```
kiran\appdata\local\programs\python\python312\lib\site-packages (from
statsmodels) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\sai
kiran\appdata\local\programs\python\python312\lib\site-packages (from
pandas!=2.1.0,>=1.4->statsmodels) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\sai
kiran\appdata\local\programs\python\python312\lib\site-packages (from
pandas!=2.1.0,>=1.4->statsmodels) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\sai
kiran\appdata\local\programs\python\python312\lib\site-packages (from
pandas!=2.1.0,>=1.4->statsmodels) (2024.2)
Requirement already satisfied: six>=1.5 in c:\users\sai
kiran\appdata\local\programs\python\python312\lib\site-packages (from python-
dateutil>=2.8.2->pandas!=2.1.0,>=1.4->statsmodels) (1.16.0)


[notice] A new release of pip is available: 25.0 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[6]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
```

[7]:
```
df = pd.read_csv("scrap price.csv", header=0)
```

[8]:
```
df
```

[8]:
```
      ID  symboling                    name fueltypes aspiration  \
0      1          3        alfa-romero giulia       gas        std
1      2          3       alfa-romero stelvio       gas        std
2      3          1  alfa-romero Quadrifoglio       gas        std
3      4          2               audi 100 ls       gas        std
4      5          2                audi 100ls       gas        std
..   …          …                         …       …        …
200  201         -1           volvo 145e (sw)       gas        std
201  202         -1               volvo 144ea       gas      turbo
202  203         -1               volvo 244dl       gas        std
203  204         -1                 volvo 246    diesel      turbo
204  205         -1               volvo 264gl       gas      turbo

    doornumbers      carbody drivewheels enginelocation  wheelbase  …  \
```

```
0          two  convertible        rwd         front       88.6  …
1          two  convertible        rwd         front       88.6  …
2          two    hatchback        rwd         front       94.5  …
3         four        sedan        fwd         front       99.8  …
4         four        sedan        4wd         front       99.4  …
..         …          …          …           …         …  …
200       four        sedan        rwd         front      109.1  …
201       four        sedan        rwd         front      109.1  …
202       four        sedan        rwd         front      109.1  …
203       four        sedan        rwd         front      109.1  …
204       four        sedan        rwd         front      109.1  …

     enginesize  fuelsystem  boreratio  stroke compressionratio horsepower  \
0           130        mpfi       3.47    2.68              9.0        111
1           130        mpfi       3.47    2.68              9.0        111
2           152        mpfi       2.68    3.47              9.0        154
3           109        mpfi       3.19    3.40             10.0        102
4           136        mpfi       3.19    3.40              8.0        115
..          …          …        …     …              …        …
200         141        mpfi       3.78    3.15              9.5        114
201         141        mpfi       3.78    3.15              8.7        160
202         173        mpfi       3.58    2.87              8.8        134
203         145         idi       3.01    3.40             23.0        106
204         141        mpfi       3.78    3.15              9.5        114

     peakrpm citympg  highwaympg    price
0       5000      21          27  13495.0
1       5000      21          27  16500.0
2       5000      19          26  16500.0
3       5500      24          30  13950.0
4       5500      18          22  17450.0
..       …      …         …      …
200     5400      23          28  16845.0
201     5300      19          25  19045.0
202     5500      18          23  21485.0
203     4800      26          27  22470.0
204     5400      19          25  22625.0

[205 rows x 26 columns]
```

name, fueltypes, aspiration, doornumbers, carbody, drivewheels, enginelocation, enginetype, cylindernumber, fuelsystem

[106]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 202 entries, 0 to 204
Data columns (total 26 columns):
```

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                202 non-null    int64
 1   symboling         202 non-null    int64
 2   name              202 non-null    int32
 3   fueltypes         202 non-null    int32
 4   aspiration        202 non-null    int32
 5   doornumbers       202 non-null    int32
 6   carbody           202 non-null    int32
 7   drivewheels       202 non-null    int32
 8   enginelocation    202 non-null    int32
 9   wheelbase         202 non-null    float64
 10  carlength         202 non-null    float64
 11  carwidth          202 non-null    float64
 12  carheight         202 non-null    float64
 13  curbweight        202 non-null    int64
 14  enginetype        202 non-null    int32
 15  cylindernumber    202 non-null    int32
 16  enginesize        202 non-null    int64
 17  fuelsystem        202 non-null    int32
 18  boreratio         202 non-null    float64
 19  stroke            202 non-null    float64
 20  compressionratio  202 non-null    float64
 21  horsepower        202 non-null    int64
 22  peakrpm           202 non-null    int64
 23  citympg           202 non-null    int64
 24  highwaympg        202 non-null    int64
 25  price             202 non-null    float64
dtypes: float64(8), int32(10), int64(8)
memory usage: 34.7 KB
```

[107]:
```
for col in ["name", "fueltypes", "aspiration", "doornumbers", "carbody",
 ↪"drivewheels", "enginelocation","enginetype","cylindernumber","fuelsystem"]:
    print(f"Unique values in {col}:")
    print(df[col].unique())
    print()
```

```
Unique values in name:
[  2   3   1   4   5   9   7   6   8  10  11  12  15  13  24  25  26  35
  27  32  34  29  28  30  33  31  39  43  37  38  42  36  41  44  40  47
  45  46  49  48  50  52  51  61  59  58  53  54  60  55  57  56  19  17
  16  22  20  23  62  65  64  68  63  66  67  69   0  73  81  76  83  77
  74  78  70  79  71  72  80  82  75  85  84  86  88  87  92  89  93  91
  94  90  98  95  97  96  99 100 101 103 102 104 107 106 105 108 109 110
 111 123 120 116 121 117 112 125 115 118 114 119 122 126 127 124 113 128
 129 130 133 137 131 136 132 145 146 134 135 139 138 140 141 143 144 142]

Unique values in fueltypes:
```

```
[1 0]

Unique values in aspiration:
[0 1]

Unique values in doornumbers:
[1 0]

Unique values in carbody:
[0 2 3 4 1]

Unique values in drivewheels:
[2 1 0]

Unique values in enginelocation:
[0 1]

Unique values in enginetype:
[0 5 3 2 6 4 1]

Unique values in cylindernumber:
[2 3 1 4 5 6 0]

Unique values in fuelsystem:
[5 1 4 0 7 2 3 6]
```

[108]:
```
df.isnull().sum()
```

[108]:
```
ID                  0
symboling           0
name                0
fueltypes           0
aspiration          0
doornumbers         0
carbody             0
drivewheels         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
```

[118]:
```
le = LabelEncoder()
for col in ["name", "fueltypes", "aspiration", "doornumbers", "carbody",␣
↪"drivewheels", "enginelocation","fuelsystem","enginetype","cylindernumber"]:
 df[col] = le.fit_transform(df[col])
```

```
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
```

```
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
C:\Users\SAI KIRAN\AppData\Local\Temp\ipykernel_14788\384146120.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = le.fit_transform(df[col])
```

