Predictive Analytics

ALY 6020, CRN 80405

Professor Vladimir Shapiro

Module 2: Assignment - Building the Car of the Future

Submitted By - Richa Umesh Rambhia

Implementation of Linear Regression - MPG Prediction

Table of Contents

Introduction

Linear Regression Algorithm

Machine Learning algorithms are classified into *supervised and unsupervised* learning, where supervised learning algorithms are further classified into **Classification & Regression** based problems. Classification problems deal with *categorical data* in order to **classify the classes** for the data points, whereas Regression problems are **prediction** based models that are *continuous* in nature and predict the output variable depending on the features of the data. [2]

Simple Linear Regression Model Linear regression models are simple methods that are used for the predictive analysis that show the linear relationship between the independent variable of the dataset and the target variable.

Multiple Linear Regression Model Simple linear regression models are used when there is only 1 independent variable to predict the target variable. However, multiple linear regression model is used when there are multiple independent variables in order to predict a single dependent variable. [2]

Problem Statement

A car manufacturer known for making large automobiles is struggling with sales and has asked for your help in designing an energy-efficient car. Using data gathered, determine which attributes may contribute to higher gas mileage so that they can design a more fuel-efficient automobile.

MPG Prediction for building the car

In this assignment, the goal is to **predict the MPG value**, i.e., miles per gallon based on the various attributes of the vehicle such that there can be recommendations made to the carmaker such that they could concentrate on those parameters in the future. In order to solve the problem of sales for the car manufacturer such that an *energy-efficient car* can be designed, it is important to understand what parameters are contributing in higher gas mileage, and thus we predict the MPG value based on the features and attributes of the dataset to understand what attributes of the car contribute to the MPG value such that the manufacturer can design and *build a fuel-efficient autombolie that would increase the sales percent.*

Analysis

Installing required packages

In [ ]:
```
!pip install pandas_profiling
!pip install featurewiz
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-w
heels/public/simple/ (https://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting pandas_profiling
  Downloading pandas_profiling-3.6.6-py2.py3-none-any.whl (324 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 324.4/324.4 kB 6.5 MB/s eta 0:00:00
Collecting ydata-profiling
  Downloading ydata_profiling-4.1.2-py2.py3-none-any.whl (345 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 345.9/345.9 kB 24.0 MB/s eta 0:00:00
Requirement already satisfied: seaborn<0.13,>=0.10.1 in /usr/local/lib/python3.9/dist-packages (from ydat
a-profiling->pandas_profiling) (0.12.2)
Collecting visions[type_image_path]==0.7.5
  Downloading visions-0.7.5-py3-none-any.whl (102 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.7/102.7 kB 9.1 MB/s eta 0:00:00
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in /usr/local/lib/python3.9/dist-packages (from ydata-p
rofiling->pandas_profiling) (6.0)
Requirement already satisfied: numpy<1.24,>=1.16.0 in /usr/local/lib/python3.9/dist-packages (from ydata-
profiling->pandas_profiling) (1.22.4)
Collecting typeguard<2.14,>=2.13.2
  Downloading typeguard-2.13.3-py3-none-any.whl (17 kB)
```

Importing libraries

In [2]:
```
import pandas as pd
import numpy as np
import pandas_profiling
import ydata_profiling
import matplotlib.pyplot as plt
import seaborn as sns
from featurewiz import featurewiz
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn import metrics
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

```
Imported version = 0.1.55.
from featurewiz import FeatureWiz
wiz = FeatureWiz(verbose=1)
X_train_selected = wiz.fit_transform(X_train, y_train)
X_test_selected = wiz.transform(X_test)
wiz.features  ### provides a list of selected features ###
```

Loading the dataset

In [3]:
```
car_data = pd.read_csv("car-1.csv")
car_data
```

Out[3]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	US Made
0	18.0	8	307.0	130	3504	12.0	70	1
1	15.0	8	350.0	165	3693	11.5	70	1
2	18.0	8	318.0	150	3436	11.0	70	1
3	16.0	8	304.0	150	3433	12.0	70	1
4	17.0	8	302.0	140	3449	10.5	70	1
...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1
394	44.0	4	97.0	52	2130	24.6	82	0
395	32.0	4	135.0	84	2295	11.6	82	1
396	28.0	4	120.0	79	2625	18.6	82	1
397	31.0	4	119.0	82	2720	19.4	82	1

398 rows × 8 columns

Table 1. Car Dataframe

Data Dictionary

(To understand each of the parameters of the dataset)

Attribute	Definition
MPG	Miles Per Gallon. Typically, the higher the number, the more fuel-efficient the vehicle is
Cylinders	The efficiency of how the motor goes through fuel, the more the more efficient
Displacement	Size of the motor
Horsepower	How powerful the motor is. Typically, the more horsepower, the less efficient
Weight	
Acceleration	How fast does it take the car to get to 100 MPH
Model Year	
US Made	

Table 2. Data Dictionary

Exploratory Data Analysis

EDA is performed on the data in order to analyze various parameters and features of the dataset and to understand the *structure* of the dataset such that various *trends and patterns* between the variables is known. Exploratory Data Analysis helps in understanding the *relationship between the various independent and dependent variables* of the dataset that would further be useful in building the model such as description analysis and statistical analysis.

Descriptive Analysis

In [ ]:
```
# displaying number of rows and columns
print("Total number of Rows and Columns:", car_data.shape)

print("\n-------------------------------------------------------")

# displaying field values/column names
print("\nColumn Names:\n")
car_data.columns
```

```
Total number of Rows and Columns: (398, 8)

-------------------------------------------------------

Column Names:
```

Out[4]:
```
Index(['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight',
       'Acceleration', 'Model Year', 'US Made'],
      dtype='object')
```

In [ ]:
```
# displaying data types
print("Data types:\n")
car_data.dtypes
```

```
Data types:
```

Out[5]:
```
MPG             float64
Cylinders         int64
Displacement    float64
Horsepower       object
Weight            int64
Acceleration    float64
Model Year        int64
US Made           int64
dtype: object
```

From the *descriptive analysis*, it is observed that there are total **398 rows of data** and **8 field values** and the data type for each of the field value is displayed in order to understand what data type values are present in the dataset.

Here, there are different types of data points that are present in the dataset which are **numerical data type** having *'int' and 'float'* values and remaining field values are of **object type**, which needs to be updated to **int/float** type as per the values present in the dataset, later in the preprocessing stage.

Statistical Analysis

In [ ]:
```
# dataset info
print("Dataset Info:\n")
car_data.info()
```

```
Dataset Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   MPG           398 non-null    float64
 1   Cylinders     398 non-null    int64
 2   Displacement  398 non-null    float64
 3   Horsepower    398 non-null    object
 4   Weight        398 non-null    int64
 5   Acceleration  398 non-null    float64
 6   Model Year    398 non-null    int64
 7   US Made       398 non-null    int64
dtypes: float64(3), int64(4), object(1)
memory usage: 25.0+ KB
```

In [ ]:
```
# describing the dataset
print("Describing the dataset:\n")
round(car_data.describe(),1)
```

```
Describing the dataset:
```

Out[7]:

	MPG	Cylinders	Displacement	Weight	Acceleration	Model Year	US Made
count	398.0	398.0	398.0	398.0	398.0	398.0	398.0
mean	23.5	5.5	193.4	2970.4	15.6	76.0	0.6
std	7.8	1.7	104.3	846.8	2.8	3.7	0.5
min	9.0	3.0	68.0	1613.0	8.0	70.0	0.0
25%	17.5	4.0	104.2	2223.8	13.8	73.0	0.0
50%	23.0	4.0	148.5	2803.5	15.5	76.0	1.0
75%	29.0	8.0	262.0	3608.0	17.2	79.0	1.0
max	46.6	8.0	455.0	5140.0	24.8	82.0	1.0

Table 3. Dataset Description

Statistical Analysis helps in understanding about each of the numerical field type based on the **total count values, minimum value, maximum value, standard deviation**, etc. giving an overall analysis of the field data points about the various rows present in the dataset.

For example, as observed in the car dataset, we see that there are multiple field values having the *minimum, maximum values* along with the *total count of values* which is **398** and *standard deviation* of the column values. It can be observed that the maximum value of *Cylinders* is **8.0** whereas the maximum value of *MPG* is **46.6**.

Thus, similarly, other parameters of the dataset can be analyzed based on their statistical values.

Data Profiling

In [50]:
```
car_data_report = car_data.profile_report(title='Car Data Analysis Report', explorative = True)
car_data_report
```

```
Summarize dataset:    0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:    0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:    0%|          | 0/1 [00:00<?, ?it/s]
```

Out[50]:

In [51]:
```
# Saving the profile report
car_data_report.to_file(output_file="Car Data Analysis Report.html")
```

```
Export report to file:    0%|          | 0/1 [00:00<?, ?it/s]
```

The data profiling report generated for the dataset helps in understanding various parameters such as the data type of the field values, the missing and duplicate values present in the dataset, the correlation between each of the field value, and the analysis of each of the field value on a individual basis based on correlation plot, histogram, and interaction graphs.

From the profiling report, it is observed that there are **5 numerical variable type and 3 categorical data type** field values present in the dataset of which the numerical data type have **integer and float values**. Also, there are **no missing values or duplicate values** present in the dataset, and the missing values visualization or plot also helps in understanding that there are no missing values present in the dataset, and for each field value a separate visualization is displayed in order to specifially analyze a particular field value.

Further cleaning of the data is implemented in the below steps.

Part 1:

Use proper data cleansing techniques to ensure you have the highest quality data to model this problem. Detail your process and discuss the decisions you made to clean the data. (Similar to that of week 1).

Data Cleaning

---

1. Checking for null values in each column of the dataset, i.e., missing or bad values
2. Replacing bad values or characters with correct values
3. Checking for data types & correcting the data type for the variables
4. Checking for outliers in the dataset

   a. *Boxplot*

   b. *Distribution Plot*

---

1. Checking for null values in each column of the dataset, i.e., missing or bad values

In [ ]:
```
# checking for missing or null values
for x in range(8):
    print("%-45s %10d" % (car_data.columns.values[x], car_data.iloc[:,x].isna().sum()))
```

```
MPG                                                    0
Cylinders                                              0
Displacement                                           0
Horsepower                                             0
Weight                                                 0
Acceleration                                           0
Model Year                                             0
US Made                                                0
```

The code above shows that there are **no missing values** present in the dataset. The **isna()** function is used in order to display and check the 'Null' or 'NA' values that are present in each of the field values of the dataset.

In [ ]:
```
# checking for unwanted characters
check_value = car_data['Horsepower'].str.contains('\?')
print(car_data[check_value])
```

```
      MPG  Cylinders  Displacement Horsepower  Weight  Acceleration  \
32   25.0          4          98.0          ?    2046          19.0
126  21.0          6         200.0          ?    2875          17.0
330  40.9          4          85.0          ?    1835          17.3
336  23.6          4         140.0          ?    2905          14.3
354  34.5          4         100.0          ?    2320          15.8
374  23.0          4         151.0          ?    3035          20.5

     Model Year  US Made
32           71        1
126          74        1
330          80        0
336          80        1
354          81        0
374          82        1
```

Here, the above code shows that there are unwanted characters present in the column of '**Horsepower**' that needs to be addressed before building the model, as the quality of data is important while training the model and any unwanted values or characters can affect the efficiency of the model.

2. Replacing bad values or characters with correct values

In [4]:
```
car_data['Horsepower'] = car_data['Horsepower'].replace('?', 0)
print("Replace successful.")
```

```
Replace successful.
```

Displaying the rows of the dataset after cleaning

In [ ]:
```
# displaying the starting rows of the dataset

print("Displaying the first 10 rows of data")
car_data.head()
```

```
Displaying the first 10 rows of data
```

Out[13]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	US Made
0	18.0	8	307.0	130	3504	12.0	70	1
1	15.0	8	350.0	165	3693	11.5	70	1
2	18.0	8	318.0	150	3436	11.0	70	1
3	16.0	8	304.0	150	3433	12.0	70	1
4	17.0	8	302.0	140	3449	10.5	70	1

Table 4. First 10 rows of the dataframe

In [ ]:
```
# displaying the end rows of the dataset

print("Displaying the last 10 rows of data")
car_data.tail()
```

```
Displaying the last 10 rows of data
```

Out[14]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	US Made
393	27.0	4	140.0	86	2790	15.6	82	1
394	44.0	4	97.0	52	2130	24.6	82	0
395	32.0	4	135.0	84	2295	11.6	82	1
396	28.0	4	120.0	79	2625	18.6	82	1
397	31.0	4	119.0	82	2720	19.4	82	1

Table 5. Last 10 rows of the dataframe

3. Checking for data types & correcting the data type for the variables

In [5]:
```
# correcting the data types for the variables of the dataset which are of object type to integer type

car_data['Horsepower'] = car_data['Horsepower'].astype(int)

print("Data Type conversion successful.")
```

```
Data Type conversion successful.
```

In [ ]:
```
# checking for the correct data type of the variable
print("Data types:\n")
car_data.dtypes
```

```
Data types:
```

Out[16]:
```
MPG             float64
Cylinders         int64
Displacement    float64
Horsepower        int64
Weight            int64
Acceleration    float64
Model Year        int64
US Made           int64
dtype: object
```

Here, the data type having **object type data** is converted to an appropriate data type, i.e., **integer data type**, whereas the field values with integer and float data type are kept the same as in the dataset which represents their correct data type.

4. Checking for outliers in the dataset

a. Boxplot

The below code creates **boxplots** for the various field values of the car dataset in order to check for outliers present in the dataset. Here, the boxplots are implemented for the variables **Cylinders**, **Displacement**, **Horsepower**, **Weight**, and **MPG**, as shown in the figures below. The outliers that are present in the dataset will not be removed as each of the data point is important for analysis and model building.

In [ ]:
```
# creating boxplot for 'Cylinders' and 'Displacement' variables

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].boxplot(car_data['Cylinders'])
axs[1].boxplot(car_data['Displacement'])
axs[0].set_title('Boxplot for Cylinders')
axs[1].set_title('Boxplot for Displacement')
axs[0].set_ylabel('Data Values')
axs[1].set_ylabel('Data Values')

plt.show()
```

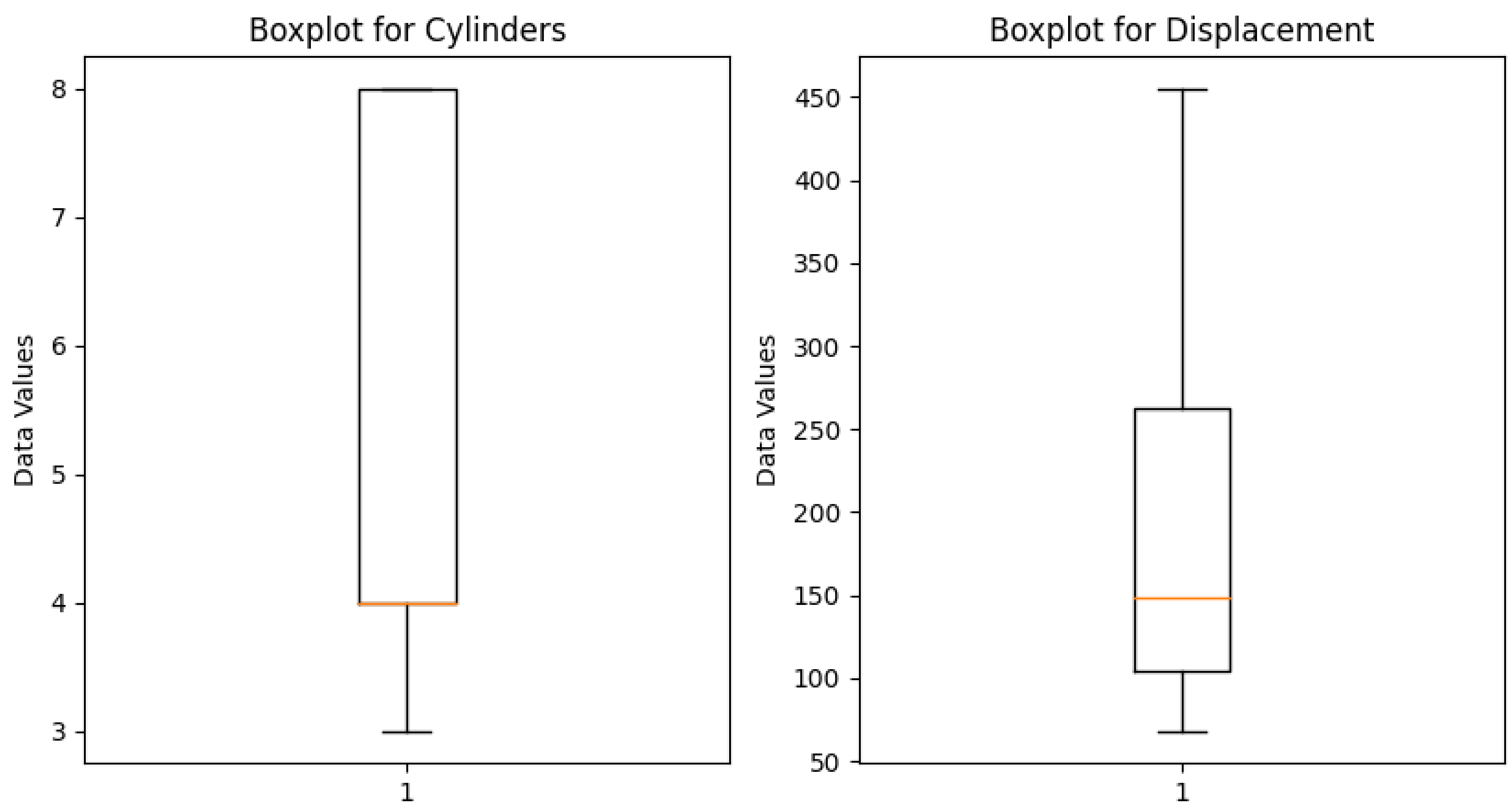


Figure 1. Boxplot for variables 'Cylinders' & 'Displacement'

In [ ]:
```
# creating boxplot for 'Horsepower' and 'Weight' variables

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].boxplot(car_data['Horsepower'])
axs[1].boxplot(car_data['Weight'])
axs[0].set_title('Boxplot for Horsepower')
axs[1].set_title('Boxplot for Weight')
axs[0].set_ylabel('Data Values')
axs[1].set_ylabel('Data Values')

plt.show()
```

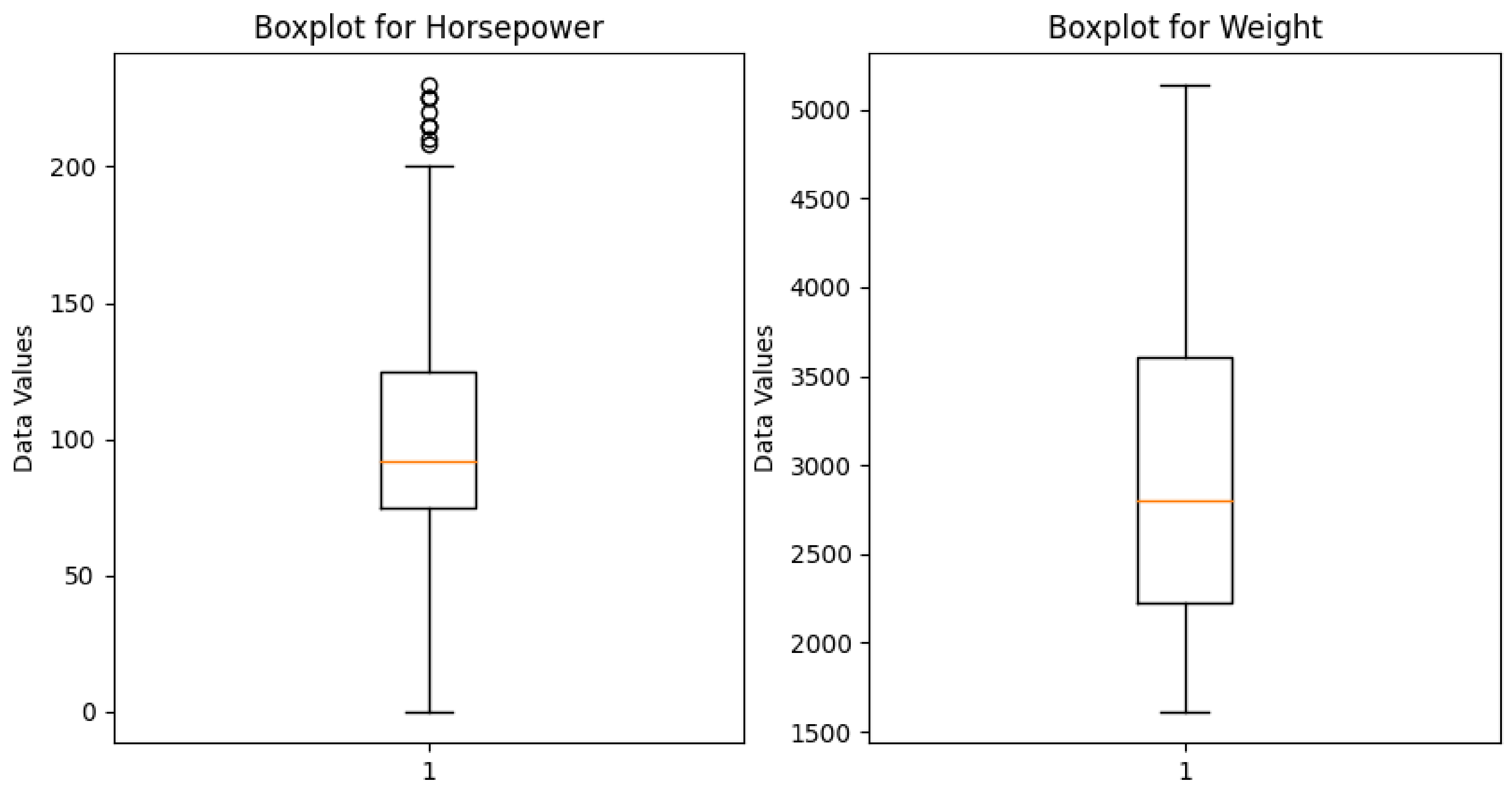


Figure 2. Boxplot for variables 'Horsepower' & 'Weight'

In [ ]:
```
# creating boxplot for 'MPG' variable

plt.boxplot(car_data['MPG'])
plt.title('Boxplot for MPG')
plt.ylabel('Data Values')

plt.show()
```

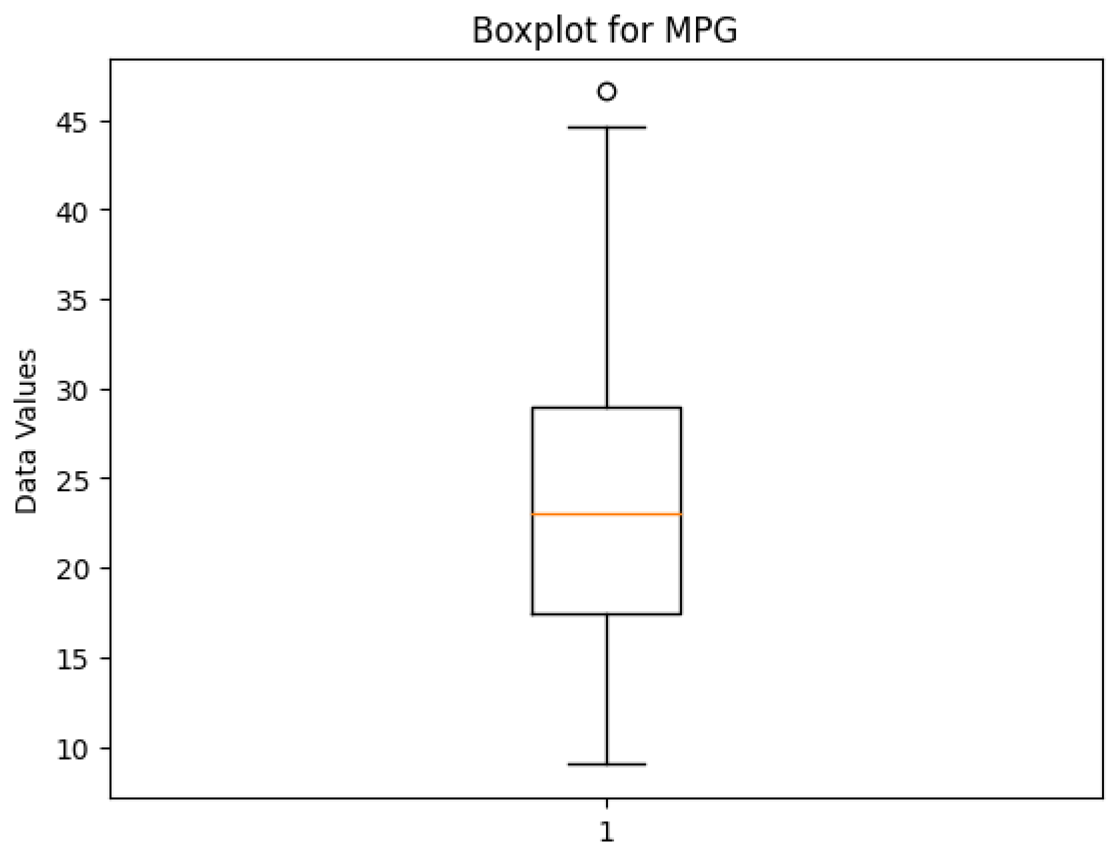


Figure 3. Boxplot for variable 'MPG'

Also, since it is known that machine learning models require *huge amount of data for training* in order to provide accuracte results, and because there is already **less amount of data points** present in our dataset, the outliers present are not excluded from the data and are considered for the training of the model.

b. Distribution Plot

The distribution plot for the various parameters of the dataset values gives an overview of the outliers that are present and the distribution of the data points across present in the dataset.

The plot below for variables *'Cylinders', 'Displacement', 'Horsepower', 'Weight', and 'MPG'* show that the data is **normally distributed** across the data points, meaning that the data points are evenly distributed around the mean value in the dataset.

In [ ]:
```
# distribution plot for Cylinder & Displacement

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(car_data['Cylinders'])
plt.subplot(1,2,2)
sns.distplot(car_data['Displacement'])
plt.show()
```

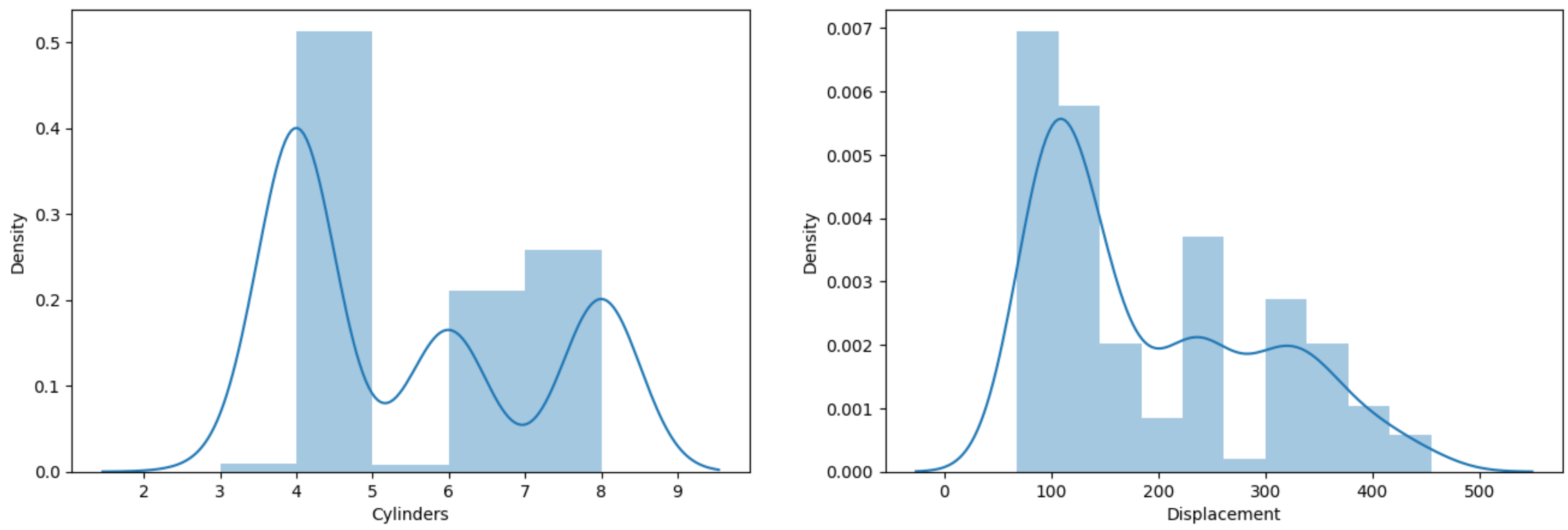


Figure 4. Distribution Plot for variables 'Cylinders' & 'Displacement'

In [ ]:
```
# distribution plot for Horsepower & Weight

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(car_data['Horsepower'])
plt.subplot(1,2,2)
sns.distplot(car_data['Weight'])
plt.show()
```

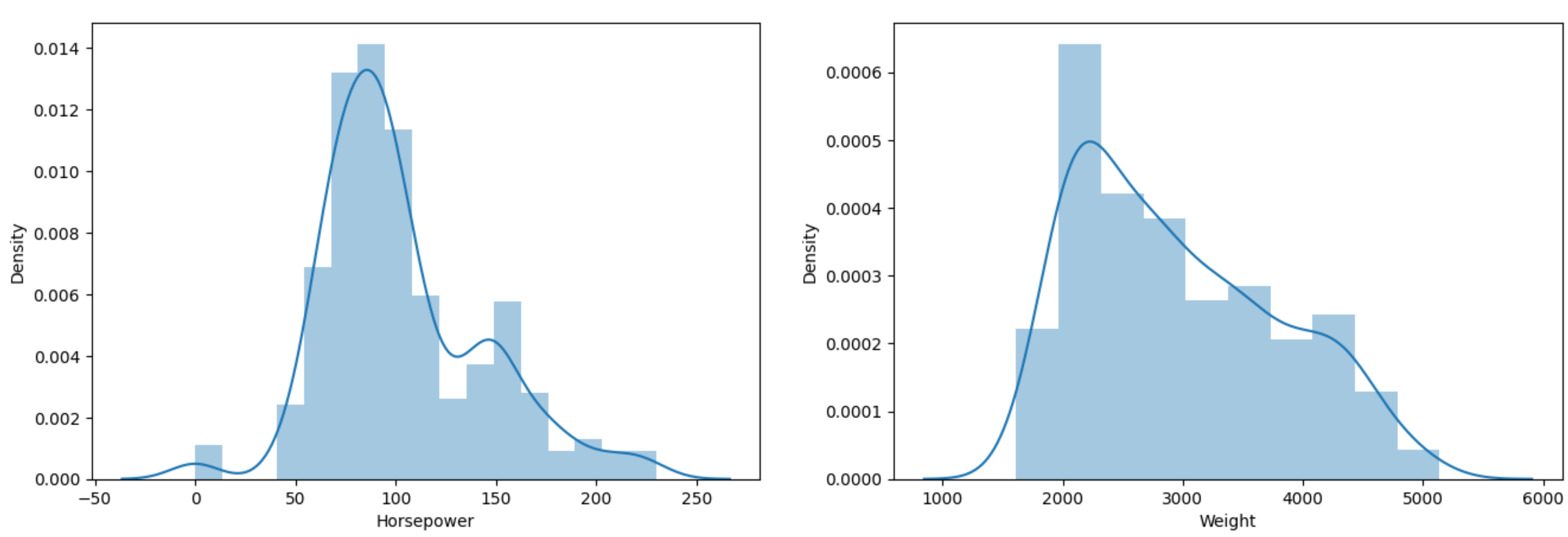


Figure 5. Distribution Plot for variables 'Horsepower' & 'Weight'

In [ ]:
```
# distribution plot for MPG

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(car_data['MPG'])
plt.show()
```

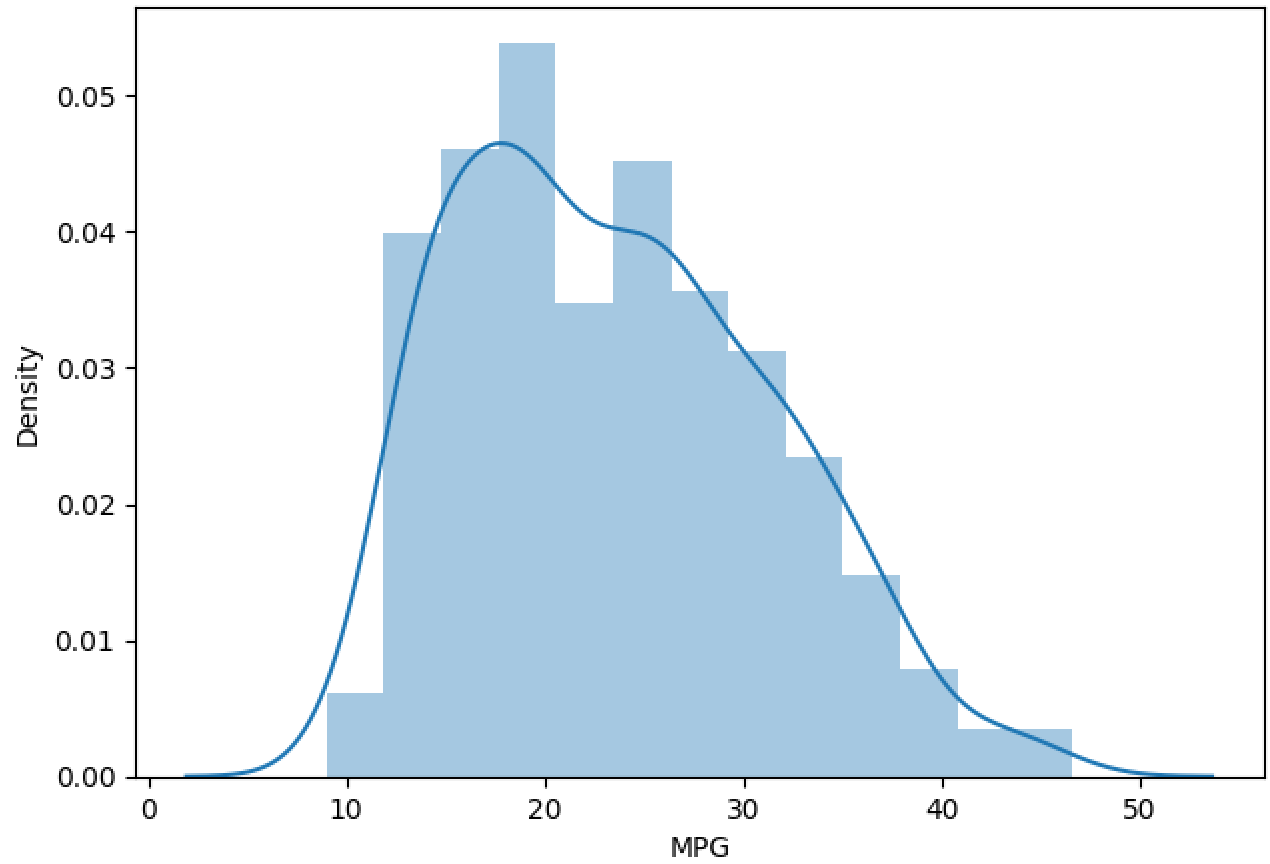


Figure 6. Distribution Plot for variable 'MPG'

Part 2:

Build a linear regression model to accurately predict miles per gallon (MPG) based on the attributes of a vehicle. Discuss the significant attributes and how they can help you build the proper car. This should help a manufacturer prioritize what to do with a car and how it will help the value.

Pre-Modeling Steps

1. Feature Selection & Extraction
2. Correlation Plot
3. Defining the features for model training
4. Spliting the dataset into train & test set

1. Feature Selection and Extraction

In [ ]:
```
# Feature Extraction

target = 'MPG'

features, train = featurewiz(car_data, target, corr_limit=0.7, verbose=2, sep=",",
header=0,test_data="", feature_engg="", category_encoders="")
```

```
############################################################################################
############       F A S T   F E A T U R E  E N G G    A N D    S E L E C T I O N ! ########
# Be judicious with featurewiz. Don't use it to create too many un-interpretable features! #
############################################################################################
featurewiz has selected 0.7 as the correlation limit. Change this limit to fit your needs...
Skipping feature engineering since no feature_engg input...
Skipping category encoding since no category encoders specified in input...
#### Single_Label Regression problem ####
    Loaded train data. Shape = (398, 8)
    Some column names had special characters which were removed...
#### Single_Label Regression problem ####
No test data filename given...
#######################################################################################
######################## C L A S S I F Y I N G  V A R I A B L E S ####################
#######################################################################################
        No variables were removed since no ID or low-information variables found in data set
GPU active on this device
    Tuning XGBoost using GPU hyper-parameters. This will take time...
    After removing redundant variables from further processing, features left = 7
```

The above code generated a feature selection & extraction report using the '**featurewiz**' function that helped in understanding which features are to be taken into consideration for the prediction of the MPG value where the target variable is the '**MPG**' column.

The features selected by the featurewiz function are as shown below.

In [ ]:
```
print("The extracted features are:")
features
```

```
The extracted features are:
```

Out[74]:
```
['Displacement', 'Model Year']
```

---

2. Corelation Plot

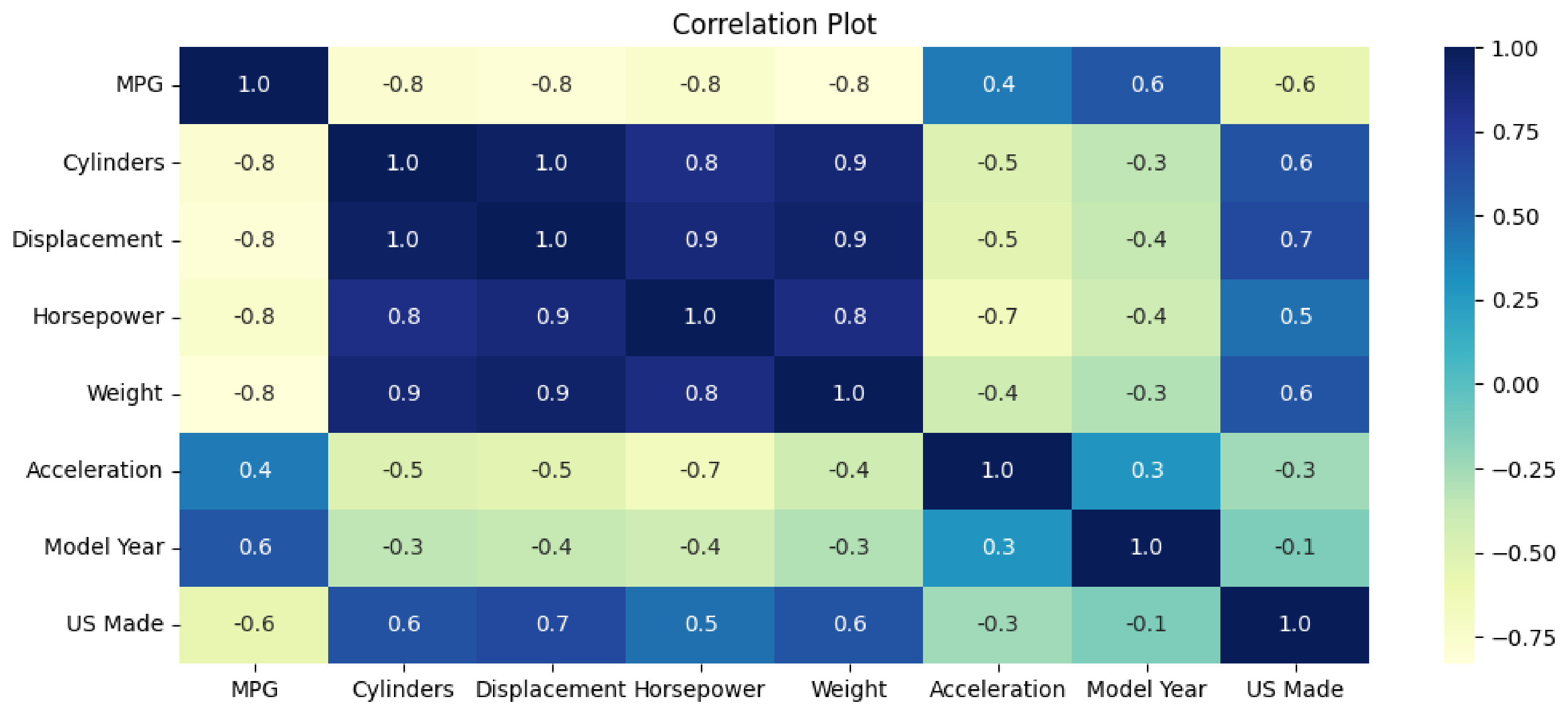
A **correlation plot** or matrix is a *visual representation of the variables* present in the dataset which helps in understanding the *relationship* between the different variables and how highly the variables are corelated to each other.

The values of the correlation plot range from **-1 to 1**, where -1 indicates a **negative correlation** between the variables, 0 indicates **no correlation**, and 1 indicates a **positive correlation**.

The variables that have positive correlation are said to be highly correlated to each and hence either of the two variables must be removed for the model building as it may lead to **multicollinearity** where the efficiency of the model may reduce.

In [ ]:
```
# plotting correlation matrix

plt.figure(figsize = (12,5))
ax = plt.subplot()
sns.heatmap(car_data.corr(),annot=True, fmt='.1f', ax=ax, cmap="YlGnBu")
ax.set_title('Correlation Plot');
```



	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	US Made
MPG	1.0	-0.8	-0.8	-0.8	-0.8	0.4	0.6	-0.6
Cylinders	-0.8	1.0	1.0	0.8	0.9	-0.5	-0.3	0.6
Displacement	-0.8	1.0	1.0	0.9	0.9	-0.5	-0.4	0.7
Horsepower	-0.8	0.8	0.9	1.0	0.8	-0.7	-0.4	0.5
Weight	-0.8	0.9	0.9	0.8	1.0	-0.4	-0.3	0.6
Acceleration	0.4	-0.5	-0.5	-0.7	-0.4	1.0	0.3	-0.3
Model Year	0.6	-0.3	-0.4	-0.4	-0.3	0.3	1.0	-0.1
US Made	-0.6	0.6	0.7	0.5	0.6	-0.3	-0.1	1.0

Figure 7. Correlation Matrix

Looking for corelations between independent & dependent variables

---

As observed in the correlation matrix above, we see that there are many variables or features that are *highly correlated* to each other and hence we need to analyze the features that are strongly correlated such that these features are excluded from the training of the model in order to avoid **multicollinearity** and *improve the efficiency of the model*. The following features are highly correlated with the other features in the dataset and can be excluded from model building.

Correlation among the variables:

1. **Cylinders**, **Displacement**, **Weight** and **Horsepower** are all highly correlated to each, with correlation values 1.0, 0.9, and 0.8
2. **US Made** is highly correlated to *Cylinders, Displacement, Weight, and Horsepower*
3. **Model Year** and **Acceleration** are correlated to each other with a correlation value of 0.3

Understanding the top features selected by Correlation Matrix

In [ ]:
```
corr_result = car_data.corr()
correlation_price = corr_result['MPG'].sort_values(ascending=False)
topfeatures = correlation_price[1:4]
print("The top features selected by correlation matrix for Price:")
print(topfeatures)
```

```
The top features selected by correlation matrix for Price:
Model Year      0.579267
Acceleration    0.420289
US Made        -0.568192
Name: MPG, dtype: float64
```

Lasso Regression to select the most important features for model training [4]

In [ ]:
```
A = car_data.drop(['MPG'], axis=1)
B = car_data['MPG']
lasso_result = Lasso(alpha=0.1)
lasso_result.fit(A, B)
coef = pd.Series(lasso_result.coef_, index=A.columns)
features_lasso = coef.abs().sort_values(ascending=False).head(3).index
print("The top three features selected by Lasso regression:")
print(features_lasso)
```

```
The top three features selected by Lasso regression:
Index(['US Made', 'Model Year', 'Acceleration'], dtype='object')
```

Features selected for model building

The features that are selected for the model building based on the **Feature Selection & Extraction, Correlation Plot, and Lasso Regression** are as follows:

Feature Selection & Extraction	Correlation Matrix	Lasso Regression
Displacement	Model Year	US Made
Model Year	Acceleration	Model Year
	US Made	Acceleration

Table 6. Features Extracted

3. Defining the features for model training

The model is trained & built on the below mentioned features that is selected from the analysis of the **Feature selection and extraction report, Correlation matrix, and Lasso regression.**

In [6]:
```
X = car_data[['Acceleration', 'Model Year', 'US Made', 'Displacement']]
y = car_data['MPG']
```

4. Splitting the dataset into train & test set

The dataset is split into training and testing data with a random split of **80%** train set and **20%** for test data.

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 42)
```

Model Building

Building the **Linear Regression** model to predict the price of the car based on the features selected for training.

Fitting the LR model

In [8]:
```
regressor_model = LinearRegression()
regressor_model.fit(X_train, y_train)
```

Out[8]:
```
▾ LinearRegression
LinearRegression()
```

Displaying the coefficients & intercepts after fitting the model

As observed from the below code, the coefficient values of the variables are either positive or negative, which indicates that the variables with **positive** value have a **positive relationship with the target variable** whereas values having a **negative sign** indicate that there is a **negative relationship between the independent variable and the target variable.** This helps in understanding which feature contributes in the prediction of the target variable, which in this case is this '**MPG**'.

In [ ]:
```
coefficients = pd.DataFrame(regressor_model.coef_, X.columns, columns=['Coefficient'])
print(coefficients)
print('Intercept: ', regressor_model.intercept_)
```

```
              Coefficient
Acceleration    -0.157292
Model Year       0.760637
US Made         -2.022210
Displacement    -0.047317
Intercept:  -21.492528165012267
```

Summary Report of the Linear Regression model [6]

The summary report of the LR model displays various results and scores of the machine learning model that helps in understanding if the trained model is efficient or not, which in this case is the Linear Regression model for prediction of the MPG value.

1. *R-squared:* It is the coefficient of determination that basically indicates the proportion of variation in the dependent variables. The **higher R-squared** value indicates a good fit model, and as it is observed from the summary report below, the R-squared value is **0.973**, which indicates it is a **good fit model** for the dataset.
2. *Adj. R-squared:* This is same as the R-squared value with a difference that while performing **multiple linear regression model** we consider the Adj. R-squared value as the addition of unnecessary variables to the model need a penalty and thus the R-squared value is adjusted for multiple features, else for a single independent variable, R-squared and Adj. R-squared value are the same. For the prediction of MPG value, the Adj. R-squared value is **0.973**.
3. *AIC & BIC values:* These values are used for the model robustness and the aim here is to minimize the AIC and BIC values in order to get an efficient model.
4. *Durbin-Watson:* This measure provides statistics for *autocorrelation in the residual*, which means that if the residual values are autocorrelated then the model will be biased which should not be the case, meaning that no value should be depending upon the other value. The ideal value for the Durbin-Watson test is from **0 to 4**, and here the Durbin-Watson test value comes to be equal to **2.122**, which is under the ideal score range. [6]
5. *coef & P>|t|:* The p-value in the summary report helps to understand which independent variables are significantly important for consideration and which are not. They are used to determine the significance of the predictor variables in the model.

   a. If **p-value is less** than the significance values of 0.05, it is considered to be **statistically significant**

   b. If the **p-value is greater** than the significance values of 0.05, it is **not considered statistically significant** which indicates that it is not contributing to the prediction of the target variables

- From the summary report below, it is observed that all the attributes considered for the training of the model have p-value less than 0.05, indicating that the attributes are statistically significant and are contributing to the prediction of the MPG value.
- Considering the statistically significant variables, there is a *positive coefficient* obtained for **Model Year** which has the **coefficient** value of 0.5089 and thus it implies that this feature will be contribute to the price of the car. The remaining three significant variables are negatively in relation with the target variable.

In [ ]:
```
model = sm.OLS(y_train, X_train).fit()
print(model.summary())
```

```
                                 OLS Regression Results                                
=======================================================================================
Dep. Variable:                    MPG   R-squared (uncentered):                   0.973
Model:                            OLS   Adj. R-squared (uncentered):              0.973
Method:                 Least Squares   F-statistic:                              2831.
Date:                Sat, 22 Apr 2023   Prob (F-statistic):                   7.35e-245
Time:                        20:26:40   Log-Likelihood:                         -899.16
No. Observations:                 318   AIC:                                      1806.
Df Residuals:                     314   BIC:                                      1821.
Df Model:                           4                                                  
Covariance Type:            nonrobust                                                  
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Acceleration    -0.2411      0.096     -2.501      0.013      -0.431      -0.051
Model Year       0.5089      0.023     22.029      0.000       0.463       0.554
US Made         -1.6535      0.637     -2.595      0.010      -2.907      -0.400
Displacement    -0.0537      0.003    -16.996      0.000      -0.060      -0.047
==============================================================================
Omnibus:                       34.044   Durbin-Watson:                   2.122
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               73.916
Skew:                           0.554   Prob(JB):                     8.90e-17
Kurtosis:                       5.086   Cond. No.                         630.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant. 
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:
```
y_pred = regressor_model.predict(X_test)
```

Evaluating the performance of the model

The various metrics for evaluating the performance of the Linear Regression model are *Mean Absolute Error, Mean Squared Error, Root Mean Squared Error, and R-squared value.*

1. **MAE:** Measures the average absolute difference between the actual values and the predicted values
2. **MSE:** Measures the average of the squared differences between the actual values and the predicted values
3. **RMSE:** Square root of the MSE value
4. **R-squared:** Measures proportion of variance in the dependent variable

In [ ]:
```
print("Model Evaluation of Linear Regression.")
print('Mean Absolute Error:', round(metrics.mean_absolute_error(y_test, y_pred),1))
print('Mean Squared Error:', round(metrics.mean_squared_error(y_test, y_pred),1))
print('Root Mean Squared Error:', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),1))
print("R-Squared value:", round(metrics.r2_score(y_test, y_pred),2))
```

```
Model Evaluation of Linear Regression.
Mean Absolute Error: 2.7
Mean Squared Error: 12.2
Root Mean Squared Error: 3.5
R-Squared value: 0.77
```

Residual Plot

In addition to the R-squared value, the residual plot helps to analyze and ensure that the data points are randomly distributed and have a constant variance.

In [ ]:
```
plt.scatter(y_pred, y_test - y_pred)
plt.xlabel('Predicted')
plt.ylabel('Residual')
plt.axhline(y=0, color='k', linestyle='--')
```

Out[32]: <matplotlib.lines.Line2D at 0x7f8e60749460>

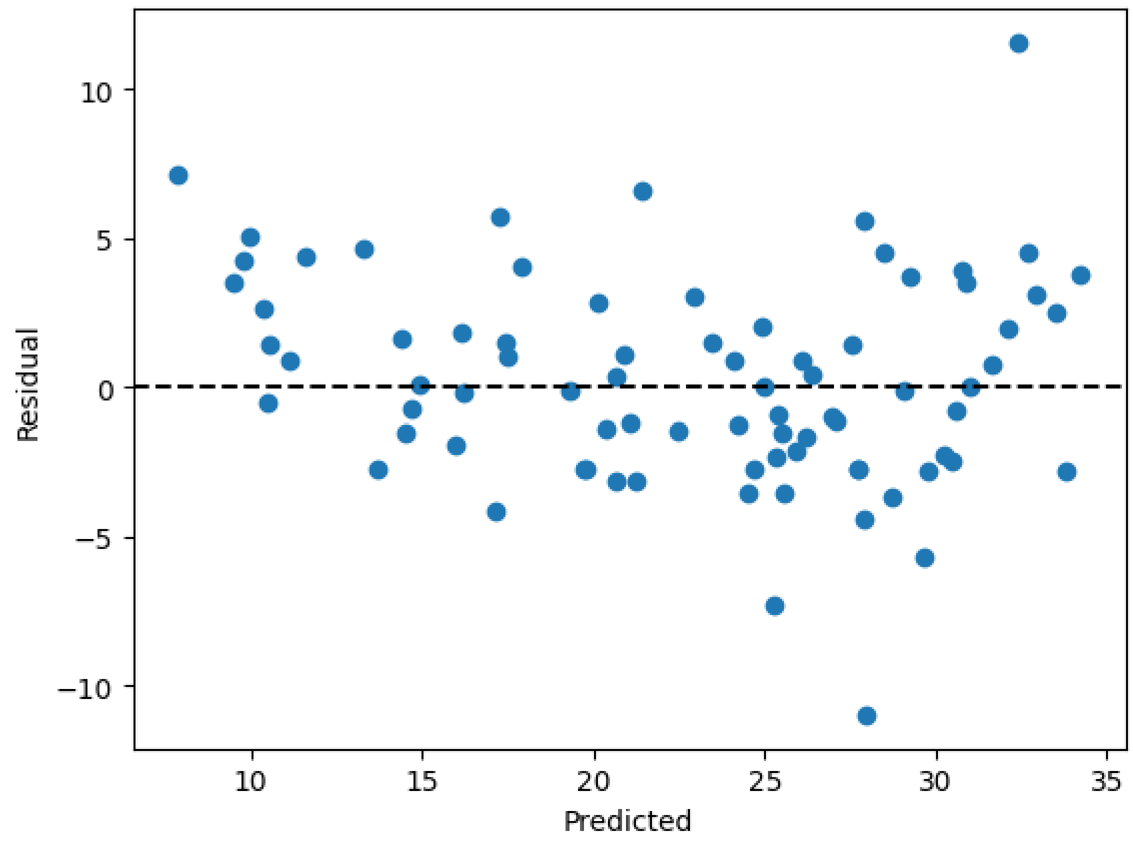


Figure 8. Residual Plot

In [ ]:
```
# accuracy of the model on training and testing set

print('Accuracy of Linear Regressor model on training set: {:.2f}'.format(regressor_model.score(X_train, y_tr
print('Accuracy of Linear Regressor model on test set:     {:.2f}'.format(regressor_model.score(X_test, y_tes

result_LR = regressor_model.score(X_test, y_test)
result_LR = round(result_LR,4)
print("Overall Accuracy of the model is: {:.2%}".format(result_LR))
```

```
Accuracy of Linear Regressor model on training set: 0.75
Accuracy of Linear Regressor model on test set:     0.77
Overall Accuracy of the model is: 77.31%
```

Overall Accuracy of the model is 77.3%

Accuracy on Training Data	Accuracy on Testing Data
75%	77%

The accuracy on training and testing data is nearly closeby, which indicates that the model efficiency is sort of good on both training and testing sets of data. Hence, there is **no issue of overfitting of the model.** Although, the accuracy slightly differs and is less for training data, there might be a possibility for **underfitting of the model** which can be *due to the less amount of data points present in the dataset.*

> Thus, based on the implementation of linear regression model for predicting the MPG value, the significant attributes are **Acceleration, Model Year, Displacement, and US Made**, of which **Model Year has a positive relation** with the MPG value indicating that this attribute highly contributes in the MPG value and thus the car manufacturer needs to consider this parameter while designing a new car as the newer car models have the most recent fuel-efficient design as compared to the older cars.

Part 3:

Run a model using forwards/backwards selections and review those results. See if they differ from Part 2 and compare them using metrics such as MSE or AIC.

Model Building with different features (Forward Selections) - Iteration 2

In this step, we run a model using the **forward selections** by *adding a new feature or attribute to the training model* in addition to the previously selected features, which in this case, the new feature added to the model is **Cylinders**.

The features selected are as follows:

1. Acceleration
2. US Made
3. Cylinders
4. Model Year
5. Displacement

Defining the features for model training

In [9]:
```
X1 = car_data[['Acceleration', 'US Made', 'Cylinders', 'Model Year', 'Displacement']]
y1 = car_data['MPG']
```

Splitting the dataset into train & test set

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.2, random_state = 42)
```

Fitting the LR model

In [11]:
```
regressor_model2 = LinearRegression()
regressor_model2.fit(X_train, y_train)
```

Out[11]:
```
▾ LinearRegression
LinearRegression()
```

Displaying the coefficients & intercepts after fitting the model

In [ ]:
```
coefficients = pd.DataFrame(regressor_model2.coef_, X1.columns, columns=['Coefficient'])
print(coefficients)
print('Intercept: ', regressor_model2.intercept_)
```

```
              Coefficient
Acceleration    -0.152927
US Made         -2.089328
Cylinders       -0.410800
Model Year       0.759842
Displacement    -0.040659
Intercept:  -20.505453561131787
```

Summary Report of the Linear Regression model

In [ ]:
```
model2 = sm.OLS(y_train, X_train).fit()
print(model2.summary())
```

```
                                 OLS Regression Results                                
=======================================================================================
Dep. Variable:                    MPG   R-squared (uncentered):                   0.973
Model:                            OLS   Adj. R-squared (uncentered):              0.973
Method:                 Least Squares   F-statistic:                              2277.
Date:                Sat, 22 Apr 2023   Prob (F-statistic):                   1.15e-243
Time:                        20:26:59   Log-Likelihood:                         -897.85
No. Observations:                 318   AIC:                                      1806.
Df Residuals:                     313   BIC:                                      1825.
Df Model:                           5                                                  
Covariance Type:            nonrobust                                                  
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Acceleration    -0.2269      0.097     -2.350      0.019      -0.417      -0.037
US Made         -1.7988      0.642     -2.802      0.005      -3.062      -0.536
Cylinders       -0.7102      0.442     -1.608      0.109      -1.579       0.159
Model Year       0.5275      0.026     20.457      0.000       0.477       0.578
Displacement    -0.0416      0.008     -5.137      0.000      -0.058      -0.026
==============================================================================
Omnibus:                       32.644   Durbin-Watson:                   2.104
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               72.500
Skew:                           0.523   Prob(JB):                     1.81e-16
Kurtosis:                       5.092   Cond. No.                         642.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

As observed from the summary report, the new feature added to the model building, i.e., **Cylinders** has a p-value of **0.109** which is less than the significance value of 0.05, and thus it can be concluded that the feature is considered **statistically significant** for the prediction of the MPG value. Although, the coefficient value of the attribute is negative and thus it is negatively related with the target variable.

In [ ]:
```
y_pred = regressor_model2.predict(X_test)
```

Evaluating the performance of the model

In [ ]:
```
print("Model Evaluation of Linear Regression.")
print('Mean Absolute Error:', round(metrics.mean_absolute_error(y_test, y_pred),1))
print('Mean Squared Error:', round(metrics.mean_squared_error(y_test, y_pred),1))
print('Root Mean Squared Error:', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),1))
print("R-Squared value:", round(metrics.r2_score(y_test, y_pred),2))
```

```
Model Evaluation of Linear Regression.
Mean Absolute Error: 2.7
Mean Squared Error: 11.9
Root Mean Squared Error: 3.4
R-Squared value: 0.78
```

Residual Plot

```
In [ ]: plt.scatter(y_pred, y_test - y_pred)
        plt.xlabel('Predicted')
        plt.ylabel('Residual')
        plt.axhline(y=0, color='k', linestyle='--')
```

```
Out[157]: <matplotlib.lines.Line2D at 0x7f8f26275820>
```

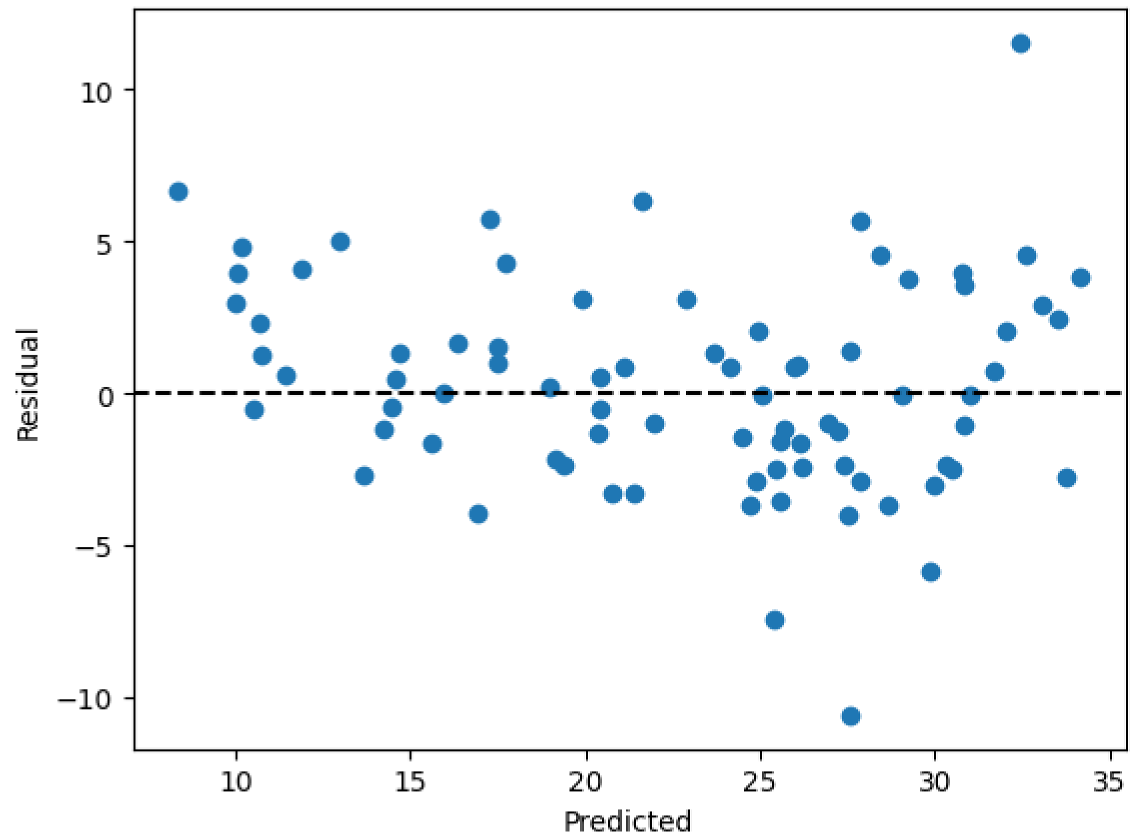


Figure 9. Residual Plot

```
In [ ]: # accuracy of the model on training and testing set

        print('Accuracy of Linear Regressor model on training set: {:.2f}'.format(regressor_model2.score(X_train, y_t
        print('Accuracy of Linear Regressor model on test set:     {:.2f}'.format(regressor_model2.score(X_test, y_te

        result_LR = regressor_model2.score(X_test, y_test)
        result_LR = round(result_LR,4)
        print("Overall Accuracy of the model is: {:.2%}".format(result_LR))
```

```
Accuracy of Linear Regressor model on training set: 0.75
Accuracy of Linear Regressor model on test set:     0.78
Overall Accuracy of the model is: 77.91%
```

Overall Accuracy of the model is 77.9%

Accuracy on Training Data	Accuracy on Testing Data
75%	78%

- The accuracy on training and testing data is nearly closeby, which indicates that the model efficiency is sort of good on both training and testing sets of data. Hence, there is **no issue of overfitting of the model.** Although, the accuracy slightly differs and is less for training data, there might be a possibility for **underfitting of the model** which can be due to the *less amount of data points present in the dataset.*
- As compared from Iteration 1 and Iteration 2, it is observed that adding a new feature to the model has slightly affected the performance of the model, and thus the results from iteration 2 differ slightly from iteration 1 where the model performance is better when an additional feature of the dataset is added.

Model Building with different features (Forward Selections) - Iteration 3

Further, in this step, we run a model using the **forward selections** by again *adding a new feature or attribute to the training model* in addition to the previously selected features, which in this case, the new feature added to the model is **Weight**.

The features selected are as follows:

1. Acceleration
2. US Made
3. Cylinders
4. Model Year
5. Displacement
6. Weight

Defining the features for model training

In [12]:
```
X2 = car_data[['Acceleration', 'US Made', 'Cylinders', 'Weight', 'Model Year', 'Displacement']]
y2 = car_data['MPG']
```

Splitting the dataset into train & test set

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X2, y2, test_size=0.2, random_state = 42)
```

Fitting the LR model

In [14]:
```
regressor_model3 = LinearRegression()
regressor_model3.fit(X_train, y_train)
```

Out[14]:
```
▾ LinearRegression
LinearRegression()
```

Displaying the coefficients & intercepts after fitting the model

In [ ]:
```
coefficients = pd.DataFrame(regressor_model3.coef_, X2.columns, columns=['Coefficient'])
print(coefficients)
print('Intercept: ', regressor_model3.intercept_)
```

```
              Coefficient
Acceleration     0.134022
US Made         -2.678088
Cylinders       -0.134196
Weight          -0.007301
Model Year       0.834213
Displacement     0.017218
Intercept:  -21.222857091198904
```

Summary Report of the Linear Regression model

```
In [ ]: model3 = sm.OLS(y_train, X_train).fit()
        print(model3.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                    MPG   R-squared (uncentered):                   0.980
Model:                            OLS   Adj. R-squared (uncentered):              0.980
Method:                 Least Squares   F-statistic:                              2609.
Date:                Sat, 22 Apr 2023   Prob (F-statistic):                  2.87e-263
Time:                        20:27:13   Log-Likelihood:                         -847.84
No. Observations:                 318   AIC:                                      1708.
Df Residuals:                     312   BIC:                                      1730.
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Acceleration     0.0560      0.087      0.645      0.519      -0.115       0.227
US Made         -2.3743      0.552     -4.300      0.000      -3.461      -1.288
Cylinders       -0.4455      0.379     -1.176      0.240      -1.191       0.300
Weight          -0.0073      0.001    -10.739      0.000      -0.009      -0.006
Model Year       0.5934      0.023     25.906      0.000       0.548       0.638
Displacement     0.0159      0.009      1.813      0.071      -0.001       0.033
==============================================================================
Omnibus:                       28.915   Durbin-Watson:                   2.192
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               48.275
Skew:                           0.562   Prob(JB):                     3.29e-11
Kurtosis:                       4.543   Cond. No.                     8.74e+03
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 8.74e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
In [ ]: y_pred = regressor_model3.predict(X_test)
```

Evaluating the performance of the model

```
In [ ]: print("Model Evaluation of Linear Regression.")
        print('Mean Absolute Error:', round(metrics.mean_absolute_error(y_test, y_pred),1))
        print('Mean Squared Error:', round(metrics.mean_squared_error(y_test, y_pred),1))
        print('Root Mean Squared Error:', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),1))
        print("R-Squared value:", round(metrics.r2_score(y_test, y_pred),2))
```

```
Model Evaluation of Linear Regression.
Mean Absolute Error: 2.3
Mean Squared Error: 8.3
Root Mean Squared Error: 2.9
R-Squared value: 0.85
```

Residual Plot

In [ ]:
```
plt.scatter(y_pred, y_test - y_pred)
plt.xlabel('Predicted')
plt.ylabel('Residual')
plt.axhline(y=0, color='k', linestyle='--')
```

Out[167]:
```
<matplotlib.lines.Line2D at 0x7f8f2524c370>
```

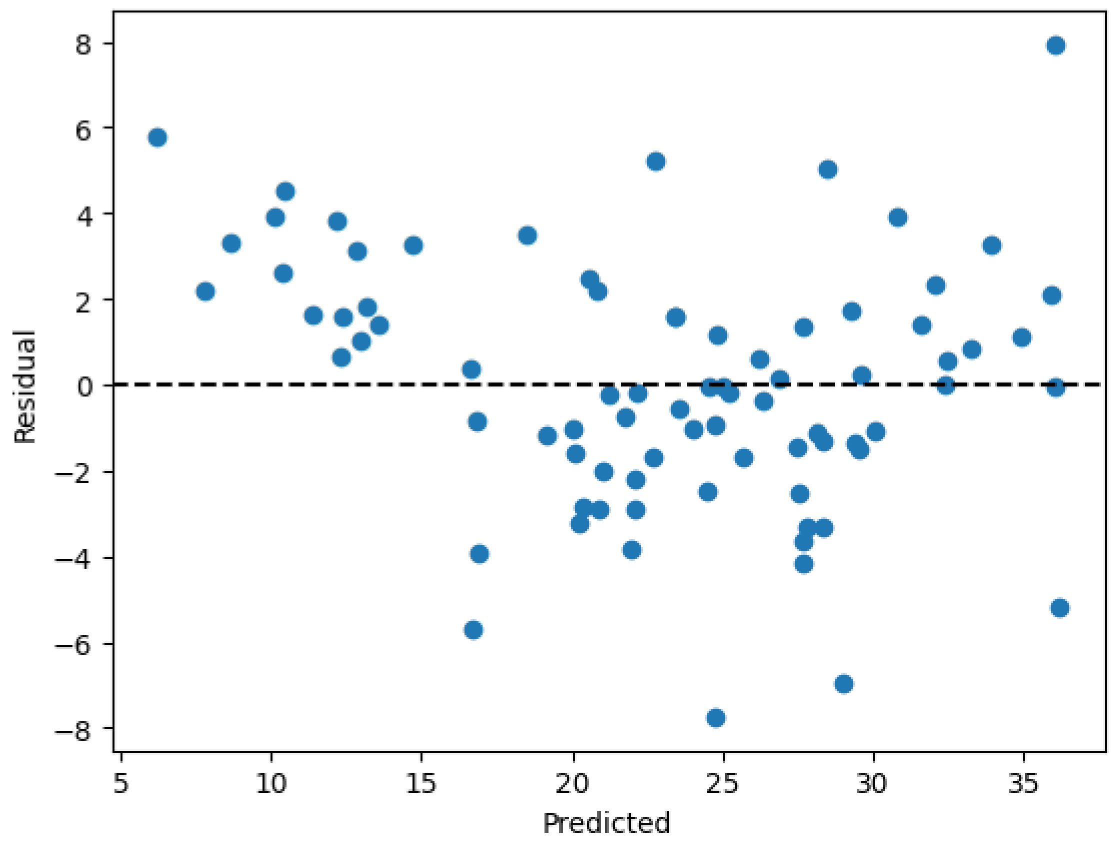


Figure 10. Residual Plot

In [ ]:
```
# accuracy of the model on training and testing set

print('Accuracy of Linear Regressor model on training set: {:.2f}'.format(regressor_model3.score(X_train, y_t
print('Accuracy of Linear Regressor model on test set:     {:.2f}'.format(regressor_model3.score(X_test, y_te

result_LR = regressor_model3.score(X_test, y_test)
result_LR = round(result_LR,4)
print("Overall Accuracy of the model is: {:.2%}".format(result_LR))
```

```
Accuracy of Linear Regressor model on training set: 0.82
Accuracy of Linear Regressor model on test set:     0.85
Overall Accuracy of the model is: 84.61%
```

Overall Accuracy of the model is 84.5%

Accuracy on Training Data	Accuracy on Testing Data
82%	85%

The accuracy on training and testing data is nearly closeby, which indicates that the model efficiency is sort of good on both training and testing sets of data. Hence, there is **no issue of overfitting of the model.** Although, the accuracy slightly differs and is less for training data, there might be a possibility for **underfitting of the model** which can be due to the *less amount of data points present in the dataset.*

Comparing results from Part 2 & Part 3

	Features Selected	Accuracy	R-sqaured value	MSE
Iteration 1	['Acceleration', 'Model Year', 'US Made', 'Displacement']	77.3%	0.77	12.2
Iteration 2	['Acceleration', 'Model Year', 'US Made', 'Displacement', 'Cylinders']	77.9%	0.78	11.9
Iteration 3	['Acceleration', 'Model Year', 'US Made', 'Displacement', 'Cylinders', 'Weight']	84.5%	0.85	8.3

Table 7. Comparing results from various iterations

> From the above table, we clearly can understand and analyze the efficiency of the model based on various parameters. It is observed that the features selected for the training of the model play an important role in the prediction of the MPG value, in a way that at each addition feature being added to the model training, the accuracy of the model would slightly increased. The highest accurate model obtained is for Iteration 3 where the features selected are Acceleration, Model Year, US Made, Displacement, Cylinders, and Weight.

> Also considering, the R-squared value and MSE value, it can be concluded that iteration 3 gave better results as compared to iteration 1 and 2, and thus we can recommend the car manufacturers to utilize this model for the prediction of the MPG value to understand the features contributing to design and build a fuel efficient vehicle.

> However, the model build can be optimized in terms of accuracy and performance. The quantity and quality of data, both play an important role in the machine learning model training phase, and thus in this case the quantity of data can be increased such that the model can have more data points for training, which will increase the efficiency of the model for prediction.

Part 4:

A clear conclusion for the carmaker would be to understand what they should prioritize in upcoming developments based on your results. Tie that into how the car industry works and how competitors have done similar tasks (this will involve research unless you really know cars).

Results

- The results from the evaluation metrics help in understanding what attributes are contributing to the predicition of the MPG value in order to recommend to the car manufacturer such that they can priortize the design for developing cars to build a fuel-efficient design.
- Considering the car industry that is constantly evolving and the task of prioritizing the fuel efficiency in vehicles has become the important feature while designing and building the car. It is important for the manufacturers to analyze and understand what parameters are to be taken into consideration that will help in improving the fuel efficiency, and thus many manyfacturers are constantly working on this where some of them have implemented electric or hybrid car design models whereas others are trying to improve the efficiency of the car engine.
- Based on the analysis and reports generated, the implementation of linear regression model for predicition of the MPG value has provided us with results that could be useful in understanding the features while designing the car for fuel efficient system.

Considering the results from Iteration 3, we can conclude that this model can be recommended to the car manufacturers as the performance of the model in iteration 3 was slightly better. The observations from the summary and evalution report are as follows.

1. As observed from the summary report, the **highest positive coefficient** value obtained is for the variable Model Year with a value of **0.5934**, which indicates that it has a *positive relation with the MPG value*. **The highest the Model Year of the car, the higher the MPG value is**, which means that it contributes in designing of a fuel efficient vehicle. Thus, the car manufacturer should prioritize the **year of the car model** parameter in the upcoming developments.

2. With respect to the remaining features of the car dataset that are taken into consideration for model training, **Acceleration and Displacement** have a positive relation with respect to the MPG value, whereas **Cylinders, Weight, and US Made** have a negative relation or coefficient value with respect to the target variable. But since the **p-value** for the **Acceleration** variable is **more than 0.05**, it is *not statistically significant*, hence the feature is not considered for the prediction of the MPG value.

3. Displacement has p-value less than 0.05, which means that it is statistically significant and contributes in the prediction of the target variable. This indicates that **higher the displacement values**, **higher is the MPG value**. Hence, it is recommended to the car manufacturer to consider the displacement values while designing the vehicle, which means that the size of the motor needs to *higher which will in turn increase the value of MPG*, making it a fuel efficient car.

4. The attributes with negative coefficient values are also considered while analyzing the prediction of the MPG value which are, **US Made, Cylinders, and Weight**, of which **Cylinders** has a p-value greater than 0.05 making it **statistically insignificant** in the prediction of the MPG value. Since, the attributes to be considered for the prediction are the values having p-value less than 0.05. we consider the features US Made and Weight for the analysis.

5. The attributes **US Made and Weight** have a coefficient value of **-2.3743** and **-0.0073** respectively, which indicates that lower the values of these attribites, higher will be the MPG value.

Conclusion

- **Linear Regression models**

  Linear regression models are the simplest method for modeling the relationship between the dependent variable and one or more independent variables which can be used to make predictions that will identify the most significant variable that contributes to the prediction of the target variable. These models assume a linear relationship between the independent and dependent variables and can be used to solve regression based problems.

- **MPG value Prediction using LR model**

  The MPG value prediction is implemented using the linear regression model which helps in understanding which parameters or features are contributing in the prediction of the MPG value, thus helping the car manufacturers in designing a fuel efficient car based on its attributes. The various parameters present in the dataset helps in analyzing the various features that could be useful in predicting the MPG value.

- **Model Performance & Accuracy**

  The model performance and accuracy is determined with the help of various evaluation metrics such as **R-squared value, MSE, MAE, etc**. which help in understanding if the model is accurate enough in order to predict the MPG value based on the features selected for training. As observed above, from the three iterations implemented, we see that the R-squared value obtained in iteration 3 is highest with a value of **0.85** which is close to 1 indicating that the model is accurate in predicting the MPG value. The highest accuracy obtained for the *train and test* dataset is in *iteration 3* with accuracy value of **82% and 85% respectively** indicating that the model can be used to predict the MPG value based on the features that are selected.

- **Recommendations**
  - Based on the *accuracy and coefficient values* obtained in **iteration 3**, we conclude that the model is not *overfitted nor underfitted* and hence can be used by the car manufacturers to predict the MPG value as the accuracy of the train and test dataset is almost closeby which is **82% and 85%** respectively, even though there are slight chances of the model being *underfitted* which is due to the **quantity of the data** that can be increased such that the accuracy on the training set increases.
  - The coefficient values along with the p-values in the summary report for the model indicate that the **Model Year** parameter has a positive coefficient value which means that the feature has a positive relationship with the MPG value and thus is contributing strongly in the prediction.
  - The analysis show that the car manufacturer should priortize the design of cars considering factors such as **lower weight, high acceleration, and higher displacement value**. Also, the model year is important to be considered as higher the value of the year is, the higher the MPG value is, i.e., the latest model year will have a fuel efficient vehicle and thus this parameter can be useful while analysis and designing of the vehicle.
  - The applications of data analysis and machine learning are well applicable in the car industry and competitors have been using the same in order to improve the performance of the vehicle and improve their designs such that an efficient vehicle can be built. For example, considering the MPG value prediction, it explains how data analysis and machine learning methods have been applied in order to analyze the features and attributes of the car such that a fuel efficient vehicle can be designed.

- **Future Scope**

  Despite the Linear Regression model is accurately able to predict the MPG value based on the features that are selected, the accuracy of the model can still be improved based on the new features selected along with eliminating the issue of multicollinearity. The issue of underfitting of the model or overfitting can be avoided by having a good quality and quantity of dataset for training model.

References

[1] Auto-mpg dataset. (2017, July 2). Kaggle. https://www.kaggle.com/datasets/uciml/autompg-dataset (https://www.kaggle.com/datasets/uciml/autompg-dataset)

[2] Mali, K. (2022). Everything you need to Know about Linear Regression! Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/ (https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/)

[3] Luna, Z. (2022, January 6). Feature Selection in Machine Learning: Correlation Matrix | Univariate Testing | RFECV. Medium. https://medium.com/geekculture/feature-selection-in-machine-learning-correlation-matrix-univariate-testing-rfecv-1186168fac12 (https://medium.com/geekculture/feature-selection-in-machine-learning-correlation-matrix-univariate-testing-rfecv-1186168fac12)

[4] Great Learning Team. (2023, January 12). What is LASSO Regression Definition, Examples and Techniques. Great Learning Blog: Free Resources What Matters to Shape Your Career! https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/#:~:text=Lasso%20regression%20is%20a%20regularization,i.e.%20models%20with%20fewer%20parameters (https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/#:~:text=Lasso%20regression%20is%20a%20regularization,i.e.%20models%20with%20fewer%20parameters)).

[5] Mawardi, D. (2018, May 29). Linear Regression in Python - Towards Data Science. Medium. https://towardsdatascience.com/linear-regression-in-python-9a1f5f000606 (https://towardsdatascience.com/linear-regression-in-python-9a1f5f000606)

[6] Mahmood, M. S. (2022, April 24). Simple Explanation of Statsmodel Linear Regression Model Summary. Medium. https://towardsdatascience.com/simple-explanation-of-statsmodel-linear-regression-model-summary-35961919868b (https://towardsdatascience.com/simple-explanation-of-statsmodel-linear-regression-model-summary-35961919868b)