2-jaamie-maarsh-joy-martin-v0-1-1

October 1, 2023

1 Assignment 2: Computization & Visualisation of DA (IE6600)

[2]:
```
# importing all the necessary libraries into the workspace
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

#This command is to ignore all the warnings
warnings.filterwarnings("ignore")

# loading/reading of the datasets into a dataframe (df)
df= pd.read_csv('/Users/jaamiemaarshj/Desktop/  DAE Course Materials/
↪Computization and Visualisation/Assignment-2/vehicles.csv', low_memory=False)

#displaying the first 5 rows of the dataframe
display(df.head())
```

```
   barrels08  barrelsA08  charge120  charge240  city08  city08U  cityA08  \
0  15.695714         0.0        0.0        0.0      19      0.0        0
1  29.964545         0.0        0.0        0.0       9      0.0        0
2  12.207778         0.0        0.0        0.0      23      0.0        0
3  29.964545         0.0        0.0        0.0      10      0.0        0
4  17.347895         0.0        0.0        0.0      17      0.0        0

   cityA08U  cityCD  cityE  …  mfrCode  c240Dscr  charge240b  c240bDscr  \
0       0.0     0.0    0.0  …      NaN       NaN         0.0        NaN
1       0.0     0.0    0.0  …      NaN       NaN         0.0        NaN
2       0.0     0.0    0.0  …      NaN       NaN         0.0        NaN
3       0.0     0.0    0.0  …      NaN       NaN         0.0        NaN
4       0.0     0.0    0.0  …      NaN       NaN         0.0        NaN

                      createdOn                    modifiedOn  startStop  \
0  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013        NaN
1  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013        NaN
2  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013        NaN
```

```
3  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013        NaN
4  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013        NaN

   phevCity  phevHwy  phevComb
0         0        0         0
1         0        0         0
2         0        0         0
3         0        0         0
4         0        0         0

[5 rows x 83 columns]
```

2 Question 1: Dataset loading , cleaning & filling missing values

[3]:
```
#gives out the count of the columns having the values (includes null values␣
↪also)
print(df.isnull().sum())

#gives the information of the columns.
df.info()
```

```
barrels08        0
barrelsA08       0
charge120        0
charge240        0
city08           0
             …
modifiedOn       0
startStop    31704
phevCity         0
phevHwy          0
phevComb         0
Length: 83, dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40081 entries, 0 to 40080
Data columns (total 83 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   barrels08        40081 non-null  float64
 1   barrelsA08       40081 non-null  float64
 2   charge120        40081 non-null  float64
 3   charge240        40081 non-null  float64
 4   city08           40081 non-null  int64
 5   city08U          40081 non-null  float64
 6   cityA08          40081 non-null  int64
 7   cityA08U         40081 non-null  float64
 8   cityCD           40081 non-null  float64
```

```
9   cityE            40081 non-null  float64
10  cityUF           40081 non-null  float64
11  co2              40081 non-null  int64
12  co2A             40081 non-null  int64
13  co2TailpipeAGpm  40081 non-null  float64
14  co2TailpipeGpm   40081 non-null  float64
15  comb08           40081 non-null  int64
16  comb08U          40081 non-null  float64
17  combA08          40081 non-null  int64
18  combA08U         40081 non-null  float64
19  combE            40081 non-null  float64
20  combinedCD       40081 non-null  float64
21  combinedUF       40081 non-null  float64
22  cylinders        39910 non-null  float64
23  displ            39912 non-null  float64
24  drive            38892 non-null  object
25  engId            40081 non-null  int64
26  eng_dscr         24182 non-null  object
27  feScore          40081 non-null  int64
28  fuelCost08       40081 non-null  int64
29  fuelCostA08      40081 non-null  int64
30  fuelType         40081 non-null  object
31  fuelType1        40081 non-null  object
32  ghgScore         40081 non-null  int64
33  ghgScoreA        40081 non-null  int64
34  highway08        40081 non-null  int64
35  highway08U       40081 non-null  float64
36  highwayA08       40081 non-null  int64
37  highwayA08U      40081 non-null  float64
38  highwayCD        40081 non-null  float64
39  highwayE         40081 non-null  float64
40  highwayUF        40081 non-null  float64
41  hlv              40081 non-null  int64
42  hpv              40081 non-null  int64
43  id               40081 non-null  int64
44  lv2              40081 non-null  int64
45  lv4              40081 non-null  int64
46  make             40081 non-null  object
47  model            40081 non-null  object
48  mpgData          40081 non-null  object
49  phevBlended      40081 non-null  bool
50  pv2              40081 non-null  int64
51  pv4              40081 non-null  int64
52  range            40081 non-null  int64
53  rangeCity        40081 non-null  float64
54  rangeCityA       40081 non-null  float64
55  rangeHwy         40081 non-null  float64
56  rangeHwyA        40081 non-null  float64
```

```
 57  trany            40070 non-null  object
 58  UCity            40081 non-null  float64
 59  UCityA           40081 non-null  float64
 60  UHighway         40081 non-null  float64
 61  UHighwayA        40081 non-null  float64
 62  VClass           40081 non-null  object
 63  year             40081 non-null  int64
 64  youSaveSpend     40081 non-null  int64
 65  guzzler          2377 non-null   object
 66  trans_dscr       15047 non-null  object
 67  tCharger         6302 non-null   object
 68  sCharger         796 non-null    object
 69  atvType          3374 non-null   object
 70  fuelType2        1547 non-null   object
 71  rangeA           1542 non-null   object
 72  evMotor          736 non-null    object
 73  mfrCode          9263 non-null   object
 74  c240Dscr         65 non-null     object
 75  charge240b       40081 non-null  float64
 76  c240bDscr        63 non-null     object
 77  createdOn        40081 non-null  object
 78  modifiedOn       40081 non-null  object
 79  startStop        8377 non-null   object
 80  phevCity         40081 non-null  int64
 81  phevHwy          40081 non-null  int64
 82  phevComb         40081 non-null  int64
dtypes: bool(1), float64(32), int64(27), object(23)
memory usage: 25.1+ MB
```

[4]:

```python
#Filling in zeros for the missing NaN values
df.fillna(0, inplace=True)
print(df.head(5))
```

```
   barrels08  barrelsA08  charge120  charge240  city08  city08U  cityA08  \
0  15.695714         0.0        0.0        0.0      19      0.0        0
1  29.964545         0.0        0.0        0.0       9      0.0        0
2  12.207778         0.0        0.0        0.0      23      0.0        0
3  29.964545         0.0        0.0        0.0      10      0.0        0
4  17.347895         0.0        0.0        0.0      17      0.0        0

   cityA08U  cityCD  cityE  …  mfrCode  c240Dscr  charge240b  c240bDscr  \
0       0.0     0.0    0.0  …        0         0         0.0          0
1       0.0     0.0    0.0  …        0         0         0.0          0
2       0.0     0.0    0.0  …        0         0         0.0          0
3       0.0     0.0    0.0  …        0         0         0.0          0
4       0.0     0.0    0.0  …        0         0         0.0          0

                     createdOn                    modifiedOn  startStop  \
```

```
0  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013          0
1  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013          0
2  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013          0
3  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013          0
4  Tue Jan 01 00:00:00 EST 2013  Tue Jan 01 00:00:00 EST 2013          0

   phevCity  phevHwy  phevComb
0         0        0         0
1         0        0         0
2         0        0         0
3         0        0         0
4         0        0         0

[5 rows x 83 columns]
```

[5]:
```
#dropping columns the necessary columns since they had contained less non-zero␣
 ↪values.
df.drop(columns=["charge120" , "guzzler" , "tCharger" , "sCharger" ,␣
 ↪"fuelType2" , "rangeA", "evMotor", "c240Dscr" , "c240bDscr"], inplace=True)

#Before dropping the total number of columns were 83 and after dropping it is 74
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40081 entries, 0 to 40080
Data columns (total 74 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   barrels08        40081 non-null  float64
 1   barrelsA08       40081 non-null  float64
 2   charge240        40081 non-null  float64
 3   city08           40081 non-null  int64
 4   city08U          40081 non-null  float64
 5   cityA08          40081 non-null  int64
 6   cityA08U         40081 non-null  float64
 7   cityCD           40081 non-null  float64
 8   cityE            40081 non-null  float64
 9   cityUF           40081 non-null  float64
 10  co2              40081 non-null  int64
 11  co2A             40081 non-null  int64
 12  co2TailpipeAGpm  40081 non-null  float64
 13  co2TailpipeGpm   40081 non-null  float64
 14  comb08           40081 non-null  int64
 15  comb08U          40081 non-null  float64
 16  combA08          40081 non-null  int64
 17  combA08U         40081 non-null  float64
 18  combE            40081 non-null  float64
 19  combinedCD       40081 non-null  float64
```

```
20  combinedUF       40081 non-null  float64
21  cylinders        40081 non-null  float64
22  displ            40081 non-null  float64
23  drive            40081 non-null  object
24  engId            40081 non-null  int64
25  eng_dscr         40081 non-null  object
26  feScore          40081 non-null  int64
27  fuelCost08       40081 non-null  int64
28  fuelCostA08      40081 non-null  int64
29  fuelType         40081 non-null  object
30  fuelType1        40081 non-null  object
31  ghgScore         40081 non-null  int64
32  ghgScoreA        40081 non-null  int64
33  highway08        40081 non-null  int64
34  highway08U       40081 non-null  float64
35  highwayA08       40081 non-null  int64
36  highwayA08U      40081 non-null  float64
37  highwayCD        40081 non-null  float64
38  highwayE         40081 non-null  float64
39  highwayUF        40081 non-null  float64
40  hlv              40081 non-null  int64
41  hpv              40081 non-null  int64
42  id               40081 non-null  int64
43  lv2              40081 non-null  int64
44  lv4              40081 non-null  int64
45  make             40081 non-null  object
46  model            40081 non-null  object
47  mpgData          40081 non-null  object
48  phevBlended      40081 non-null  bool
49  pv2              40081 non-null  int64
50  pv4              40081 non-null  int64
51  range            40081 non-null  int64
52  rangeCity        40081 non-null  float64
53  rangeCityA       40081 non-null  float64
54  rangeHwy         40081 non-null  float64
55  rangeHwyA        40081 non-null  float64
56  trany            40081 non-null  object
57  UCity            40081 non-null  float64
58  UCityA           40081 non-null  float64
59  UHighway         40081 non-null  float64
60  UHighwayA        40081 non-null  float64
61  VClass           40081 non-null  object
62  year             40081 non-null  int64
63  youSaveSpend     40081 non-null  int64
64  trans_dscr       40081 non-null  object
65  atvType          40081 non-null  object
66  mfrCode          40081 non-null  object
67  charge240b       40081 non-null  float64
```

```
 68   createdOn        40081 non-null   object
 69   modifiedOn       40081 non-null   object
 70   startStop        40081 non-null   object
 71   phevCity         40081 non-null   int64
 72   phevHwy          40081 non-null   int64
 73   phevComb         40081 non-null   int64
dtypes: bool(1), float64(31), int64(27), object(15)
memory usage: 22.4+ MB
```

