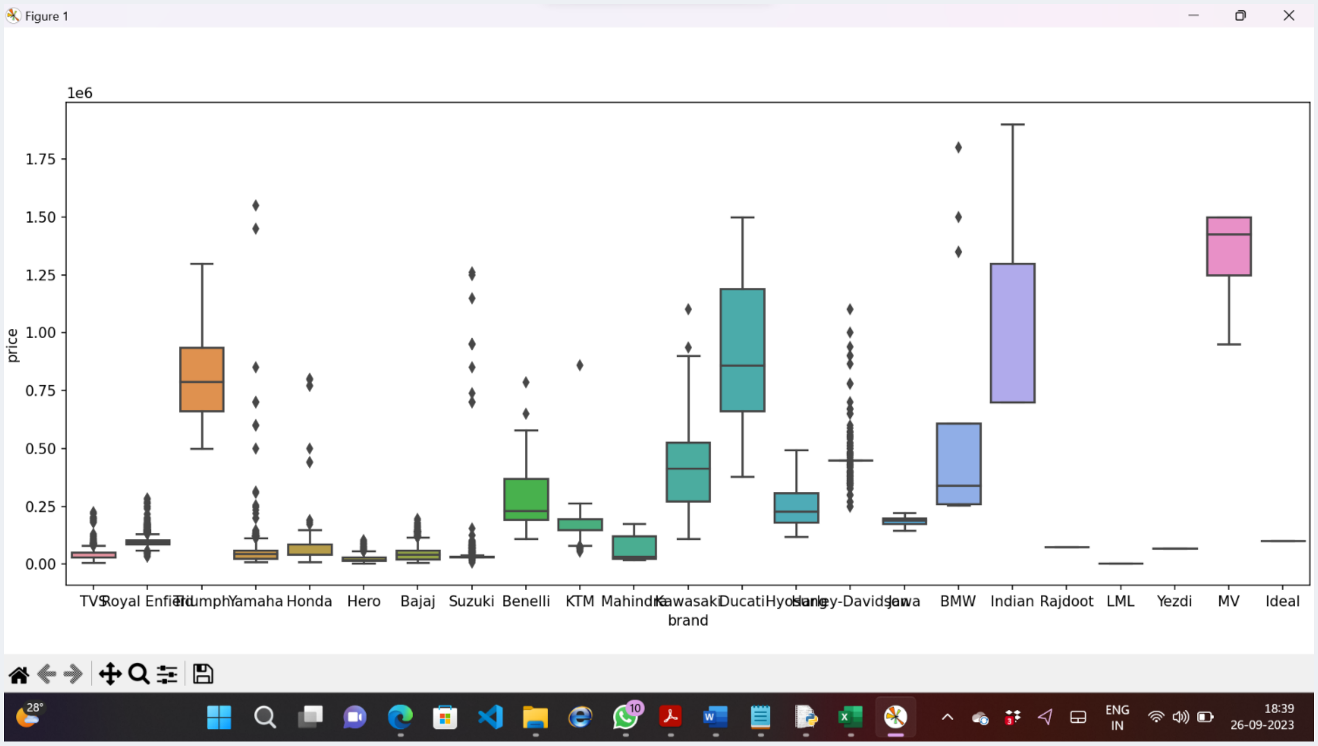
A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset. It provides a summary of key statistical measures and helps you visualize the spread and central tendency of the data. Box plots are particularly useful for identifying outliers and comparing distributions between different groups or categories within the data.

The box's vertical length indicates the variability or spread of the data within the interquartile range.
The longer the whiskers, the more spread out the data is beyond the central 50%.
The median line within the box represents the central value of the data.
Outliers are displayed as individual points outside the whiskers

Box plots are particularly useful for comparing the distributions of multiple datasets or categories side by side, making it easy to see differences in spread and central tendency. They are commonly used in data analysis, especially when dealing with numerical data, to gain insights into data distribution and identify potential outliers.

Here we have plotted the box graph of brand v/s price by making use of **seaborn** library function called **boxplot()**

OUTPUT



CODE 6:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style

dataset=pd.read_csv("Used_Bikes.csv")
style.use("dark_background")

#pair plot
g=sns.pairplot(dataset)
g.map_upper(sns.scatterplot,color='red')
g.map_lower(sns.scatterplot, color='green')
g.map_diag(plt.hist)
plt.show()
```

A pair plot is a type of data visualization technique commonly used in data analysis, particularly in exploratory data analysis (EDA). It is a grid of scatterplots and histograms that allows you to visualize pairwise relationships between multiple variables in a dataset. Pair plots are especially useful when you want to quickly identify patterns, correlations, and distributions in multivariate data.

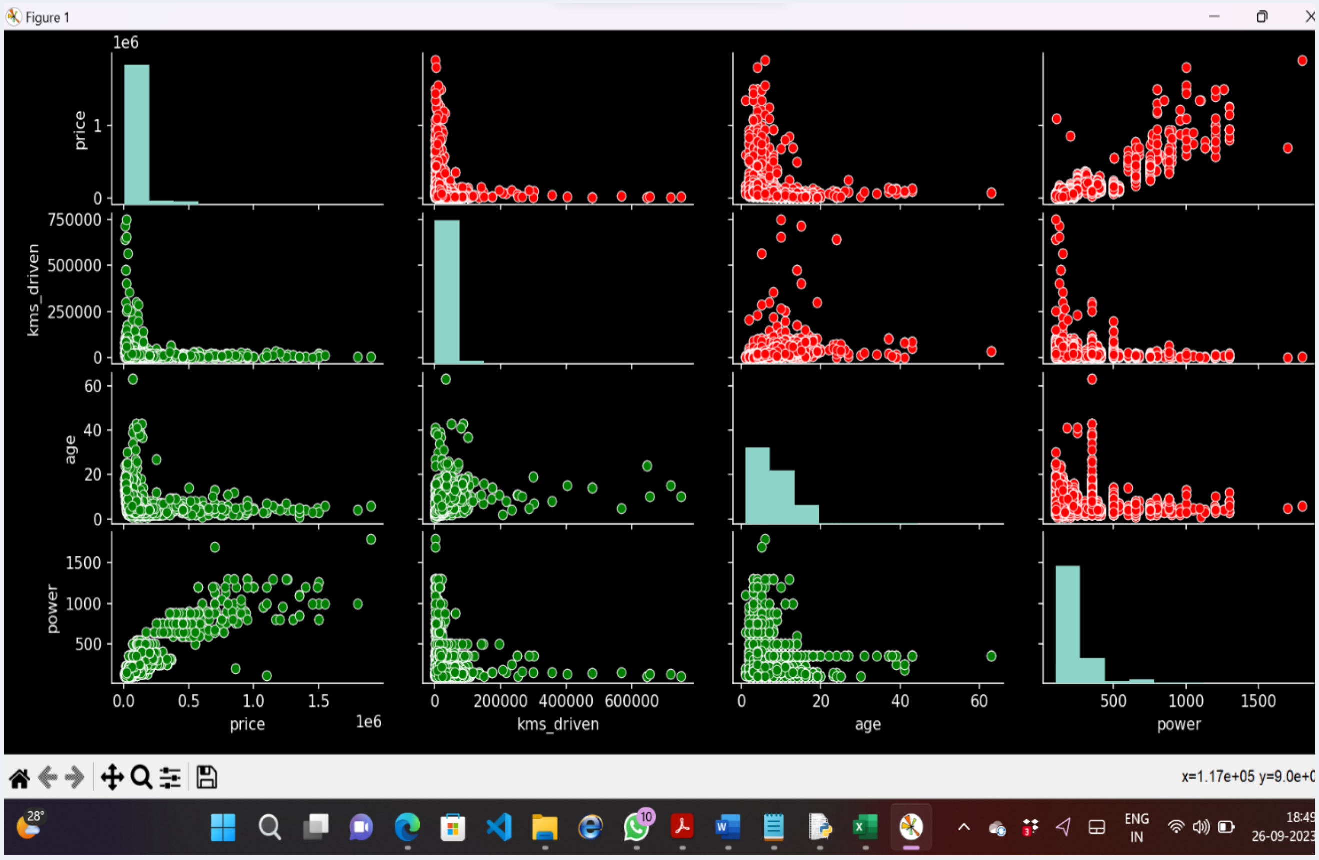
Here's how a pair plot typically works:

Scatterplots: Along the diagonal of the grid, you'll typically see histograms or kernel density plots for each variable, showing the distribution of that variable. This provides a univariate view of each variable.

Off-Diagonal Scatterplots: In the lower-left half (below the diagonal) and upper-right half (above the diagonal) of the grid, you'll see scatterplots that display the relationships between pairs of variables. Each scatterplot represents the relationship between one variable (e.g., X-axis) and another variable (e.g., Y-axis).

To plot the pairplot we imported the library called seaborn then we called the **pairplot()** function then we modified the diagonal plots as the histogram and upper diagonal and lower diagonal plots as the scatter plots using **map_upper()** and **map_lower()** functions and then we used the **show()** function to display graphs

OUTPUT



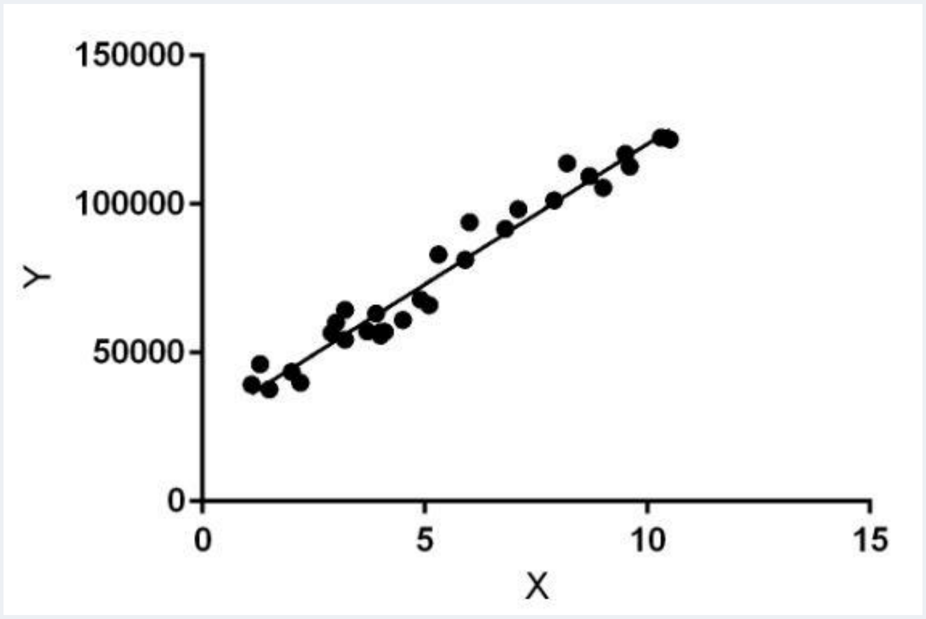
STEP 4: DATA MODELLING

Data modeling in machine learning (ML) refers to the process of creating and designing a mathematical or computational representation of a real-world problem or system. It is a fundamental step in the ML workflow and plays a central role in building predictive or descriptive models. Data modeling involves transforming raw data into a format that can be used by ML algorithms to make predictions, classifications, or gain insights.

The dataset that I have chosen is the "**used bike dataset**" this dataset lay under the a **regression** problem and I have chosen the **Linear Regression** model to predict the price.

LINEAR REGRESSION MODEL:

Linear regression is one of the fundamental and widely used supervised machine learning techniques used for predicting a continuous target variable based on one or more independent features. It establishes a linear relationship between the input features and the output variable. Linear regression is commonly used for tasks such as predicting house prices, estimating the impact of variables on a target, or modeling trends over time.



Objective: The objective of linear regression is to find the values of the coefficients (b0, b1, b2, ..., bn) that minimize the error or the difference between the predicted values and the actual values in the training dataset. Commonly used error metrics include Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Assumptions: Linear regression relies on certain assumptions, including the linearity of the relationship, independence of errors, homoscedasticity (constant variance of errors), and normally distributed errors.

Here are the key components and characteristics of a linear regression model in machine learning:

Linear Relationship: Linear regression assumes that there is a linear relationship between the input features (predictors or independent variables) and the target variable (dependent variable). Mathematically, it can be represented as:

$y = b0 + b1 * x1 + b2 * x2 + ... + bn * xn$

where:

y is the target variable.
x1, x2, ..., xn are the input features.
b0 is the intercept (the value of y when all features are zero).
b1, b2, ..., bn are the coefficients that represent the impact of each feature on the target variable.

Simple and Multiple Linear Regression: Linear regression can be categorized into two main types:

- **Simple Linear Regression:** In this case, there is only one input feature (x) that is used to predict the target variable (y). The relationship is a straight line.
- **Multiple Linear Regression:** Multiple input features (x1, x2, ..., xn) are used to predict the target variable. The relationship is a hyperplane in higher-dimensional space.

Model Evaluation: Linear regression models are evaluated using various metrics, including R-squared (R^2), which measures the proportion of variance in the target variable explained by the model, and the aforementioned MSE and MAE.

Cost function

The cost function or the loss function is nothing but the error or difference between the predicted value y_predict and the true value Y. It is the **Mean Squared Error (MSE)** between the predicted value and the true value. The cost function (J) can be written as:

$$\text{Cost function}(J) = \frac{1}{n}\sum_{n}^{i}(\hat{y_i} - y_i)^2$$

If the cost function of any line is minimum then it will be the best fit line

Linear regression is a simple yet powerful tool for modeling relationships between variables when there is a linear association. However, it may not perform well when the relationship between variables is highly non-linear or when there are interactions between features. In such cases, more complex regression models or other machine learning techniques may be more suitable.

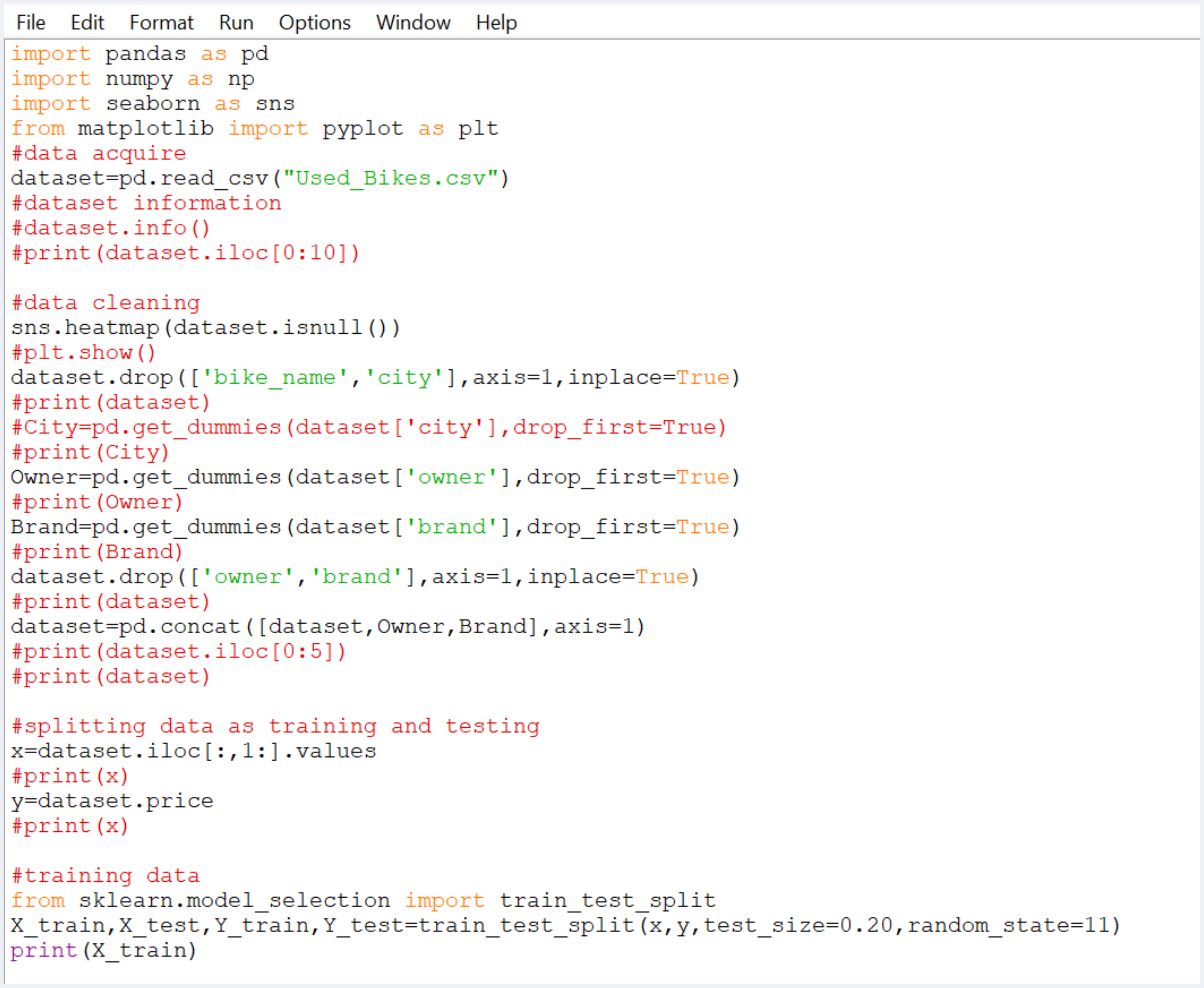
SPLITTING DATASET AS TESTING AND TRAINING DATA

Before implementing any model, we need to split the dataset to train and test sets. We use **train_test_split** class from **sklearn.model** selection library to split our dataset.

Random state ensures that the splits that you generate are reproducible. Scikit-learn uses random permutations to generate the splits. The random state that you provide is used as a seed to the random number generator. This ensures that the random numbers are generated in the same order.

The **test_size** mainly specify the percent of data instance which goes for testing.

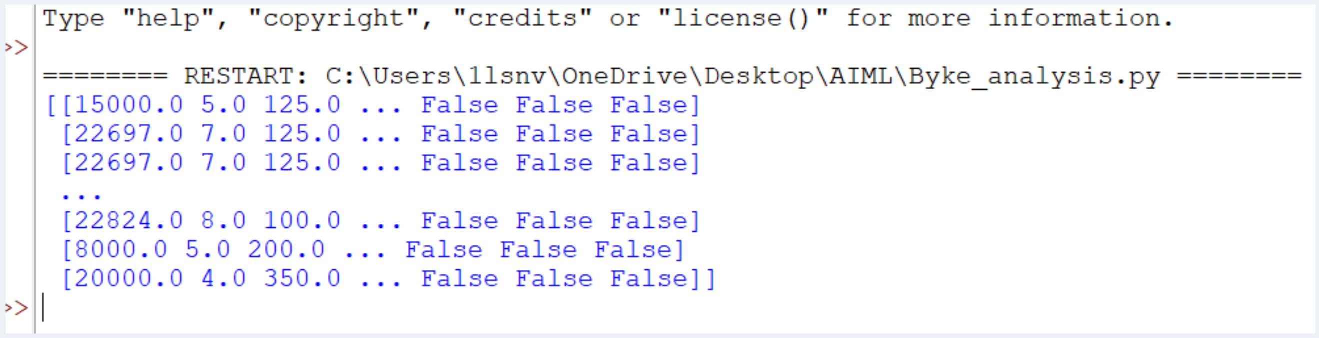
CODE:



```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
#data acquire
dataset=pd.read_csv("Used_Bikes.csv")
#dataset information
#dataset.info()
#print(dataset.iloc[0:10])

#data cleaning
sns.heatmap(dataset.isnull())
#plt.show()
dataset.drop(['bike_name','city'],axis=1,inplace=True)
#print(dataset)
#City=pd.get_dummies(dataset['city'],drop_first=True)
#print(City)
Owner=pd.get_dummies(dataset['owner'],drop_first=True)
#print(Owner)
Brand=pd.get_dummies(dataset['brand'],drop_first=True)
#print(Brand)
dataset.drop(['owner','brand'],axis=1,inplace=True)
#print(dataset)
dataset=pd.concat([dataset,Owner,Brand],axis=1)
#print(dataset.iloc[0:5])
#print(dataset)

#splitting data as training and testing
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)

#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
print(X_train)
```

Here we have given test_size as 0.20 and random_state as 11 then we have printed the X_train value using **print ()** function.

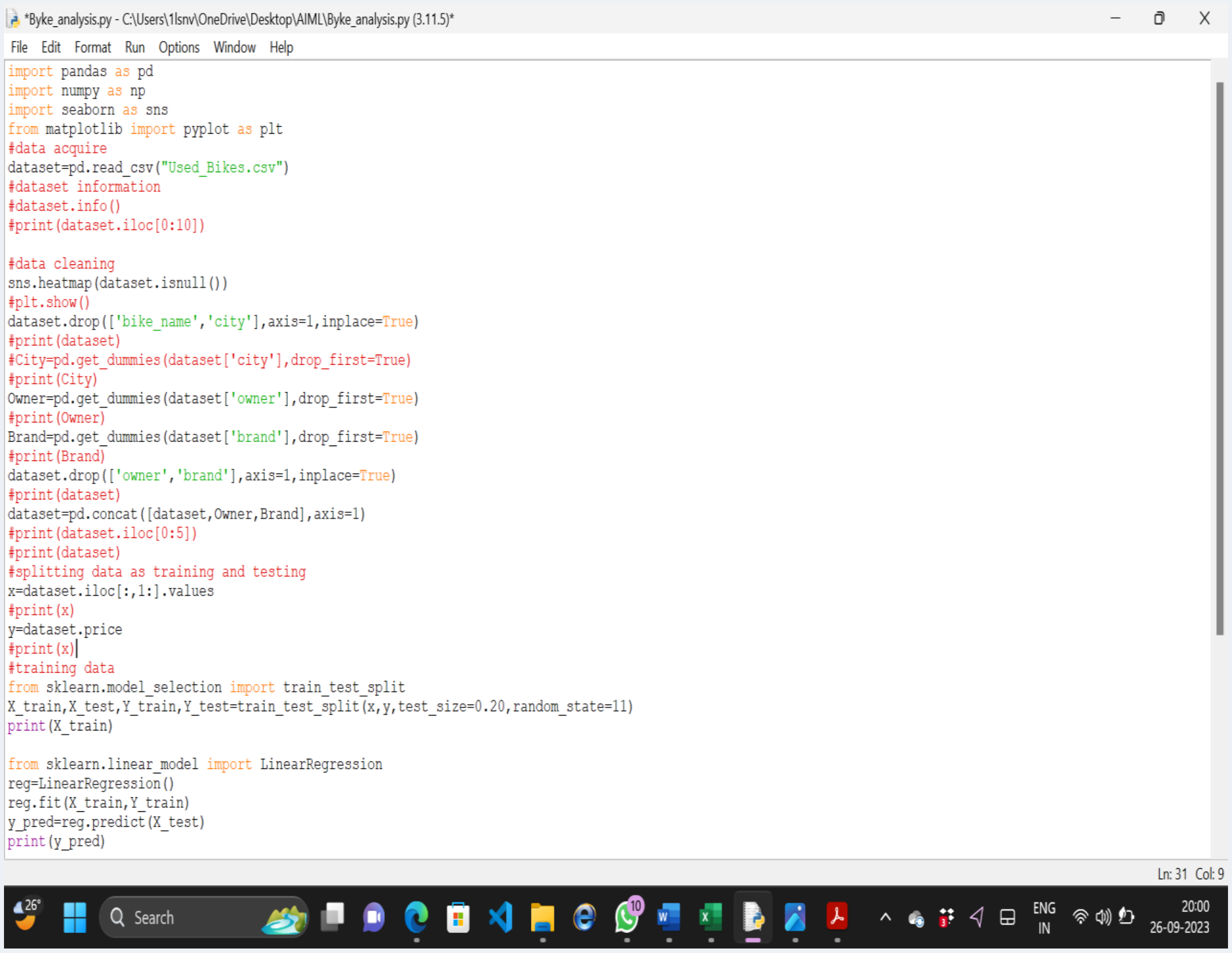
OUTPUT



```
Type "help", "copyright", "credits" or "license()" for more information.
>>
======== RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py ========
[[15000.0 5.0 125.0 ... False False False]
 [22697.0 7.0 125.0 ... False False False]
 [22697.0 7.0 125.0 ... False False False]
 ...
 [22824.0 8.0 100.0 ... False False False]
 [8000.0 5.0 200.0 ... False False False]
 [20000.0 4.0 350.0 ... False False False]]
>>
```

Then we import the LinearRegression class from sklearn.linear_model
Then we initialize the variable called reg and assign the **LinearRegression()** method. Then we train the model using the **fit()** method. we predict the values using the **predict()** method.

CODE :

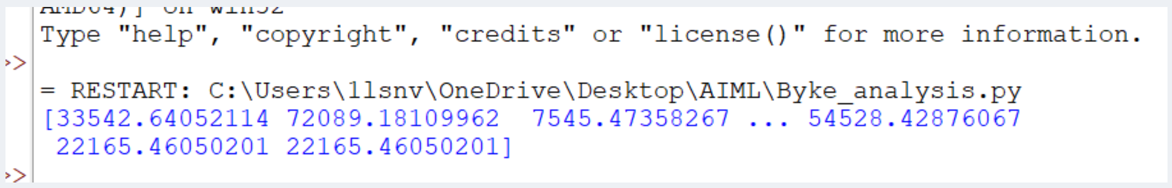


Byke_analysis.py - C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py (3.11.5)

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
#data acquire
dataset=pd.read_csv("Used_Bikes.csv")
#dataset information
#dataset.info()
#print(dataset.iloc[0:10])

#data cleaning
sns.heatmap(dataset.isnull())
#plt.show()
dataset.drop(['bike_name','city'],axis=1,inplace=True)
#print(dataset)
#City=pd.get_dummies(dataset['city'],drop_first=True)
#print(City)
Owner=pd.get_dummies(dataset['owner'],drop_first=True)
#print(Owner)
Brand=pd.get_dummies(dataset['brand'],drop_first=True)
#print(Brand)
dataset.drop(['owner','brand'],axis=1,inplace=True)
#print(dataset)
dataset=pd.concat([dataset,Owner,Brand],axis=1)
#print(dataset.iloc[0:5])
#print(dataset)
#splitting data as training and testing
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)
#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred=reg.predict(X_test)
print(y_pred)
```

Ln: 31 Col: 9

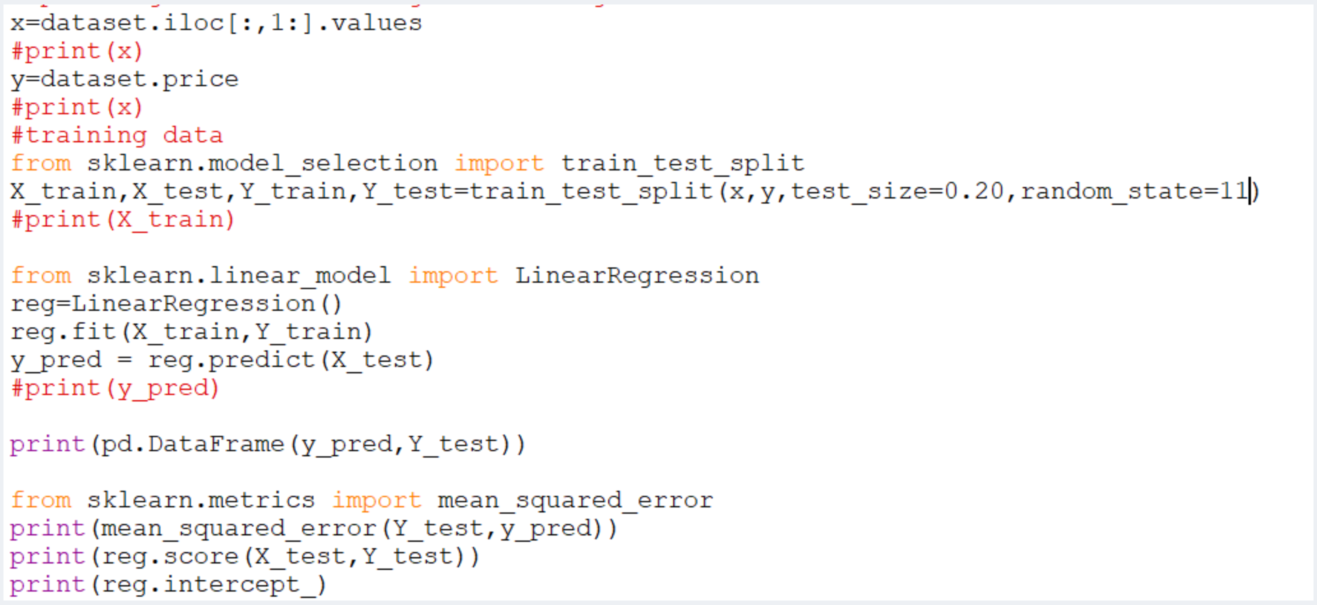
OUTPUT



```
Type "help", "copyright", "credits" or "license()" for more information.
>
= RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py
[33542.64052114 72089.18109962  7545.47358267 ... 54528.42876067
 22165.46050201 22165.46050201]
>
```

STEP 5: ACCURACY CHECK

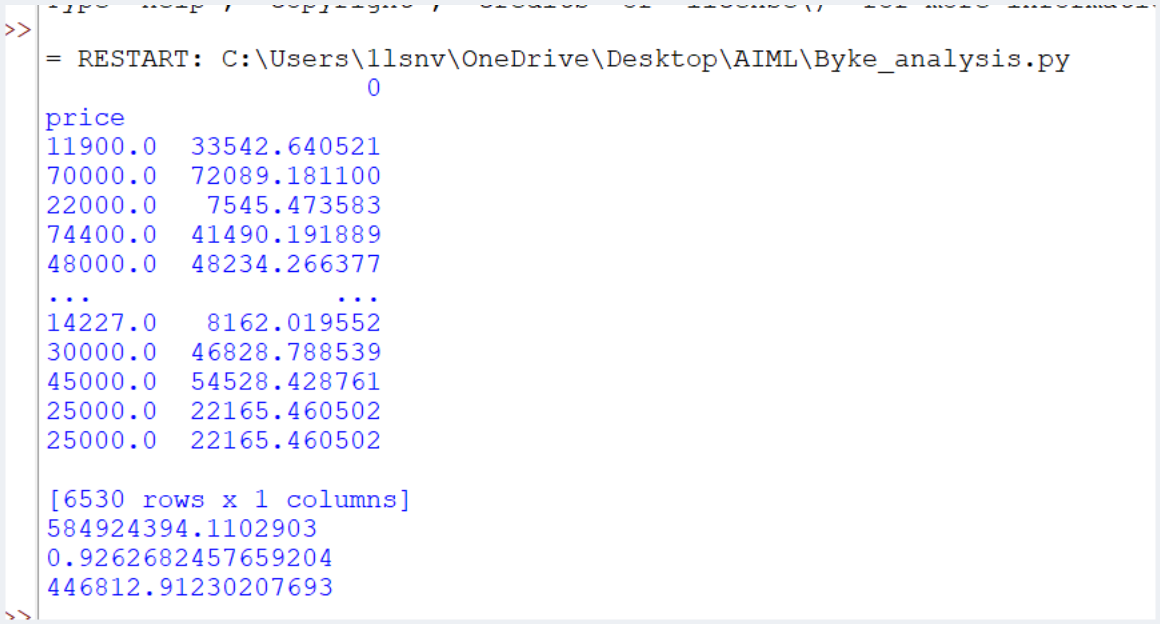
CODE:



```
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)
#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
#print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred = reg.predict(X_test)
#print(y_pred)

print(pd.DataFrame(y_pred,Y_test))

from sklearn.metrics import mean_squared_error
print(mean_squared_error(Y_test,y_pred))
print(reg.score(X_test,Y_test))
print(reg.intercept_)
```

Here in this code the accuracy is predicted using the **score()** method and the mean square error is calculated using **mean_square_error()** method and by using **reg.intercept_** we get the value about the intercept. Mean square error is calculated using the class mean_square_error from sklearn.metrics.

OUTPUT

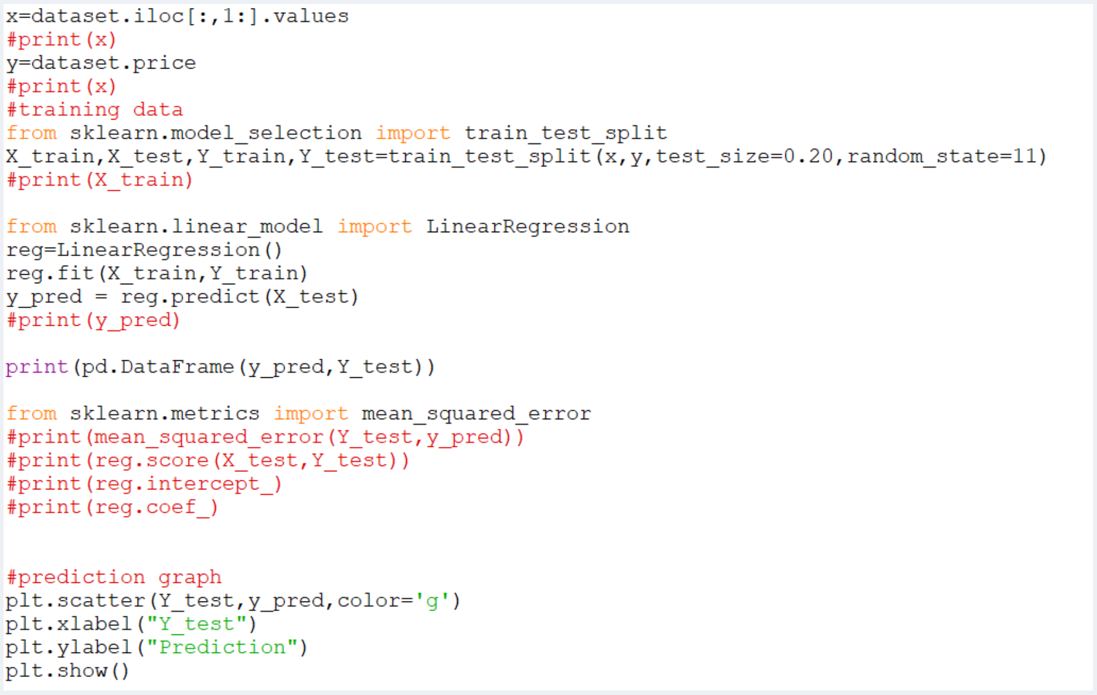


```
>>
   = RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py
                           0
   price
   11900.0  33542.640521
   70000.0  72089.181100
   22000.0   7545.473583
   74400.0  41490.191889
   48000.0  48234.266377
   ...               ...
   14227.0   8162.019552
   30000.0  46828.788539
   45000.0  54528.428761
   25000.0  22165.460502
   25000.0  22165.460502

   [6530 rows x 1 columns]
   584924394.1102903
   0.9262682457659204
   446812.91230207693
>>
```

This model gives the accuracy about 92% and has the mean square error of about 5849424394 and the intercept is 446812

GRAPHICAL REPRESENTATION

CODE:



```
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)
#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
#print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred = reg.predict(X_test)
#print(y_pred)

print(pd.DataFrame(y_pred,Y_test))

from sklearn.metrics import mean_squared_error
#print(mean_squared_error(Y_test,y_pred))
#print(reg.score(X_test,Y_test))
#print(reg.intercept_)
#print(reg.coef_)


#prediction graph
plt.scatter(Y_test,y_pred,color='g')
plt.xlabel("Y_test")
plt.ylabel("Prediction")
plt.show()
```

OUTPUT

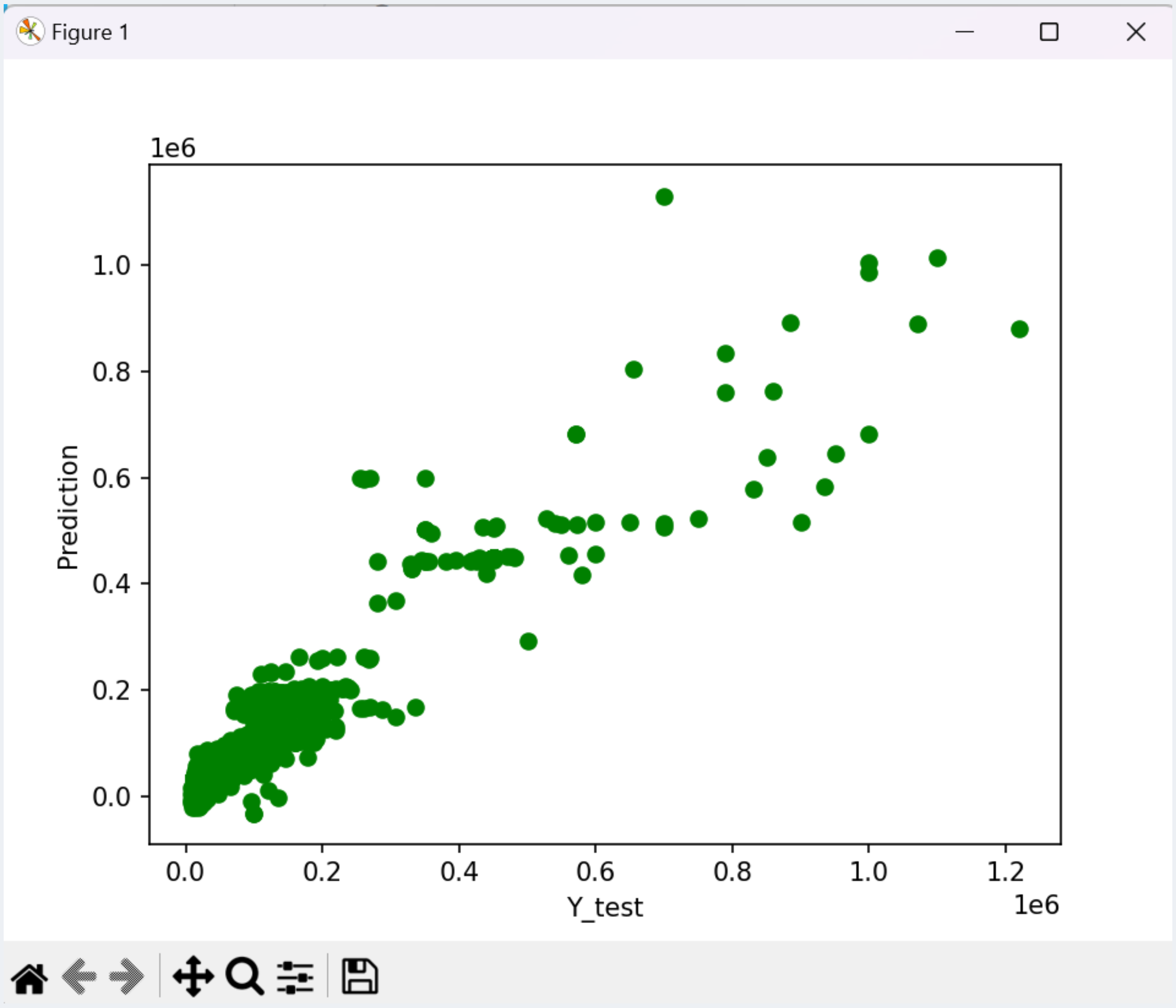
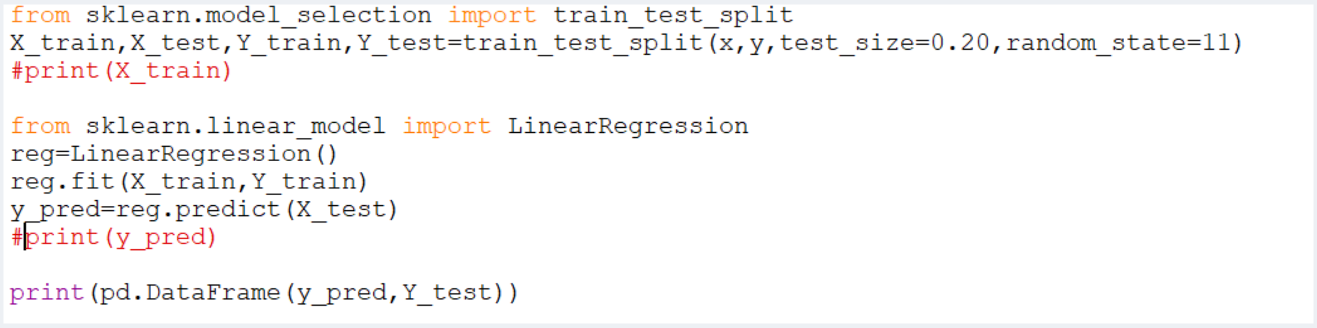


Figure 1
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Prediction
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Y_test
1e6

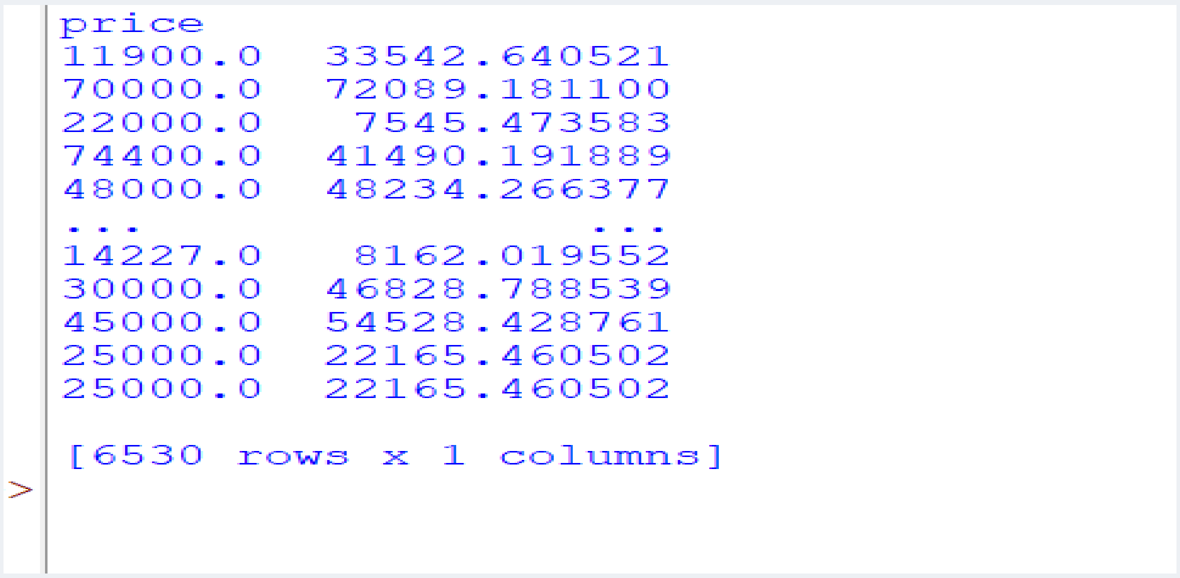
STEP 6: TESTING DATA

CODE: Comparing predicted y value and actual y value



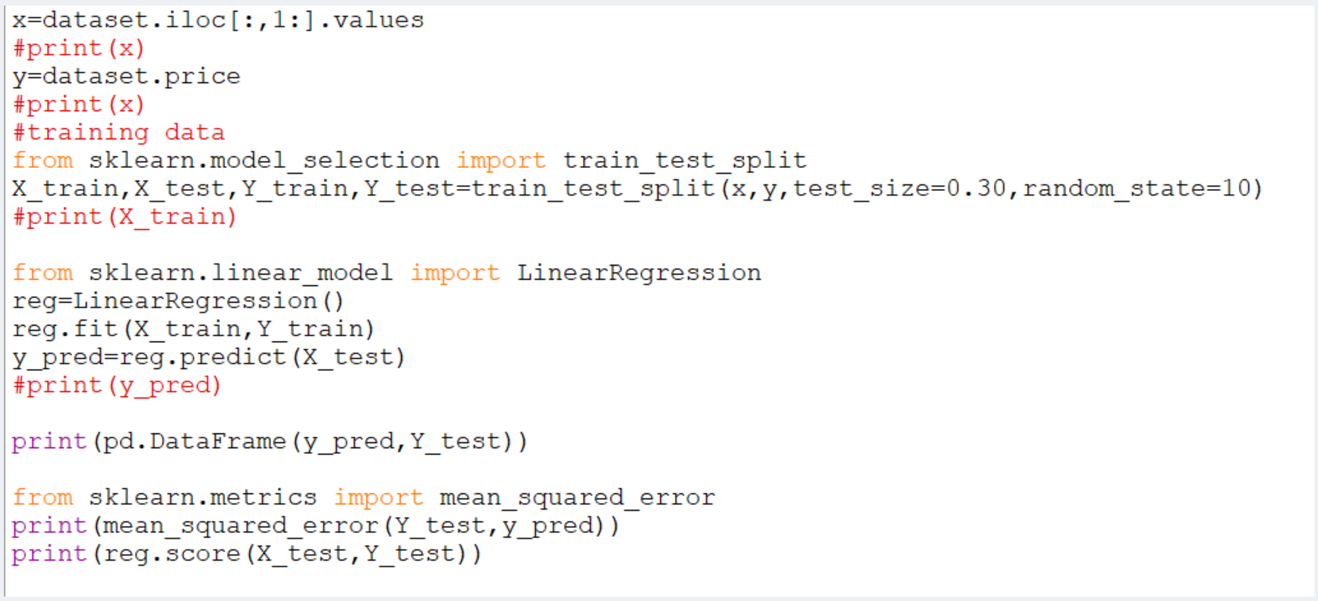
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
#print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred=reg.predict(X_test)
#print(y_pred)

print(pd.DataFrame(y_pred,Y_test))
```

OUTPUT



```
price
11900.0   33542.640521
70000.0   72089.181100
22000.0    7545.473583
74400.0   41490.191889
48000.0   48234.266377
...                ...
14227.0    8162.019552
30000.0   46828.788539
45000.0   54528.428761
25000.0   22165.460502
25000.0   22165.460502

[6530 rows x 1 columns]
>
```

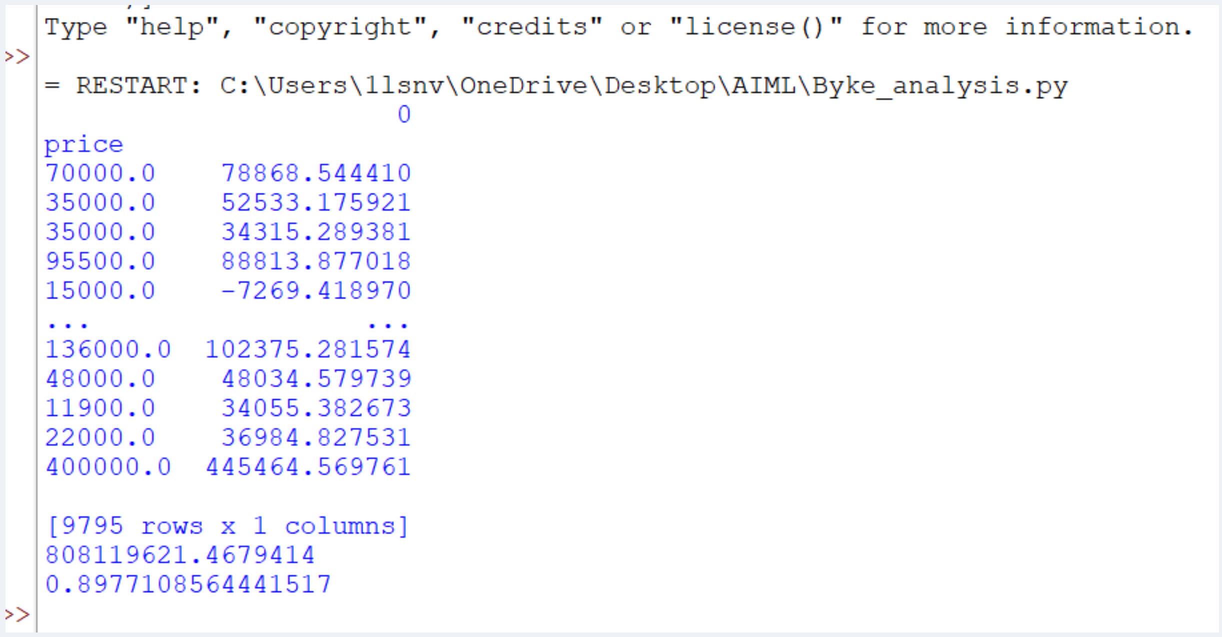
CODE: Testing by changing test size and random state



```
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)
#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.30,random_state=10)
#print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred=reg.predict(X_test)
#print(y_pred)

print(pd.DataFrame(y_pred,Y_test))

from sklearn.metrics import mean_squared_error
print(mean_squared_error(Y_test,y_pred))
print(reg.score(X_test,Y_test))
```

Here we changed the test size as 0.30 and random state as 10
We check the accuracy by importing mean_score_error class from sklearn.metrics and by using the **score()** method .

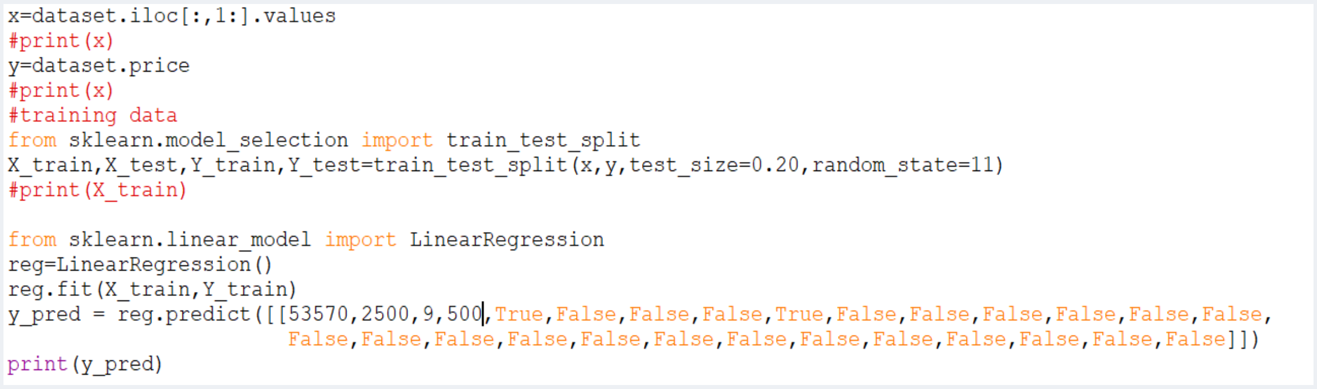
OUTPUT



```
Type "help", "copyright", "credits" or "license()" for more information.
>>
= RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py
                      0
price
70000.0    78868.544410
35000.0    52533.175921
35000.0    34315.289381
95500.0    88813.877018
15000.0    -7269.418970
...                 ...
136000.0  102375.281574
48000.0    48034.579739
11900.0    34055.382673
22000.0    36984.827531
400000.0  445464.569761

[9795 rows x 1 columns]
808119621.4679414
0.8977108564441517
>>
```

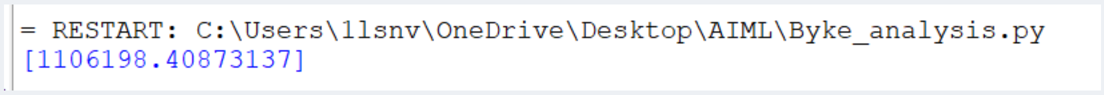
Here we can observe the accuracy is about 89% and mean square error is around 808119621.

CODE:



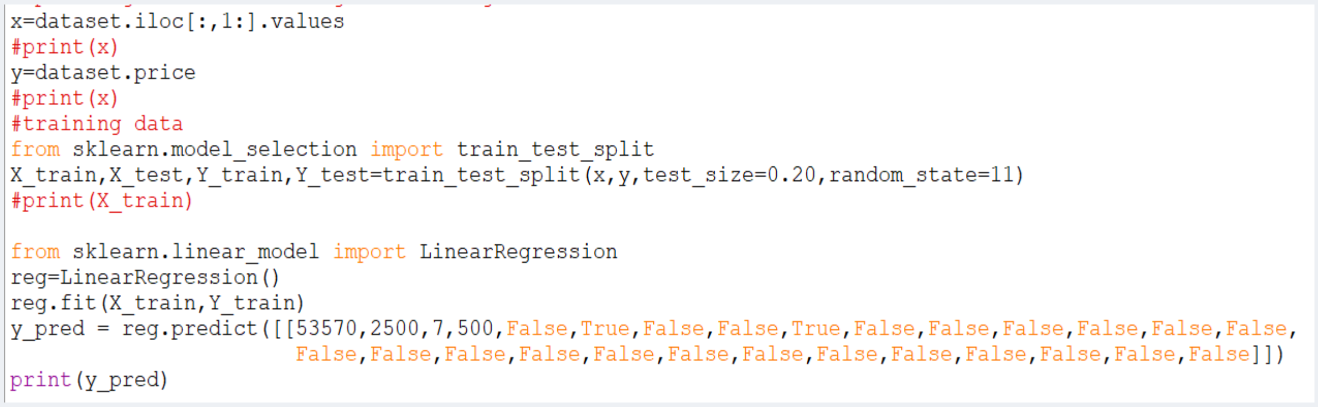
```
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)
#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
#print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred = reg.predict([[53570,2500,9,500,True,False,False,False,True,False,False,False,False,False,False,
                       False,False,False,False,False,False,False,False,False,False,False,False,False]])
print(y_pred)
```

OUTPUT



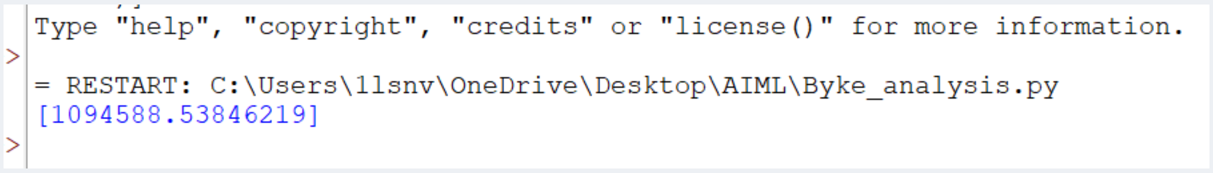
```
= RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py
[1106198.40873137]
```

CODE:



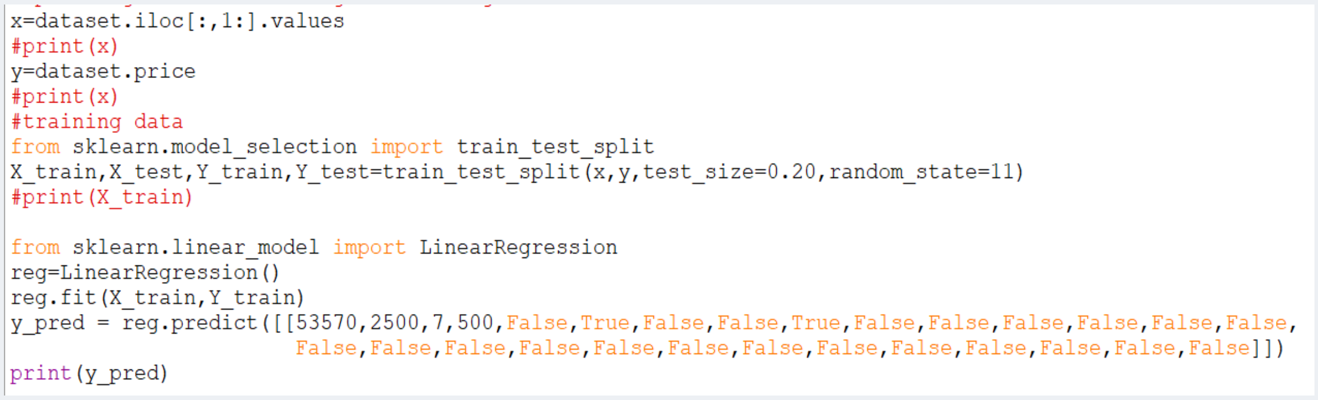
```
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)
#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
#print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred = reg.predict([[53570,2500,7,500,False,True,False,False,True,False,False,False,False,False,False,
                       False,False,False,False,False,False,False,False,False,False,False,False,False]])
print(y_pred)
```

OUTPUT



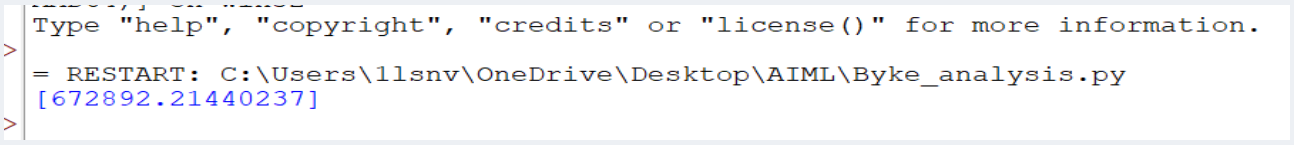
```
  Type "help", "copyright", "credits" or "license()" for more information.
>
  = RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py
  [1094588.53846219]
>
```

CODE:



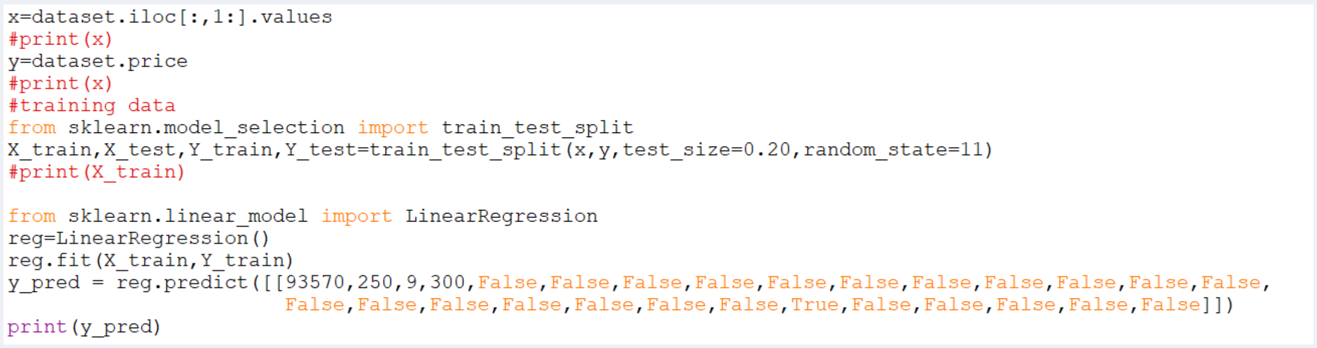
```
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)
#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
#print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred = reg.predict([[53570,2500,7,500,False,True,False,False,True,False,False,False,False,False,False,
                       False,False,False,False,False,False,False,False,False,False,False,False,False]])
print(y_pred)
```

OUTPUT



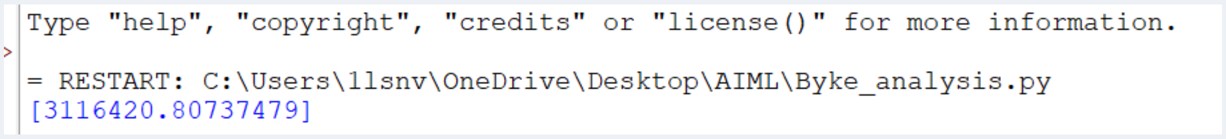
```
 Type "help", "copyright", "credits" or "license()" for more information.
>
 = RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py
 [672892.21440237]
>
```

CODE:



```
x=dataset.iloc[:,1:].values
#print(x)
y=dataset.price
#print(x)
#training data
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20,random_state=11)
#print(X_train)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
y_pred = reg.predict([[93570,250,9,300,False,False,False,False,False,False,False,False,False,False,False,
                       False,False,False,False,False,False,False,True,False,False,False,False,False]])
print(y_pred)
```

OUTPUT



```
 Type "help", "copyright", "credits" or "license()" for more information.
>
 = RESTART: C:\Users\1lsnv\OneDrive\Desktop\AIML\Byke_analysis.py
 [3116420.80737479]
```

FUTURE ENHANCEMENT

Analyzing a used bike dataset using a linear regression model is a valuable application of artificial intelligence and machine learning (AI/ML) with potential for various future advancements and improvements. Here are some future scope areas and potential models that can enhance the accuracy of the analysis:

Non-linear Models: While linear regression is a good starting point, the relationship between bike attributes and price may not always be strictly linear. Future work could involve experimenting with non-linear models like:

Polynomial Regression: This extends linear regression to capture higher-order polynomial relationships.
Decision Trees and **Random Forests**: These models can handle non-linear relationships and interactions between features effectively.

Ensemble Techniques: You can explore ensemble techniques to combine multiple models for improved accuracy. For example:

Gradient Boosting Algorithms (e.g., XGBoost, LightGBM): These algorithms often provide better predictive performance by combining the outputs of multiple weak models.
Stacking: Stacking multiple diverse models and using a meta-learner to make predictions can enhance accuracy.

CONCLUSION

our analysis of the used bike dataset using a linear regression model has provided valuable insights into the factors influencing the resale prices of bicycles. Through this project, we aimed to understand how various features, such as bike age, brand, condition, and mileage, contribute to the pricing of second-hand bikes. Our findings and the model's performance shed light on several key points:

Feature Significance: The linear regression model allowed us to identify the significance of different features. We found that bike age and mileage had a strong negative correlation with the resale price, indicating that older bikes with higher mileage tend to be priced lower.

Brand Influence: Our analysis highlighted that certain bike brands commanded premium prices in the used bike market.