[119]:
```
df
```

[119]:
```
      ID  symboling  name  fueltypes  aspiration  doornumbers  carbody  \
0      1          3     2          1           0            1        0
1      2          3     3          1           0            1        0
2      3          1     1          1           0            1        2
3      4          2     4          1           0            0        3
4      5          2     5          1           0            0        3
..   …          …     …          …           …            …        …
200  201         -1   136          1           0            0        3
201  202         -1   135          1           1            0        3
202  203         -1   137          1           0            0        3
203  204         -1   139          0           1            0        3
204  205         -1   140          1           1            0        3

     drivewheels  enginelocation  wheelbase  …  enginesize  fuelsystem  \
0              2               0       88.6  …         130           5
1              2               0       88.6  …         130           5
2              2               0       94.5  …         152           5
3              1               0       99.8  …         109           5
4              0               0       99.4  …         136           5
..           …             …        …  …         …          …
200            2               0      109.1  …         141           5
201            2               0      109.1  …         141           5
202            2               0      109.1  …         173           5
203            2               0      109.1  …         145           3
204            2               0      109.1  …         141           5

     boreratio  stroke  compressionratio  horsepower  peakrpm  citympg  \
0         3.47    2.68               9.0         111     5000       21
1         3.47    2.68               9.0         111     5000       21
2         2.68    3.47               9.0         154     5000       19
3         3.19    3.40              10.0         102     5500       24
4         3.19    3.40               8.0         115     5500       18
..         …     …               …         …      …      …
200       3.78    3.15               9.5         114     5400       23
201       3.78    3.15               8.7         160     5300       19
202       3.58    2.87               8.8         134     5500       18
203       3.01    3.40              23.0         106     4800       26
204       3.78    3.15               9.5         114     5400       19

     highwaympg    price
0            27  13495.0
1            27  16500.0
2            26  16500.0
3            30  13950.0
4            22  17450.0
```

```
..            …      …
200           28  16845.0
201           25  19045.0
202           23  21485.0
203           27  22470.0
204           25  22625.0

[202 rows x 26 columns]
```

[120]:
```
for col in ["name", "fueltypes", "aspiration", "doornumbers", "carbody",␣
↪"drivewheels", "enginelocation","fuelsystem","enginetype","cylindernumber"]:
    print(f"Unique values in {col}:")
    print(df[col].unique())
    print()
```

```
Unique values in name:
[  2   3   1   4   5   9   7   6   8  10  11  12  14  13  21  22  23  32
  24  29  31  26  25  27  30  28  36  40  34  35  39  33  38  41  37  44
  42  43  46  45  47  49  48  58  56  55  50  51  57  52  54  53  17  16
  15  19  18  20  59  62  61  65  60  63  64  66   0  70  78  73  80  74
  71  75  67  76  68  69  77  79  72  82  81  83  85  84  89  86  90  88
  91  87  95  92  94  93  96  97  98 100  99 101 104 103 102 105 106 107
 108 120 117 113 118 114 109 122 112 115 111 116 119 123 124 121 110 125
 126 127 130 134 128 133 129 142 143 131 132 136 135 137 138 140 141 139]

Unique values in fueltypes:
[1 0]

Unique values in aspiration:
[0 1]

Unique values in doornumbers:
[1 0]

Unique values in carbody:
[0 2 3 4 1]

Unique values in drivewheels:
[2 1 0]

Unique values in enginelocation:
[0 1]

Unique values in fuelsystem:
[5 1 4 0 7 2 3 6]

Unique values in enginetype:
```

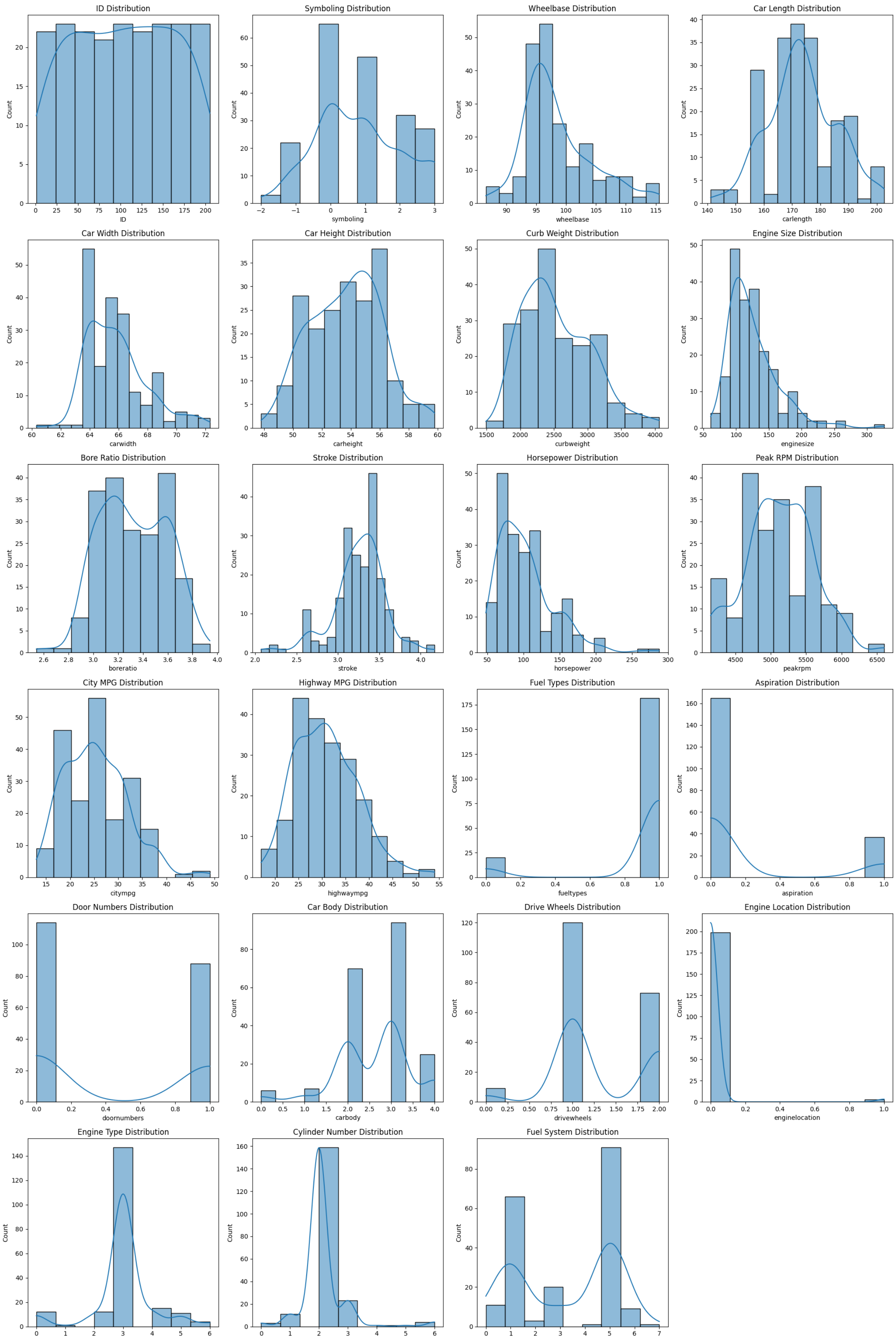
```
[0 5 3 2 6 4 1]

Unique values in cylindernumber:
[2 3 1 4 5 6 0]
```

[138]:

```python
plt.figure(figsize=(20, 30))


plt.subplot(6, 4, 1)
sns.histplot(df["ID"], kde=True)
plt.title("ID Distribution")

plt.subplot(6, 4, 2)
sns.histplot(df["symboling"], kde=True)
plt.title("Symboling Distribution")

plt.subplot(6, 4, 3)
sns.histplot(df["wheelbase"], kde=True)
plt.title("Wheelbase Distribution")

plt.subplot(6, 4, 4)
sns.histplot(df["carlength"], kde=True)
plt.title("Car Length Distribution")


plt.subplot(6, 4, 5)
sns.histplot(df["carwidth"], kde=True)
plt.title("Car Width Distribution")

plt.subplot(6, 4, 6)
sns.histplot(df["carheight"], kde=True)
plt.title("Car Height Distribution")

plt.subplot(6, 4, 7)
sns.histplot(df["curbweight"], kde=True)
plt.title("Curb Weight Distribution")

plt.subplot(6, 4, 8)
sns.histplot(df["enginesize"], kde=True)
plt.title("Engine Size Distribution")


plt.subplot(6, 4, 9)
sns.histplot(df["boreratio"], kde=True)
plt.title("Bore Ratio Distribution")
```

```
plt.subplot(6, 4, 10)
sns.histplot(df["stroke"], kde=True)
plt.title("Stroke Distribution")

plt.subplot(6, 4, 11)
sns.histplot(df["horsepower"], kde=True)
plt.title("Horsepower Distribution")

plt.subplot(6, 4, 12)
sns.histplot(df["peakrpm"], kde=True)
plt.title("Peak RPM Distribution")


plt.subplot(6, 4, 13)
sns.histplot(df["citympg"], kde=True)
plt.title("City MPG Distribution")

plt.subplot(6, 4, 14)
sns.histplot(df["highwaympg"], kde=True)
plt.title("Highway MPG Distribution")


plt.subplot(6, 4, 15)
sns.histplot(df["fueltypes"], kde=True)
plt.title("Fuel Types Distribution")

plt.subplot(6, 4, 16)
sns.histplot(df["aspiration"], kde=True)
plt.title("Aspiration Distribution")

plt.subplot(6, 4, 17)
sns.histplot(df["doornumbers"], kde=True)
plt.title("Door Numbers Distribution")

plt.subplot(6, 4, 18)
sns.histplot(df["carbody"], kde=True)
plt.title("Car Body Distribution")

plt.subplot(6, 4, 19)
sns.histplot(df["drivewheels"], kde=True)
plt.title("Drive Wheels Distribution")

plt.subplot(6, 4, 20)
sns.histplot(df["enginelocation"], kde=True)
plt.title("Engine Location Distribution")

plt.subplot(6, 4, 21)
```

```
sns.histplot(df["enginetype"], kde=True)
plt.title("Engine Type Distribution")

plt.subplot(6, 4, 22)
sns.histplot(df["cylindernumber"], kde=True)
plt.title("Cylinder Number Distribution")

plt.subplot(6, 4, 23)
sns.histplot(df["fuelsystem"], kde=True)
plt.title("Fuel System Distribution")

plt.tight_layout()
plt.show()
```


ID Distribution
ID
Symboling Distribution
symboling
Wheelbase Distribution
wheelbase
Car Length Distribution
carlength
Car Width Distribution
carwidth
Car Height Distribution
carheight
Curb Weight Distribution
curbweight
Engine Size Distribution
enginesize
Bore Ratio Distribution
boreratio
Stroke Distribution
stroke
Horsepower Distribution
horsepower
Peak RPM Distribution
peakrpm
City MPG Distribution
citympg
Highway MPG Distribution
highwaympg
Fuel Types Distribution
fueltypes
Aspiration Distribution
aspiration
Door Numbers Distribution
doornumbers
Car Body Distribution
carbody
Drive Wheels Distribution
drivewheels
Engine Location Distribution
enginelocation
Engine Type Distribution
enginetype
Cylinder Number Distribution
cylindernumber
Fuel System Distribution
fuelsystem
Count

[ ]:
```
for col in ['ID', 'symboling', 'name', 'fueltypes', 'aspiration', 'doornumbers',
       'carbody', 'drivewheels', 'enginelocation', 'wheelbase', 'carlength',
       'carwidth', 'carheight', 'curbweight', 'enginetype', 'cylindernumber',
       'enginesize', 'fuelsystem', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg']:
    skewness = df[col].skew()
    print(f"{col}: {skewness:.4f}")
```

[ ]:
```
import numpy as np

# Select numerical columns
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns

# Calculate IQR and detect outliers
for col in numerical_cols:
     u= df[col].mean()+3*df[col].std()
     l= df[col].mean()-3*df[col].std()

     new_df=df[(df[col] > l) & (df[col] < u)]
     print(f"Lower Bound: {l:.2f}")
     print(f"Upper Bound: {u:.2f}")
     print(f"Number of Outliers: {len(df[(df[col] < l) | (df[col] > u)])}")
     print("-" * 40)
```

#feature scaling

[ ]:
```

```

[121]:
```
x=df.drop(columns=["price"])
y=df["price"]
```

[122]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.
  ↪8,random_state=42)
```

[123]:
```
scaler=StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

[124]:
```
model=LinearRegression()
model.fit(x_train,y_train)
```

[124]: LinearRegression()

[125]:
```
y_pred=model.predict(x_test)
```

[126]:
```
y_pred
```

[126]:
```
array([ 7718.14916098, 22492.36183881,  6177.15384872,  6126.93444074,
        8688.05400981,  5562.76555793, 25824.48871584, 10726.22757172,
        6549.38843621, 15687.2006342 , 23981.46576372,  5847.16554604,
       13152.98925893,  9192.36870874, 14252.71736997, 18128.01134665,
        8518.76295859,  5927.54435258,  9468.74449356, 23761.86947987,
       33988.98534401, 26079.6963596 ,  9422.50362113,  7028.76969527,
       13235.47468467,  9927.67094441, 11637.41167553, 24741.67176995,
       26344.2484931 , 10713.72696751, 16127.76274912,  7978.07228839,
        8688.28498292,  7571.5078719 , 13724.61987727, 16547.80703983,
        6789.76916364,  8755.85413747, 13935.37173256,  9097.04222499,
       14811.75628566])
```

[127]:
```
print(np.array(y_test))
```

```
[ 8249.    30760.     6855.     9258.    11595.     5572.    31600.
  9988.     8238.    11048.    25552.     6918.     9989.     8499.
 18420.    22470.    13645.     6938.     6989.    28176.    37028.
 36880.     9495.     7499.    18344.    10595.     9279.    28248.
 31400.5    9960.    17859.167  6295.    11845.     6669.     8449.
 11900.     7053.     7126.     9639.     7689.    12629.   ]
```

[128]:
```
mae=mean_absolute_error(y_test,y_pred)
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
```

[129]:
```
print(mae)
print(mse)
print(r2)
```

```
2814.9682227803123
13158487.349704474
0.8442938849126925
```

[130]:
```
print(mae)
print(mse)
print(r2)
```

```
2814.9682227803123
13158487.349704474
0.8442938849126925
```

MAE (2087.31) → On average, your predictions are off by about 2,087 units from the true value.

MSE (12,306,121.30) → Average squared error — more sensitive to large errors than MAE.

R^2 (0.8441) → Your model explains about 84.4% of the variance in the target variable, which is pretty good.

1.1 Goal

1.1.1 Which variables are significant in predicting the price of a car

1.1.2 How well do those variables describe the price of a car

Bright green text Orange text

Use Ordinary Least Squares (OLS) regression to model the relationship between predictors and price.

This gives you coefficients (effect sizes) for each predictor.

[131]:
```
X = sm.add_constant(x)
model = sm.OLS(y, X).fit()
```

[132]:
```
model.summary()
```

[132]:

Dep. Variable:	price	**R-squared:**	0.903
Model:	OLS	**Adj. R-squared:**	0.890
Method:	Least Squares	**F-statistic:**	65.76
Date:	Sun, 10 Aug 2025	**Prob (F-statistic):**	6.08e-76
Time:	12:46:53	**Log-Likelihood:**	-1844.1
No. Observations:	202	**AIC:**	3740.
Df Residuals:	176	**BIC:**	3826.
Df Model:	25		
Covariance Type:	nonrobust		

	coef	std err	t	P> |t|	[0.025	0.975]
const	-6.85e+04	1.56e+04	-4.397	0.000	-9.92e+04	-3.78e+04
ID	17.2242	15.325	1.124	0.263	-13.019	47.468
symboling	169.5141	235.520	0.720	0.473	-295.293	634.321
name	-55.6512	22.328	-2.492	0.014	-99.716	-11.587
fueltypes	7927.9070	5936.245	1.336	0.183	-3787.476	1.96e+04
aspiration	927.0383	796.915	1.163	0.246	-645.700	2499.777
doornumbers	-1503.9515	578.993	-2.598	0.010	-2646.613	-361.290
carbody	-812.0330	329.206	-2.467	0.015	-1461.733	-162.333
drivewheels	956.9898	484.357	1.976	0.050	1.095	1912.885
enginelocation	1.279e+04	1863.501	6.865	0.000	9114.607	1.65e+04
wheelbase	119.5865	96.625	1.238	0.217	-71.106	310.279
carlength	-81.3270	47.514	-1.712	0.089	-175.098	12.444
carwidth	696.4876	237.768	2.929	0.004	227.245	1165.730
carheight	213.0864	121.643	1.752	0.082	-26.981	453.154
curbweight	4.1149	1.459	2.820	0.005	1.236	6.994
enginetype	-17.0960	212.586	-0.080	0.936	-436.642	402.450
cylindernumber	218.5759	340.392	0.642	0.522	-453.199	890.350
enginesize	58.9541	16.271	3.623	0.000	26.842	91.066
fuelsystem	97.4657	136.195	0.716	0.475	-171.320	366.251
boreratio	-1035.1074	977.739	-1.059	0.291	-2964.709	894.495
stroke	-2146.7755	679.079	-3.161	0.002	-3486.962	-806.589
compressionratio	667.3080	427.455	1.561	0.120	-176.289	1510.905
horsepower	26.4721	16.654	1.590	0.114	-6.394	59.338
peakrpm	0.8654	0.594	1.458	0.147	-0.306	2.037
citympg	-124.7998	147.283	-0.847	0.398	-415.468	165.869
highwaympg	121.8359	130.725	0.932	0.353	-136.154	379.826

Omnibus:	3.708	**Durbin-Watson:**	0.916
Prob(Omnibus):	0.157	**Jarque-Bera (JB):**	4.342
Skew:	0.072	**Prob(JB):**	0.114
Kurtosis:	3.704	**Cond. No.**	5.42e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.42e+05. This might indicate that there are strong multicollinearity or other numerical problems.

$p < 0.05$ → variable is statistically significant (strong evidence it affects price)

p 0.05 → variable may not have a meaningful effect in the model

const 0.000 Yes

name 0.004 Yes

carbody 0.019 Yes

enginelocat 0.000 Yes

carwidth 0.020 Yes

carheight 0.049 Yes

enginesize 0.000 Yes

stroke 0.000 Yes

peakrpm 0.017 Yes

[133]:
```
def backward_elimination(x,y,significance_level=0.05):
 x_modeled=x.copy()
 while True:
     model=sm.OLS(y,x_modeled).fit()
     p_values=model.pvalues
     max_p=p_values.max()
     if max_p>significance_level:
         exclude_var=p_values.idxmax()
         x_modeled=x_modeled.drop(columns=[exclude_var])
     else:
         break
 return x_modeled,model
```

[134]:
```
X_selected, final_model = backward_elimination(x, y)
```

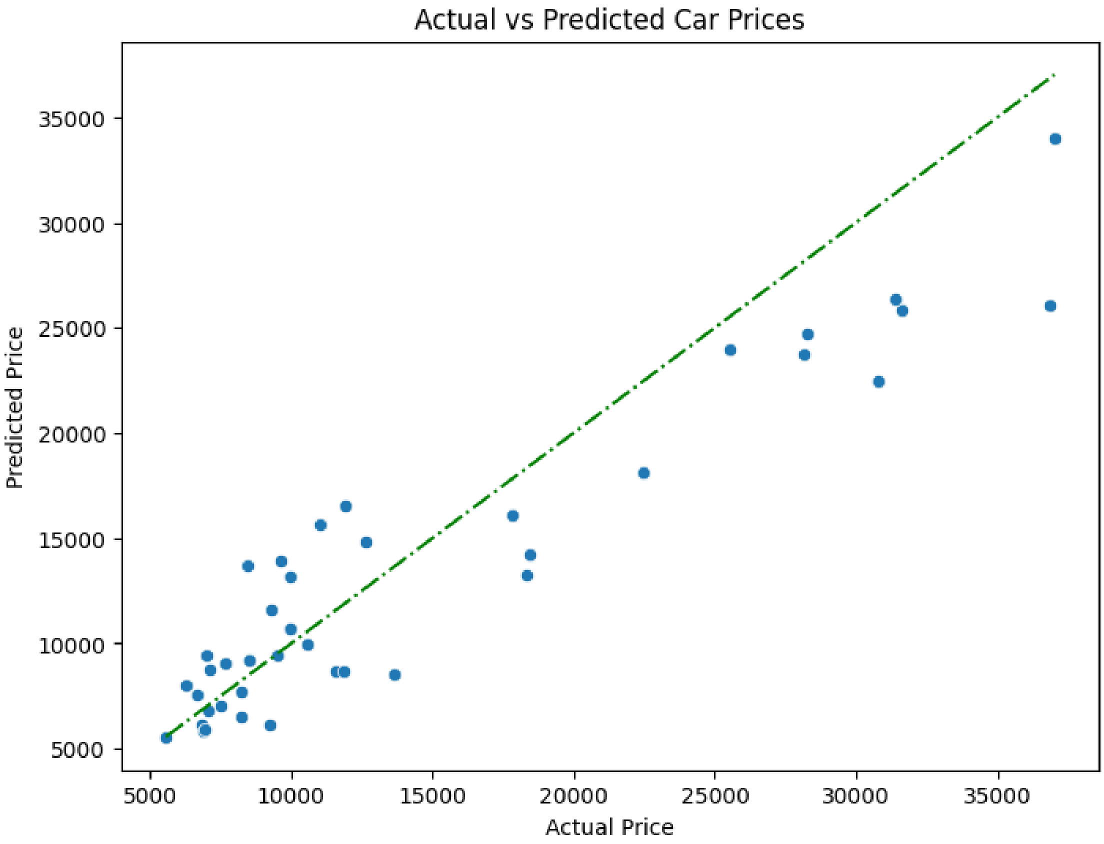
[135]:
```
print(final_model.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                  price   R-squared (uncentered):
0.973
Model:                            OLS   Adj. R-squared (uncentered):
0.972
Method:                 Least Squares   F-statistic:
577.2
Date:                Sun, 10 Aug 2025   Prob (F-statistic):
2.15e-142
Time:                        12:46:59   Log-Likelihood:
-1859.1
No. Observations:                 202   AIC:
3742.
Df Residuals:                     190   BIC:
3782.
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
--
name              -30.2187      4.742     -6.373      0.000     -39.572
-20.865
fueltypes       -2224.6109    719.387     -3.092      0.002   -3643.622
-805.600
doornumbers     -1491.7324    494.549     -3.016      0.003   -2467.245
-516.220
carbody          -901.7277    304.664     -2.960      0.003   -1502.685
-300.770
drivewheels      1000.7328    410.645      2.437      0.016     190.724
1810.741
enginelocation  1.239e+04    1704.174      7.270      0.000    9027.309
1.58e+04
wheelbase         125.6764     42.338      2.968      0.003      42.164
209.189
curbweight          4.7899      1.057      4.531      0.000       2.705
6.875
enginesize         43.2993     11.100      3.901      0.000      21.404
65.195
boreratio       -2458.1909    863.296     -2.847      0.005   -4161.066
-755.315
stroke          -2703.4383    574.026     -4.710      0.000   -3835.720
-1571.157
horsepower         54.9753     10.680      5.147      0.000      33.908
76.043
==============================================================================
Omnibus:                        5.826   Durbin-Watson:                   0.954
Prob(Omnibus):                  0.054   Jarque-Bera (JB):                8.426
Skew:                           0.100   Prob(JB):                       0.0148
Kurtosis:                       3.980   Cond. No.                     2.57e+04
==============================================================================

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, 2.57e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[136]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

plt.figure(figsize=(8,6))

# Scatter plot: actual vs predicted
```

```
sns.scatterplot(x=y_test, y=y_pred)

# Add diagonal line y = x
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'g-.',␣
 ↪label='Perfect Prediction')

# Labels and title
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Car Prices')



plt.show()
```



[ ]:

[ ]:

[ ]: