Introduction to Data Science (S1-22_DSECLZG532)-ASSIGNMENT

Group No : IDS Group 168

Group Member Names:

1. BAGAVATH P (2022da04561)
2. PRAHARA D P (2022da04594)
3. NIKITHA V (2022da04721)
4. AADIL AHMAD NENGROO (2022da04705)

1. Business Understanding

Students are expected to identify a data analytics task of your choice. You have to detail the Business Understanding part of your problem under this heading which basically addresses the following questions.

1. What is the business problem that you are trying to solve?
2. What data do you need to answer the above problem?
3. What are the different sources of data?
4. What kind of analytics task are you performing?

Score: 1 Mark in total (0.25 mark each)

```
1. What is the business problem that you are trying to solve?
--The business problem we are trying to solve is to analyze the motorcycle
market in India. We want to understand the current trends and patterns in
the Indian motorcycle market, and determining the most profitable
motorcycle segment in India.

2. What data do you need to answer the above problem?
--We need data of different types of motorcycles available, their features
and prices.

3. What are the different sources of data?
--The data can be gathered from a variety of sources, including government
statistics, industry reports, consumer surveys, and online data sources.

4. What kind of analytics task are you performing?
--The analytics task that we are performing is 'Descriptive Analytics',
which involves using data to describe the current situation in the
motorcycle market in India. We are also using 'Predictive Analytics' to
identify potentially most profitable motorcycle segment in India.
```

2. Data Acquisition

For the problem identified , find an appropriate data set (Your data set must be unique) from any public data source.

2.1 Download the data directly

In [2]:
```
! pip install --trusted-host pypi.org --trusted-host pypi.python.org --truste
```

```
Requirement already satisfied: opendatasets in c:\users\p.puttaiahgowda\anac
onda3\lib\site-packages (0.1.22)
Requirement already satisfied: click in c:\users\p.puttaiahgowda\anaconda3\l
ib\site-packages (from opendatasets) (8.0.4)
Requirement already satisfied: kaggle in c:\users\p.puttaiahgowda\anaconda3
\lib\site-packages (from opendatasets) (1.5.13)
Requirement already satisfied: tqdm in c:\users\p.puttaiahgowda\anaconda3\li
b\site-packages (from opendatasets) (4.64.1)
Requirement already satisfied: colorama in c:\users\p.puttaiahgowda\anaconda
3\lib\site-packages (from click->opendatasets) (0.4.5)
Requirement already satisfied: certifi in c:\users\p.puttaiahgowda\anaconda3
\lib\site-packages (from kaggle->opendatasets) (2022.9.14)
Requirement already satisfied: six>=1.10 in c:\users\p.puttaiahgowda\anacond
a3\lib\site-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: python-slugify in c:\users\p.puttaiahgowda\an
aconda3\lib\site-packages (from kaggle->opendatasets) (5.0.2)
Requirement already satisfied: urllib3 in c:\users\p.puttaiahgowda\anaconda3
\lib\site-packages (from kaggle->opendatasets) (1.26.11)
Requirement already satisfied: requests in c:\users\p.puttaiahgowda\anaconda
3\lib\site-packages (from kaggle->opendatasets) (2.28.1)
Requirement already satisfied: python-dateutil in c:\users\p.puttaiahgowda\a
naconda3\lib\site-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\p.puttaiahgow
da\anaconda3\lib\site-packages (from python-slugify->kaggle->opendatasets)
(1.3)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\p.puttai
ahgowda\anaconda3\lib\site-packages (from requests->kaggle->opendatasets)
(2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\p.puttaiahgowda\anac
onda3\lib\site-packages (from requests->kaggle->opendatasets) (3.3)
```

In [ ]:
```python
import opendatasets as od
dataset  = "https://www.kaggle.com/datasets/yashwanthkumarmn/motorcycles-in-i
od.download(dataset)
```

In [ ]:

```
import os
data_dir = '.\motorcycles-in-india'
os.listdir(data_dir)
```

2.2 Code for converting the above downloaded data into a dataframe

In [3]:

```
##---------Type the code below this line------------------##
import pandas as pd
df = pd.read_csv('bike_dataset.csv')
df
```

Out[3]:

	model_name	price	CC	mileage	type_of_bike	weight_in_kg	
0	Gravton Motors Quanta	99000	NaN	320.0	Electric Bike	100	https://www.carandbike.c m
1	Simple Energy One	109999	NaN	236.0	Electric Bike	110	https://www.carandbike. ene
2	Okaya Classiq	69900	NaN	200.0	Electric Bike	95	https://www.carandbike. b
3	Oben Electric Rorr	102999	NaN	200.0	Electric Bike	120	https://www.carandbike ele
4	Ola Electric S1	85099	NaN	181.0	Electric Bike	121	https://www.carandbi elec
...	...	...	...	...	...	...	
356	Aprilia RSV4	2369000	1099.0	12.0	Petrol Bike	202	https://www.carandbike.
357	Harley-Davidson Sportster S	1551000	1252.0	11.8	Petrol Bike	228	https://www.carandbike. da
358	Suzuki Hayabusa	1640000	1340.0	11.0	Petrol Bike	266	https://www.carandbike. bike
359	Ducati Hypermotard 950	1402278	937.0	9.0	Petrol Bike	176	https://www.carandbike. bike
360	Harley-Davidson CVO Limited	4999000	1923.0	8.0	Petrol Bike	411	https://www.carandbike. da

361 rows × 9 columns

2.3 Confirm the data has been downloaded correctly by displaying the first 5 and last 5 records.

In [4]:
```
##---------Type the code below this line------------------##
#First 5 records
df.head()
```

Out[4]:

	model_name	price	CC	mileage	type_of_bike	weight_in_kg	
0	Gravton Motors Quanta	99000	NaN	320.0	Electric Bike	100	https://www.carandbike.com/gra motors-l
1	Simple Energy One	109999	NaN	236.0	Electric Bike	110	https://www.carandbike.com/si energy-bi
2	Okaya Classiq	69900	NaN	200.0	Electric Bike	95	https://www.carandbike.com/o bikes/c
3	Oben Electric Rorr	102999	NaN	200.0	Electric Bike	120	https://www.carandbike.com/ electric-bi
4	Ola Electric S1	85099	NaN	181.0	Electric Bike	121	https://www.carandbike.cor electric-bik

In [5]:
```
#Last 5 records
df.tail()
```

Out[5]:

	model_name	price	CC	mileage	type_of_bike	weight_in_kg	
356	Aprilia RSV4	2369000	1099.0	12.0	Petrol Bike	202	https://www.carandbike.c
357	Harley-Davidson Sportster S	1551000	1252.0	11.8	Petrol Bike	228	https://www.carandbike.c davi
358	Suzuki Hayabusa	1640000	1340.0	11.0	Petrol Bike	266	https://www.carandbike.c bikes
359	Ducati Hypermotard 950	1402278	937.0	9.0	Petrol Bike	176	https://www.carandbike.c bikes
360	Harley-Davidson CVO Limited	4999000	1923.0	8.0	Petrol Bike	411	https://www.carandbike.c davi

2.4 Display the column headings, statistical information, description and statistical summary of the data.

In [6]:
```
##---------Type the code below this line------------------##
##Column Headings
df.columns.tolist()
```

Out[6]:
```
['model_name',
 'price',
 'CC',
 'mileage',
 'type_of_bike',
 'weight_in_kg',
 'links',
 'acceleration_speed',
 'top_speed']
```

In [7]:
```
df.shape
```

Out[7]:
```
(361, 9)
```

In [8]:
```
df.describe()
```

Out[8]:

	price	CC	mileage	weight_in_kg	acceleration_speed	top_speed
count	3.610000e+02	304.000000	361.000000	361.000000	170.000000	200.000000
mean	8.399079e+05	680.973684	44.681413	178.839335	4.193412	99.338650
std	1.052083e+06	547.744364	39.890270	73.839516	2.369334	39.631992
min	3.800000e+04	87.800000	8.000000	55.000000	1.010000	25.000000
25%	1.000000e+05	164.425000	20.000000	118.000000	2.800000	79.500000
50%	2.420000e+05	618.000000	30.000000	169.000000	3.215000	100.000000
75%	1.459000e+06	1051.500000	55.000000	216.000000	5.075000	129.115000
max	7.990000e+06	2458.000000	320.000000	433.000000	13.800000	200.000000

In [9]:
```
##Statistical Description for all columns
df.describe(include='all')
```

Out[9]:

	model_name	price	CC	mileage	type_of_bike	weight_in_kg	
count	361	3.610000e+02	304.000000	361.000000	361	361.000000	
unique	361	NaN	NaN	NaN	2	NaN	
top	Gravton Motors Quanta	NaN	NaN	NaN	Petrol Bike	NaN	https:/
freq	1	NaN	NaN	NaN	304	NaN	
mean	NaN	8.399079e+05	680.973684	44.681413	NaN	178.839335	
std	NaN	1.052083e+06	547.744364	39.890270	NaN	73.839516	
min	NaN	3.800000e+04	87.800000	8.000000	NaN	55.000000	
25%	NaN	1.000000e+05	164.425000	20.000000	NaN	118.000000	
50%	NaN	2.420000e+05	618.000000	30.000000	NaN	169.000000	
75%	NaN	1.459000e+06	1051.500000	55.000000	NaN	216.000000	
max	NaN	7.990000e+06	2458.000000	320.000000	NaN	433.000000	

In [10]:
```
## Information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 361 entries, 0 to 360
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   model_name          361 non-null    object 
 1   price               361 non-null    int64  
 2   CC                  304 non-null    float64
 3   mileage             361 non-null    float64
 4   type_of_bike        361 non-null    object 
 5   weight_in_kg        361 non-null    int64  
 6   links               361 non-null    object 
 7   acceleration_speed  170 non-null    float64
 8   top_speed           200 non-null    float64
dtypes: float64(4), int64(2), object(3)
memory usage: 25.5+ KB
```

2.5 Write your observations from the above.

1. Size of the dataset
2. What type of data attributes are there?
3. Is there any null data that has to be cleaned?

Score: 2 Marks in total (0.25 marks for 2.1, 0.25 marks for 2.2, 0.5 marks for 2.3, 0.25 marks for 2.4, 0.75 marks for 2.5)

```
--------------Type the answers below this line--------------
1. Size of the dataset: The dataset contains 645 rows and 8 columns.

2. Type of data attributes: The dataset contains numerical and categorical
data attributes.

3. Null data that has to be cleaned: Yes, there are some null values
present in the dataset which has to be cleaned.
```

3. Data Preparation

```
If input data is numerical or categorical, do 3.1, 3.2 and 3.4
If input data is text, do 3.3 and 3.4
```

3.1 Check for

- duplicate data
- missing data
- data inconsistencies

In [11]:
```
##---------Type the code below this line------------------##
# Check for duplicate data
duplicates = df.duplicated()
print(df[duplicates])
```

```
Empty DataFrame
Columns: [model_name, price, CC, mileage, type_of_bike, weight_in_kg, links,
acceleration_speed, top_speed]
Index: []
```

In [12]:
```
# Missing Data

# Print out the missing rows
print(df[df.isnull().any(axis=1)])
```

```
                       model_name    price      CC  mileage   type_of_bike  \
0            Gravton Motors Quanta    99000     NaN    320.0  Electric Bike
1                Simple Energy One   109999     NaN    236.0  Electric Bike
2                    Okaya Classiq    69900     NaN    200.0  Electric Bike
3              Oben Electric Rorr    102999     NaN    200.0  Electric Bike
4                  Ola Electric S1    85099     NaN    181.0  Electric Bike
..                             ...      ...     ...      ...            ...
356                   Aprilia RSV4  2369000  1099.0     12.0    Petrol Bike
357  Harley-Davidson Sportster S   1551000  1252.0     11.8    Petrol Bike
358                 Suzuki Hayabusa  1640000  1340.0     11.0    Petrol Bike
359        Ducati Hypermotard 950  1402278   937.0      9.0    Petrol Bike
360  Harley-Davidson CVO Limited   4999000  1923.0      8.0    Petrol Bike

     weight_in_kg                                              links  \
0             100  https://www.carandbike.com/gravton-motors-bike... (https://www.carandbike.com/gravton-motors-bike...)
1             110  https://www.carandbike.com/simple-energy-bikes... (https://www.carandbike.com/simple-energy-bikes...)
2              95     https://www.carandbike.com/okaya-bikes/classiq (https://www.carandbike.com/okaya-bikes/classiq)
3             120  https://www.carandbike.com/oben-electric-bikes... (https://www.carandbike.com/oben-electric-bikes...)
4             121   https://www.carandbike.com/ola-electric-bikes/s1 (https://www.carandbike.com/ola-electric-bikes/s1)
..            ...                                                ...
356           202      https://www.carandbike.com/aprilia-bikes/rsv4 (https://www.carandbike.com/aprilia-bikes/rsv4)
357           228  https://www.carandbike.com/harley-davidson-bik... (https://www.carandbike.com/harley-davidson-bik...)
358           266   https://www.carandbike.com/suzuki-bikes/hayabusa (https://www.carandbike.com/suzuki-bikes/hayabusa)
359           176  https://www.carandbike.com/ducati-bikes/hyperm... (https://www.carandbike.com/ducati-bikes/hyperm...)
360           411  https://www.carandbike.com/harley-davidson-bik... (https://www.carandbike.com/harley-davidson-bik...)

     acceleration_speed  top_speed
0                   4.2       70.0
1                   3.6      100.0
2                   NaN       25.0
3                   3.0      100.0
4                   2.9      116.0
..                  ...        ...
356                 NaN        NaN
357                 NaN        NaN
358                 NaN        NaN
359                 NaN        NaN
360                 NaN        NaN

[223 rows x 9 columns]
```

In [13]:
```
#data inconsistencies - removing the column with links, since it doesn't play
# drop the column 'Links'
df.drop(['links','CC'], inplace=True, axis=1)
df
```

Out[13]:

	model_name	price	mileage	type_of_bike	weight_in_kg	acceleration_speed	top_speed
0	Gravton Motors Quanta	99000	320.0	Electric Bike	100	4.2	70.0
1	Simple Energy One	109999	236.0	Electric Bike	110	3.6	100.0
2	Okaya Classiq	69900	200.0	Electric Bike	95	NaN	25.0
3	Oben Electric Rorr	102999	200.0	Electric Bike	120	3.0	100.0
4	Ola Electric S1	85099	181.0	Electric Bike	121	2.9	116.0
...	...	...	...	...	...	...	...
356	Aprilia RSV4	2369000	12.0	Petrol Bike	202	NaN	NaN
357	Harley-Davidson Sportster S	1551000	11.8	Petrol Bike	228	NaN	NaN
358	Suzuki Hayabusa	1640000	11.0	Petrol Bike	266	NaN	NaN
359	Ducati Hypermotard 950	1402278	9.0	Petrol Bike	176	NaN	NaN
360	Harley-Davidson CVO Limited	4999000	8.0	Petrol Bike	411	NaN	NaN

361 rows × 7 columns

3.2 Apply techiniques

- to remove duplicate data
- to impute or remove missing data
- to remove data inconsistencies

In [14]:
```
##---------Type the code below this line------------------##
#remove duplicate data
df = df.drop_duplicates()
```

In [15]:
```
#remove duplicate data
df.drop_duplicates(keep = False, inplace = True)
print(df)
# Print the shape of the dataset
print("Shape of dataset after removing duplicate data:", df.shape)
```

```
                      model_name    price  mileage   type_of_bike  \
0          Gravton Motors Quanta    99000    320.0  Electric Bike
1              Simple Energy One   109999    236.0  Electric Bike
2                  Okaya Classiq    69900    200.0  Electric Bike
3             Oben Electric Rorr   102999    200.0  Electric Bike
4                Ola Electric S1    85099    181.0  Electric Bike
..                           ...      ...      ...            ...
356                Aprilia RSV4  2369000     12.0    Petrol Bike
357  Harley-Davidson Sportster S  1551000     11.8    Petrol Bike
358              Suzuki Hayabusa  1640000     11.0    Petrol Bike
359         Ducati Hypermotard 950  1402278      9.0    Petrol Bike
360  Harley-Davidson CVO Limited  4999000      8.0    Petrol Bike

     weight_in_kg  acceleration_speed  top_speed
0             100                 4.2       70.0
1             110                 3.6      100.0
2              95                 NaN       25.0
3             120                 3.0      100.0
4             121                 2.9      116.0
..            ...                 ...        ...
356           202                 NaN        NaN
357           228                 NaN        NaN
358           266                 NaN        NaN
359           176                 NaN        NaN
360           411                 NaN        NaN

[361 rows x 7 columns]
Shape of dataset after removing duplicate data: (361, 7)
```

In [16]:
```
#removing missing data
df.dropna(inplace=True)
```

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
model_name            0
price                 0
mileage               0
type_of_bike          0
weight_in_kg          0
acceleration_speed    0
top_speed             0
dtype: int64
```

3.3 Encode categorical data

In [18]:
```
# ---------Type the code below this line------------------##
df.dtypes
```

Out[18]:
```
model_name              object
price                    int64
mileage                float64
type_of_bike            object
weight_in_kg             int64
acceleration_speed     float64
top_speed              float64
dtype: object
```

In [19]:
```
#dummies = pd.get_dummies(df.type_of_bike)
#df = pd.concat([df, dummies], axis=1)
#df.drop(df.columns[7:],axis=1,inplace=True)
#df
one_hot_encoded_data = pd.get_dummies(df, columns = ['type_of_bike'])
print(one_hot_encoded_data)
df=one_hot_encoded_data.copy()
df.drop(df.columns[8:],axis=1,inplace=True)
df
```

```
                        model_name   price  mileage  weight_in_kg  \
0              Gravton Motors Quanta   99000    320.0           100
1                  Simple Energy One  109999    236.0           110
3                 Oben Electric Rorr  102999    200.0           120
4                    Ola Electric S1   85099    181.0           121
5                Ola Electric S1 Pro  120149    181.0           125
..                               ...     ...      ...           ...
195                    Kawasaki Z250  308000     26.0           168
196               Kawasaki Ninja 300  337000     26.0           179
197                FB Mondial HPS 300  337000     26.0           135
198     Royal Enfield Interceptor 650  281518     25.5           202
199  Royal Enfield Continental GT 650  298079     25.5           198

     acceleration_speed  top_speed  type_of_bike_Electric Bike  \
0                  4.20       70.0                           1
1                  3.60      100.0                           1
3                  3.00      100.0                           1
4                  2.90      116.0                           1
5                  2.90      116.0                           1
..                  ...        ...                         ...
195                1.01      200.0                           0
196                2.01      182.0                           0
197                3.00      141.0                           0
198                1.90      170.0                           0
199                1.79      161.0                           0

     type_of_bike_Petrol Bike
0                           0
1                           0
3                           0
4                           0
5                           0
..                        ...
195                         1
196                         1
197                         1
198                         1
199                         1

[170 rows x 8 columns]
```

Out[19]:

	model_name	price	mileage	weight_in_kg	acceleration_speed	top_speed	type_of_bike_
0	Gravton Motors Quanta	99000	320.0	100	4.20	70.0	
1	Simple Energy One	109999	236.0	110	3.60	100.0	
3	Oben Electric Rorr	102999	200.0	120	3.00	100.0	
4	Ola Electric S1	85099	181.0	121	2.90	116.0	
5	Ola Electric S1 Pro	120149	181.0	125	2.90	116.0	
...	...	...	...	...	...	...	
195	Kawasaki Z250	308000	26.0	168	1.01	200.0	
196	Kawasaki Ninja 300	337000	26.0	179	2.01	182.0	
197	FB Mondial HPS 300	337000	26.0	135	3.00	141.0	
198	Royal Enfield Interceptor 650	281518	25.5	202	1.90	170.0	
199	Royal Enfield Continental GT 650	298079	25.5	198	1.79	161.0	

170 rows × 8 columns

In [20]:
```
list(df)
```

Out[20]:
```
['model_name',
 'price',
 'mileage',
 'weight_in_kg',
 'acceleration_speed',
 'top_speed',
 'type_of_bike_Electric Bike',
 'type_of_bike_Petrol Bike']
```

3.4 Text data

1. Remove special characters
2. Change the case (up-casing and down-casing).
3. Tokenization — process of discretizing words within a document.
4. Filter Stop Words.

```
# ---------Type the code below this line------------------##

There is no text data in the analysis.
```

3.4 Report

Mention and justify the method adopted

- to remove duplicate data, if present
- to impute or remove missing data, if present
- to remove data inconsistencies, if present

OR for textdata

- How many tokens after step 3?
- how may tokens after stop words filtering?

If the any of the above are not present, then also add in the report below.

Score: 2 Marks (based on the dataset you have, the data prepreation you had to do and report typed, marks will be distributed between 3.1, 3.2, 3.3 and 3.4)

```
##---------Type the code below this line------------------##
The method adopted to remove duplicate data is to use the
'drop_duplicates()' feature from Pandas as it can be used to detect and
remove duplicate rows.
Syntax:  DataFrame.drop_duplicates(subset=None, keep='first',
inplace=False)
```

```
The method adopted to impute or remove missing data is to use the 'fillna'
feature from Pandas. This feature can be used to replace missing data with
a given value. This feature can also be used to remove missing data
altogether by setting the missing values to 'NaN' and then dropping those
rows or columns.
```

```
since we just wanted to remove the column with links and no other data
inconsistencies was there no explicit feature or method is used.
```

3.5 Identify the target variables.

- Separate the data from the target such that the dataset is in the form of (X,y) or (Features, Label)
- Discretize / Encode the target variable or perform one-hot encoding on the target or any other as and if required.
- Report the observations

Score: 1 Mark

Our target variable is "Price"

In [21]:
```
X =df.drop(['price'], axis=1)
y =df.price

#Creating dataset in the form of (X,y)
df1 = (X,y)
print(df1)
```

```
(                          model_name  mileage  weight_in_kg  \
0              Gravton Motors Quanta    320.0           100   
1                  Simple Energy One    236.0           110   
3                 Oben Electric Rorr    200.0           120   
4                    Ola Electric S1    181.0           121   
5                Ola Electric S1 Pro    181.0           125   
..                               ...      ...           ...   
195                    Kawasaki Z250     26.0           168   
196               Kawasaki Ninja 300     26.0           179   
197               FB Mondial HPS 300     26.0           135   
198     Royal Enfield Interceptor 650     25.5           202   
199  Royal Enfield Continental GT 650     25.5           198   

     acceleration_speed  top_speed  type_of_bike_Electric Bike  \
0                  4.20       70.0                           1   
1                  3.60      100.0                           1   
3                  3.00      100.0                           1   
4                  2.90      116.0                           1   
5                  2.90      116.0                           1   
..                  ...        ...                         ...   
195                1.01      200.0                           0   
196                2.01      182.0                           0   
197                3.00      141.0                           0   
198                1.90      170.0                           0   
199                1.79      161.0                           0   

     type_of_bike_Petrol Bike  
0                           0  
1                           0  
3                           0  
4                           0  
5                           0  
..                        ...  
195                         1  
196                         1  
197                         1  
198                         1  
199                         1  

[170 rows x 7 columns], 0       99000
1      109999
3      102999
4       85099
5      120149
        ...  
195    308000
196    337000
197    337000
198    281518
199    298079
Name: price, Length: 170, dtype: int64)
```

```
we need not encode target variable "price" as it a numerical data not a
categorical or nominal one. So one-hot encoding is not needed.
```

```
The above are the features columns we need to analyze with target.
```

4. Data Exploration using various plots

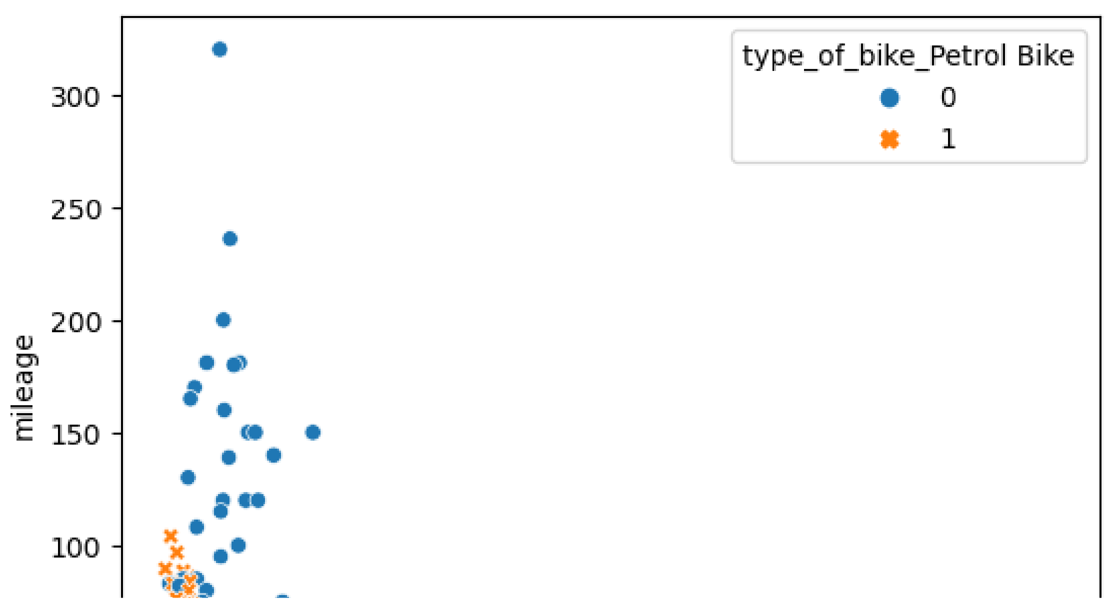
4.1 Scatter plot of each quantitative attribute with the target.

Score: 1 Mark

```
A scatter plot is a type of data visualization that can be used to explore
the relationship between two variables, such as two wheeler price, CC,
mileage, weight_in_kg, acceleration_speed, and top_speed. A scatter plot
can help to identify trends and correlations between different variables
and can help to uncover relationships that might not be immediately
obvious. It can also be used to identify outliers, which can help to inform
decisions about pricing, design and performance.
```

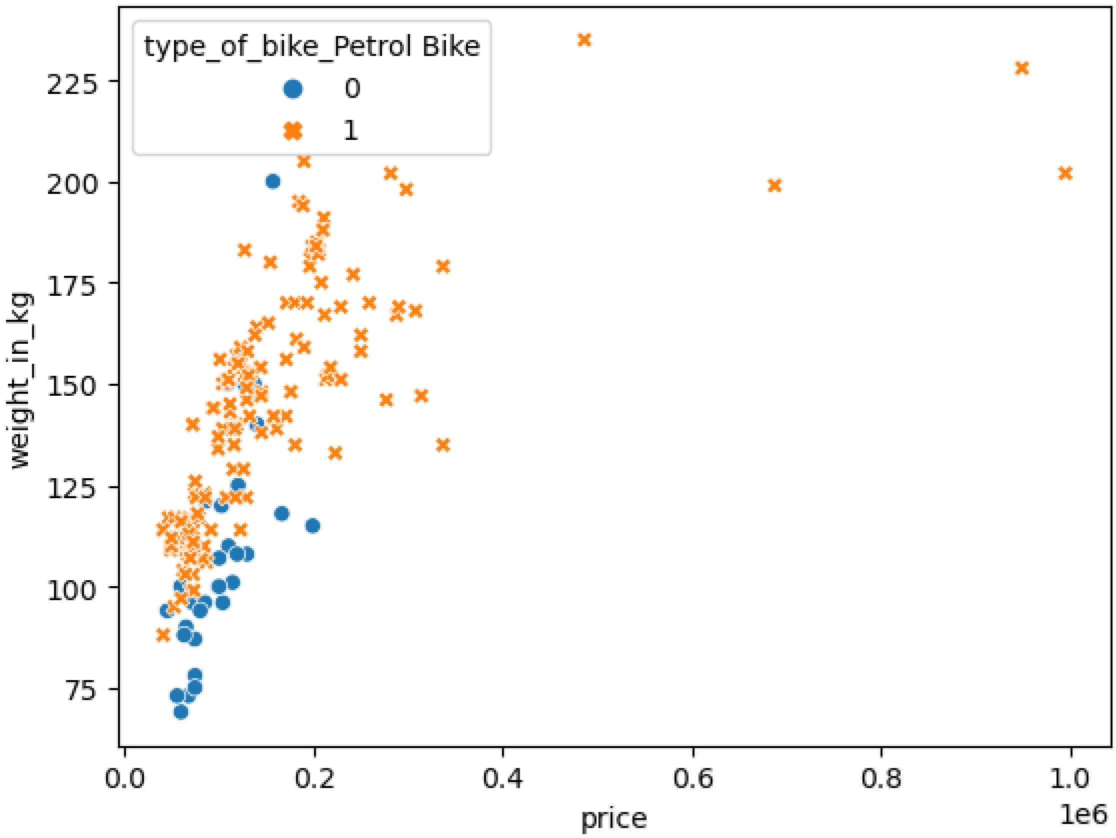
In [22]:
```
##---------Type the code below this line------------------##
# importing packages
import matplotlib.pyplot as plt
import seaborn as sns

sns.scatterplot(x="price", y="mileage", data=df, hue = "type_of_bike_Petrol B
```

Out[22]: `<AxesSubplot:xlabel='price', ylabel='mileage'>`



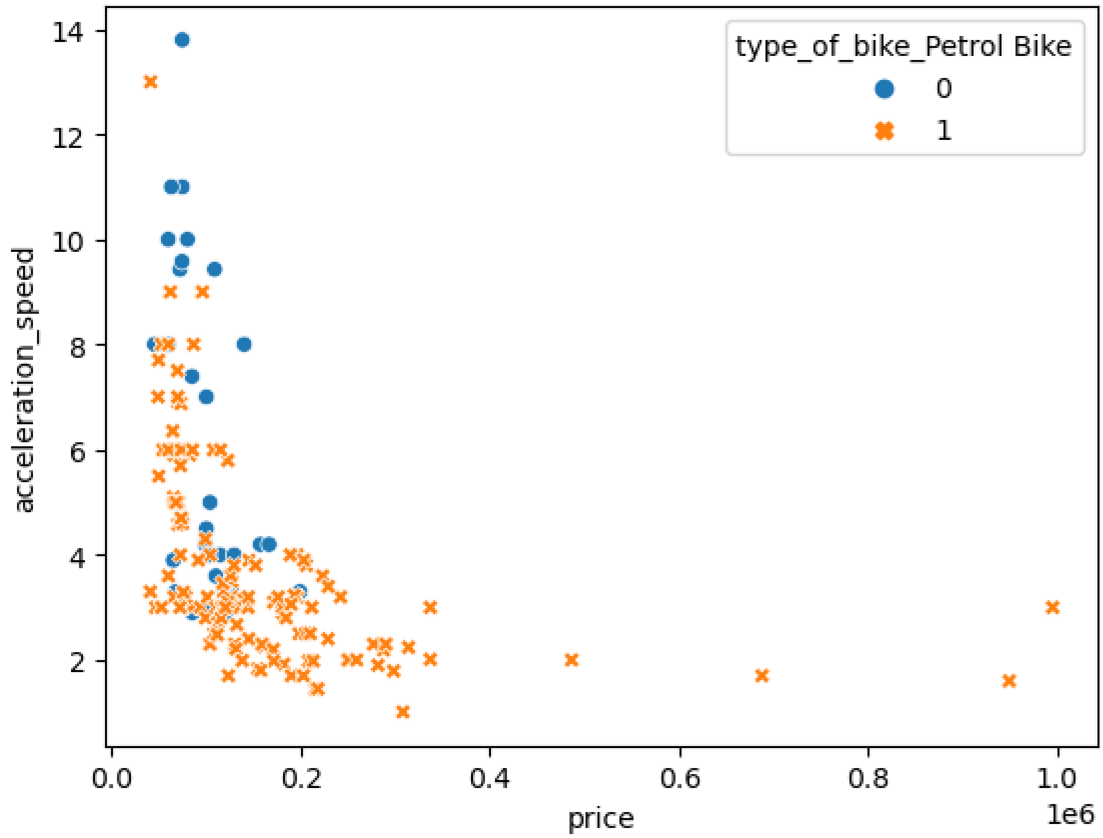
In [23]:
```
sns.scatterplot(x="price", y="weight_in_kg", data=df, hue = "type_of_bike_Pet
```

Out[23]:
```
<AxesSubplot:xlabel='price', ylabel='weight_in_kg'>
```



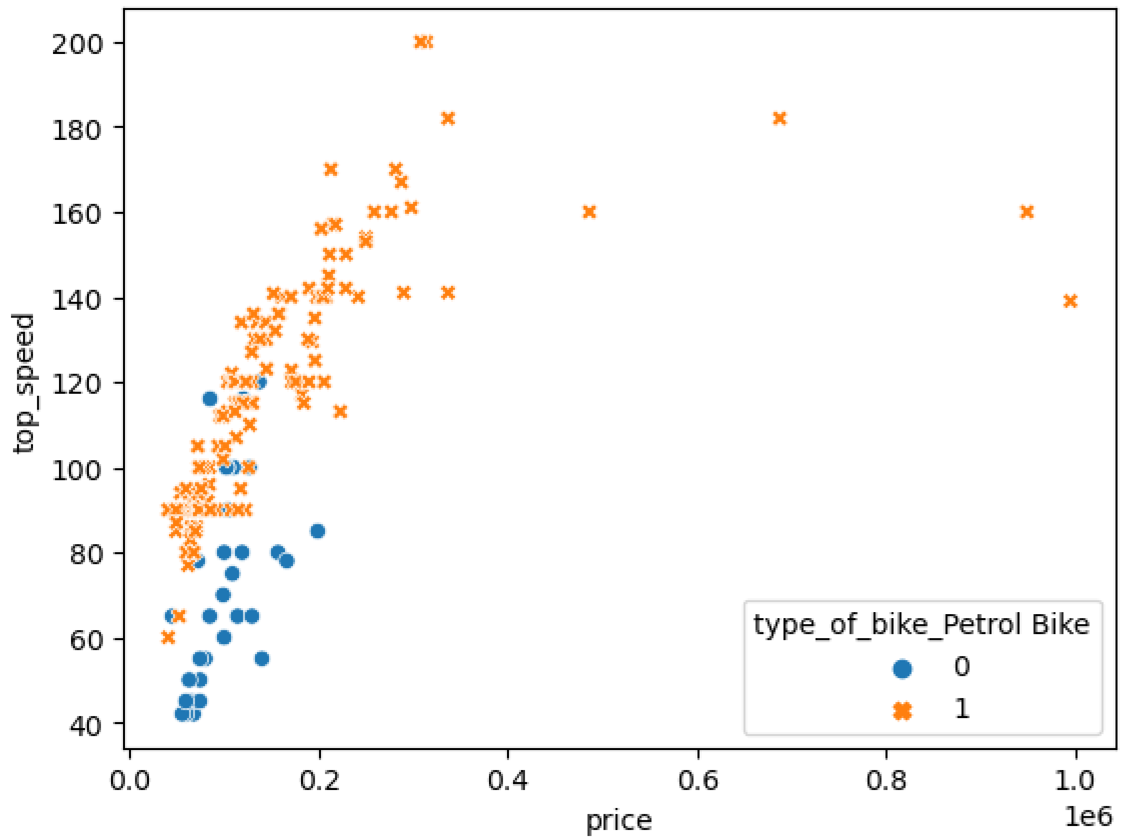
In [24]:
```
sns.scatterplot(x="price", y="acceleration_speed", data=df, hue = "type_of_bi
```

Out[24]: <AxesSubplot:xlabel='price', ylabel='acceleration_speed'>



In [25]:
```
sns.scatterplot(x="price", y="top_speed", data=df, hue = "type_of_bike_Petrol
```

Out[25]:
```
<AxesSubplot:xlabel='price', ylabel='top_speed'>
```



4.2 EDA using visuals

- Use (minimum) 2 plots (pair plot, heat map, correlation plot, regression plot...) to identify the optimal set of attributes that can be used for classification.
- Name them, explain why you think they can be helpful in the task and perform the plot as well. Unless proper justification for the choice of plots given, no credit will be awarded.

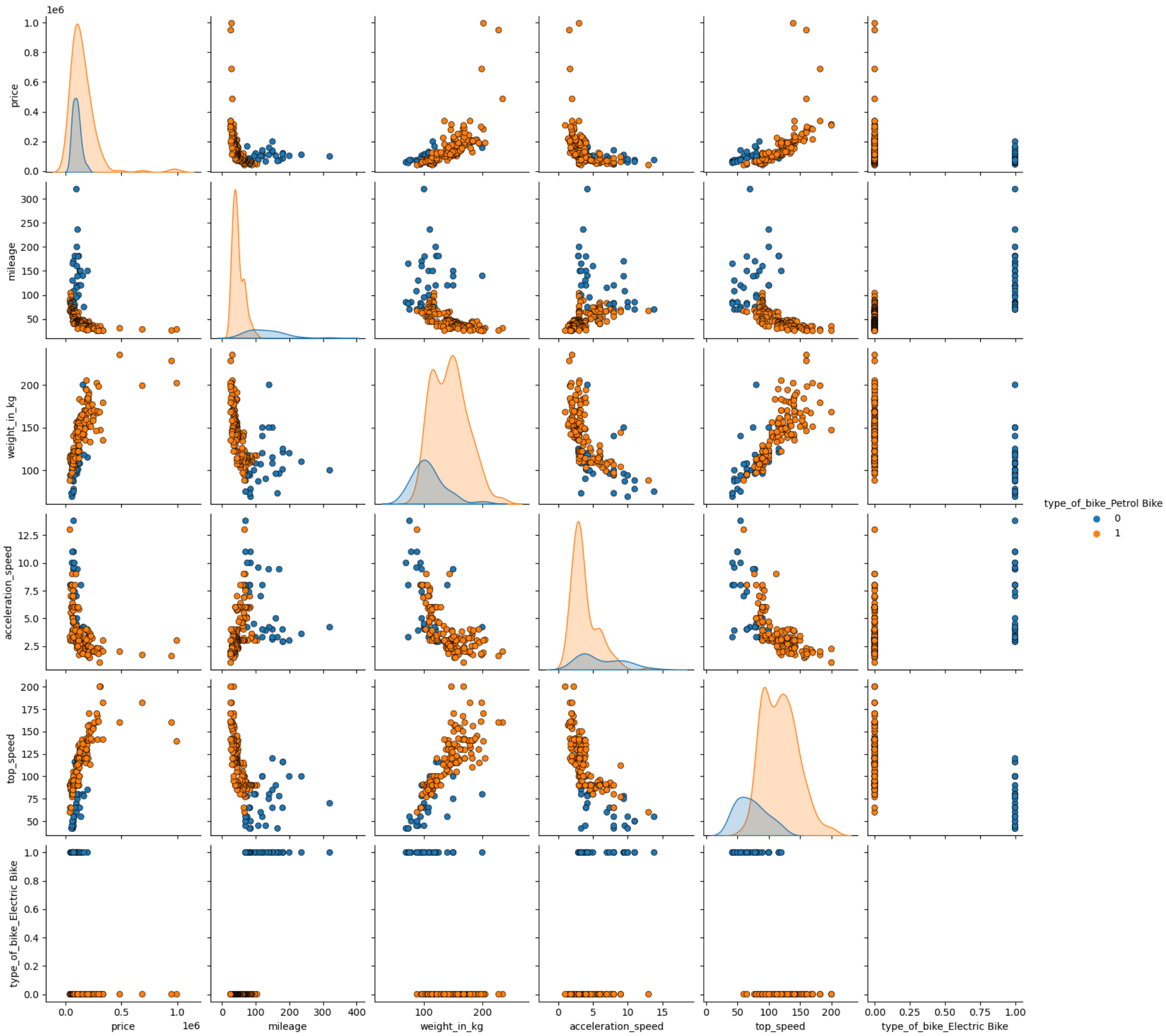
Score: 2 Marks

Pair Plot - Why?:

```
A pair plot is a type of visualization that allows you to quickly vis
ualize relationships between multiple variables in a dataset. When ap
plied to two wheeler prices, CC, mileage, weight_in_kg, acceleration_
speed, and top_speed, a pair plot can help you identify patterns betw
een the different variables. For example, you may be able to see that
the weight_in_kg and top_speed are correlated, or that the CC and mil
eage are correlated.
```

Knowing these relationships can help you better understand how different variables affect the two wheeler price.

In [26]:
```
##---------Type the code below this line------------------##
#PairPlots
sns.pairplot(df, hue = 'type_of_bike_Petrol Bike', plot_kws = { 'edgecolor':
plt.show()
```

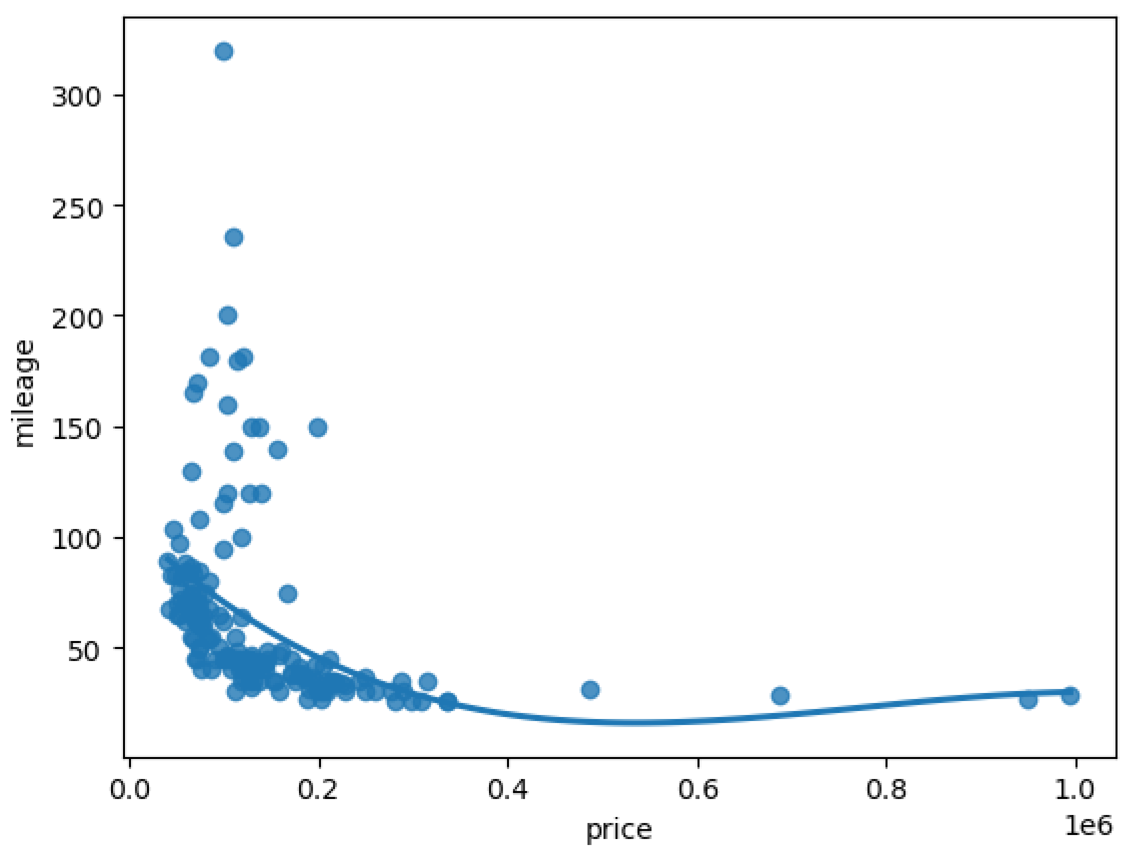


Regression plots - Why?

Regression Plots are used to visualize the relationship between two o
r more numerical variables. In this case, the regression plot can be
used to visualize the relationship between the two wheeler price and
the other variables such as CC, mileage, type of vehicle, weight in k
g, acceleration speed, and top speed. The regression plot can help id
entify any linear or non-linear relationships between the two wheeler
price and the other variables and can also provide insights into whic
h of the variables have a greater impact on the two wheeler price.
Here The order parameter is used to control the order of polynomial r
egression, which can be used to fit nonlinear relationships between t
he variables. By changing the order parameter, it is possible to fit
more complex models with higher accuracy. This can help in better und
erstanding the underlying relationships between the variables.
Seaborn's CI (confidence interval) regression plots are used to visua
lize the uncertainty in a regression line. They show the 95% confiden
ce interval around the regression line, which is a measure of the unc
ertainty in the regression line. This can help identify areas of high
uncertainty or areas where the regression line may not be a good fit
for the data.

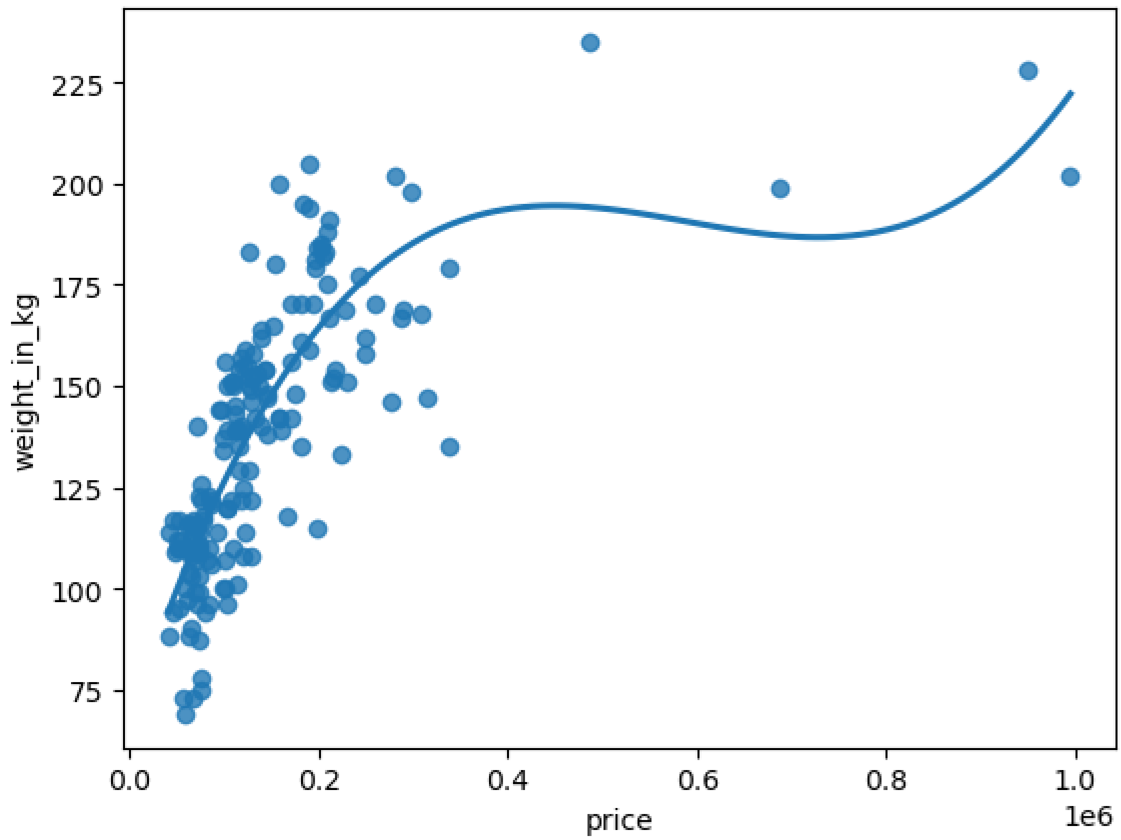
In [27]:
```
sns.regplot(x="price", y="mileage", data=df, order=3, ci=None)
```

Out[27]: <AxesSubplot:xlabel='price', ylabel='mileage'>



In [28]:
```
sns.regplot(x="price", y="weight_in_kg", data=df, order=3, ci=None)
```

Out[28]: <AxesSubplot:xlabel='price', ylabel='weight_in_kg'>

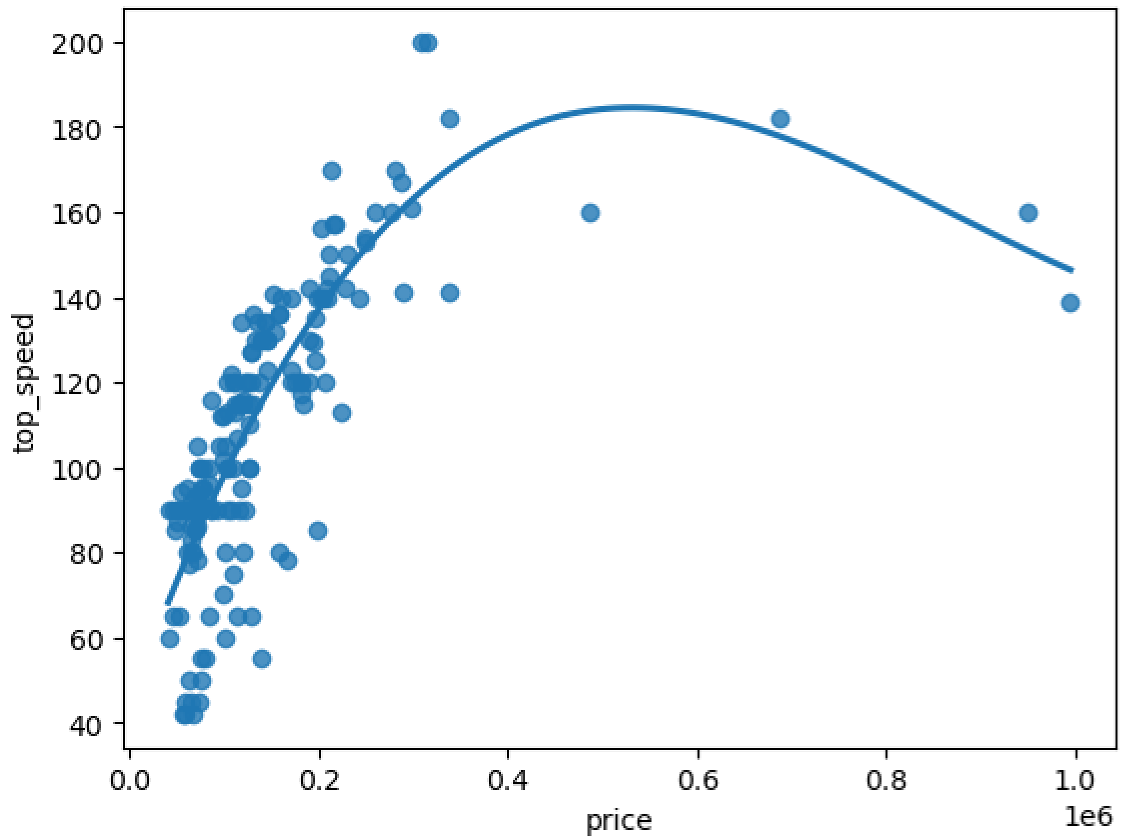


In [29]:
```
sns.regplot(x="price", y="acceleration_speed", data=df, order=3, ci=None)
```

Out[29]: <AxesSubplot:xlabel='price', ylabel='acceleration_speed'>

In [30]:
```
sns.regplot(x="price", y="top_speed", data=df, order=3, ci=None)
```

Out[30]: <AxesSubplot:xlabel='price', ylabel='top_speed'>



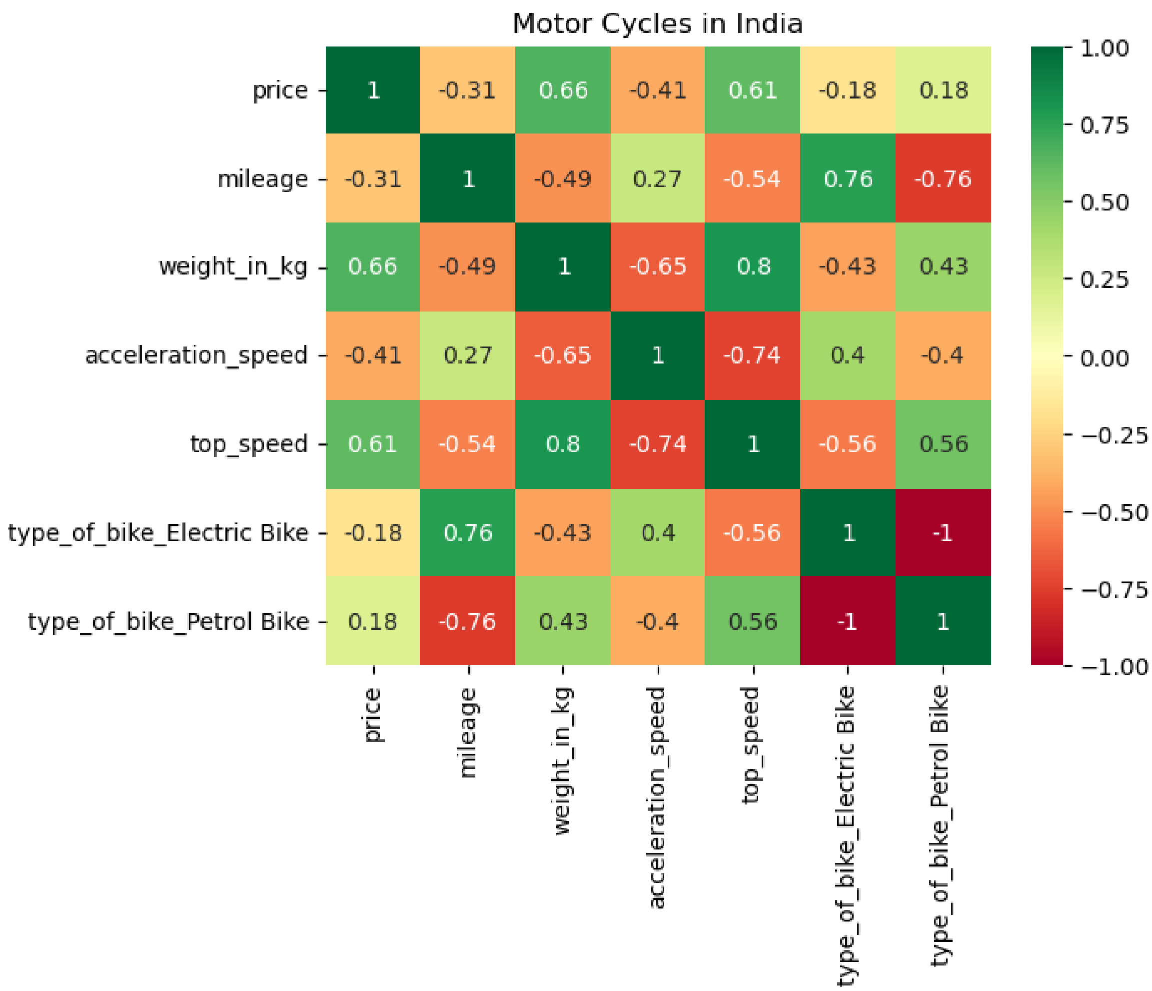
In [31]:
```
##Finding the correlation
correlation = df.corr()
correlation
```

Out[31]:

	price	mileage	weight_in_kg	acceleration_speed	top_speed	type_o
price	1.000000	-0.305958	0.662138	-0.410462	0.609977	
mileage	-0.305958	1.000000	-0.490213	0.269118	-0.538411	
weight_in_kg	0.662138	-0.490213	1.000000	-0.654655	0.802899	
acceleration_speed	-0.410462	0.269118	-0.654655	1.000000	-0.736749	
top_speed	0.609977	-0.538411	0.802899	-0.736749	1.000000	
type_of_bike_Electric Bike	-0.177314	0.759700	-0.432894	0.402208	-0.556962	
type_of_bike_Petrol Bike	0.177314	-0.759700	0.432894	-0.402208	0.556962	

In [32]:
```
#Heat Map for finding corrleation between features