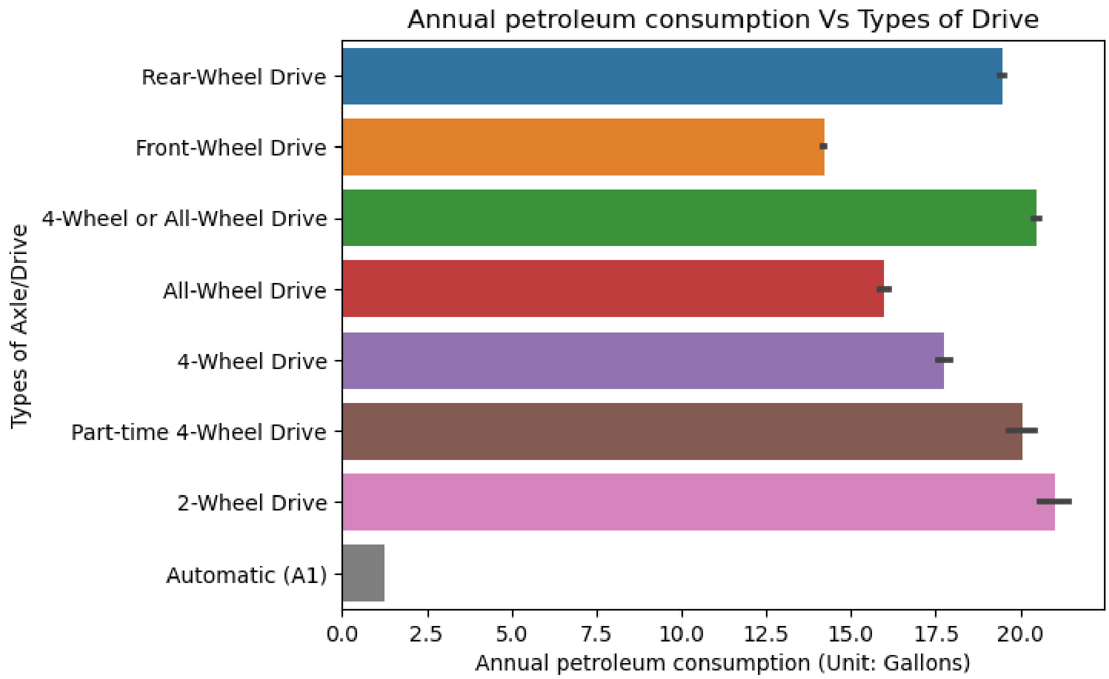
3 Question 2: Using matplotlib use charts of your choice and create visualizations(use at least 20 features) and create 15 charts.

4 Chart Type: Bar Chart

[14]:
```
#Chart 1: Comparision of the annual petroleum consumption Vs Drive␣
↪(2-Wheeldrive & 4-Wheeldrive)

#Dropping the zero values from the column 'drive'
Dropped_values_drive = [0]
df = df[~df['drive'].isin(Dropped_values_drive)]

# plotting a bar chart
sns.barplot(x=df['barrels08'], y=df['drive'])

print('Chart-1: Bar Chart: annual petroleum consumption Vs Drive ')
print("--------------------------------------------------------------")

# Labeling of X&Y axis and the title
plt.xlabel('Annual petroleum consumption (Unit: Gallons)')
plt.ylabel('Types of Axle/Drive')
plt.title('Annual petroleum consumption Vs Types of Drive')

# Displays the bar chart
plt.show()

#Insights:
#1. From the comparision of vehicle using different drives, it is found that␣
↪the "Automatic" drive vehicle uses very less fuel when compared to all other␣
↪drive vehicles.
#2. The fuel usage is almost equal to that of partial or complete 4-wheel drive
#3. Among the 2 wheel drive vehicles, the front wheel drive is the most␣
↪efficient.
#4. Automatic vehicles consume approxmately 13% less when compared to other␣
↪axle vehicles.
```

```
Chart-1: Bar Chart: annual petroleum consumption Vs Drive
```

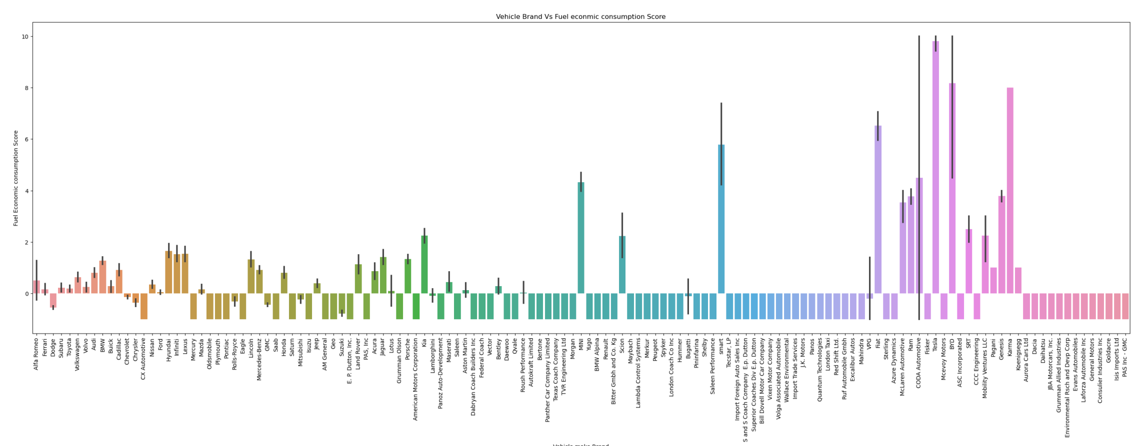
```
-------------------------------------------------------------
```



[12]:
```python
# Chart 2: Chart comparision of the Fuel Economic score with the car brand names.

#setting up the size of the graph so that it can accomodate all the x axis labels.
plt.figure(figsize=(35, 10))

#Setting up the bar chart
sns.barplot(x=df["make"], y=df["feScore"])

print('Chart-2: Bar Chart- Chart comparision of the Fuel Economic score with the car brand names')

# labelling the axes and the title
plt.xlabel('Vehicle make Brand')
plt.ylabel('Fuel Economic consumption Score')
plt.title('Vehicle Brand Vs Fuel econmic consumption Score')

print("-------------------------------------------------------------")
print(" FEscore range: 1-Worst Fuel Economy" , "10-Best Fuel Economy")
print("-------------------------------------------------------------")
```

```
#Rotating the x axis labels to accomodate all the brand names and avoid overlapping
plt.xticks(rotation=90)

# Show the plot
plt.show()


#Insights
# 1. Tesla has the maximum/best fuel economy when compared to its competetors, as it has the maximum intensity at score 10
# and the second best is found to be BYD, a touch over 8.
# 2. The average FE score across top brands with large sample size is 3.
# 3. Companies like Ford and Aston Martin has the least fuel economy score of decimals over 0.
```

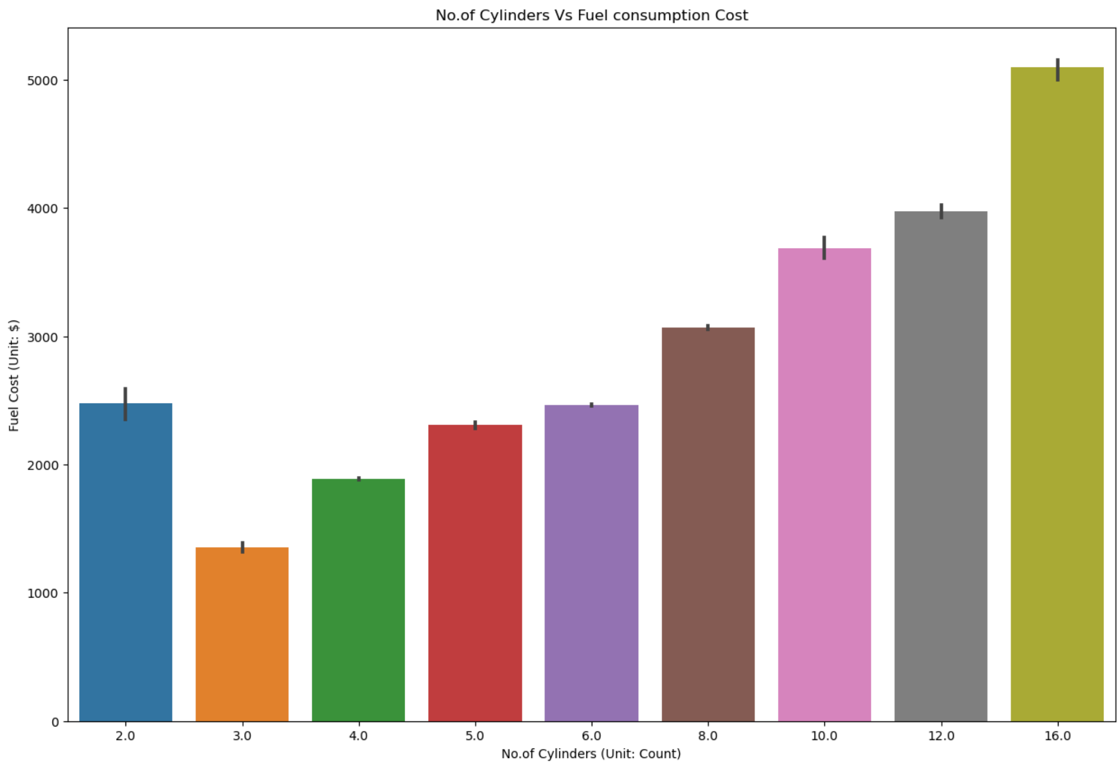
```
Chart-2: Bar Chart- Chart comparision of the Fuel Economic score with the car
brand names
-------------------------------------------------------------
 FEscore range: 1-Worst Fuel Economy 10-Best Fuel Economy
-------------------------------------------------------------
```



[13]:
```
# Chart 3: Chart comparision of the Fuel consumption cost with the number of cylinders present in the engine.

#Dropping the zero values from the column 'cylinders' as they dont have any effect on the analysis
Dropped_values_cylinders = [0]
df = df[~df['cylinders'].isin(Dropped_values_cylinders )]
```

```
#setting up the size of the graph so that it can accomodate all the x axis␣
↪labels.
plt.figure(figsize=(15, 10))

sns.barplot(x=df["cylinders"], y=df["fuelCost08"])

print('Chart-3: Bar Chart- Chart comparision of the Fuel cost with the␣
↪corresponding number of cylinders')
print("----------------------------------------------------------------------------------------

# Adds the necessary labels and the title
plt.xlabel('No.of Cylinders (Unit: Count)')
plt.ylabel('Fuel Cost (Unit: $)')
plt.title('No.of Cylinders Vs Fuel consumption Cost')

# Displaying the plot
plt.show()

#Insights:
# 1. The fuel cost incured rises exponentially, when the number of cylinders in␣
↪the engine of the car tends to increase.
# 2. Out of the lot, the engine have 3 cylinders is more fuel efficient when␣
↪compared to the others.
```

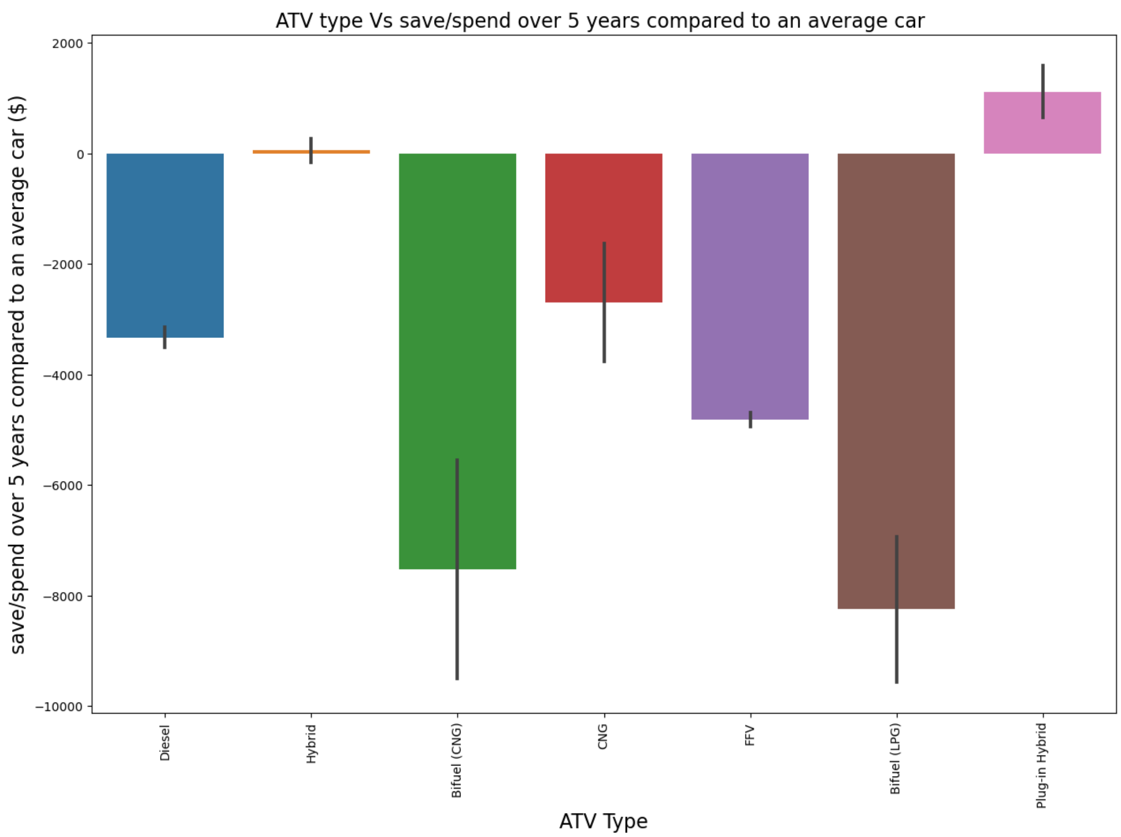
```
Chart-3: Bar Chart- Chart comparision of the Fuel cost with the corresponding
number of cylinders
--------------------------------------------------------------------------------
-----------------
```



[18]:
```
# Chart 4: Chart comparision between ATV types and the amount of spending/
↪saving to an average car.

#Dropping the zero values from the column 'atvType'
Dropped_values_atvType = [0]
df = df[~df['atvType'].isin(Dropped_values_atvType )]

#setting up the size of the graph for better readability
plt.figure(figsize=(15, 10))

#plotting the bar plot
sns.barplot(x=df["atvType"], y=df["youSaveSpend"])

print('Chart-4: Bar Chart- Chart comparision between ATV types and the amount␣
↪of spending/saving to an average car')
print("-----------------------------------------------------------------------------

# labelling the axes and the title
plt.xlabel('ATV Type', fontsize=16)
plt.ylabel('save/spend over 5 years compared to an average car ($)' ,␣
↪fontsize=16)
```

```
plt.title('ATV type Vs save/spend over 5 years compared to an average car ' ,␣
    ↪fontsize=16)

#Rotating the x axis labels to accomodate all the brand names
plt.xticks(rotation=90)

# Plotting the chart
plt.show()


#Insights
# 1. ATVs which are plug-in hybrid are the ones on which there is savings and␣
    ↪the rest of the fuel types incur a loss
# over the 5 year span.
# 2. Biofuels suchs are LPG and CNG have incurred heavy losses and will␣
    ↪continue to be so if it is used over long
# periods of time.
```

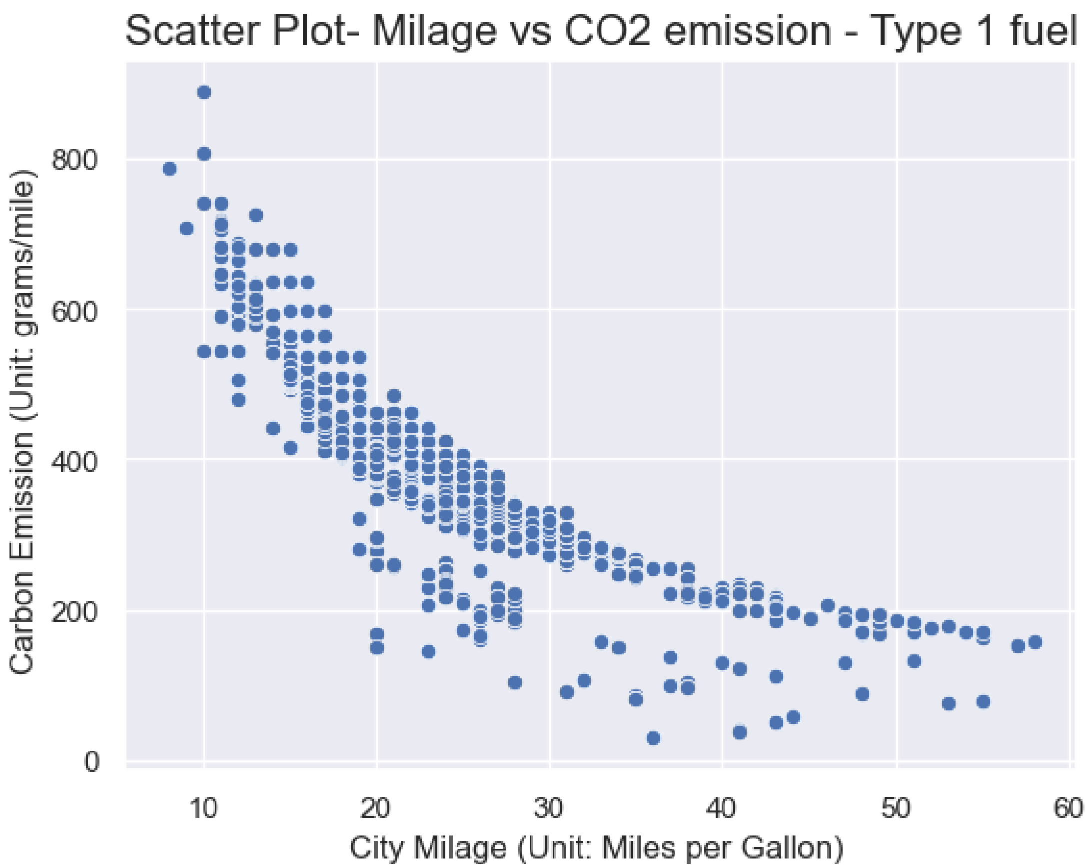
```
Chart-4: Bar Chart- Chart comparision between ATV types and the amount of
spending/saving to an average car
--------------------------------------------------------------------------------
---------------------------
```



5 Chart type: Scatter Plots

[22]:
```
#Chart 5: tailpipe CO2 in grams/mile  vs City Milage - For type 1 fuel

#Plotting the scatterplot with the below axes variables
sns.scatterplot(x= "city08", y= "co2TailpipeGpm", data= df)
sns.set(style="dark")

print('Chart-5: Scatter Plot- tailpipe CO2 in grams/mile  vs City Milage - For␣
↪type 1 fuel')
print("-----------------------------------------------------------------------------------------

#Adding the necessary X&Y axis labels
plt.xlabel('City Milage (Unit: Miles per Gallon) ', fontsize=12)
plt.ylabel('Carbon Emission (Unit: grams/mile) ', fontsize=12)
plt.title('Scatter Plot- Milage vs CO2 emission - Type 1 fuel', fontsize=16)

#adding grid to the chart for more readability
plt.grid(True)

#Showing the Scatter plot
plt.show()

#insights
# 1. from the graph, it is found that wheneven the milage inside the city comes␣
↪down there is a significant increase
# in the carbon emission in the atmosphere.
```

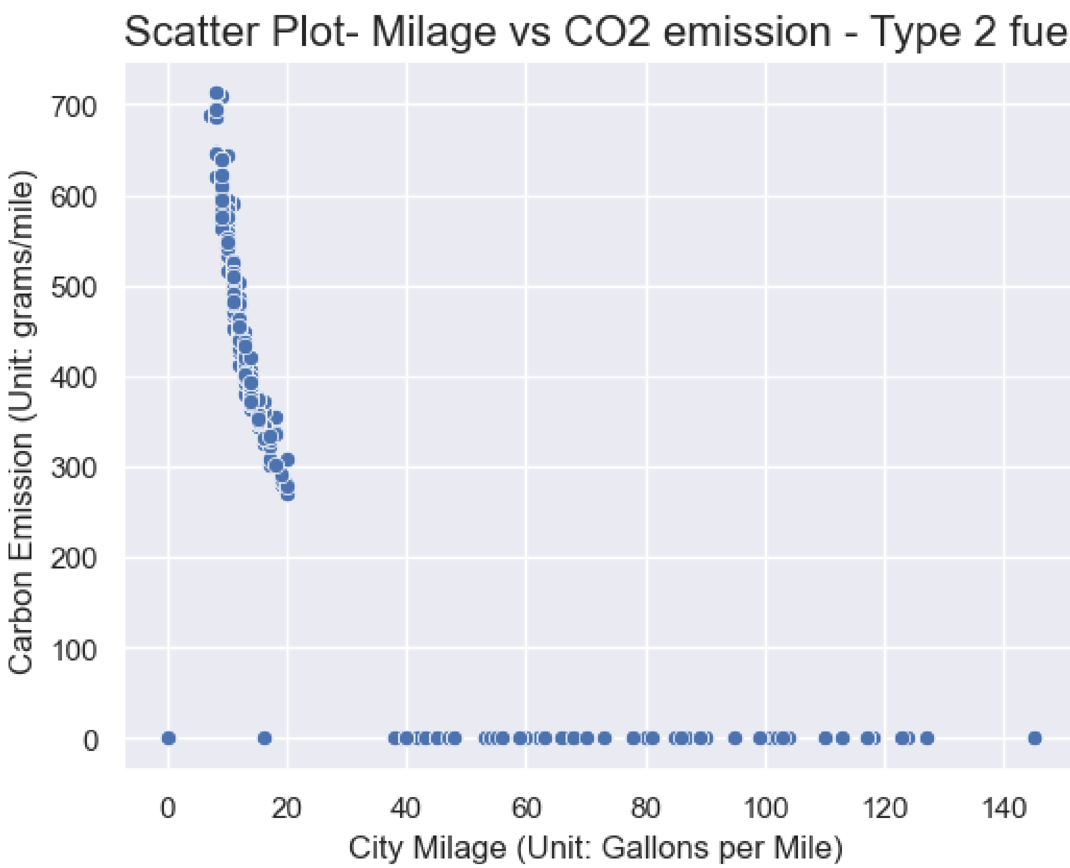
```
Chart-5: Scatter Plot- tailpipe CO2 in grams/mile  vs City Milage - For type 1
fuel
--------------------------------------------------------------------------------
---------------------------
```



[23]:
```
#tailpipe CO2 in grams/mile  vs City Milage - For type 2 fuel

#Plotting the scatterplot with the below variables
sns.scatterplot(x= "cityA08", y= "co2TailpipeAGpm", data= df)
sns.set(style="darkgrid")

print('Chart-6: Scatter Plot- Carbon dioxide emission vs City Milage - For type␣
 ↪2 fuel')
print("------------------------------------------------------------------------------------


#Adding the necessary X&Y axis labels
plt.xlabel('City Milage (Unit: Gallons per Mile) ' , fontsize=12)
plt.ylabel('Carbon Emission (Unit: grams/mile) ', fontsize=12)
plt.title('Scatter Plot- Milage vs CO2 emission - Type 2 fuel', fontsize=16)

#Adding grid to the scatter plot for more readability and accuracy
plt.grid(True)
```

```
#showing the scatter plot
plt.show()

#insights
# 1. from the graph, it is found that wheneven the milage inside the city comes␣
↪down there is a significant increase
# in the carbon emission in the atmosphere even for the auxillary fuel.
```

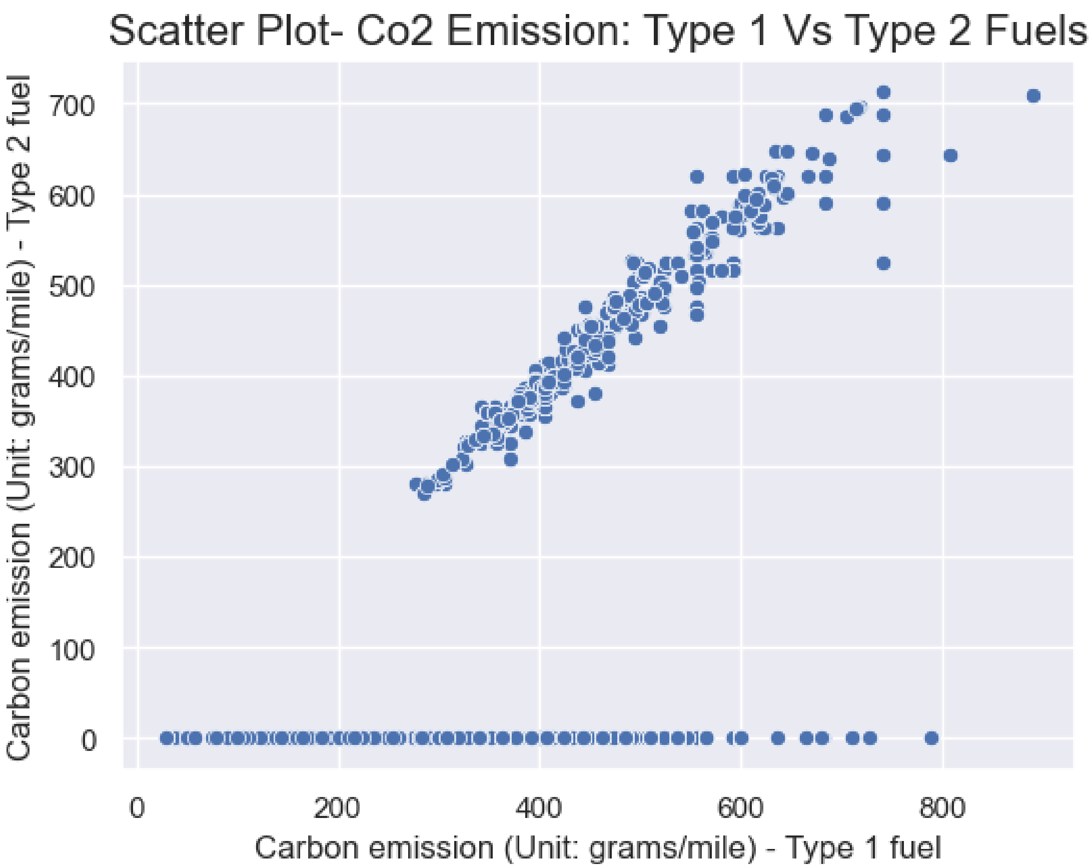
```
Chart-6: Scatter Plot- Carbon dioxide emission vs City Milage - For type 2 fuel
--------------------------------------------------------------------------------
---------------------------
```



[27]:
```
#tailpipe CO2 in grams/mile  vs tailpipe CO2 in grams/mile - For type 1 fuel Vs␣
↪Type 2

#Plotting the scatterplot for the below variables
sns.scatterplot(x= "co2TailpipeGpm", y= "co2TailpipeAGpm", data= df)
```

```
print('Chart-7: Scatter Plot- Visualisation of Carbon emission comparison - For␣
 ↪type 1 fuel Vs Type 2 ')
print("-----------------------------------------------------------------------------------

#Adding the necessary X&Y axis labels
plt.xlabel('Carbon emission (Unit: grams/mile) - Type 1 fuel', fontsize=12)
plt.ylabel('Carbon emission (Unit: grams/mile) - Type 2 fuel' , fontsize=12)
plt.title('Scatter Plot- Co2 Emission: Type 1 Vs Type 2 Fuels', fontsize=16)

#Adding grid to the scatter plot for more readability and accuracy
plt.grid(True)

#Showing the scatter plot
plt.show()

#insights
# 1. from the graph, it is found that the mean emission value for both the fuel␣
 ↪types lies between 250-600 grams/mile.
```

```
Chart-7: Scatter Plot- Visualisation of Carbon emission comparison - For type 1
fuel Vs Type 2
--------------------------------------------------------------------------------
---------------------------
```

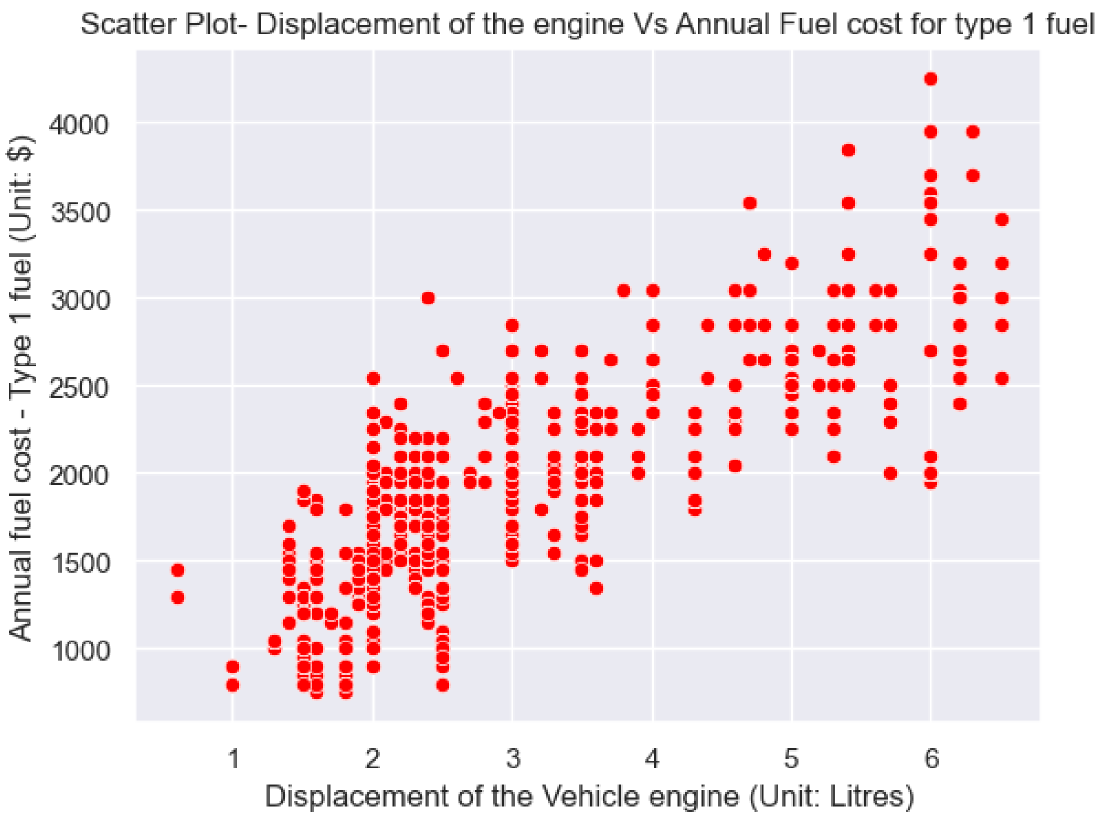


[30]:

```
#Displacement of the engine Vs Annual Fuel cost for type 1 fuel

#Plotting the scatter plot for the below variables
sns.scatterplot(x= "displ", y= "fuelCost08", c='red' , data= df)
sns.set(style="dark")

print('Chart-8: Scatter Plot- Displacement of the engine Vs Annual Fuel cost␣
  ↪for type 1 fuel ')
print("--------------------------------------------------------------------------------------

#Labelling of axes & title
plt.xlabel('Displacement of the Vehicle engine (Unit: Litres)')
plt.ylabel('Annual fuel cost - Type 1 fuel (Unit: $)')
plt.title('Scatter Plot- Displacement of the engine Vs Annual Fuel cost for␣
  ↪type 1 fuel')

#setting the grid for more readability and accuracy
plt.grid(True)
```

```
#Showing the plotted graph
plt.show()

#Insights:
# 1. There is found to to be a positive corelation between the label, whenever␣
↪the displacement of the engine increases
# the annual fuel cost also increases.
# 2. The plots are mostly found to be more effective when the displacement is␣
↪between 1.5-2.5 Litres
```

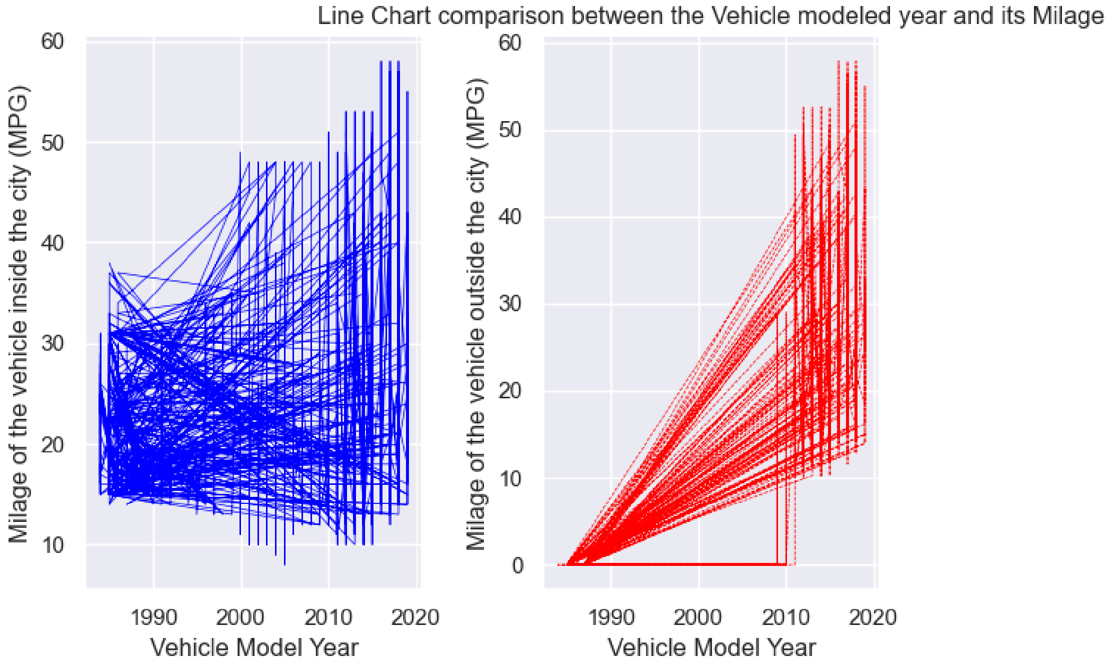
```
Chart-8: Scatter Plot- Displacement of the engine Vs Annual Fuel cost for type 1
fuel
--------------------------------------------------------------------------------
--------------------------
```



6 Chart type: Line Chart

[38]:
```
# Visualization between the Vehicle model Year Vs Milage of the vehicle (MPG)

print('Chart-9: Line Chart- Vehicle model Year Vs Milage of the vehicle (MPG) ')
print("--------------------------------------------------------------------------")

#Subplot 1: Vehicle model Year Vs Milage of the vehicle inside City limits (MPG)
plt.subplot(1, 2, 1)

#Labelling the axes
plt.plot(df["year"] , df["city08"] , color='blue', linestyle='-', linewidth=0.5)
plt.xlabel('Vehicle Model Year')
plt.ylabel('Milage of the vehicle inside the city (MPG)')

#setting the grid for more readability and accuracy
plt.grid(True)

#subplot 2: Vehicle model Year Vs Milage of the vehicle outside city limits␣
↪(MPG)

plt.subplot(1, 2, 2)
plt.plot(df["year"] , df["city08U"] , color='red', linestyle='--', linewidth=0.
↪5)
plt.plot

#Labelling the axes
plt.xlabel('Vehicle Model Year')
plt.ylabel('Milage of the vehicle outside the city (MPG)')

plt.title('Line Chart comparison between the Vehicle modeled year and its␣
↪Milage')

#setting the grid for more readability and accuracy
plt.grid(True)

#adjusts the spacing between the 2 charts
plt.tight_layout()

#Displaying the graph
plt.show()

#insights
# 1. It is found that as the make of the vehicle is newer and newer the milage␣
↪for the vehicle, irrespective of being inside/outside the city keeps on␣
↪increasing.
# 2. There is a sudden increase in the vehicle milage from 2003 to 2013
```

```
Chart-9: Line Chart- Vehicle model Year Vs Milage of the vehicle (MPG)
---------------------------------------------------------------------------
```

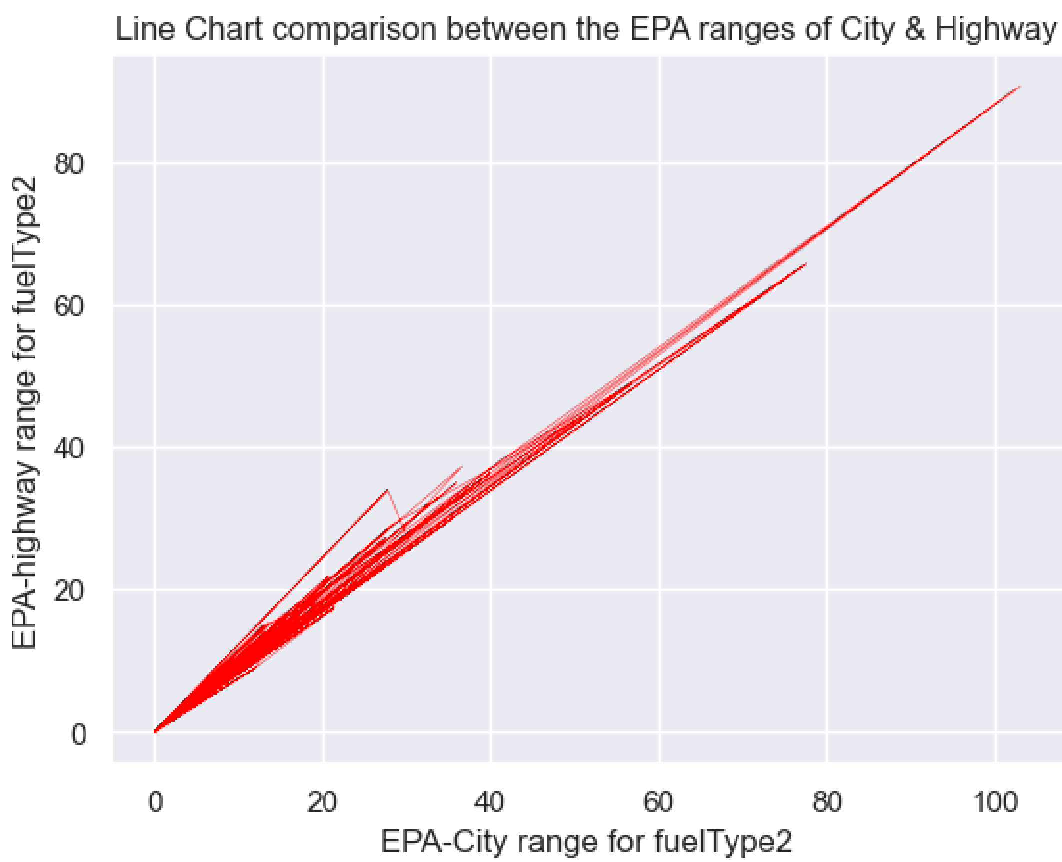


[39]:

```python
#Chart 10: Line Chart comparison between the EPA for fueltype2

#Plotting of the line chart between the EPA ranges
plt.plot(df["rangeCityA"] , df["rangeHwyA"] , color='red', linestyle='-',␣
    ↪linewidth=0.3)

print('Chart-10: Line Chart- Vehicle model Year Vs EPA range for Highway and␣
    ↪city ')
print("---------------------------------------------------------------------------")

#labeling the axes
plt.xlabel('EPA-City range for fuelType2')
plt.ylabel('EPA-highway range for fuelType2')

#Labeling the title of the line graph
plt.title('Line Chart comparison between the EPA ranges of City & Highway')

#setting the grid for more readability and accuracy
plt.grid(True)

#Displaying the chart
plt.show()
```

```
#Insights:
# 1. The chart infers to the fact these 2 fields are dependent of each other␣
↪and also has a positive slope for the same
# types of cars.
```

```
Chart-10: Line Chart- Vehicle model Year Vs EPA range for Highway and city
--------------------------------------------------------------------------
```



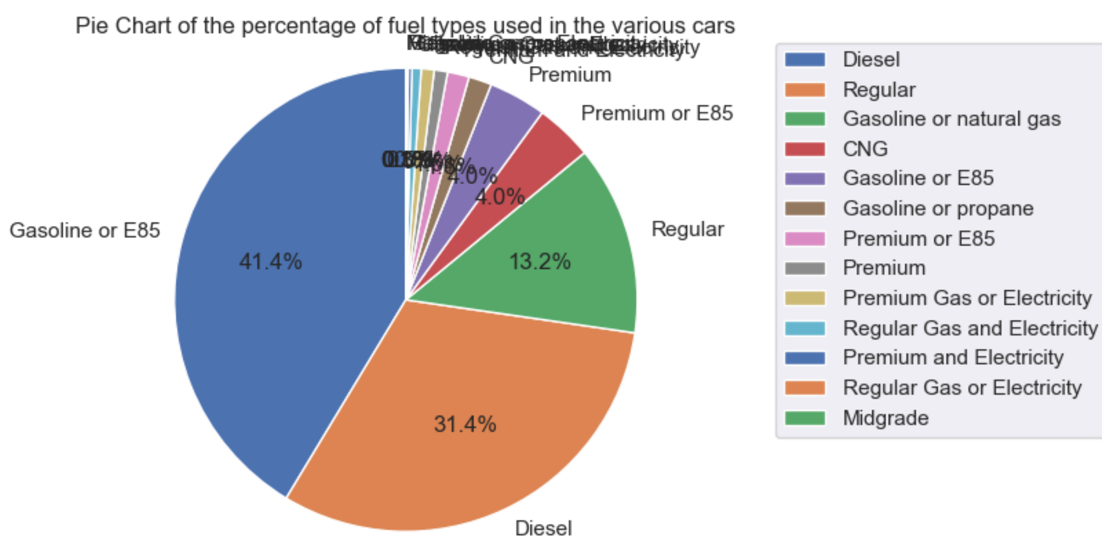
7 Chart type: Pie Chart

[41]:
```
#Chart 11: A pie chart to represent the different types of fuel used and it␣
↪percentage of usage across all the cars.

print('Chart-11: Pie Chart- represent the different types of fuel used for the␣
↪Vehicles under study ')
print("----------------------------------------------------------------------------------------

# Count the categories
```

```
No_of_Fueltype_counts = df['fuelType'].value_counts()

# Calculate proportions
total_Fueltype_count = No_of_Fueltype_counts.sum()
Fueltype_proportions = No_of_Fueltype_counts/ total_Fueltype_count

#Displaying the necessary calculated values.
print("the total specific fuel type counts:", No_of_Fueltype_counts)
print(" The total count of the types:" , total_Fueltype_count)
print("Catagory Proportions" , Fueltype_proportions)

# Create a pie chart
plt.pie(Fueltype_proportions, labels=Fueltype_proportions.index, autopct='%1.
 ↪1f%%', startangle=90)

#To maintain an equal aspect ratio so that the pie chart is neat and circular
plt.axis('equal')

# Add a title
plt.title('Pie Chart of the percentage of fuel types used in the various cars')

#altering the font size of the contents for visual appeal
plt.rcParams['font.size'] = 12

#Displaying the legend/labels of the pie chart.
unique_fuelType = df['fuelType'].unique()
plt.legend(unique_fuelType, loc='upper left', bbox_to_anchor=(1.025, 1.025))

# plot the chart
plt.show()

#Insights:
# 1. Regular gas is the most used fuel (approx 65%) across all the cars which␣
 ↪are manufactured between the year 1985-2010
# and the best being premium (approx 28%).
# 2. The usage of electricity as the fuel is constanly on the rise with almost␣
 ↪168 vehicle run solely on and approximately
# around 100 vehicles run with a combination of other fuels.
```

```
Chart-11: Pie Chart- represent the different types of fuel used for the Vehicles
under study
--------------------------------------------------------------------------------
-------------
the total specific fuel type counts: Gasoline or E85                1287
Diesel                           976
Regular                          412
Premium or E85                   125
```

```
Premium                          125
CNG                               50
Premium and Electricity           47
Regular Gas and Electricity       29
Premium Gas or Electricity        28
Gasoline or natural gas           20
Gasoline or propane                8
Regular Gas or Electricity         3
Midgrade                           2
Name: fuelType, dtype: int64
 The total count of the types: 3112
Catagory Proportions Gasoline or E85                    0.413560
Diesel                         0.313625
Regular                        0.132391
Premium or E85                 0.040167
Premium                        0.040167
CNG                            0.016067
Premium and Electricity        0.015103
Regular Gas and Electricity    0.009319
Premium Gas or Electricity     0.008997
Gasoline or natural gas        0.006427
Gasoline or propane            0.002571
Regular Gas or Electricity     0.000964
Midgrade                       0.000643
Name: fuelType, dtype: float64
```



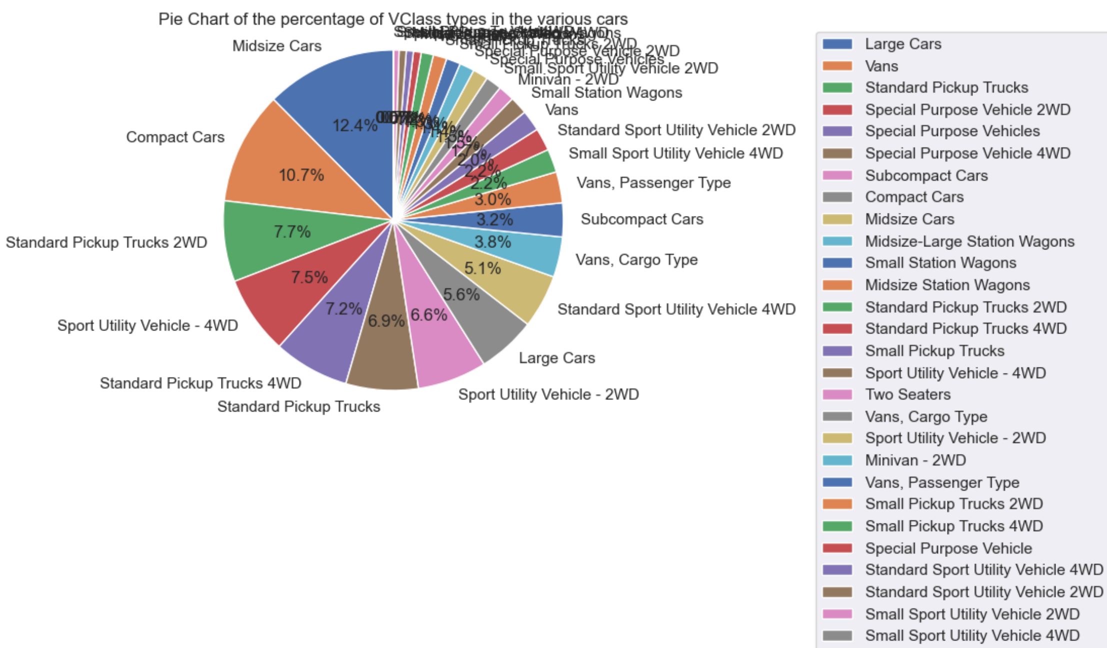
[42]:
```
# Chart 12: Pie Chart - Finding out the percentage of Vehicle class under study
```

```python
print('Chart-12: Pie Chart- Finding out the percentage of Vehicle class under study ')
print("-----------------------------------------------------------------------------")

#Count the categories
No_of_VClass_counts = df['VClass'].value_counts()

# Calculate proportions
total_VClass_count = No_of_VClass_counts.sum()
VClass_proportions = No_of_VClass_counts/ total_VClass_count

#Displaying the necessary calculated values.
print("the total specific VClass counts:", No_of_VClass_counts)
print(" The total count of the Class types:" , total_VClass_count)
print("Catagory Proportions of VClass" , VClass_proportions)

# Create a pie chart
plt.pie(VClass_proportions, labels=VClass_proportions.index, autopct='%1.1f%%', startangle=90)

#To maintain an equal aspect ratio so that the pie chart is neat and circular
plt.axis('equal')
# Add a title
plt.title('Pie Chart of the percentage of VClass types in the various cars')

#reduce the font size of the contents for visual appeal
plt.rcParams['font.size'] = 12

#Displaying the legend/labels of the pie chart.
unique_VClass= df['VClass'].unique()
plt.legend(unique_VClass, loc='upper left', bbox_to_anchor=(1.325, 1.025))

# plot the chart
plt.show()

#Insights:
# 1. According to the chart, Compact cars (14.3%) and sub compact cars(12.6%) are the most preferred Vehicle class types,
# whereas special purpose vehicles are very low on number (approx less than 1% of the total class on display).
# 2. When analysing the vans and the utility vehicle types, they seems to be scarsely present from approximately 2-6%
# individually.
```

```
Chart-12: Pie Chart- Finding out the percentage of Vehicle class under study
-----------------------------------------------------------------------------
the total specific VClass counts: Midsize Cars                          387
```

```
Compact Cars                          334
Standard Pickup Trucks 2WD            239
Sport Utility Vehicle - 4WD           232
Standard Pickup Trucks 4WD            224
Standard Pickup Trucks                214
Sport Utility Vehicle - 2WD           205
Large Cars                            173
Standard Sport Utility Vehicle 4WD    158
Vans, Cargo Type                      119
Subcompact Cars                       100
Vans, Passenger Type                   92
Small Sport Utility Vehicle 4WD        69
Standard Sport Utility Vehicle 2WD     67
Vans                                   62
Small Station Wagons                   53
Minivan - 2WD                          48
Small Sport Utility Vehicle 2WD        47
Special Purpose Vehicles               45
Special Purpose Vehicle 2WD            44
Small Pickup Trucks 2WD                42
Small Pickup Trucks                    40
Two Seaters                            36
Midsize Station Wagons                 23
Midsize-Large Station Wagons           21
Special Purpose Vehicle 4WD            21
Small Pickup Trucks 4WD                16
Special Purpose Vehicle                 1
Name: VClass, dtype: int64
 The total count of the Class types: 3112
Catagory Proportions of VClass Midsize Cars                          0.124357
Compact Cars                          0.107326
Standard Pickup Trucks 2WD            0.076799
Sport Utility Vehicle - 4WD           0.074550
Standard Pickup Trucks 4WD            0.071979
Standard Pickup Trucks                0.068766
Sport Utility Vehicle - 2WD           0.065874
Large Cars                            0.055591
Standard Sport Utility Vehicle 4WD    0.050771
Vans, Cargo Type                      0.038239
Subcompact Cars                       0.032134
Vans, Passenger Type                  0.029563
Small Sport Utility Vehicle 4WD       0.022172
Standard Sport Utility Vehicle 2WD    0.021530
Vans                                  0.019923
Small Station Wagons                  0.017031
Minivan - 2WD                         0.015424
Small Sport Utility Vehicle 2WD       0.015103
Special Purpose Vehicles              0.014460
```

```
Special Purpose Vehicle 2WD          0.014139
Small Pickup Trucks 2WD              0.013496
Small Pickup Trucks                  0.012853
Two Seaters                          0.011568
Midsize Station Wagons               0.007391
Midsize-Large Station Wagons         0.006748
Special Purpose Vehicle 4WD          0.006748
Small Pickup Trucks 4WD              0.005141
Special Purpose Vehicle              0.000321
Name: VClass, dtype: float64
```

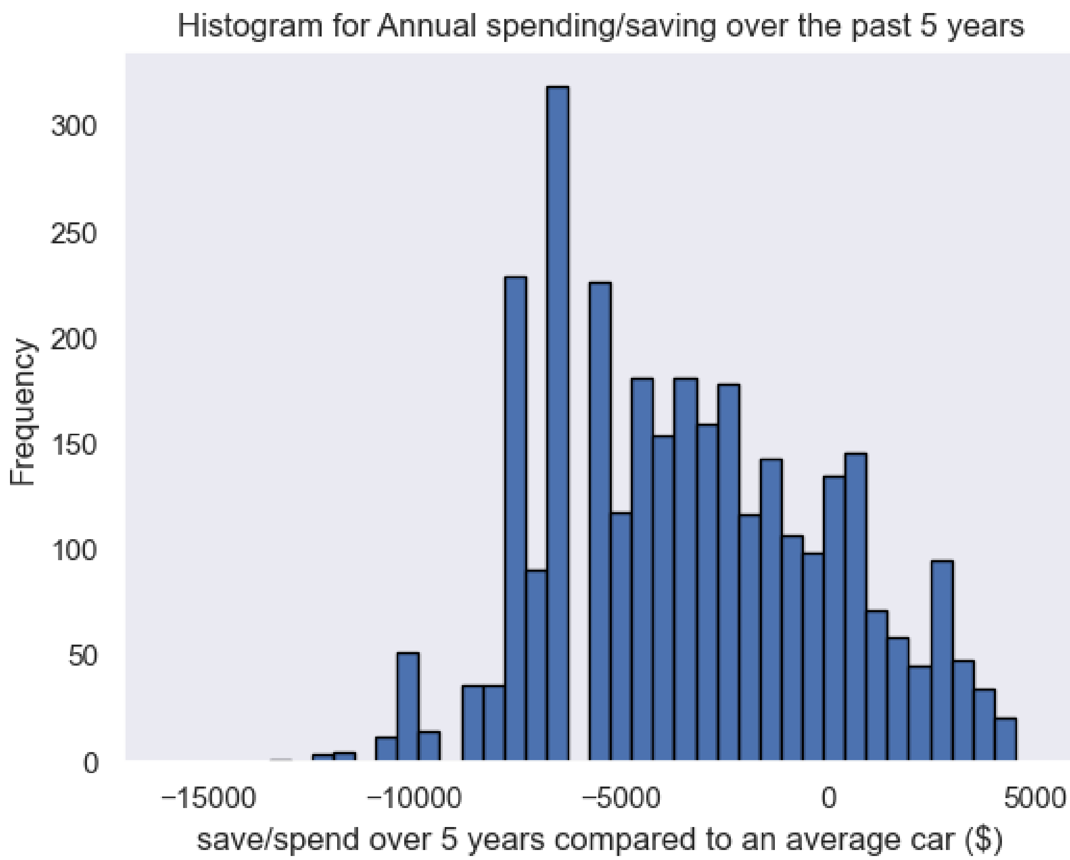


8 Chart type : Histogram

[44]:
```python
#Chart 13: A histogram chart comparison between the spending/saving

print('Chart-13: Histogram - comparison between the spending/saving  ')
print("-----------------------------------------------------------------------------")

#Plotting the histogram for the below variables
plt.hist(df["youSaveSpend"] , bins=35, edgecolor='black')

#Customising up the x axis range
plt.xlim(-17000, 6000, 6000)

#Labelling the axis & title
```

```
plt.xlabel('save/spend over 5 years compared to an average car ($)')
plt.ylabel('Frequency')
plt.title('Histogram for Annual spending/saving over the past 5 years')

#Showing the histogram
plt.show()

#Insights:
# 1. The major data distribution falls between the -12000(Spending) to␣
↪+2000(Saving) but the majority percentage of
# car owners tend to spend over the past 5 year period rather than saving.
```

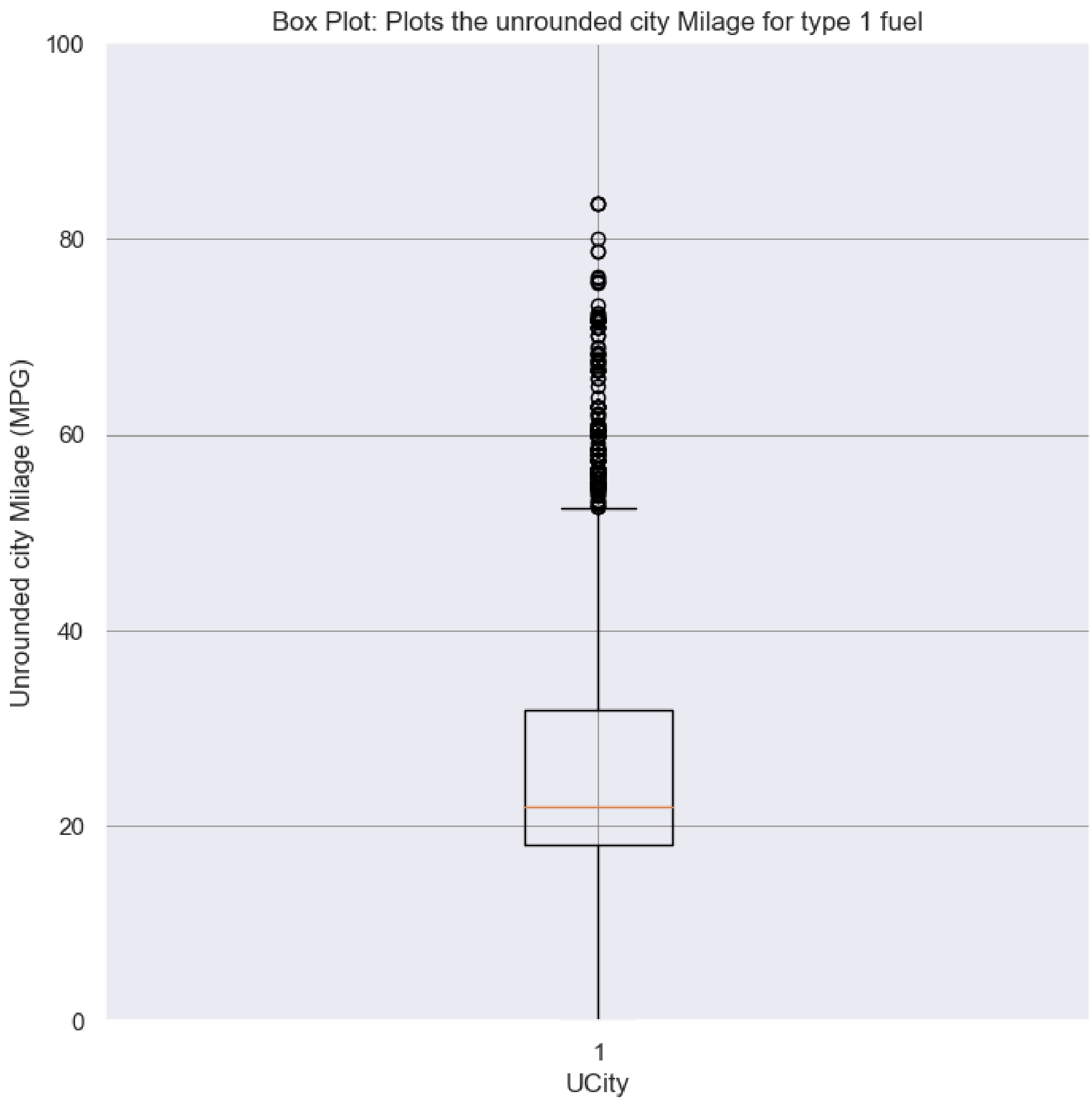
```
Chart-13: Histogram - comparison between the spending/saving
--------------------------------------------------------------------------
```



9 Chart Type: Box Chart

[45]:
```
#Chart 14: Box Chart to analyse the unrounded city Milage for type 1 fuel.

print('Chart-14: Box Chart - analyse the unrounded city Milage for type 1 fuel␣
↪')
print("-----------------------------------------------------------------------------")

# defining the dimentions and plotting the variable for the box chart
plt.figure(figsize=(8, 8))
plt.boxplot(df["UCity"])

#adding grids to the chart for better readability and asthetic appeal
plt.grid(True, linestyle='-', color='grey', linewidth=0.5)

#Customising the y axis limit
plt.ylim(0, 100)

# Adding labels and title
plt.xlabel('UCity')
plt.ylabel('Unrounded city Milage (MPG)')
plt.title('Box Plot: Plots the unrounded city Milage for type 1 fuel')

# Show the box plot
plt.show()

#Insights:
# 1. The average value of the milage for unrounded city for fuel type 1 is␣
↪approximately 21 Miles/Gallon, where falls
# the majority of the data.
# 2. There are few extreme outliers which are present when analyzing the graph,␣
↪for instance the data points containing
# the values above 39 upto 220 are some.
```

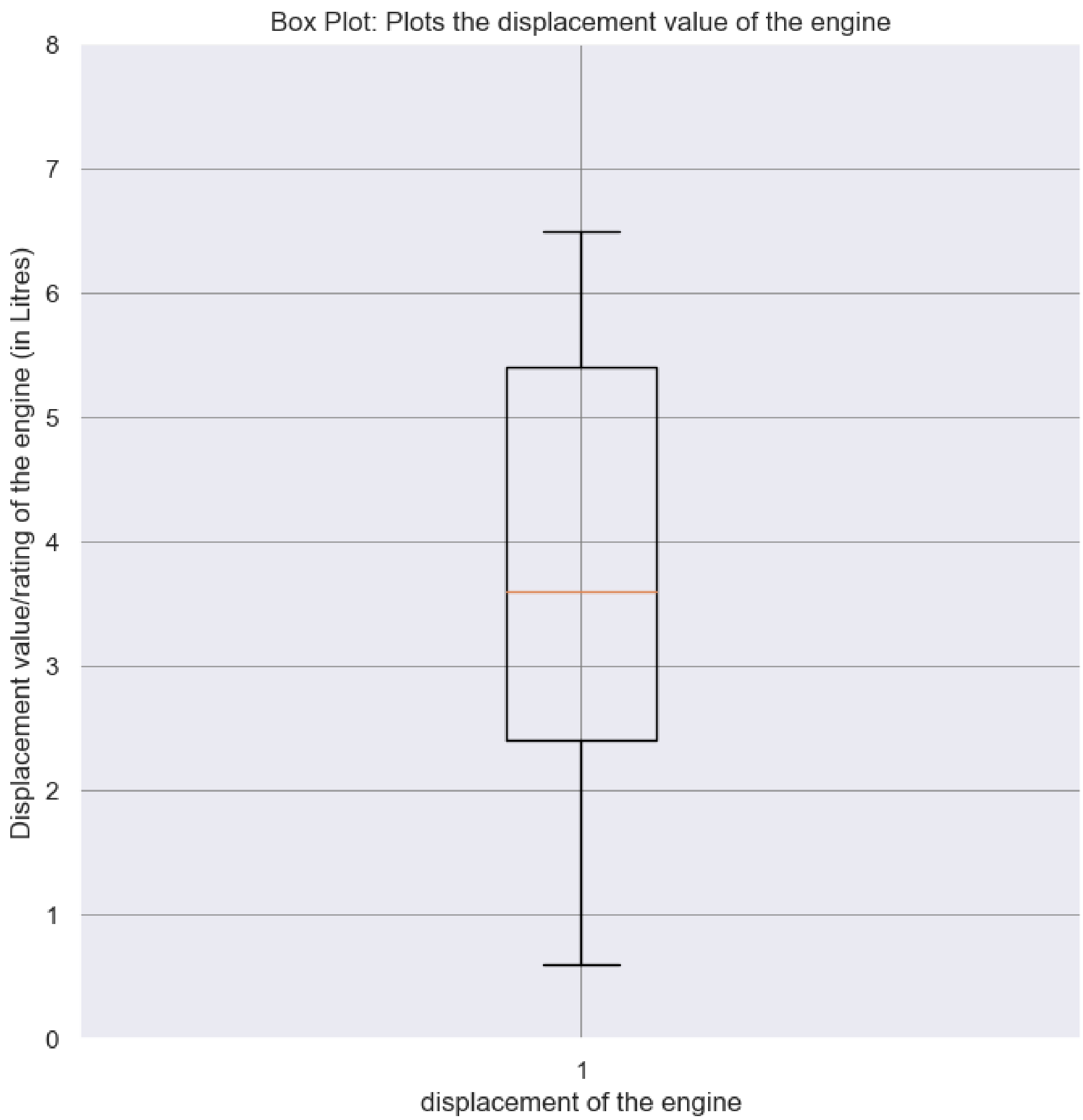
```
Chart-14: Box Chart - analyse the unrounded city Milage for type 1 fuel
-----------------------------------------------------------------------------
```



[46]:
```
#Chart 15: Box Chart to analyse the displacement of the engines for vehicles␣
↪under study.

print('Chart-15: Box Chart - analyse the displacement of the engines for␣
↪vehicles under study')
print("---------------------------------------------------------------------------'

# defining the dimentions and the variable for the box chart
plt.figure(figsize=(8, 8))
plt.boxplot(df["displ"])

#adding grids to the chart for better readability
```

```
plt.grid(True, linestyle='-', color='grey', linewidth=0.5)

#Setting up customised limits
plt.ylim(0, 8)

# Adding labels and title
plt.xlabel('displacement of the engine')
plt.ylabel('Displacement value/rating of the engine (in Litres)')
plt.title('Box Plot: Plots the displacement value of the engine')

# Show the box plot
plt.show()


#Insights:
# 1. The mean value of displacement of the engine is approximately 3.5 Litres
 ↪across all the vehicles under consideration.
# 2. The maximum range value of displacement is between 2.5-5.5 litres.
```

Chart-15: Box Chart - analyse the displacement of the engines for vehicles under study
----

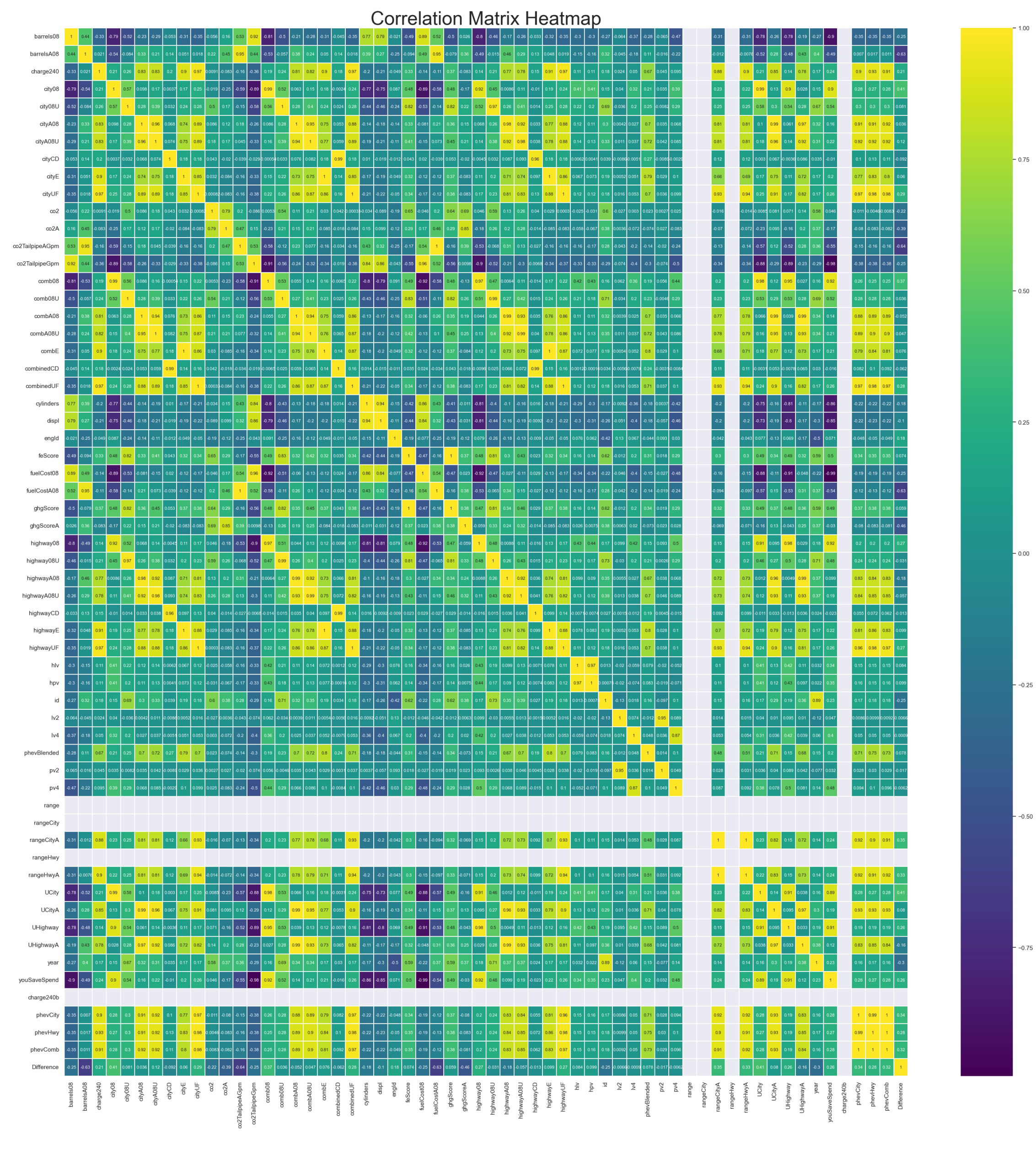


10 Question 3: Correlation Matrix

[48]:
```
#Creating the correlation matrix, which consists of only numerical columns
correlation_heatmap = df.corr(numeric_only = True)

print('Question 3: Correlation heatmap of the Vehicle dataset')
print("--------------------------------------------------------")

# Setting the dimensions of the figure
plt.figure(figsize=(35, 35))
```

```
#Plotting of the heatmap
sns.heatmap(correlation_heatmap, cmap='viridis', linewidth=.
 ↪25,annot=True,annot_kws={'size': 8})

#Printing the title of the heatmap
plt.title('Correlation Matrix Heatmap',fontsize=35)

#Displaying the heatmap
plt.show()
```

Question 3: Correlation heatmap of the Vehicle dataset



11 Question 4: Conclusion for Data analysis and visualization

[504]:
```
print("Final Conclusion & Insights on the Vehicle dataset Visualisation:")
print("-----------------------------------------------------------------")
print(" ")
print(" 1) It can be infered that, in a scenario where a vehicle with the␣
↪automatic axle had less fuel consumption which in reduced the expense on the␣
↪same. Also, even though the various brands or make used the same interior␣
↪composition like cylinder - Engine displacement combination, it tend to␣
↪produced contrasting results from each other on the Fuel efficiency and the␣
↪Fuel economy score.")
print(" ")
print(" 2) The most number of prefered type of vehicles where the compact and␣
↪the midsized cars which constituted more than 25% of the total cars which␣
↪were on display.")
print(" ")
print(" 3) When coming to ATVs, Electric vehicles gave a positive yield or␣
↪saved money on the fuel due to its cost. The same could not be said about␣
↪the Bifuel Vehicles as it incurred heavy spending when comparing over a␣
↪5-year period.")
print(" ")
print(" 4) Finally, There is found to be a positive correlation between the␣
↪fuel consumption with the Co2 emission as there"
"found to increase when ever there is an increase in the former. This act␣
↪inturn increase the total value/total miles travelled of the "
"but when looking indepth, the milage still decreases and the Fuel efficiency␣
↪score too.")
print(" ")
print("-----------X-------------X---------------X--------------X---------")
```

```
Final Conclusion & Insights on the Vehicle dataset Visualisation:
-----------------------------------------------------------------

 1) It can be infered that, in a scenario where a vehicle with the automatic
axle had less fuel consumption which in reduced the expense on the same. Also,
even though the various brands or make used the same interior composition like
cylinder - Engine displacement combination, it tend to produced contrasting
results from each other on the Fuel efficiency and the Fuel economy score.

 2) The most number of prefered type of vehicles where the compact and the
midsized cars which constituted more than 25% of the total cars which were on
display.

 3) When coming to ATVs, Electric vehicles gave a positive yield or saved money
```

on the fuel due to its cost. The same could not be said about the Bifuel Vehicles as it incurred heavy spending when comparing over a 5-year period.

4) Finally, There is found to be a positive correlation between the fuel consumption with the Co2 emission as therefound to increase when ever there is an increase in the former. This act inturn increase the total value/total miles travelled of the but when looking indepth, the milage still decreases and the Fuel efficiency score too.

-----------X-------------X---------------X--------------X---------