Supervised Learning- Regression

Car Pricing</h2>

Problem Description: A Chinese automobile company aspires to enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts.

In [13]:
```python
# importing necessary libraries
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn import linear_model
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```python
# loading the dataset
df=pd.read_csv("C://Users//EliteBook//Desktop//ds//project//CarPr
df
```

Out[2]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody
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
...	...	...	...	...	...	...	...
200	201	-1	volvo 145e (sw)	gas	std	four	sedan
201	202	-1	volvo 144ea	gas	turbo	four	sedan
202	203	-1	volvo 244dl	gas	std	four	sedan
203	204	-1	volvo 246	diesel	turbo	four	sedan
204	205	-1	volvo 264gl	gas	turbo	four	sedan

205 rows × 26 columns

Exploratory Analysis</h2>

In [4]:
```python
df.shape
```

Out[4]:
```
(205, 26)
```

In [5]:
```python
df.head()
```

Out[5]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	dri
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

In [6]:
```python
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
```

In [7]:
```python
df.columns
```

Out[7]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiratio
n',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation',
'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'engin
etype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio',
'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'h
ighwaympg',
       'price'],
      dtype='object')
```

In [8]:
```python
# Count missing values in each column
df.isnull().sum()
```

Out[8]:
```
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
```

In [11]:
```python
# Convert categorical variables using Label Encoding
label_encoder = LabelEncoder()
categorical_cols = ['fueltype', 'aspiration', 'doornumber', 'carb
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])
```

In [12]:
```python
# Display the cleaned dataset
df.head()
```

Out[12]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drive
0	1	3	alfa-romero giulia	1	0	1	0	
1	2	3	alfa-romero stelvio	1	0	1	0	
2	3	1	alfa-romero Quadrifoglio	1	0	1	2	
3	4	2	audi 100 ls	1	0	0	3	
4	5	2	audi 100ls	1	0	0	3	

5 rows × 26 columns

In [25]:
```python
# Split the 'CarName' column into 'CarName' and 'CompanyName'
df[['CarName', 'CompanyName']] = df['CarName'].str.split(' ', 1, 
```

In [26]:
```python
df.head()
```

Out[26]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivew
0	1	3	alfa-romero	1	0	1	0	
1	2	3	alfa-romero	1	0	1	0	
2	3	1	alfa-romero	1	0	1	2	
3	4	2	audi	1	0	0	3	
4	5	2	audi	1	0	0	3	

5 rows × 27 columns

In [27]:
```python
for col in df.describe(include = 'object').columns:
    print(col)
    print(df[col].unique())
```

```
CarName
['alfa-romero' 'audi' 'bmw' 'chevrolet' 'dodge' 'honda' 'isuzu'
'jaguar'
 'maxda' 'mazda' 'buick' 'mercury' 'mitsubishi' 'Nissan' 'nissan'
 'peugeot' 'plymouth' 'porsche' 'porcshce' 'renault' 'saab' 'suba
ru'
 'toyota' 'toyouta' 'vokswagen' 'volkswagen' 'vw' 'volvo']
CompanyName
['giulia' 'stelvio' 'Quadrifoglio' '100 ls' '100ls' 'fox' '5000'
'4000'
 '5000s (diesel)' '320i' 'x1' 'x3' 'z4' 'x4' 'x5' 'impala' 'monte
carlo'
 'vega 2300' 'rampage' 'challenger se' 'd200' 'monaco (sw)' 'colt
hardtop'
 'colt (sw)' 'coronet custom' 'dart custom' 'coronet custom (sw)'
'civic'
 'civic cvcc' 'accord cvcc' 'accord lx' 'civic 1500 gl' 'accord'
 'civic 1300' 'prelude' 'civic (auto)' 'MU-X' 'D-Max ' 'D-Max V-C
ross'
 'xj' 'xf' 'xk' 'rx3' 'glc deluxe' 'rx2 coupe' 'rx-4' '626' 'glc'
 'rx-7 gs' 'glc 4' 'glc custom l' 'glc custom' 'electra 225 custo
m'
 'century luxus (sw)' 'century' 'skyhawk' 'opel isuzu deluxe' 'sk
ylark'
 'century special' 'regal sport coupe (turbo)' 'cougar' 'mirage'
'lancer'
 'outlander' 'g4' 'mirage g4' 'montero' 'pajero' 'versa' 'gt-r'
'rogue'
 'latio' 'titan' 'leaf' 'juke' 'note' 'clipper' 'nv200' 'dayz' 'f
uga'
 'otti' 'teana' 'kicks' '504' '304' '504 (sw)' '604sl' '505s turb
o diesel'
 'fury iii' 'cricket' 'satellite custom (sw)' 'fury gran sedan'
'valiant'
 'duster' 'macan' 'panamera' 'cayenne' 'boxter' '12tl' '5 gtl' '9
9e'
 '99le' '99gle' None 'dl' 'brz' 'baja' 'r1' 'r2' 'trezia' 'tribec
a'
 'corona mark ii' 'corona' 'corolla 1200' 'corona hardtop'
 'corolla 1600 (sw)' 'carina' 'mark ii' 'corolla' 'corolla liftba
ck'
 'celica gt liftback' 'corolla tercel' 'corona liftback' 'starle
t'
 'tercel' 'cressida' 'celica gt' 'rabbit' '1131 deluxe sedan' 'mo
del 111'
 'type 3' '411 (sw)' 'super beetle' 'dasher' 'rabbit custom' '145
e (sw)'
 '144ea' '244dl' '245' '264gl' 'diesel' '246']
```

In [15]:
```python
len(df.CarName.unique())
```

Out[15]:
```
147
```

In [16]:
```python
df.describe()
```

Out[16]:

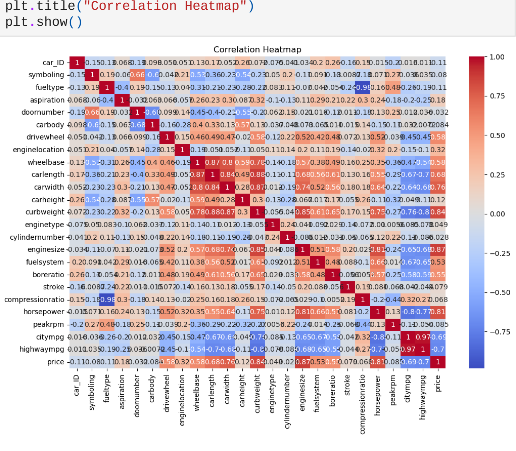
	car_ID	symboling	fueltype	aspiration	doornumber	carbody	c
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	2
mean	103.000000	0.834146	0.902439	0.180488	0.439024	2.614634	
std	59.322565	1.245307	0.297446	0.385535	0.497483	0.859081	
min	1.000000	-2.000000	0.000000	0.000000	0.000000	0.000000	
25%	52.000000	0.000000	1.000000	0.000000	0.000000	2.000000	
50%	103.000000	1.000000	1.000000	0.000000	0.000000	3.000000	
75%	154.000000	2.000000	1.000000	0.000000	1.000000	3.000000	
max	205.000000	3.000000	1.000000	1.000000	1.000000	4.000000	

8 rows × 25 columns

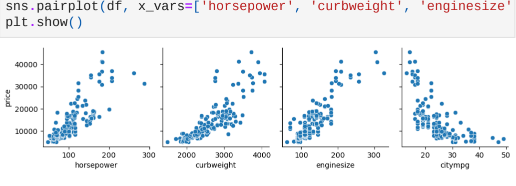
Explore relationships between variables using visualizations.

In [17]:
```python
# Correlation matrix
corr_matrix = df.corr()
```

In [18]:
```python
# Heatmap of correlations
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



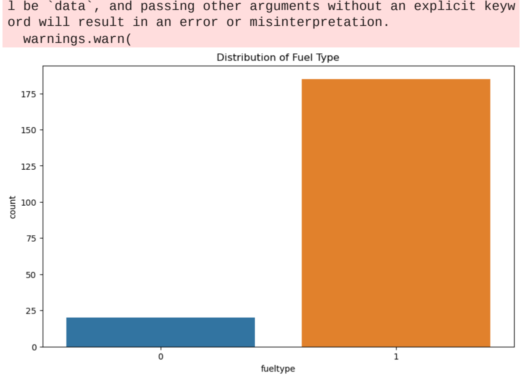
In [28]:
```python
# Scatter plots
sns.pairplot(df, x_vars=['horsepower', 'curbweight', 'enginesize'
plt.show()
```



Analyze categorical variables and their distribution.

In [29]:
```python
# Countplot of a categorical variable
plt.figure(figsize=(10, 6))
sns.countplot(df['fueltype'])
plt.title("Distribution of Fuel Type")
plt.show()
```

```
C:\Users\EliteBook\anaconda3\lib\site-packages\seaborn\_decorator
s.py:36: FutureWarning: Pass the following variable as a keyword
arg: x. From version 0.12, the only valid positional argument wil
l be `data`, and passing other arguments without an explicit keyw
ord will result in an error or misinterpretation.
  warnings.warn(
```



Based on our analysis, we found the following distribution of fuel types in the dataset:

- Gas (Petrol) Cars: [belowe 25]
- Diesel Cars: [Above 175]

Data Preprocessing</h2>

In [30]:
```python
# There are non numeric values for some columns. They should be c
le = preprocessing.LabelEncoder()
le.fit(df.fueltype)
df.fueltype = le.transform(df.fueltype)
```

In [31]:
```python
le.fit(df.aspiration)
df.aspiration = le.transform(df.aspiration)
```

In [32]:
```python
le.fit(df.doornumber)
df.doornumber = le.transform(df.doornumber)
```

In [33]:
```python
le.fit(df.carbody)
df.carbody = le.transform(df.carbody)
```

In [34]:
```python
le.fit(df.drivewheel)
df.drivewheel = le.transform(df.drivewheel)
```

In [35]:
```python
le.fit(df.enginelocation)
df.enginelocation = le.transform(df.enginelocation)
```

In [37]:
```python
le.fit(df.enginetype)
df.enginetype = le.transform(df.enginetype)
```

In [38]:
```python
le.fit(df.cylindernumber)
df.cylindernumber = le.transform(df.cylindernumber)
```

In [39]:
```python
le.fit(df.fuelsystem)
df.fuelsystem = le.transform(df.fuelsystem)
```

In [42]:
```python
df.head()
```

Out[42]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivew
0	1	3	alfa-romero	1	0	1	0	
1	2	3	alfa-romero	1	0	1	0	
2	3	1	alfa-romero	1	0	1	2	
3	4	2	audi	1	0	0	3	
4	5	2	audi	1	0	0	3	

5 rows × 27 columns

In [41]:
```python
df[df.duplicated()]
```

Out[41]:

car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewh

0 rows × 27 columns

Modelling using Linear Regression</h2>

In [43]:
```python
reg = linear_model.LinearRegression()

ind_var = df[[ 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'w
       'carlength', 'carwidth', 'carheight', 'curbweight', 'engin
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'hi
dep_var = df.price

reg.fit(ind_var,dep_var)
```

Out[43]:
```
LinearRegression()
```

In [44]:
```python
reg.coef_
```

Out[44]:
```
array([-9.21677014e+03,  2.42352286e+02, -5.13930613e+02, -8.1913
9857e+02,
        1.16616708e+03,  1.03974427e+04,  1.10593867e+02, -2.9110
2195e+01,
        7.12602036e+02,  1.84232716e+02,  2.35662037e+00,  1.9094
8576e+02,
        1.28166071e+02,  1.01622195e+02, -1.48294905e+02, -2.7445
0799e+03,
       -2.82460445e+03, -5.48092812e+02,  2.06295041e+01,  2.1044
0991e+00,
       -1.45461967e+02,  1.35183217e+02])
```

In [45]:
```python
reg.intercept_
```

Out[45]:
```
-49313.25002503388
```

In [51]:
```python
reg.predict([[1,0,1,0,2,0,88.6,168.8,64.1,48.8,2548,0,0,13,0,1,0,
```

```
C:\Users\EliteBook\anaconda3\lib\site-packages\sklearn\base.py:45
0: UserWarning: X does not have valid feature names, but LinearRe
gression was fitted with feature names
  warnings.warn(
```

Out[51]:
```
array([12962.15203553])
```

In [49]:
```python
df['price'][0]
```

Out[49]:
```
13495.0
```

Summery</h2>

The predicted price is similar to actual price. So the model works fine