plot = sns.heatmap(correlation,cmap="RdYlGn", annot= True)
plot.set_title("Motor Cycles in India")
```

Out[32]: Text(0.5, 1.0, 'Motor Cycles in India')



Heat Map - why?

From the above correlation analysis and heatmap, it is clear that none of the features/columns are highly co-related to each other. Basically By using heat map we can find the correlated features.

5. Data Wrangling

5.1 Univariate Filters

Numerical and Categorical Data

- Identify top 5 significant features by evaluating each feature independently with respect to the target variable by exploring

1. Mutual Information (Information Gain)
2. Gini index
3. Gain Ratio
4. Chi-Squared test
5. Fisher Score (From the above 5 you are required to use only any **two**)

For Text data

1. Stemming / Lemmatization.
2. Forming n-grams and storing them in the document vector.
3. TF-IDF (From the above 2 you are required to use only any **two**)

Score: 3 Marks

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 170 entries, 0 to 199
Data columns (total 8 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   model_name                  170 non-null    object 
 1   price                       170 non-null    int64  
 2   mileage                     170 non-null    float64
 3   weight_in_kg                170 non-null    int64  
 4   acceleration_speed          170 non-null    float64
 5   top_speed                   170 non-null    float64
 6   type_of_bike_Electric Bike  170 non-null    uint8  
 7   type_of_bike_Petrol Bike    170 non-null    uint8  
dtypes: float64(3), int64(2), object(1), uint8(2)
memory usage: 13.7+ KB
```

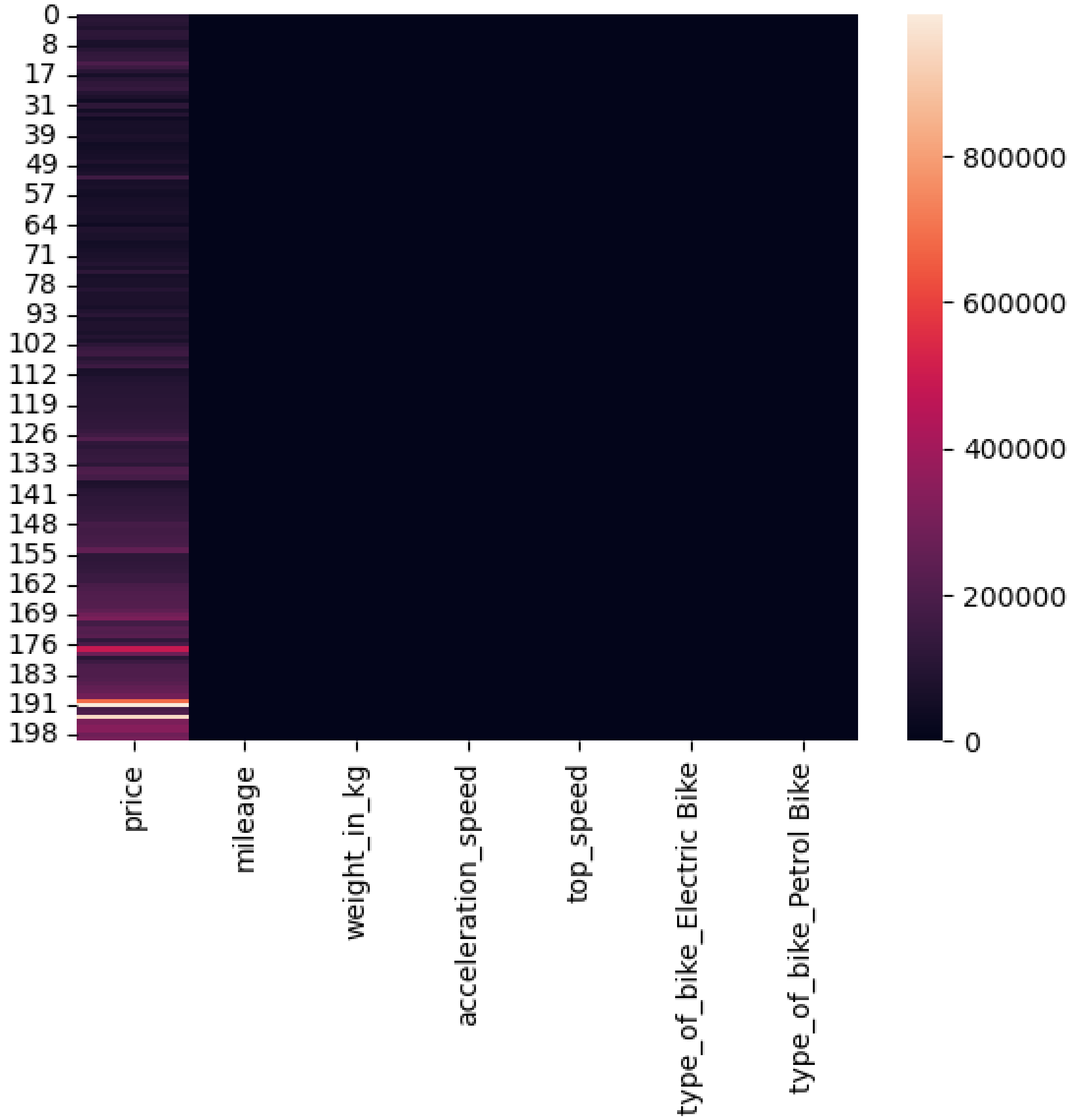
In [34]:
```
feature_sel =df.copy()
feature_sel.drop(['model_name'], axis=1, inplace=True)
feature_sel
```

Out[34]:

	price	mileage	weight_in_kg	acceleration_speed	top_speed	type_of_bike_Electric Bike	type_
0	99000	320.0	100	4.20	70.0	1	
1	109999	236.0	110	3.60	100.0	1	
3	102999	200.0	120	3.00	100.0	1	
4	85099	181.0	121	2.90	116.0	1	
5	120149	181.0	125	2.90	116.0	1	
...	...	...	...	...	...	...	
195	308000	26.0	168	1.01	200.0	0	
196	337000	26.0	179	2.01	182.0	0	
197	337000	26.0	135	3.00	141.0	0	
198	281518	25.5	202	1.90	170.0	0	
199	298079	25.5	198	1.79	161.0	0	

170 rows × 7 columns

In [35]:
```
ax = sns.heatmap(feature_sel)
```



In [36]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import mutual_info_classif
```

In [37]:
```
feature_sel.head()
```

Out[37]:

	price	mileage	weight_in_kg	acceleration_speed	top_speed	type_of_bike_Electric Bike	type_of
0	99000	320.0	100	4.2	70.0	1	
1	109999	236.0	110	3.6	100.0	1	
3	102999	200.0	120	3.0	100.0	1	
4	85099	181.0	121	2.9	116.0	1	
5	120149	181.0	125	2.9	116.0	1	

In [38]:
```
def show_top_univariate_filters(X, y, func, top_k):
    #apply SelectKBest class to extract top k best features
    bestfeatures = SelectKBest(score_func=func, k=top_k)
    fit = bestfeatures.fit(X,y)

    dfscores = pd.DataFrame(fit.scores_)
    dfcolumns = pd.DataFrame(X.columns)

    #concat two dataframes for better visualization
    featureScores = pd.concat([dfcolumns,dfscores],axis=1)
    featureScores.columns = ['Specs','Score']  #naming the dataframe columns
    print(featureScores.nlargest(top_k,'Score'))  #print 10 best features
```

In [39]:
```
#Chi-Squared test
show_top_univariate_filters(feature_sel, y, chi2, 5)
```

```
                Specs         Score
0               price  1.765914e+07
1             mileage  4.986263e+03
4           top_speed  1.566767e+03
2        weight_in_kg  1.298425e+03
3  acceleration_speed  2.233312e+02
```

In [40]:
```
 # Mutual Information
show_top_univariate_filters(feature_sel, y, mutual_info_classif, 5)
```

```
                     Specs     Score
0                    price  0.616667
6   type_of_bike_Petrol Bike  0.588889
4                top_speed  0.380556
3       acceleration_speed  0.213889
1                  mileage  0.130556
```

5.2 Report observations

Write your observations from the results of each method. Clearly justify your choice of the method.

Score 1 mark

```
##---------Type the code below this line------------------##
From the two methods which we have choosen, we have got five important
features:
    price,
    mileage,
    top_speed,
    weight_in_kg,
    acceleration_speed
where this features help us to understand the current trends and patterns
in the Indian motorcycle business.
```

6. Implement Machine Learning Techniques

Use any 2 ML algorithms

1. Classification -- Decision Tree classifier
2. Clustering -- kmeans
3. Association Analysis
4. Anomaly detection
5. Textual data -- Naive Bayes classifier (not taught in this course)

A clear justification have to be given for why a certain algorithm was chosen to address your problem.

Score: 4 Marks (2 marks each for each algorithm)

6.1 ML technique 1 + Justification

In [41]:

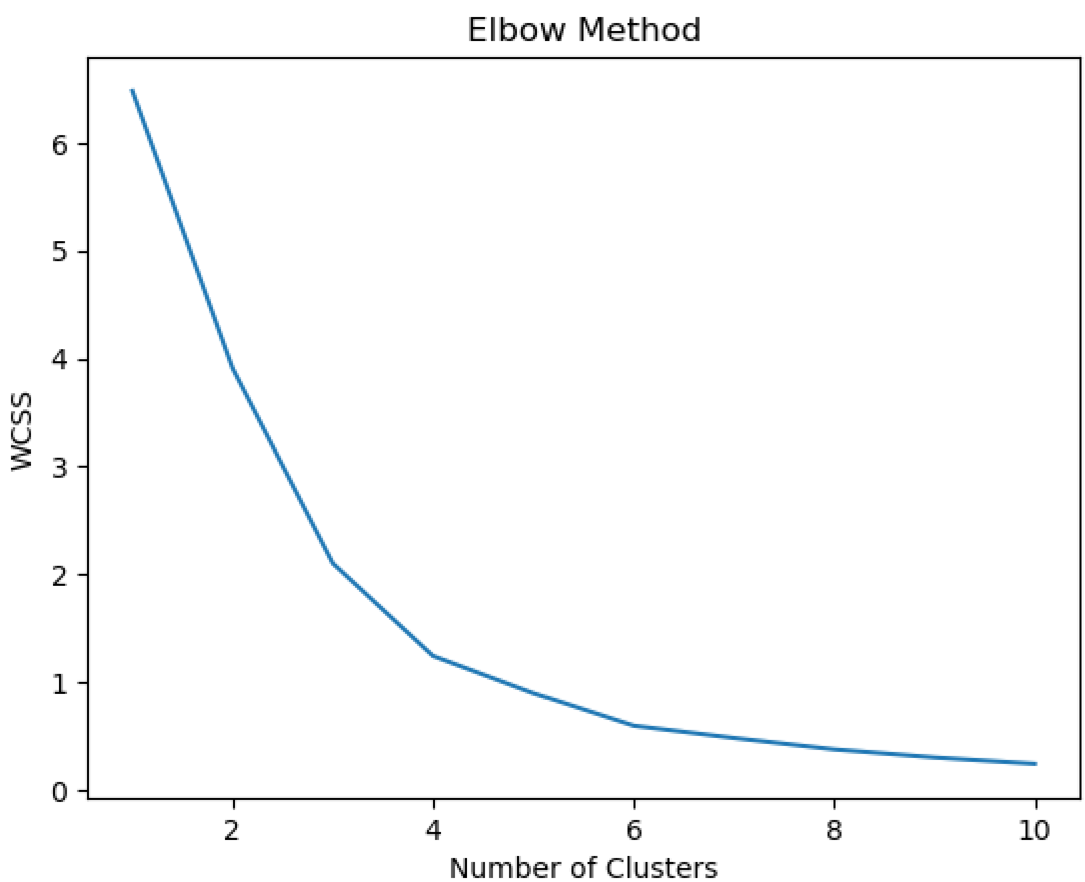
```
##---------Type the code below this line------------------##
from sklearn.preprocessing import MinMaxScaler
X = MinMaxScaler().fit_transform(df[['price','mileage']])
X
```

Out[41]:

```
array([[6.09994068e-02, 1.00000000e+00],
       [7.25262679e-02, 7.14770798e-01],
       [6.51903258e-02, 5.92529711e-01],
       [4.64312738e-02, 5.28013582e-01],
       [8.31633840e-02, 5.28013582e-01],
       [7.69959527e-02, 5.24617997e-01],
       [3.26931501e-02, 4.90662139e-01],
       [2.80295869e-02, 4.73684211e-01],
       [6.60989346e-02, 4.56706282e-01],
       [9.29243790e-02, 4.22750424e-01],
       [1.00812613e-01, 4.22750424e-01],
       [1.65798580e-01, 4.22750424e-01],
       [1.21782927e-01, 3.88794567e-01],
       [7.12162782e-02, 3.85398981e-01],
       [2.53676879e-02, 3.54838710e-01],
       [6.46663299e-02, 3.20882852e-01],
       [9.03326955e-02, 3.20882852e-01],
       [1.03967068e-01, 3.20882852e-01],
       [6.20463506e-02, 3.03904924e-01],
```

In [42]:
```
# determine the optimal number of clusters using the elbow method
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

```
C:\Users\p.puttaiahgowda\Anaconda3\lib\site-packages\sklearn\cluster\_kmean
s.py:1036: UserWarning: KMeans is known to have a memory leak on Windows wit
h MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

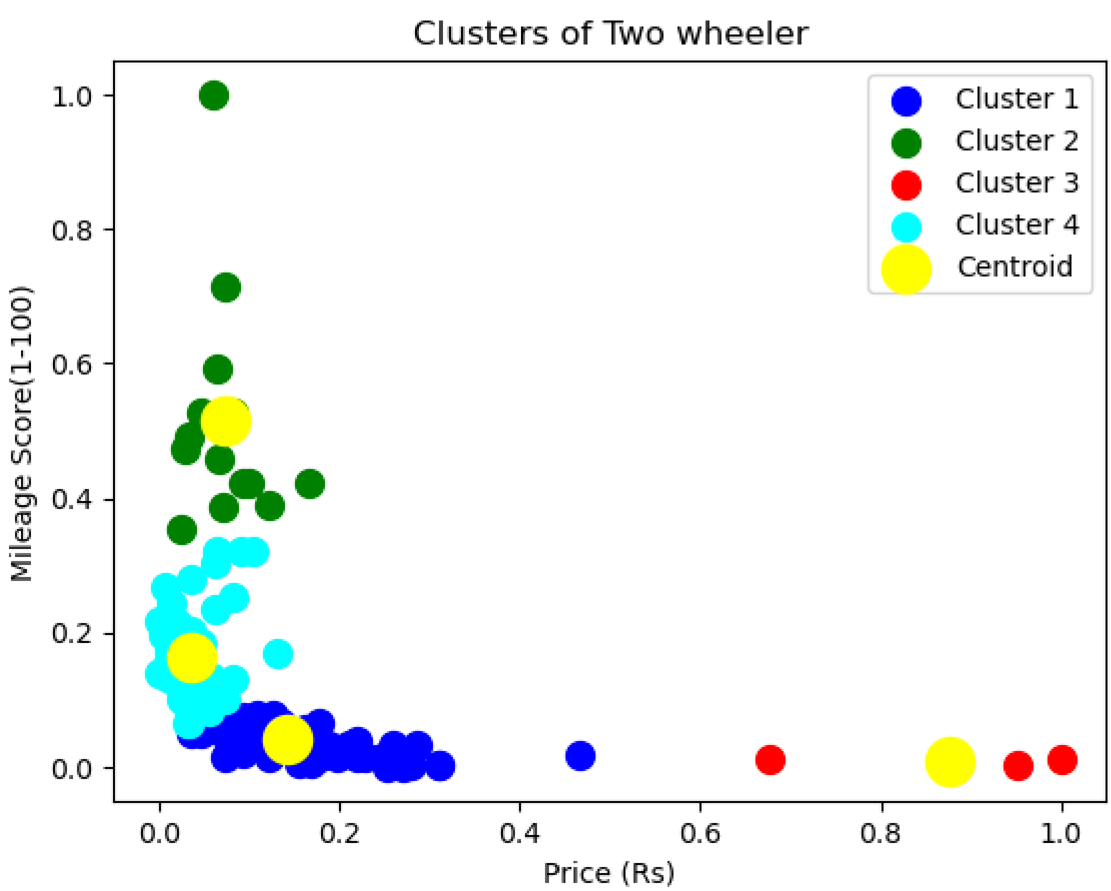


From the above plot, we can see the elbow point is at 4. So the number of clusters here will be 4.

In [43]:
```
##---------Type the code below this line------------------##
#training the K-means model on a dataset
kmeans = KMeans(n_clusters=4, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(X)
y_predict
```

Out[43]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 3, 3, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0])
```

In [44]:
```
#visulaizing the clusters
plt.scatter(X[y_predict == 0, 0], X[y_predict == 0, 1], s = 100, c = 'blue',
plt.scatter(X[y_predict == 1, 0], X[y_predict == 1, 1], s = 100, c = 'green',
plt.scatter(X[y_predict== 2, 0], X[y_predict == 2, 1], s = 100, c = 'red', la
plt.scatter(X[y_predict == 3, 0], X[y_predict == 3, 1], s = 100, c = 'cyan',
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s =
plt.title('Clusters of Two wheeler')
plt.xlabel('Price (Rs)')
plt.ylabel('Mileage Score(1-100)')
plt.legend()
plt.show()
```



The output image is clearly showing the four different clusters with different colors. The clusters are formed between two parameters of the dataset: Price and Mileage of the vehicle.

We can also observe some points from the above patterns, which are given below:

Cluster1 shows the bike with very low price but average mileage, so we can categorize these as Second choice.
Cluster2 shows the bike with very high price but very low mileage, so we can categorize these as Last choice.
Cluster3 shows the bike with medium price but very low mileage, so we can be categorize these as Third choice.
Cluster4 shows the bike with low price but high mileage so they can be categorized as target, and these vehicles can be the most profitable two wheeler bikes for the customers. we can categorize these as First choice.

