ML ASSIGNMENT 2

Implementation of Linear and Multiple Linear Regression Models to Predict a Continuous Output and Evaluate Their Performance

In [2]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Load CSV

In [11]:
```
df = pd.read_csv("D:\Machine Learning\CarData\CarPrice_Assignment.csv")
df.head()
```

Out[11]:

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

5 rows × 26 columns

Understanding Data

In [12]:
```
df.shape
```

Out[12]: (205, 26)

In [13]:
```
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 [14]:
```
df.describe()
```

Out[14]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	engin
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.00
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.90
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.64
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.00
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.00
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.00
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.00
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.00

Check Missing Values

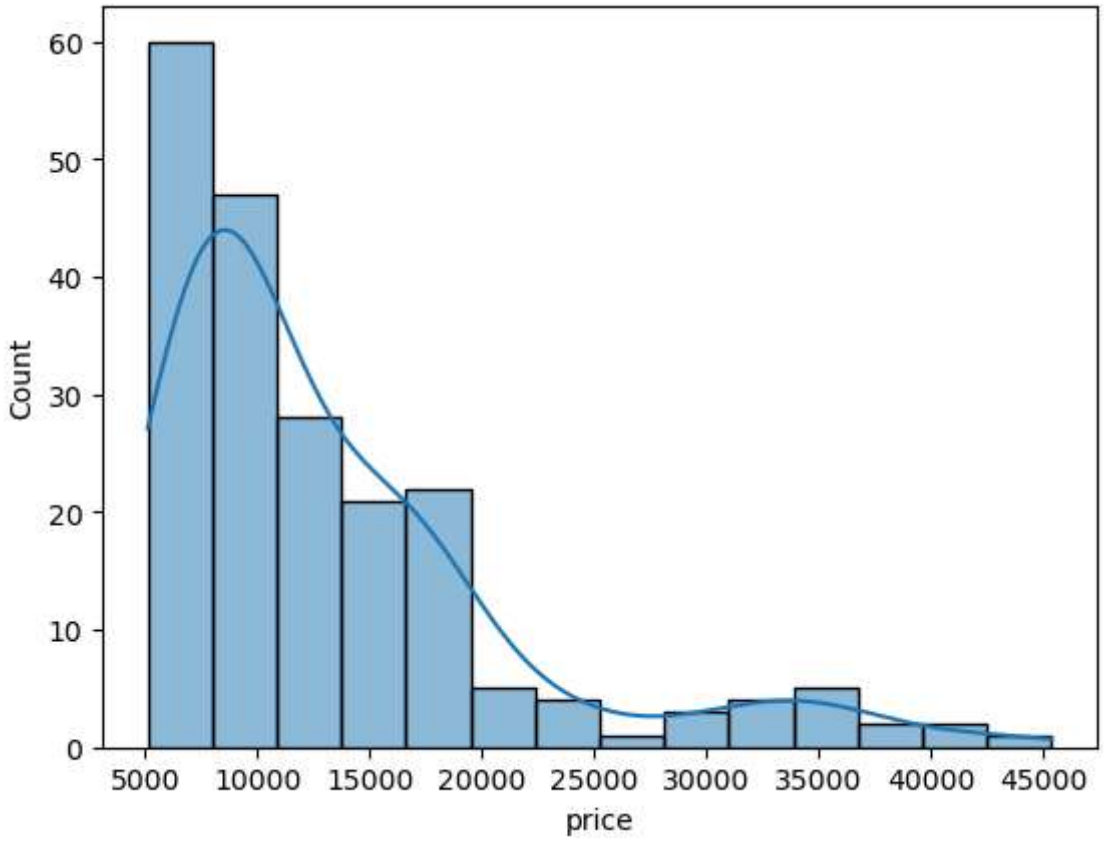
In [15]:
```
# Check null values
df.isnull().sum()
```

Out[15]:
```
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
```

EDA (Exploratory Data Analysis)

1. Target Value Distribution

In [17]:
```
sns.histplot(df['price'], kde=True)
# sns.title("Car Price Distribution")
plt.show()
```

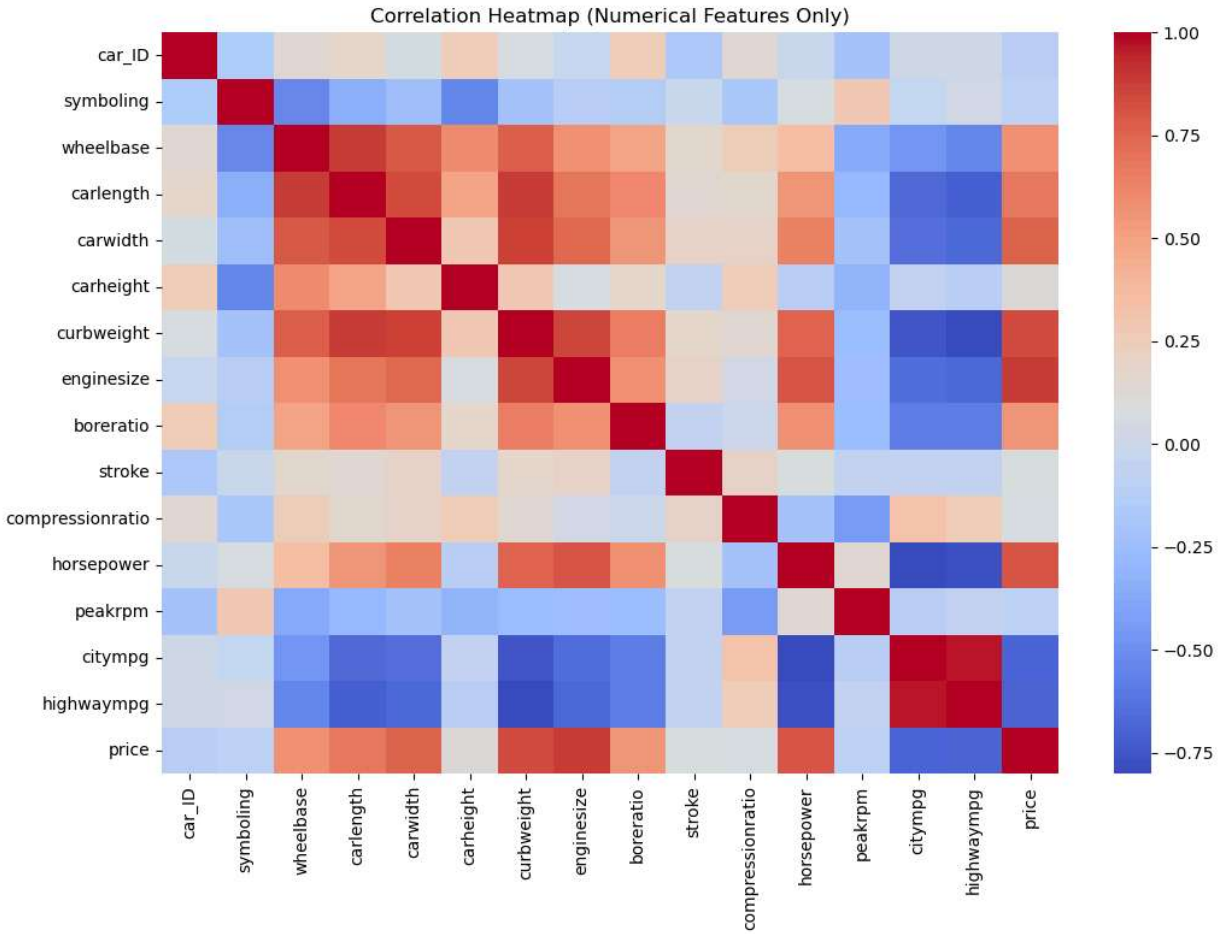


2. Correlation Heatmap

In [20]:
```
# plt.figure(figsize=(12,8))
# sns.heatmap(df.corr(), cmap='coolwarm')
# plt.title("Correlation Heatmap")
# plt.show()

numeric_df = df.select_dtypes(include=['int64', 'float64'])

plt.figure(figsize=(12,8))
sns.heatmap(numeric_df.corr(), cmap='coolwarm', annot=False)
plt.title("Correlation Heatmap (Numerical Features Only)")
plt.show()
```



Data Preprocessing

Drop Unnecessary Columns

In [22]:
```
df.drop(['car_ID', 'CarName'], axis = 1, inplace=True)
```

Handle Categorical Variables

In [23]:
```
categorical_cols = df.select_dtypes(include='object').columns
categorical_cols
```

Out[23]:
```
Index(['fueltype', 'aspiration', 'doornumber', 'carbody', 'drivewheel',
       'enginelocation', 'enginetype', 'cylindernumber', 'fuelsystem'],
      dtype='object')
```

Encode Categorical Features

In [24]:
```
le = LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])
```

Feature Target split

In [25]:
```
X = df.drop('price', axis=1)
y = df['price']

# X -> Independent Variables
# y -> Dependent Variables
```

Train Test Split

In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Feature Scaling

In [31]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Train Linear Regression

In [32]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[32]:
```
LinearRegression
LinearRegression()
```

Model Prediction

In [34]:
```
y_pred = lr.predict(X_test)
```

Model Evaluation

In [35]:
```
mse = mean_squared_error(y_test, y_pred)
mse
```

Out[35]: 15916389.725439584

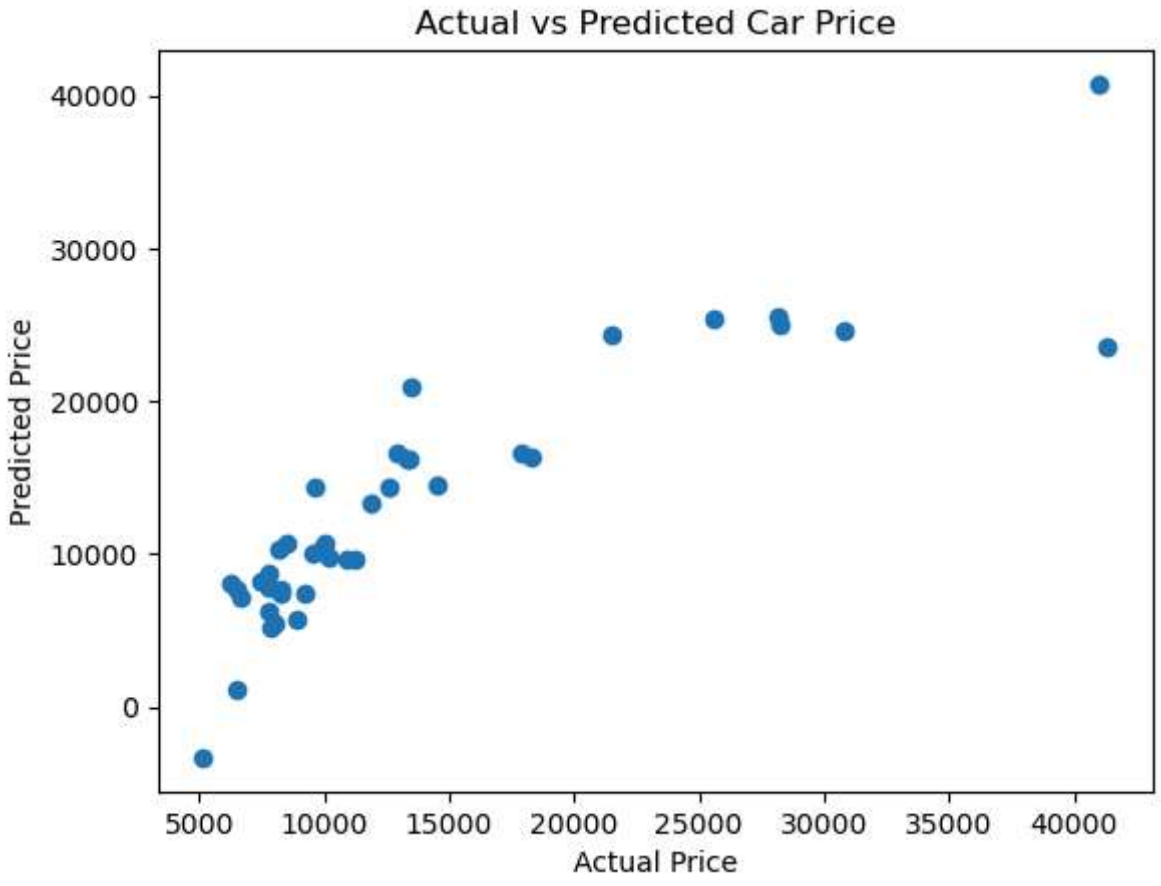
R^2 Score

In [37]:
```
r2 = r2_score(y_test, y_pred)
r2
```

Out[37]: 0.7983838478445057

Actual VS Predicted Plot

In [38]:
```
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Car Price")
plt.show()
```



Model Coeeficients

In [41]:
```
coefficients = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': lr.coef_
})

coefficients.sort_values(by='Coefficient', ascending=False)
```

Out[41]:

	Feature	Coefficient
14	enginesize	3789.156570
18	compressionratio	1682.480348
6	enginelocation	1462.682327
9	carwidth	1261.844659
7	wheelbase	1207.404187
1	fueltype	1043.355337
11	curbweight	923.788072
20	peakrpm	868.239122
5	drivewheel	592.571216
19	horsepower	532.894984
22	highwaympg	532.862799
12	enginetype	391.096736
0	symboling	263.952074
2	aspiration	248.836087
10	carheight	228.049694
15	fuelsystem	-132.679899
13	cylindernumber	-362.163353
16	boreratio	-645.515786
8	carlength	-672.656809
17	stroke	-673.834047
3	doornumber	-730.658459
4	carbody	-872.164796
21	citympg	-1592.354554

In [ ]:
```
```