```
# Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


# Load the dataset

df = pd.read_csv("/content/CarPrice_Assignment.csv")


# Display the first few rows of the dataset

df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewl
0	1	3	alfa-romero giulia	gas	std	two	convertible	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	
3	4	2	audi 100 ls	gas	std	four	sedan	
4	5	2	audi 100ls	gas	std	four	sedan	

5 rows × 26 columns

```
# Check the shape (number of rows and columns) of the dataset

df.shape
```

```
(205, 26)
```

```
# Display information about the dataset (data types, non-null counts, etc.)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

```
# Check for missing values in the dataset
```

```
df.isnull().sum()
```

```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

```
# Get the column names

df.columns
```

```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

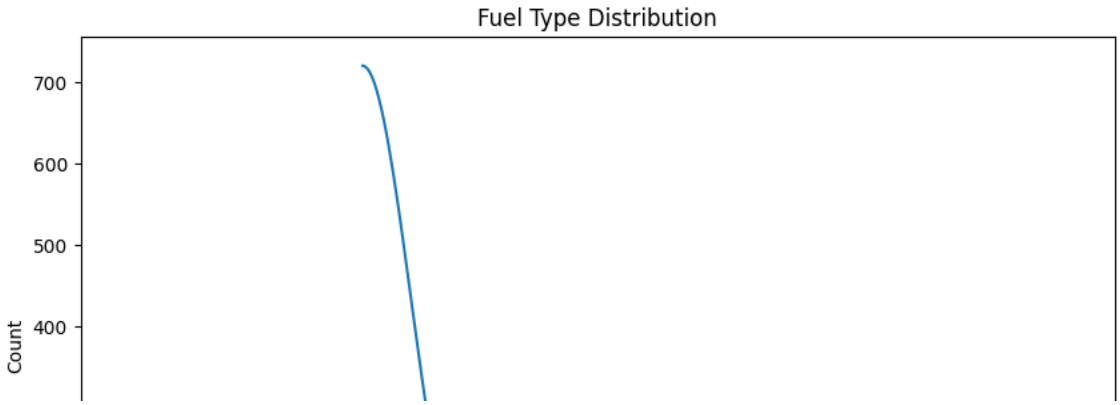
```
# Descriptive statistics of the dataset

df.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweig
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.5658
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.6802
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.0000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.0000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.0000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.0000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.0000

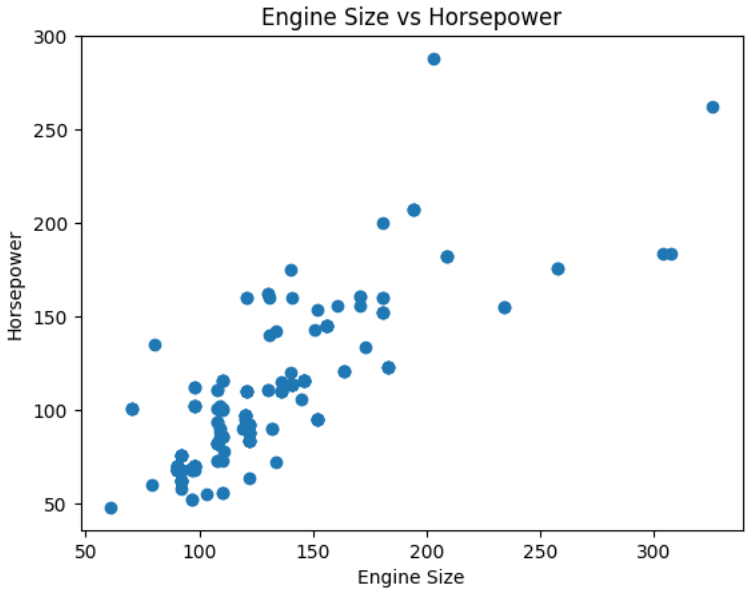
```
# Visualize the distribution of the 'fueltype' column using a histogram

plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="fueltype", bins=20, kde=True)
plt.title("Fuel Type Distribution")
plt.show()
```




```
# Create a scatter plot of 'enginesize' vs. 'horsepower'

plt.scatter(df['enginesize'], df['horsepower'])
plt.xlabel('Engine Size')
plt.ylabel('Horsepower')
plt.title('Engine Size vs Horsepower')
plt.show()
```




```
# Encode categorical variables using LabelEncoder

le = LabelEncoder()
var_mod = df.select_dtypes(include='object').columns

for i in var_mod:
    df[i] = le.fit_transform(df[i])


# Split the dataset into training and testing sets

X = df.drop(['price'], axis=1)
y = df['price']
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.20)


# Create a Linear Regression model

model = LinearRegression()
model.fit(x_train, y_train)
```

▾ LinearRegression
LinearRegression()

```
# Make predictions on the test set

y_pred = model.predict(x_test)


# Calculate Mean Squared Error and R-squared for evaluation

mse = mean_squared_error(y_test, y_pred)
```

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)


print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 6470794.75295691
R-squared: 0.9220495970783854
```

```
# Define new car features for prediction

new_car_features = [4000, 0, 2, 3, 0, 96.0, 172.0, 65.4, 2221, 120, 4, 3.46, 3.19, 9.0, 68, 5500, 31, 38, 0, 0, 0, 0, 0, 0, 0]


# Predict the price for the new car features

new_car_price = model.predict([new_car_features])
print("Predicted Price:", new_car_price[0])
```

```
Predicted Price: 21974014.45782642
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
```