6.2 ML technique 2 + Justification

Classification -- Decision Tree classifier

In [45]:
```
##---------Type the code below this line------------------##
!pip install --trusted-host pypi.org --trusted-host pypi.python.org --trusted
```

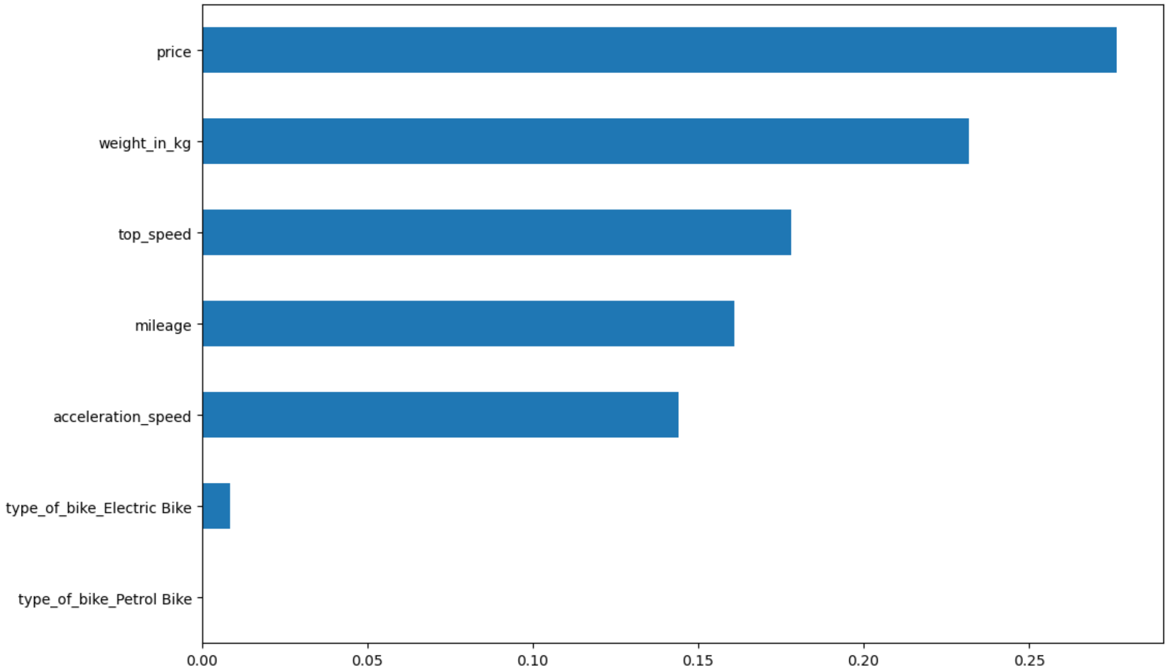
```
Requirement already satisfied: mlxtend in c:\users\p.puttaiahgowda\anaconda3
\lib\site-packages (0.21.0)
Requirement already satisfied: scipy>=1.2.1 in c:\users\p.puttaiahgowda\anac
onda3\lib\site-packages (from mlxtend) (1.9.1)
Requirement already satisfied: joblib>=0.13.2 in c:\users\p.puttaiahgowda\an
aconda3\lib\site-packages (from mlxtend) (1.1.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\p.puttaiahgow
da\anaconda3\lib\site-packages (from mlxtend) (1.0.2)
Requirement already satisfied: pandas>=0.24.2 in c:\users\p.puttaiahgowda\an
aconda3\lib\site-packages (from mlxtend) (1.4.4)
Requirement already satisfied: numpy>=1.16.2 in c:\users\p.puttaiahgowda\ana
conda3\lib\site-packages (from mlxtend) (1.21.5)
Requirement already satisfied: setuptools in c:\users\p.puttaiahgowda\anacon
da3\lib\site-packages (from mlxtend) (63.4.1)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\p.puttaiahgowda
\anaconda3\lib\site-packages (from mlxtend) (3.5.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\p.puttaiahgowda
\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\p.puttaiahgowda
\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\p.puttaiahgowda\a
naconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (21.3)
Requirement already satisfied: cycler>=0.10 in c:\users\p.puttaiahgowda\anac
onda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\p.puttaiahgowda
\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in c:\users\p.puttaiahgowda\ana
conda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (9.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\p.puttaiahgo
wda\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\p.puttaiahgowda\anac
onda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2022.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\p.puttaiahgo
wda\anaconda3\lib\site-packages (from scikit-learn>=1.0.2->mlxtend) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\p.puttaiahgowda\anaconda
3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend)
(1.16.0)
```

In [46]:
```
##---------Type the code below this line------------------##
import joblib
import sys
import numpy as np

sys.modules['sklearn.externals.joblib'] = joblib
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

In [47]:
```
def prepare_decision_tree(X, y, show_matrix=False, show_accuracy=True, show_r
    #Split the data into training and testing set
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size =0.3)

    #Construct decision tree
    dt = DecisionTreeClassifier(random_state=100)
    dt.fit(X_train, y_train)

    #Use the decision tree for prediction on test data
    y_pred = dt.predict(X_test)

    #Prepare the confusion matrix
    actuals = np.array(y_test)
    predictions = np.array(y_pred)

    if show_matrix:
        print("Confusion Matrix : ")
        print(confusion_matrix(actuals, predictions), "\n")

    #Compute accuracy
    if show_accuracy:
        print ("Accuracy : ", accuracy_score(y_test,y_pred)*100, "\n")

    #Generate classification report
    if show_report:
        print("Classification Report : \n", classification_report(y_test, y_p

    #Show the important features visually
    if show_visual:
        importances=pd.Series(dt.feature_importances_, index=X.columns).sort_
        importances.plot(kind='barh', figsize=(12,8))

    return dt
```

In [48]:
```
dt = prepare_decision_tree(feature_sel,y, show_visual = True)
```

```
Accuracy :  0.0
```



7. Conclusion

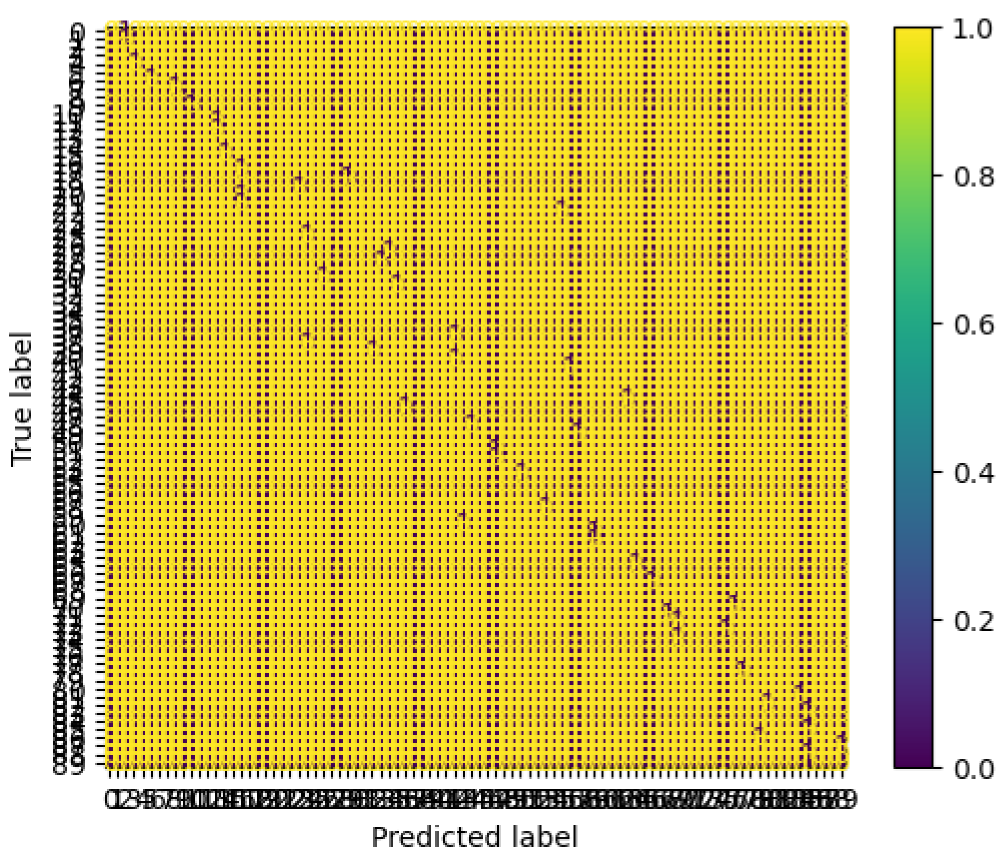
Compare the performance of the ML techniques used.

Derive values for preformance study metrics like accuracy, precision, recall, F1 Score, AUC-ROC etc to compare the ML algos and plot them. A proper comparision based on different metrics should be done and not just accuracy alone, only then the comparision becomes authentic. You may use Confusion matrix, classification report, Word cloud etc as per the requirement of your application/problem.

Score 1 Mark

In [50]:
```
##---------Type the code below this line------------------##
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size =0.3)

    #Construct decision tree
dt = DecisionTreeClassifier(random_state=100)
dt.fit(X_train, y_train)

    #Use the decision tree for prediction on test data
y_pred = dt.predict(X_test)

    #Prepare the confusion matrix
actuals = np.array(y_test)
predictions = np.array(y_pred)

print("Confusion Matrix : ")

cm = confusion_matrix(actuals, predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

```
Confusion Matrix :
```



In [51]:
```
Accuracy = metrics.accuracy_score(actuals, predictions)
print("Accuracy:",Accuracy)
```

```
Accuracy: 0.0196078431372549
```

In [52]:
```
print("Sensitivity_recall:", metrics.recall_score(y_test, y_pred, average='we
```

```
Sensitivity_recall: 0.0196078431372549

C:\Users\p.puttaiahgowda\Anaconda3\lib\site-packages\sklearn\metrics\_classi
fication.py:1318: UndefinedMetricWarning: Recall is ill-defined and being se
t to 0.0 in labels with no true samples. Use `zero_division` parameter to co
ntrol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [53]:
```
print("F1 Score:", metrics.f1_score(y_test, y_pred, average='weighted'))
```

```
F1 Score: 0.00980392156862745
```

In [54]:
```
print("Precision:", metrics.precision_score(y_test, y_pred, average='weighted
```

```
Precision: 0.0065359477124183

C:\Users\p.puttaiahgowda\Anaconda3\lib\site-packages\sklearn\metrics\_classi
fication.py:1318: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` paramete
r to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

8. Solution

What is the solution that is proposed to solve the business problem discussed in Section 1. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

Score 2 Marks

--------------Type the answers below this line--------------

```
##---------Type the answer below this line------------------##
From the above analysis, we can see one of the Best two wheeler available
in the market which gives high mileage at a low cost. From the Cluster
Analysis, the Cluster 4 types vehicles are the best to buy for the
customers. This is the business problem which we solved by giving some
current trends and patterns in the Indian motorcycle business.
The quality of the dataset was a challenge. As there was missing values,
outliers, and errors in the data that need to be dealt with before
analysis.
```

The observation we made was while selecting the appropriate model for the analysis, considering the type of data, the problem statement, and the objective, could affect the outcome of the project.

In [ ]: