In [1]:
```
# Import necessary libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
# Ignore warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
# Load the dataset
df = pd.read_csv(r"C:\Users\USER\Downloads\auto.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
1	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
4	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	15250

5 rows × 26 columns

In [5]:
```
# Create column names
col_names = [
    "symboling", "normalized-losses", "make", "fuel-type", "aspiration", "num-of-doors",
    "body-style", "drive-wheels", "engine-location", "wheel-base", "length", "width",
    "height", "curb-weight", "engine-type", "num-of-cylinders", "engine-size",
    "fuel-system", "bore", "stroke", "compression-ratio", "horsepower", "peak-rpm",
    "city-mpg", "highway-mpg", "price"
]

df.columns = col_names
```

In [6]:
```
# Replace '?' with NaN
df = df.replace('?', np.NaN)
```

In [7]:
```
df.head()          #Confirming column name and replacement
```

Out[7]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compress
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
1	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	
2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	
3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	
4	2	NaN	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	

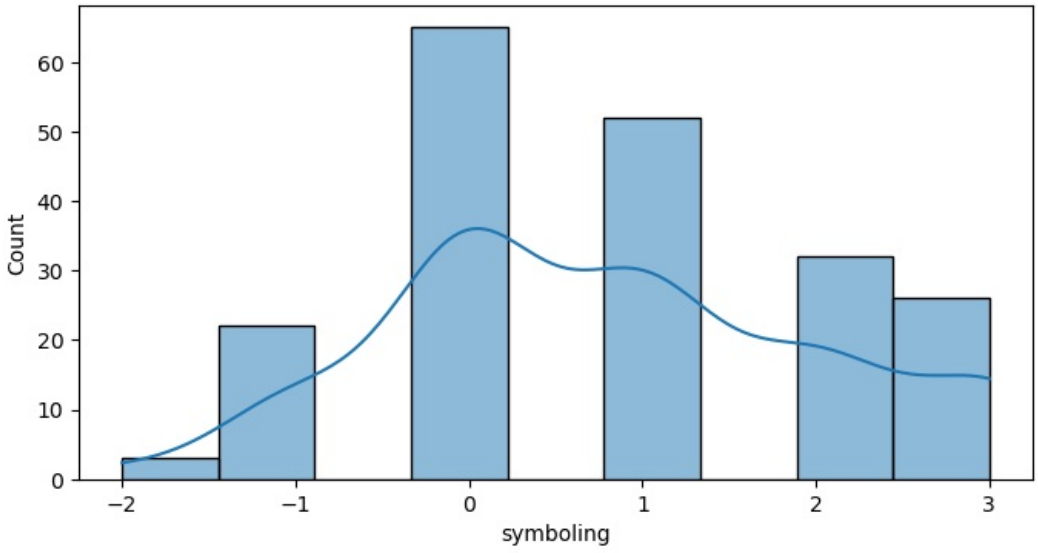
5 rows × 26 columns

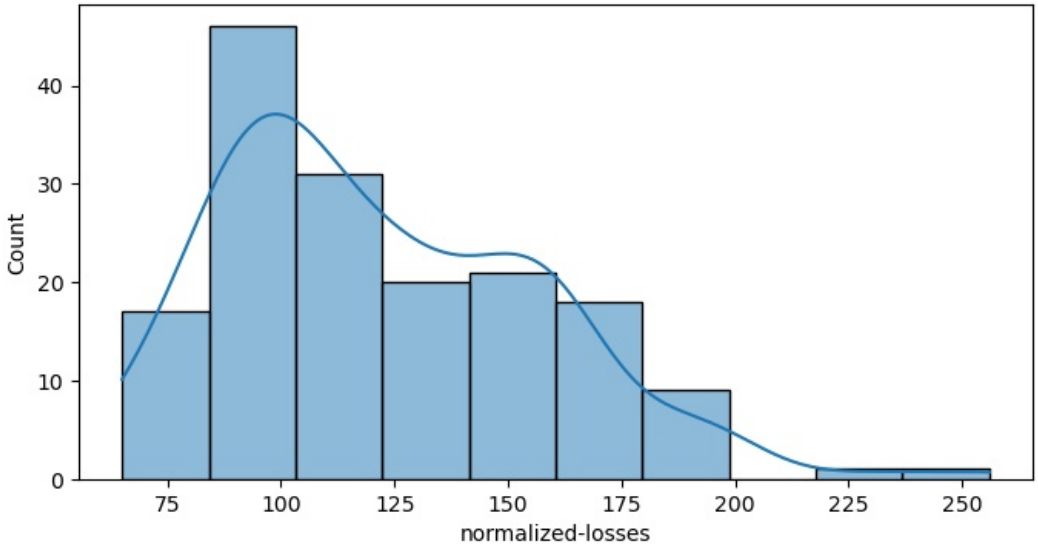
In [8]:
```
# Convert columns to appropriate data types
df['normalized-losses'] = df['normalized-losses'].astype(float)
df['bore'] = df['bore'].astype(float)
df['stroke'] = df['stroke'].astype(float)
df['horsepower'] = df['horsepower'].astype(float)
df['peak-rpm'] = df['peak-rpm'].astype(float)
df['price'] = df['price'].astype(float)
```

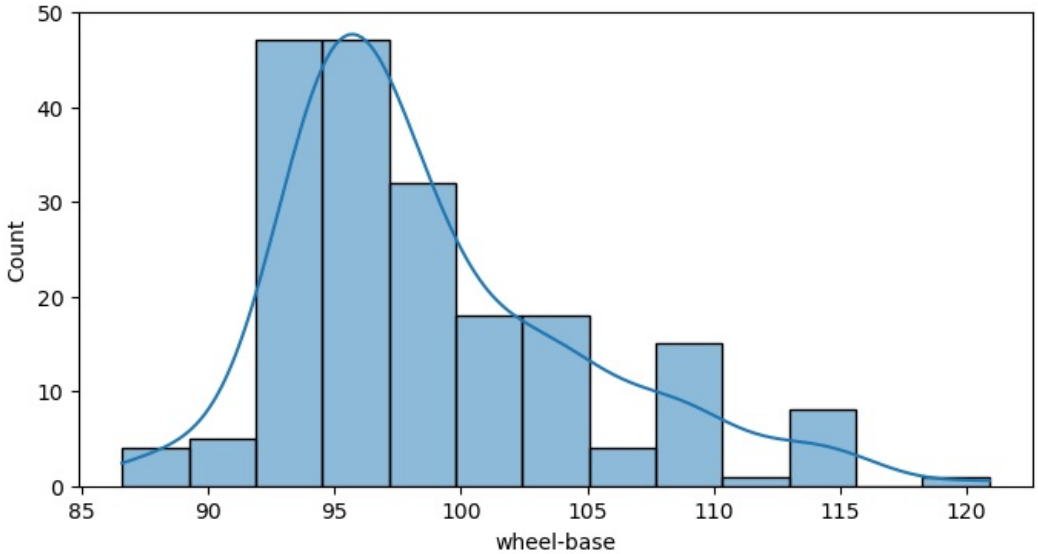
In [9]:
```
# Drop rows where the target variable 'price' is NaN
df = df.dropna(subset=['price'])
```

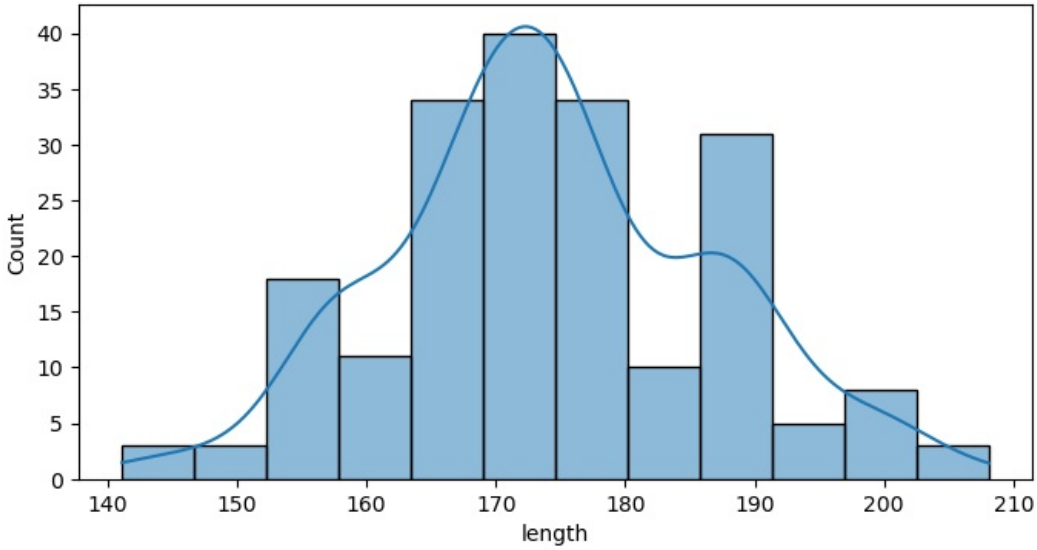
In [10]:
```
# Univariate Analysis - Distribution of numeric features
numeric_features = df.select_dtypes(include=['int64', 'float64']).columns
```

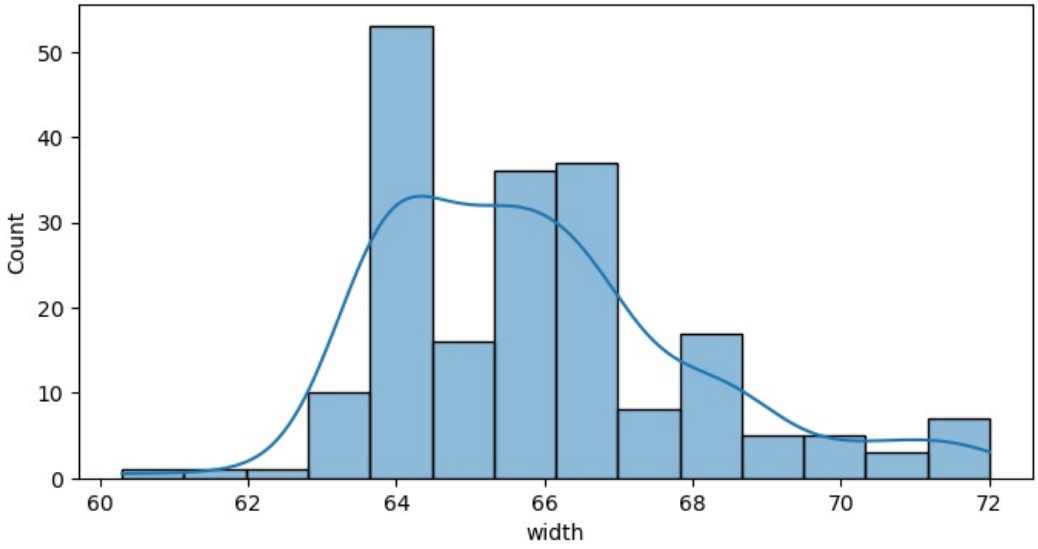
```
for col in numeric_features:
    plt.figure(figsize=(8, 4))
    sns.histplot(df[col], kde=True)
```

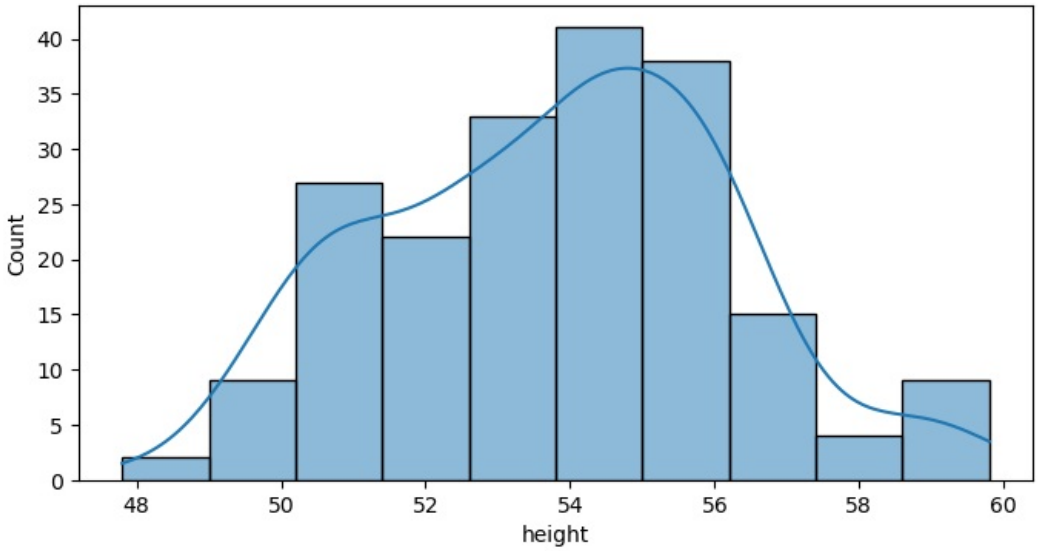


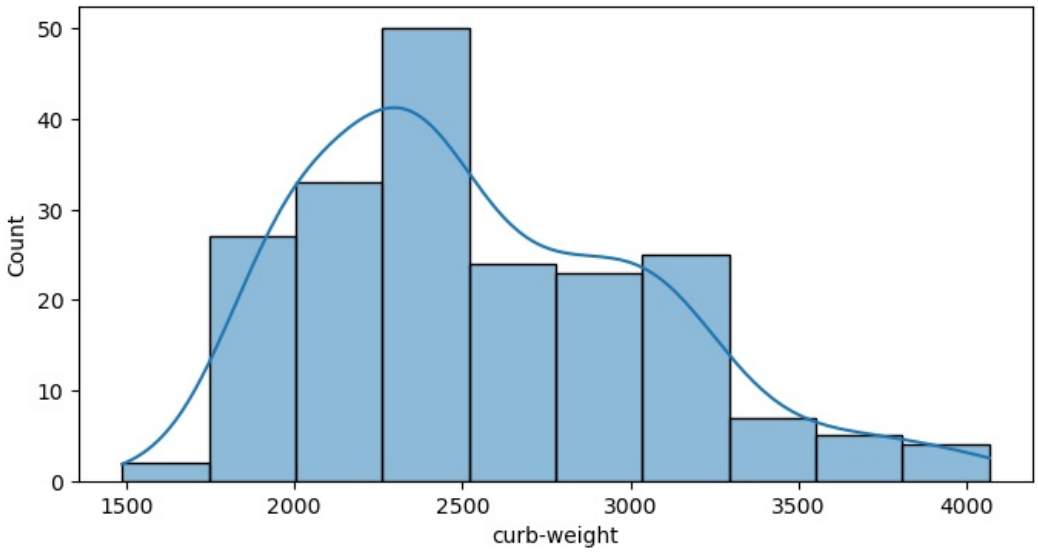


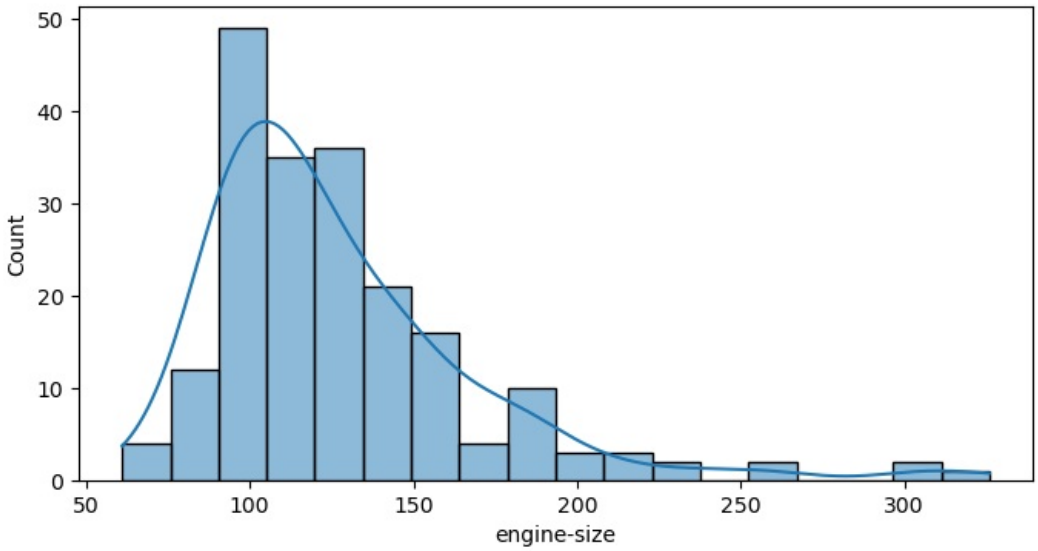


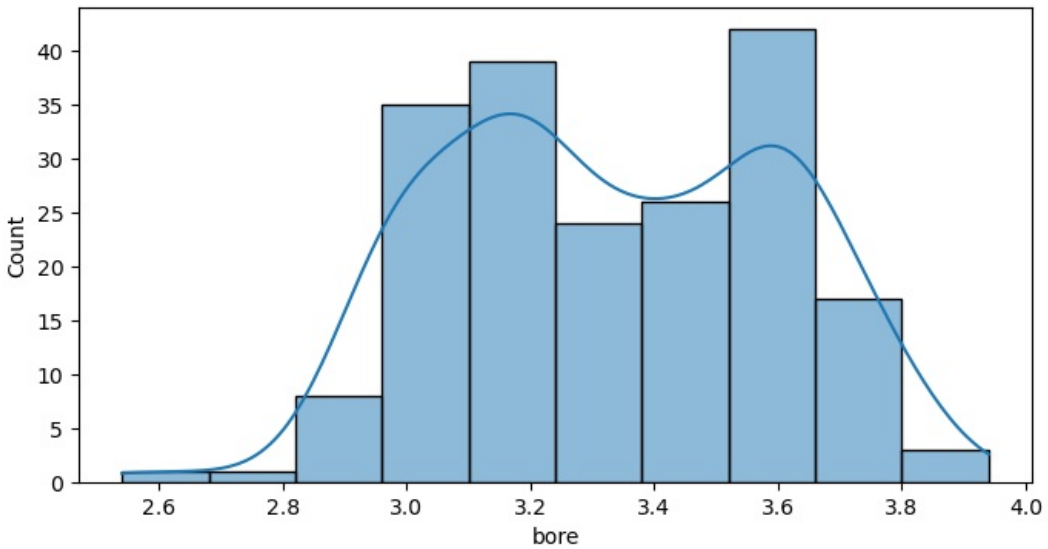

Count
length
0
5
10
15
20
25
30
35
40
140
150
160
170
180
190
200
210

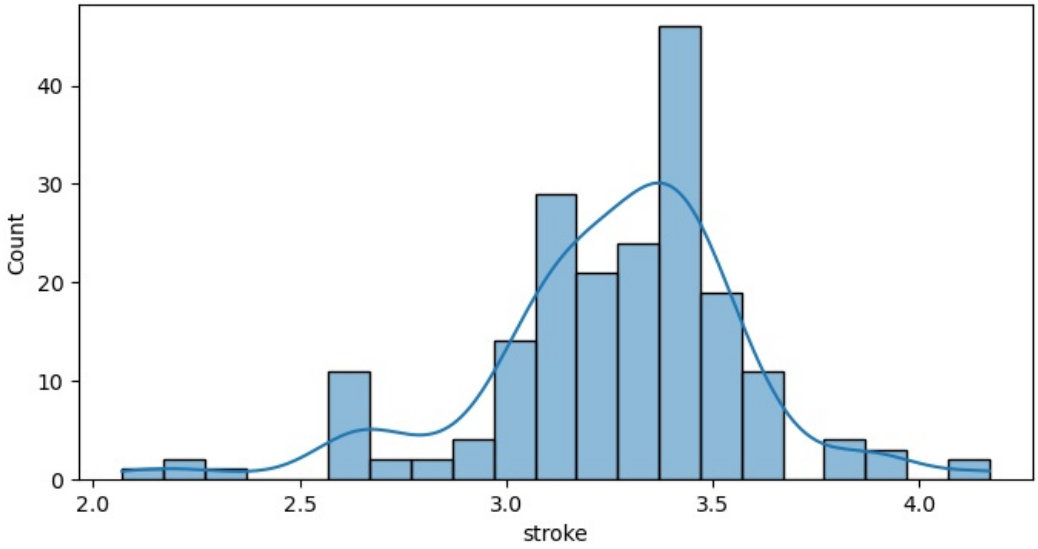

Count
width
0
10
20
30
40
50
60
62
64
66
68
70
72

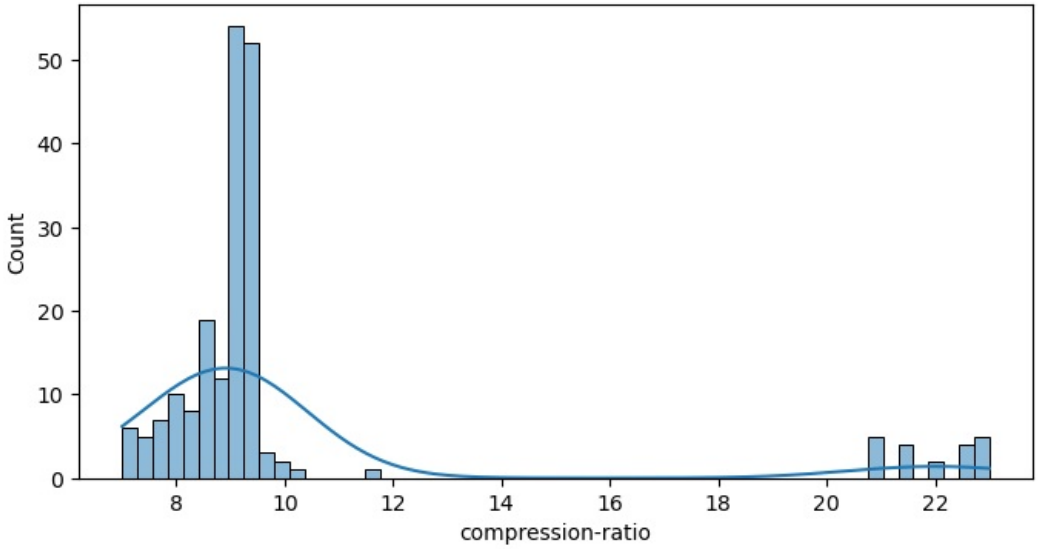

Count
height
0
5
10
15
20
25
30
35
40
48
50
52
54
56
58
60

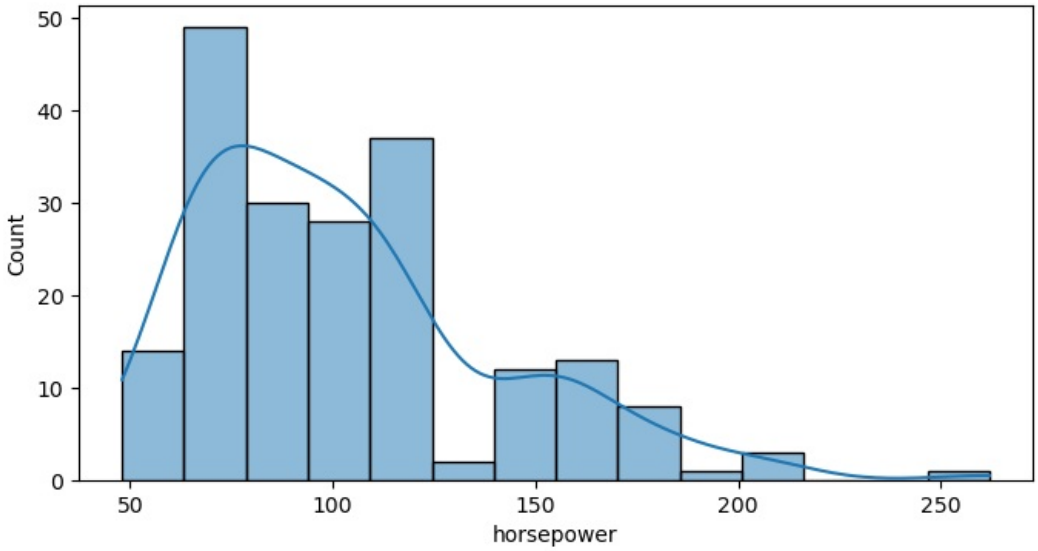

Count
curb-weight

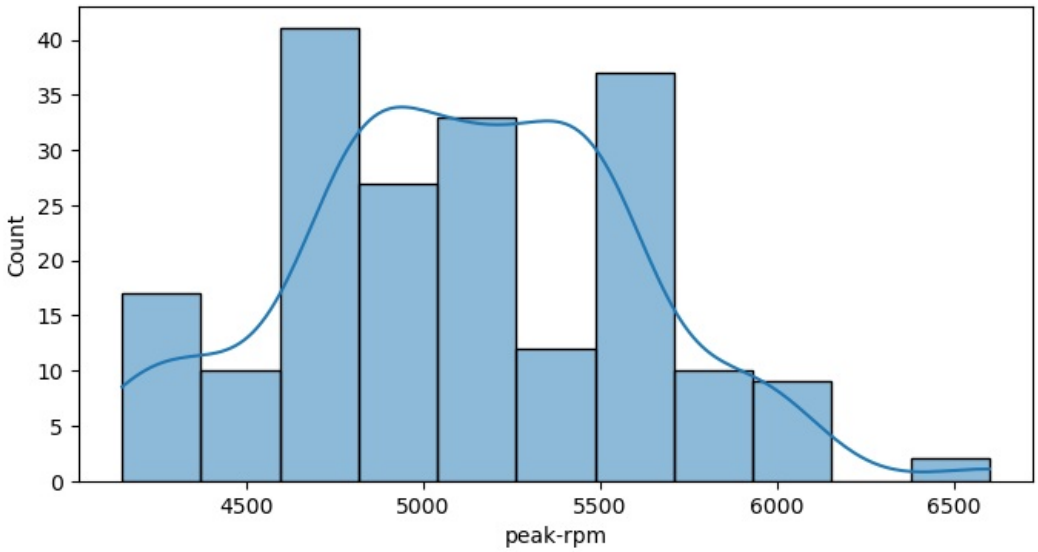

Count
engine-size

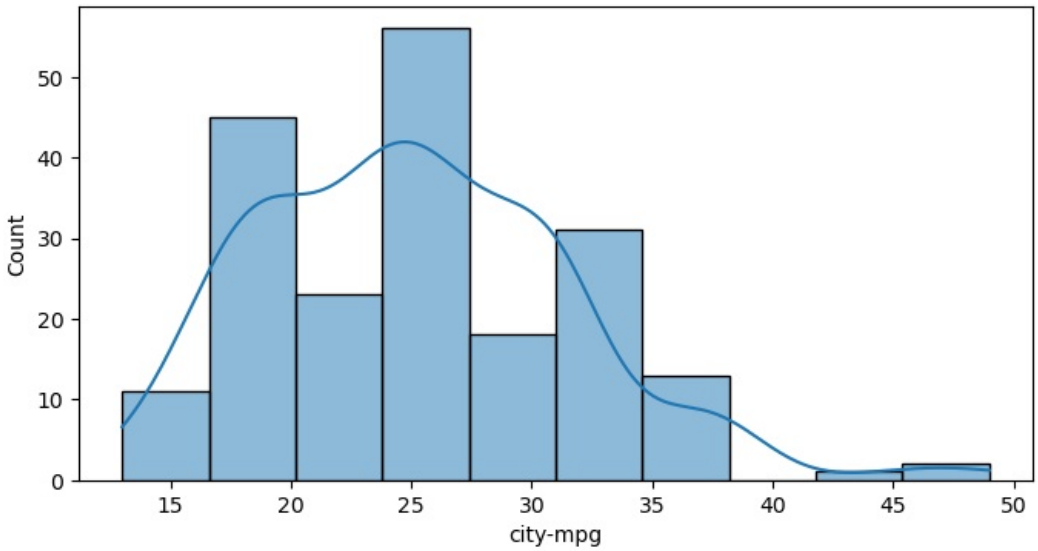

Count
bore

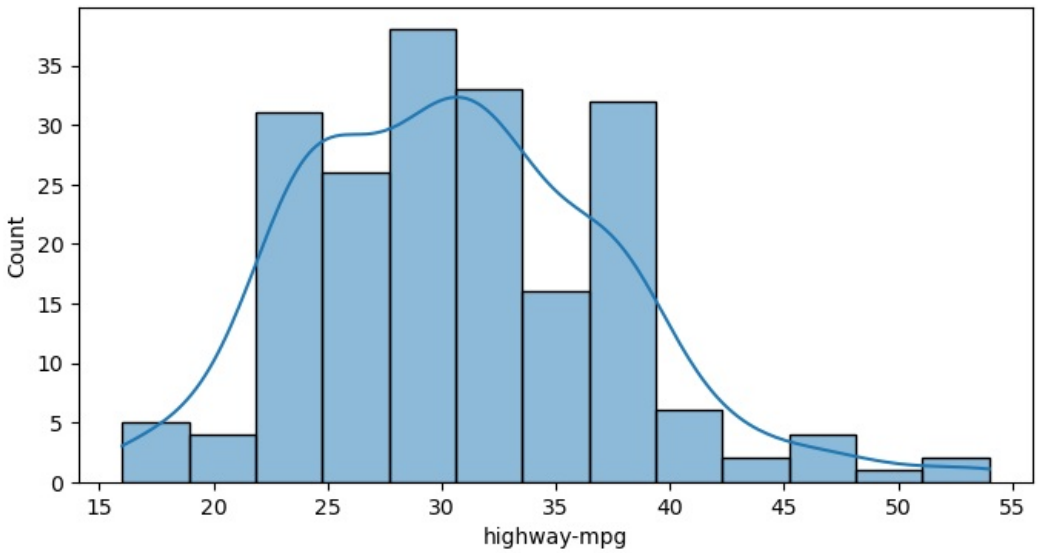

Count
stroke
0
10
20
30
40
2.0
2.5
3.0
3.5
4.0

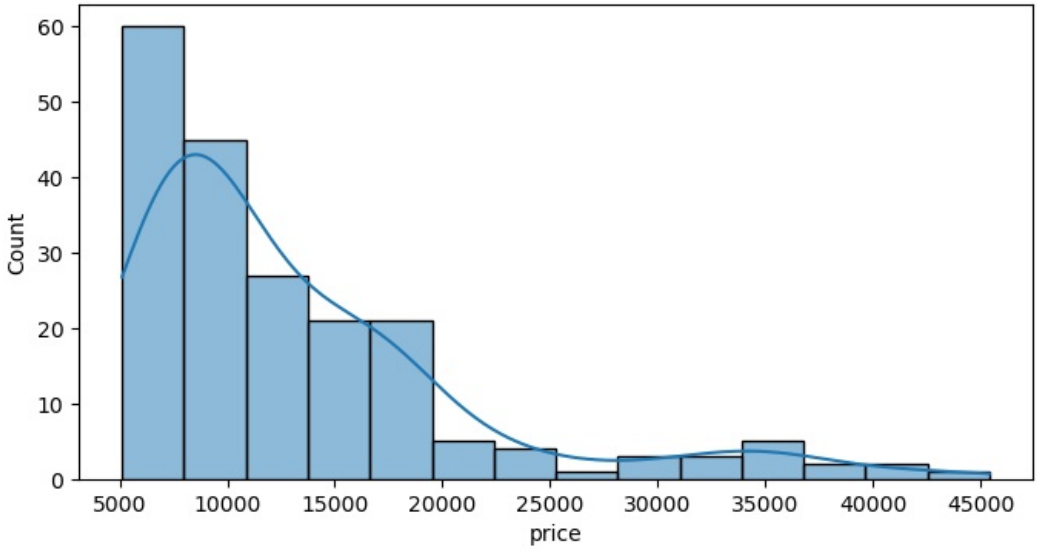

Count
compression-ratio
0
10
20
30
40
50
8
10
12
14
16
18
20
22


Count
horsepower
0
10
20
30
40
50
50
100
150
200
250

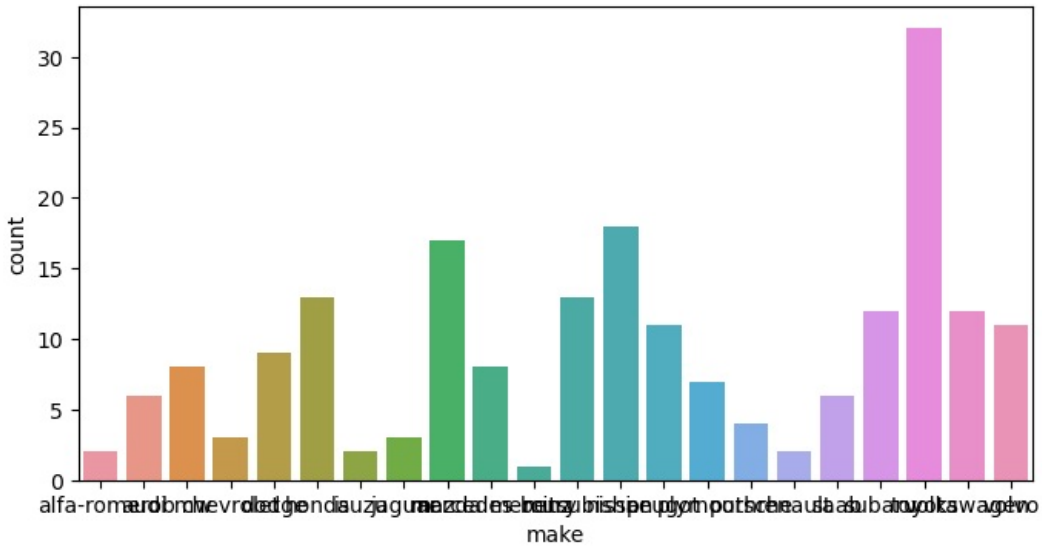

Count
peak-rpm
4500
5000
5500
6000
6500

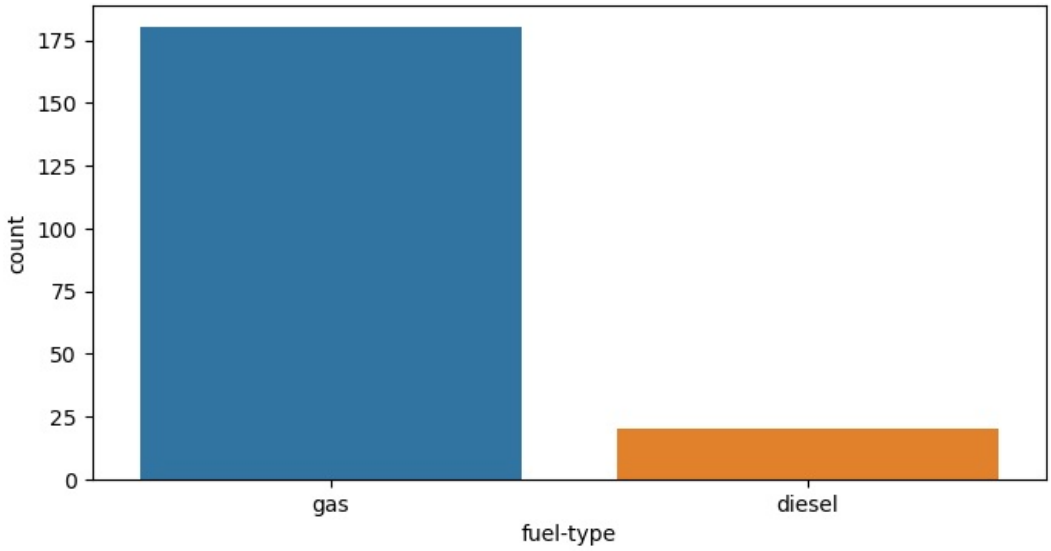

Count
city-mpg
15
20
25
30
35
40
45
50

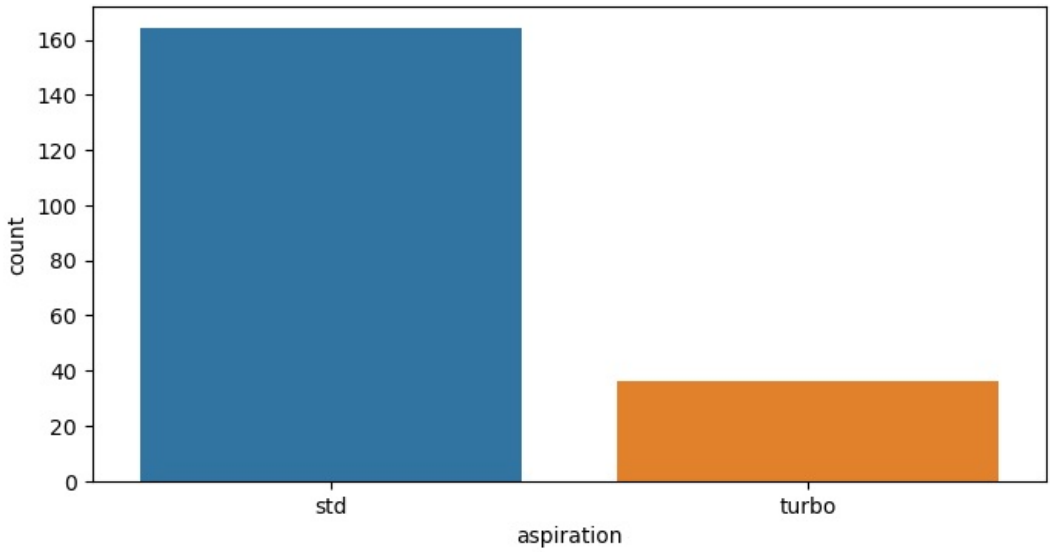

Count
highway-mpg
15
20
25
30
35
40
45
50
55

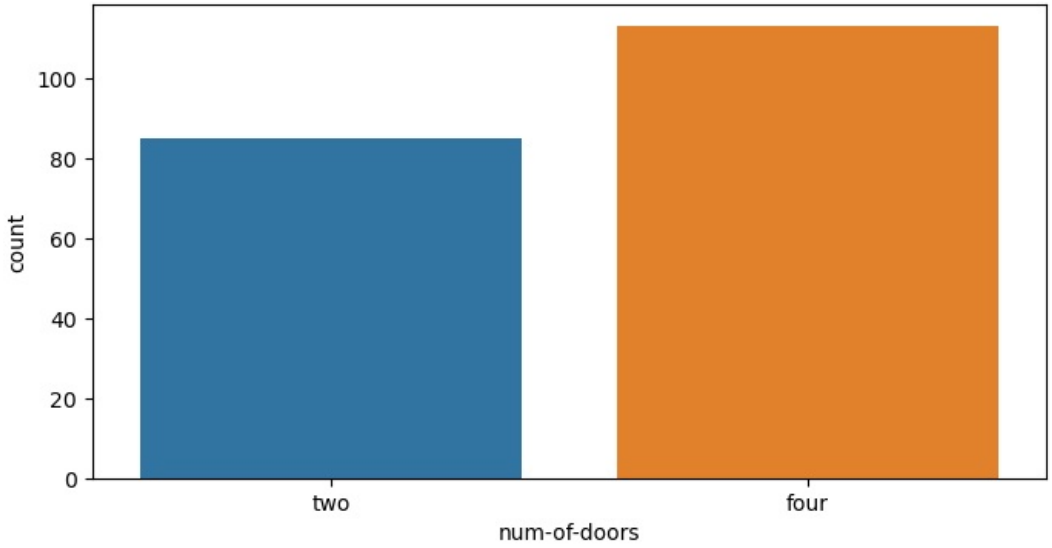


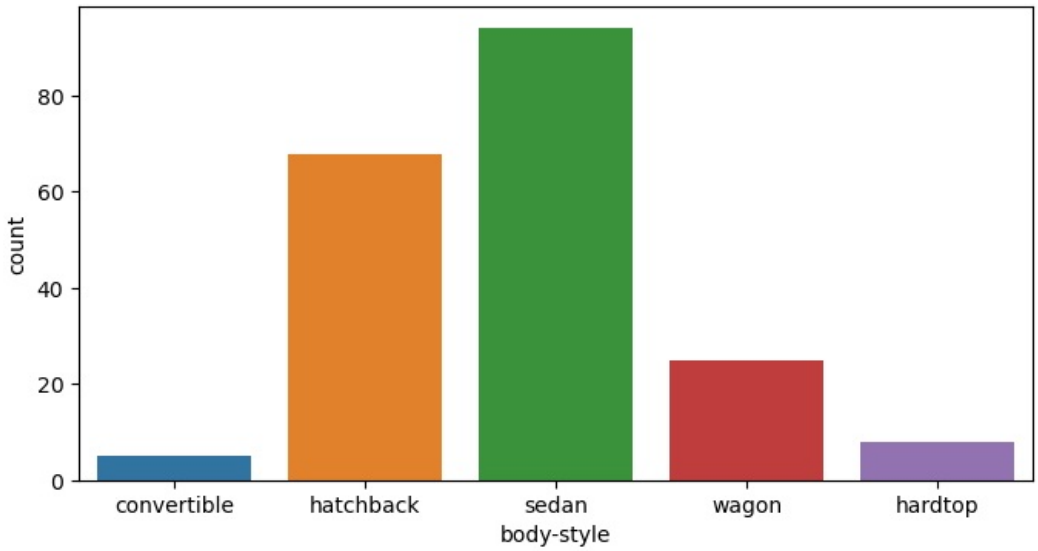
In [11]:
```
categorical_features = df.select_dtypes(include=['object']).columns
for col in categorical_features:
    plt.figure(figsize=(8, 4))
    sns.countplot(x=df[col])
```

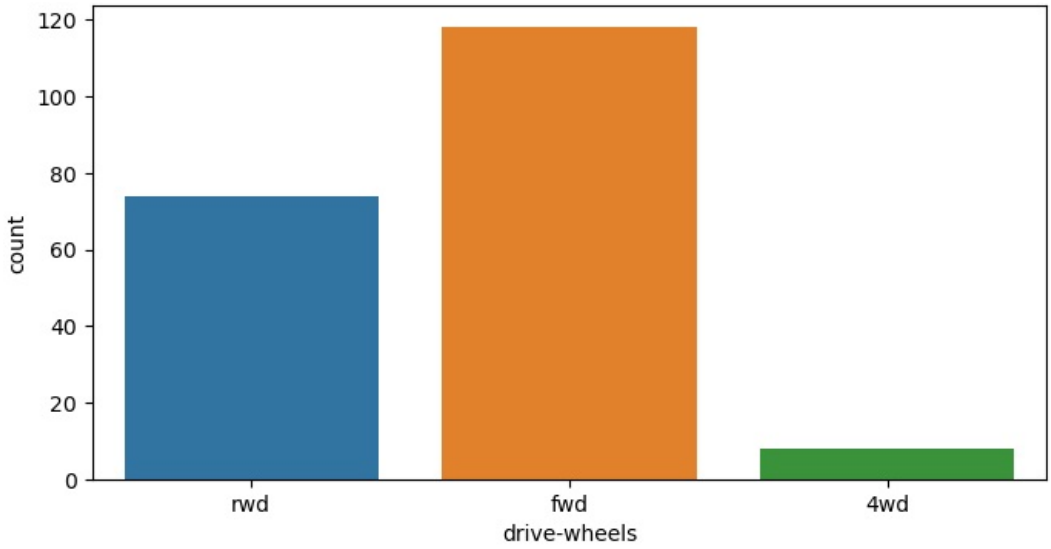


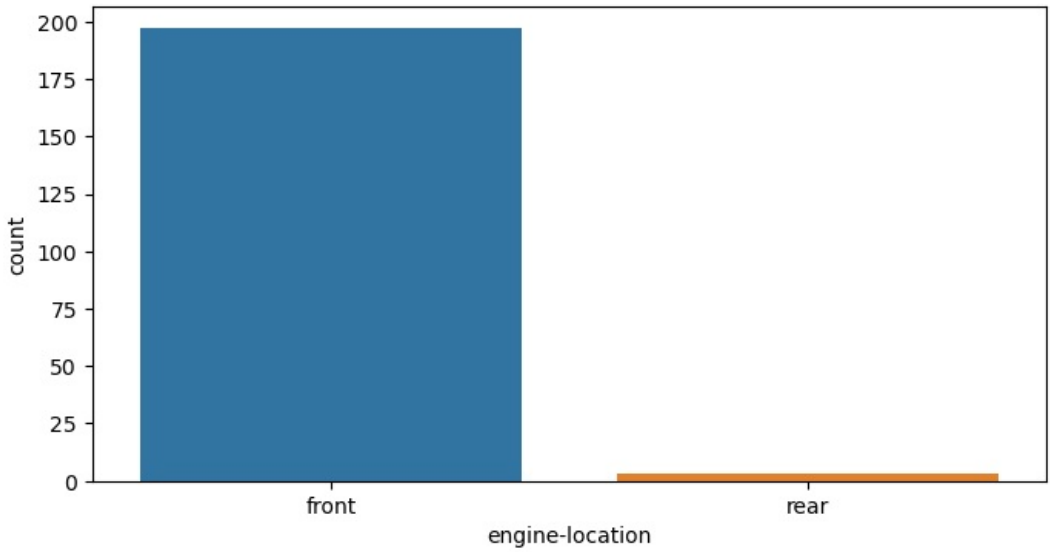

175
150
125
100
75
50
25
0
count
gas
diesel
fuel-type

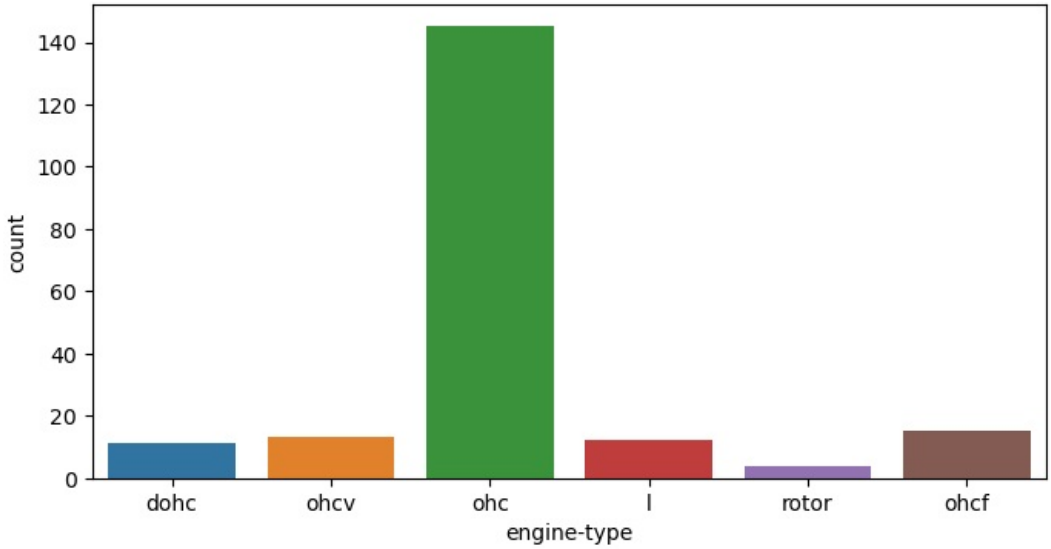

160
140
120
100
80
60
40
20
0
count
std
turbo
aspiration

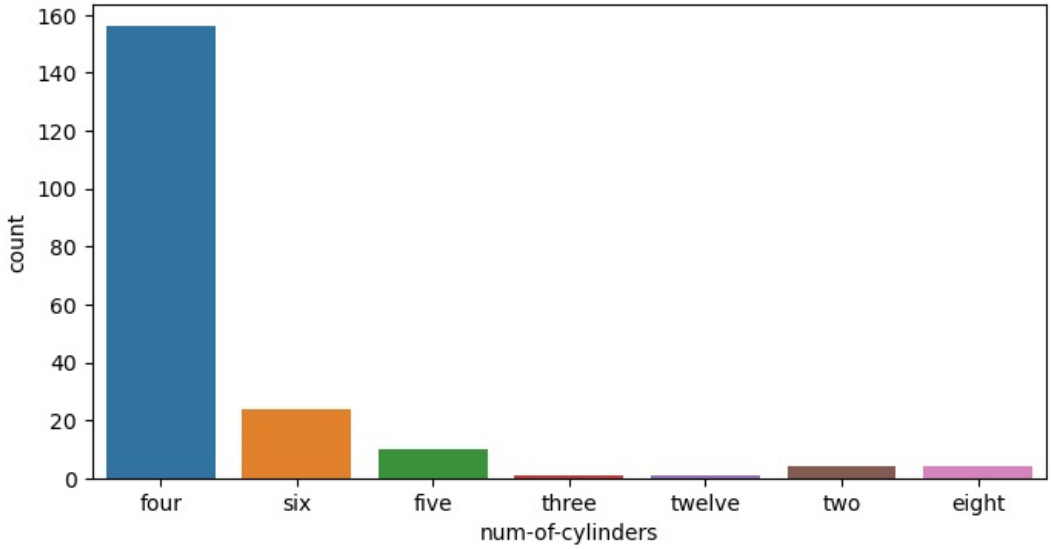

100
80
60
40
20
0
count
two
four
num-of-doors

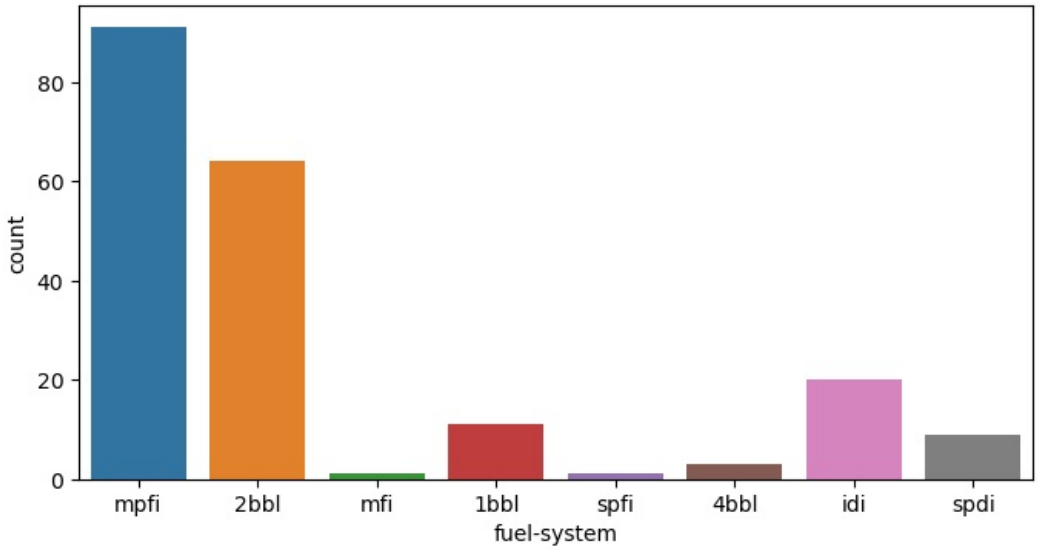

80
60
40
20
0
count
convertible
hatchback
sedan
wagon
hardtop
body-style


120
100
80
60
40
20
0
count
rwd
fwd
4wd
drive-wheels

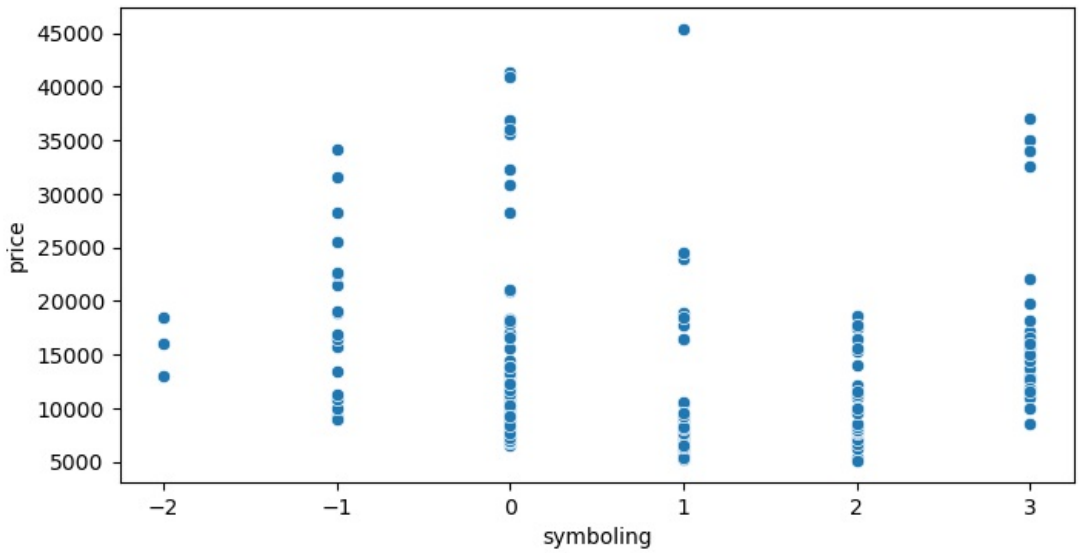

200
175
150
125
100
75
50
25
0
count
front
rear
engine-location

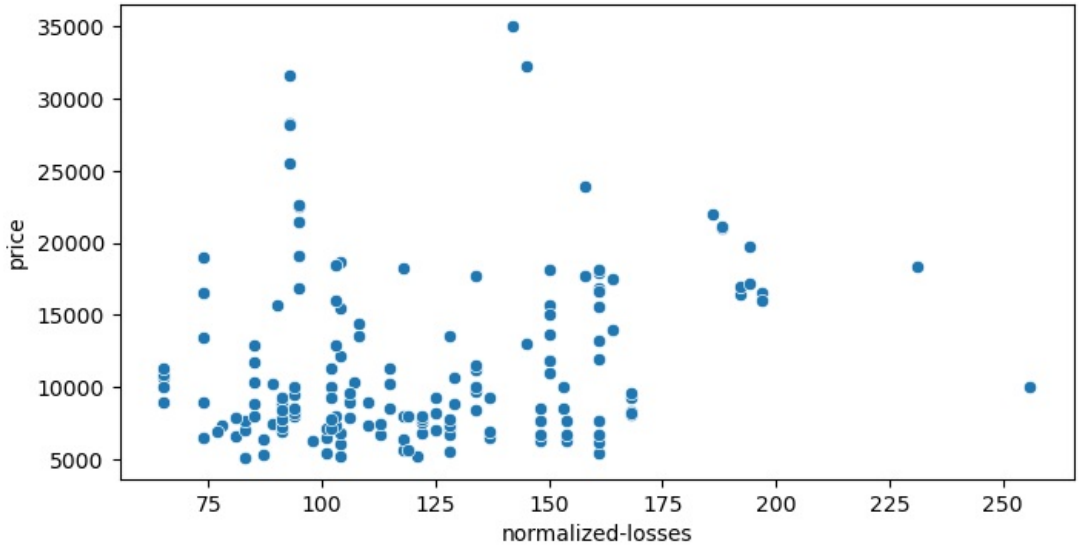

count
0
20
40
60
80
100
120
140
dohc
ohcv
ohc
l
rotor
ohcf
engine-type

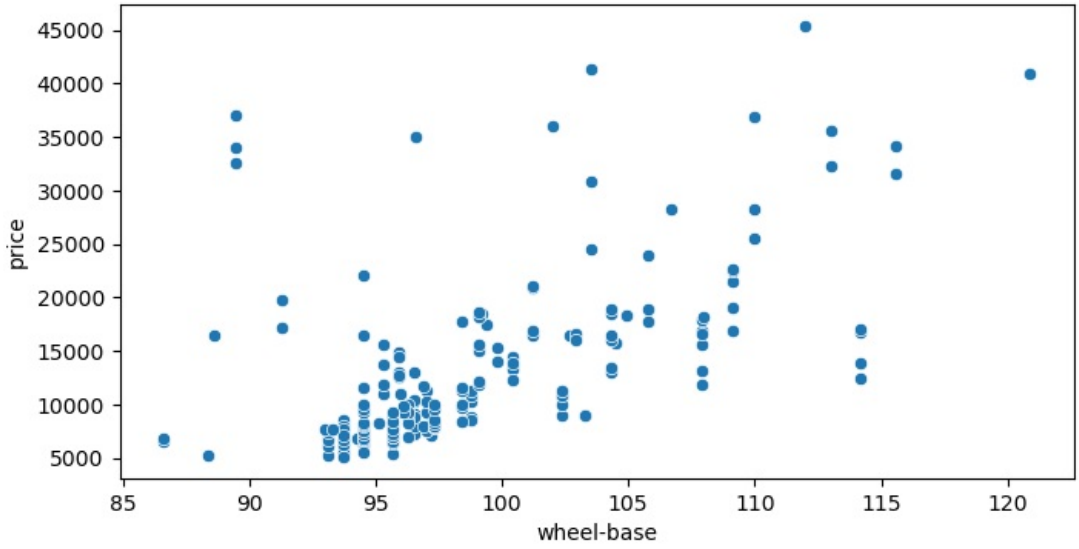

count
0
20
40
60
80
100
120
140
160
four
six
five
three
twelve
two
eight
num-of-cylinders

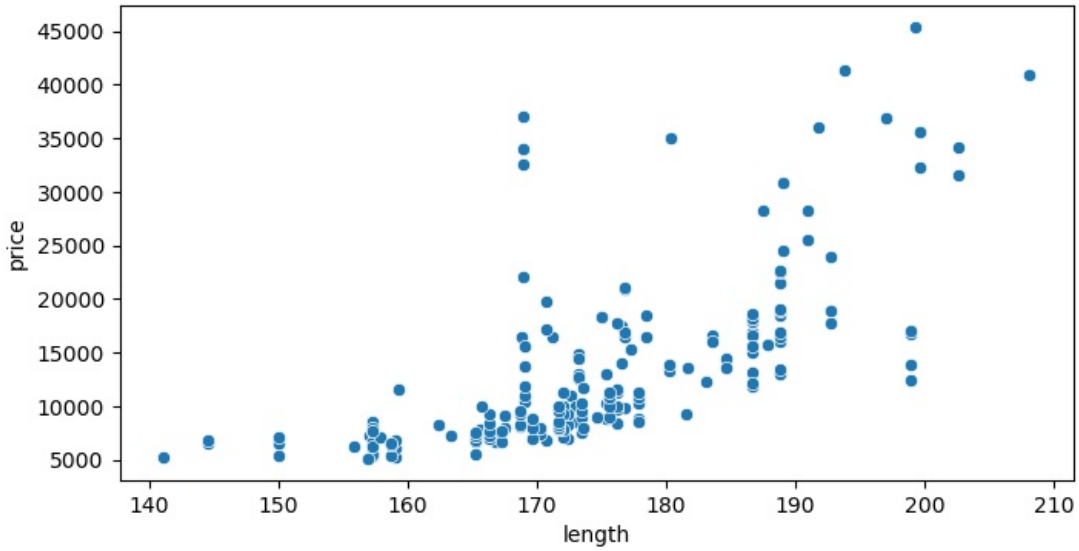


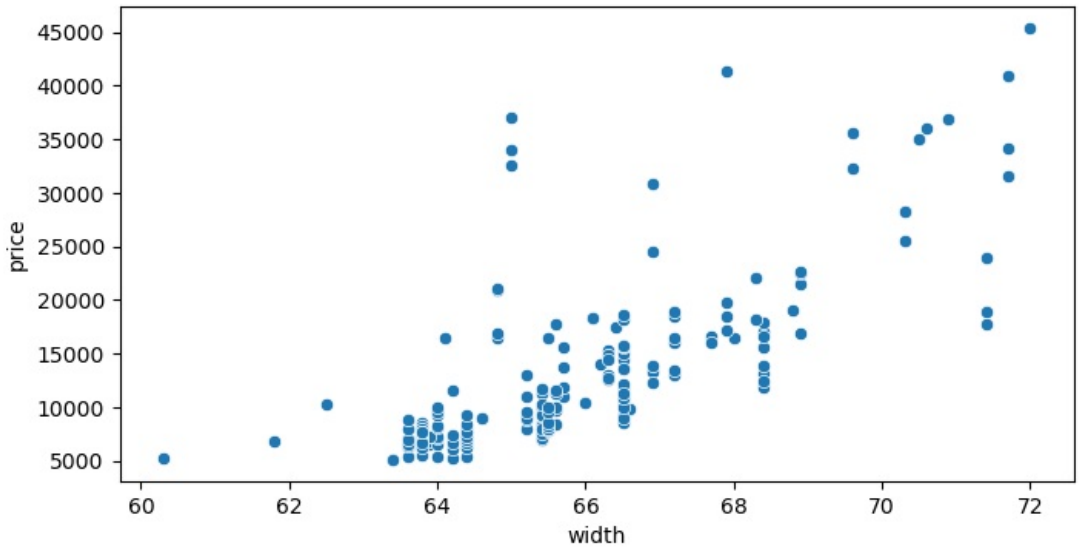
In [12]:
```
# Bivariate Analysis - Relationship with target variable
for col in numeric_features:
    plt.figure(figsize=(8, 4))
    sns.scatterplot(x=df[col], y=df['price'])
```

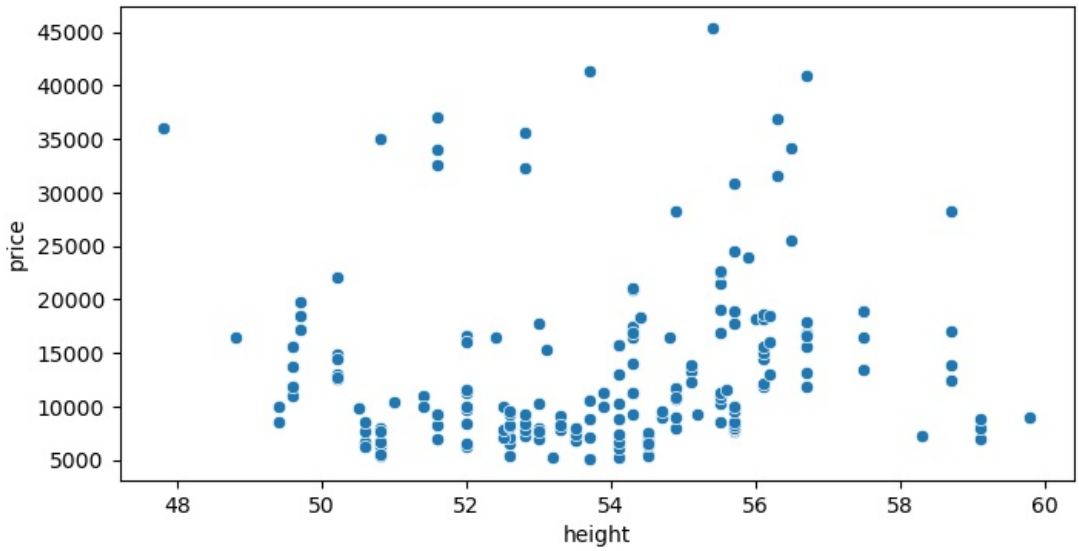


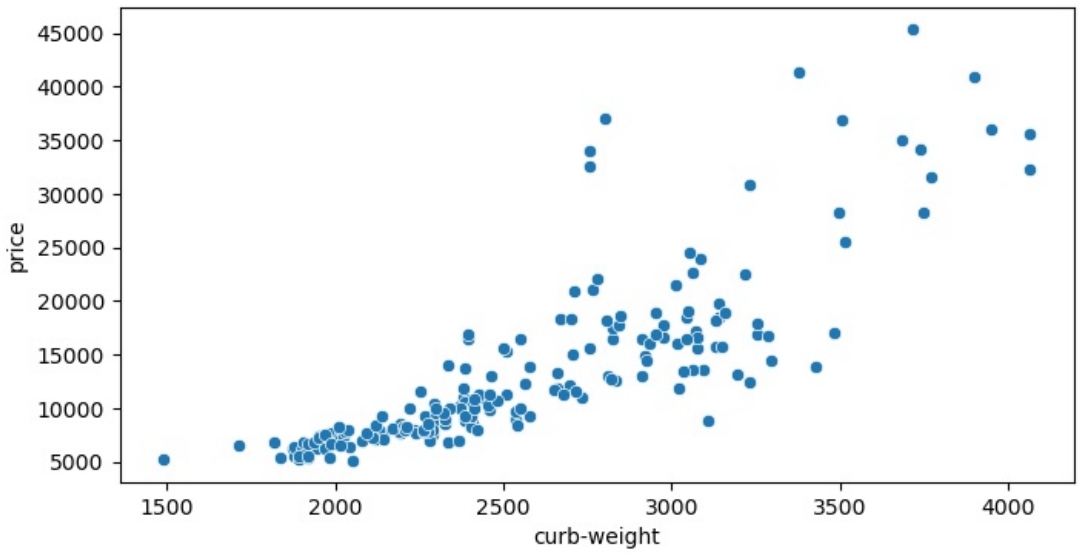

price
35000
30000
25000
20000
15000
10000
5000
75
100
125
150
175
200
225
250
normalized-losses

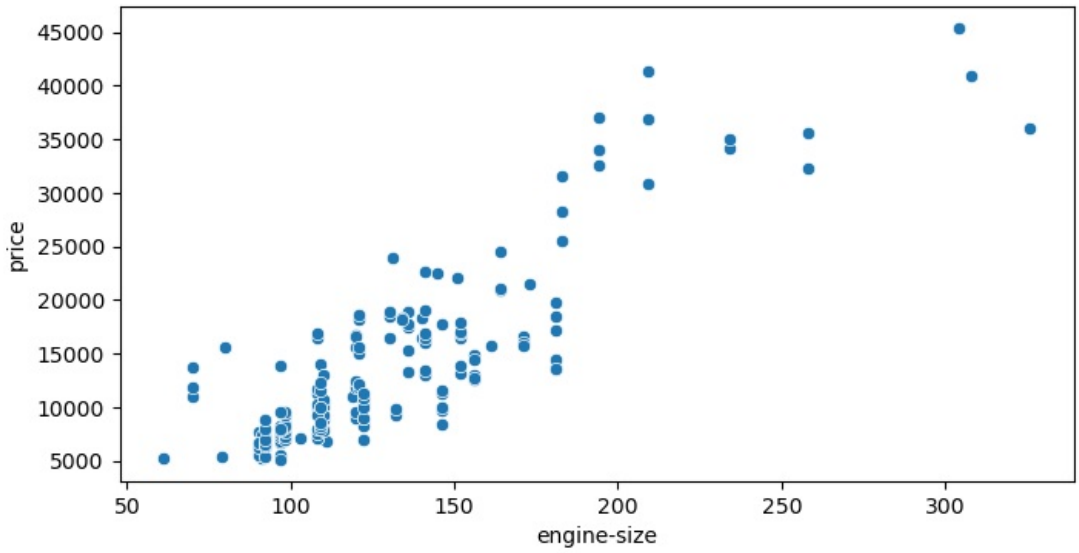

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
85
90
95
100
105
110
115
120
wheel-base

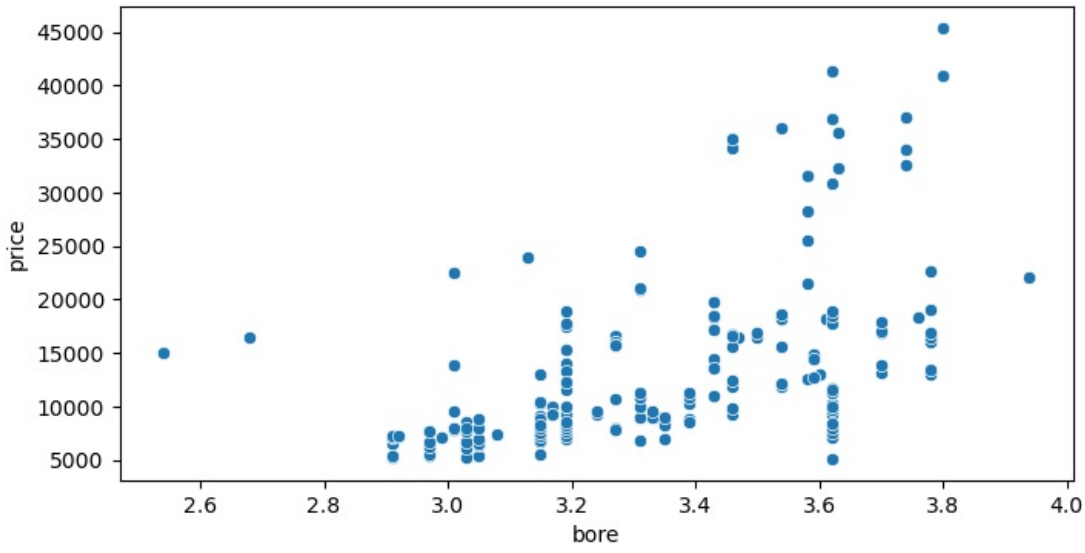

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
140
150
160
170
180
190
200
210
length

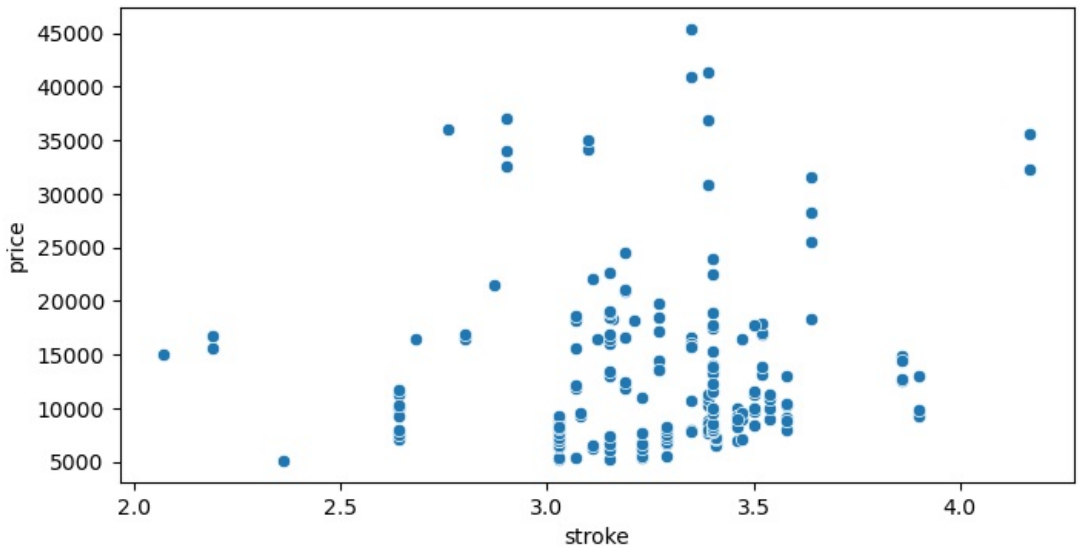

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
60
62
64
66
68
70
72
width

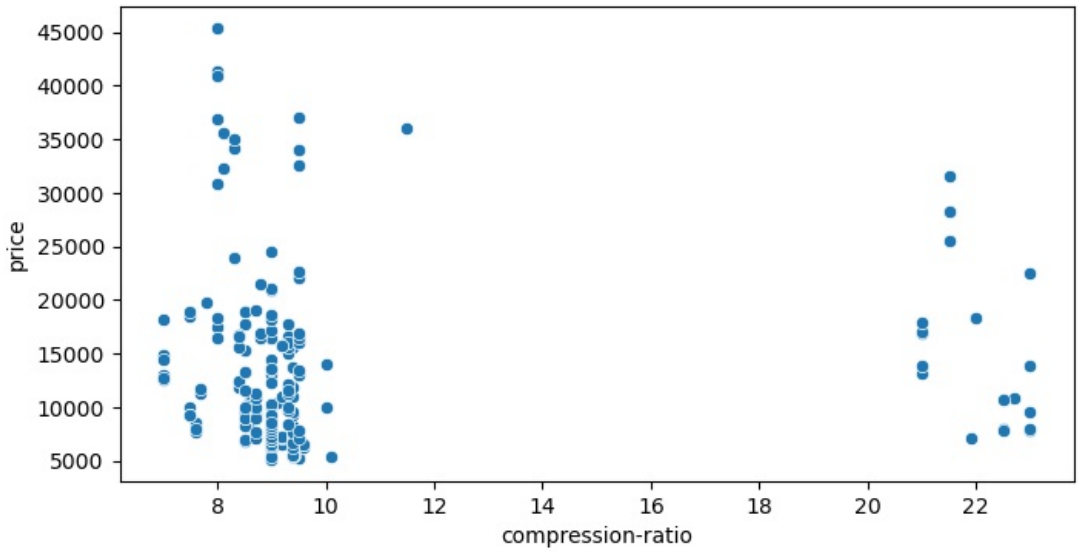

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
48
50
52
54
56
58
60
height

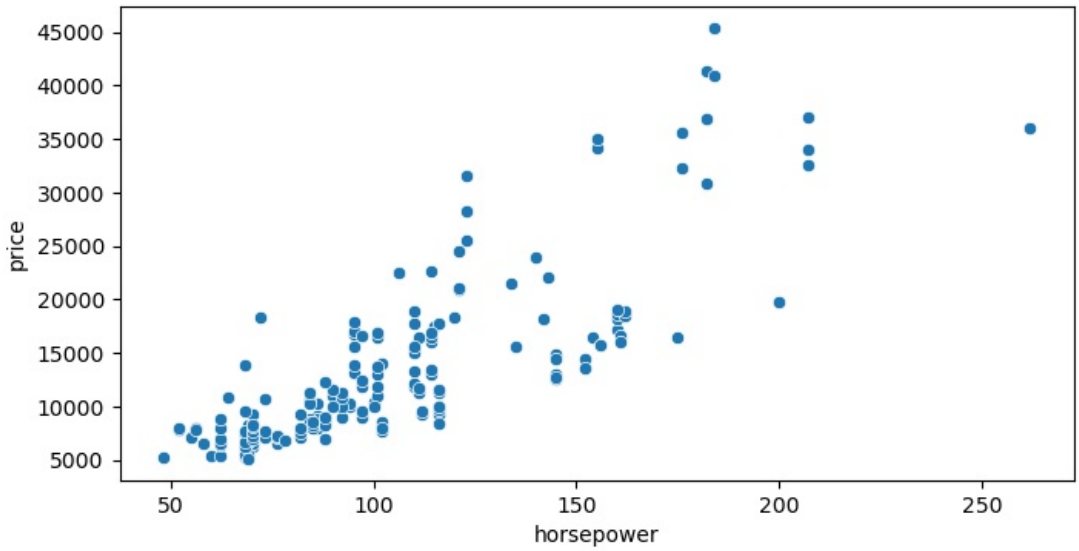

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
1500
2000
2500
3000
3500
4000
curb-weight

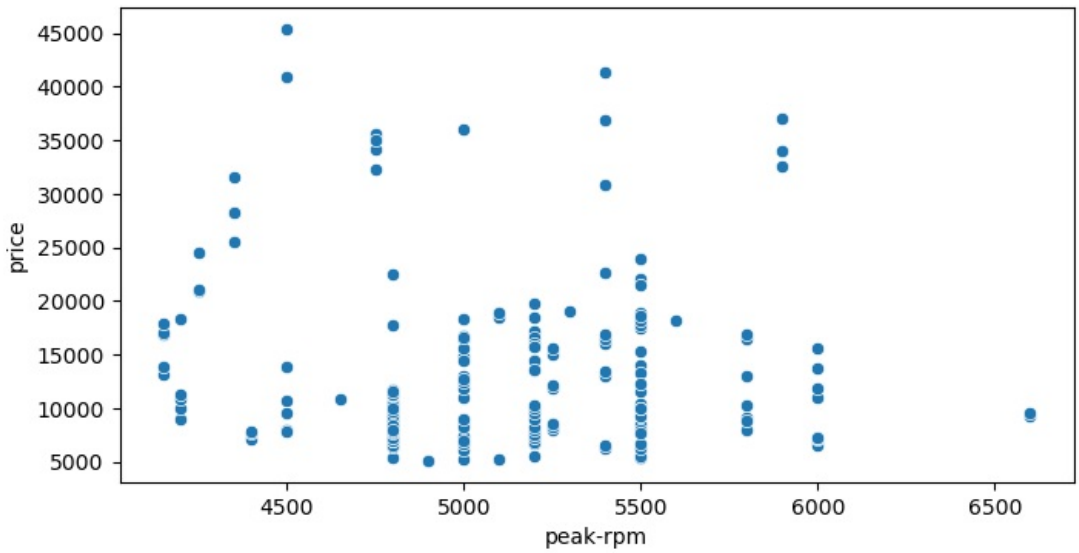

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
50
100
150
200
250
300
engine-size

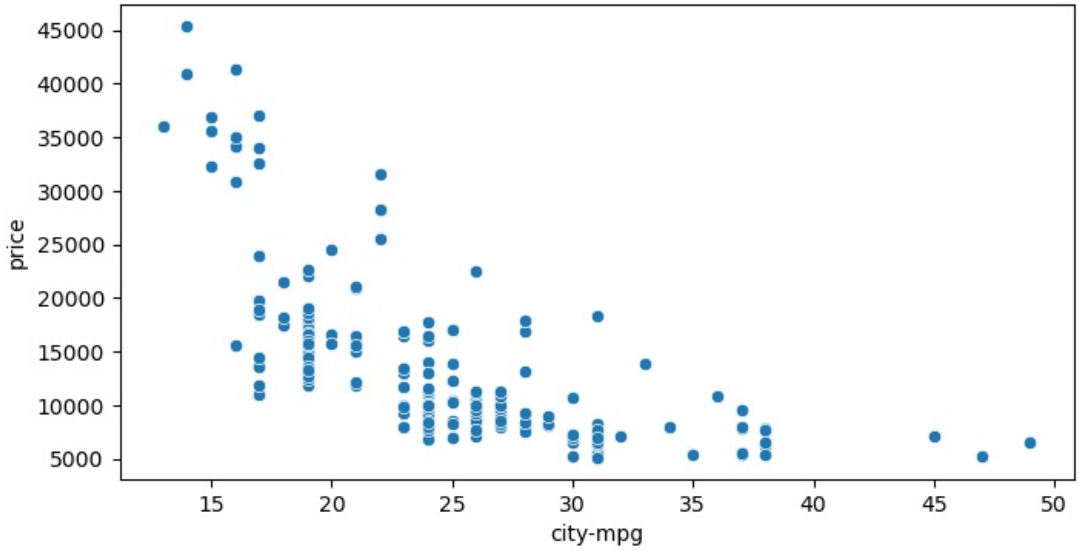

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
bore

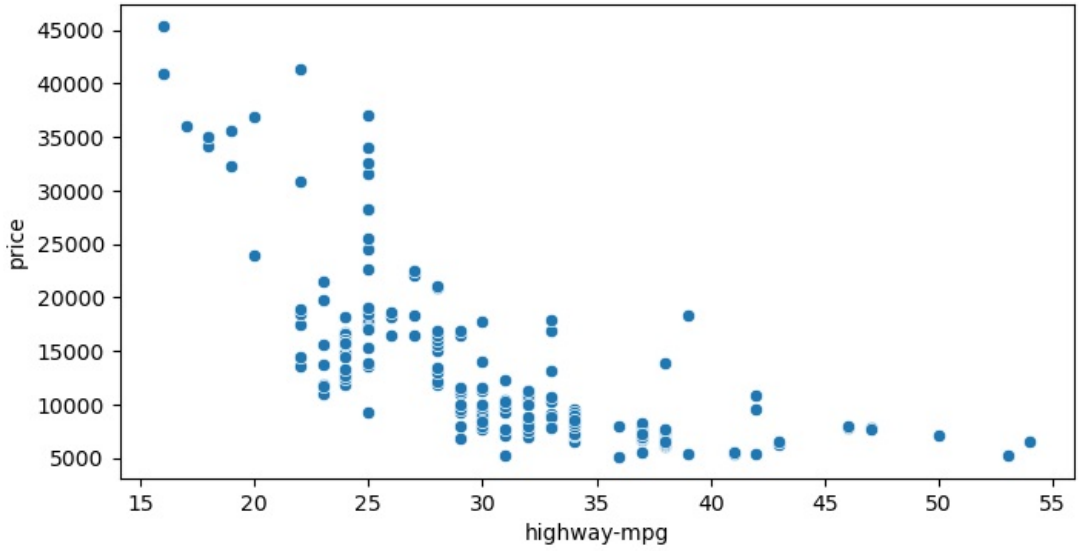

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
2.0
2.5
3.0
3.5
4.0
stroke

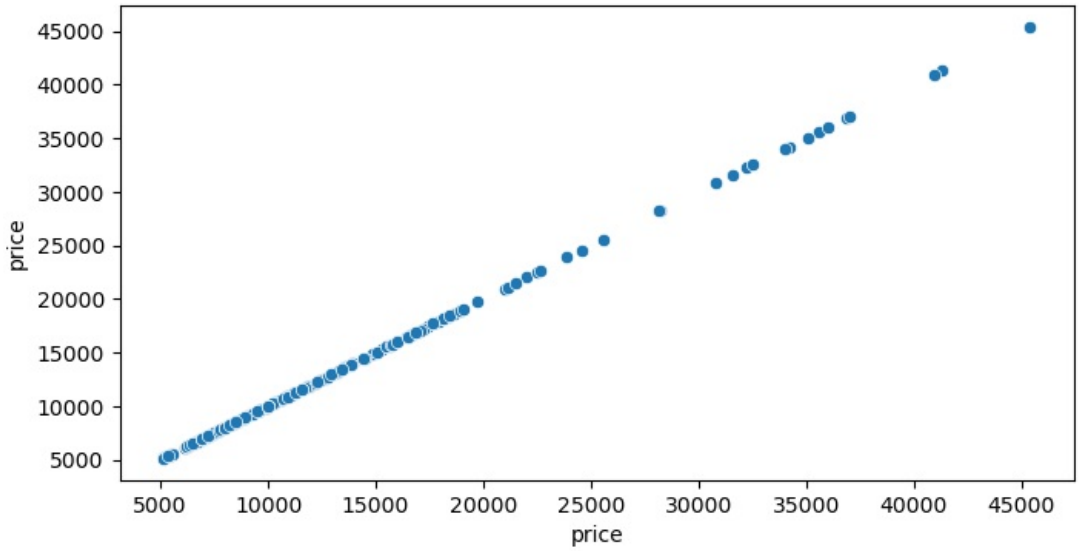

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
8
10
12
14
16
18
20
22
compression-ratio


price
45000
40000
35000
30000
25000
20000
15000
10000
5000
50
100
150
200
250
horsepower

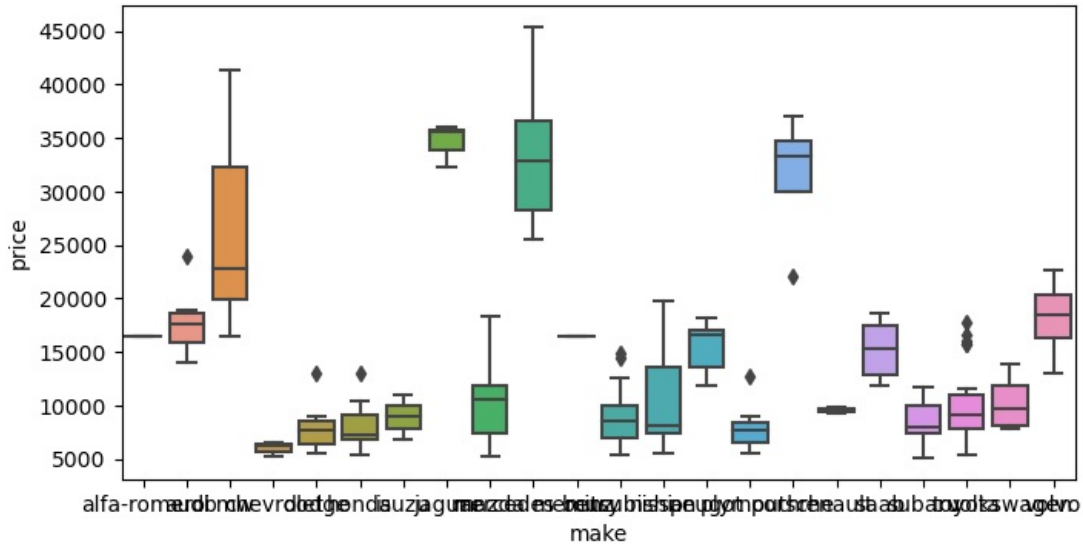

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
4500
5000
5500
6000
6500
peak-rpm

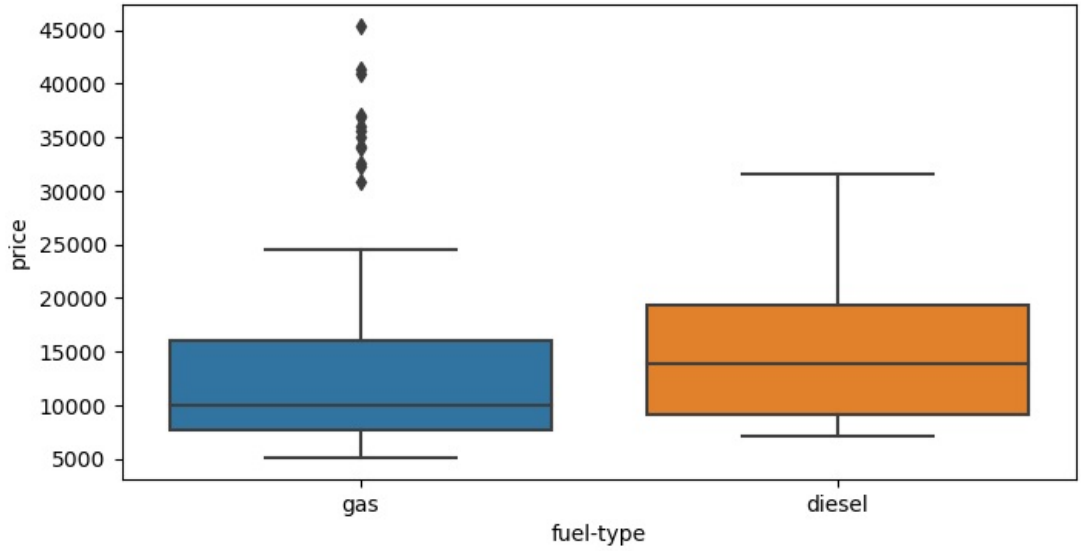


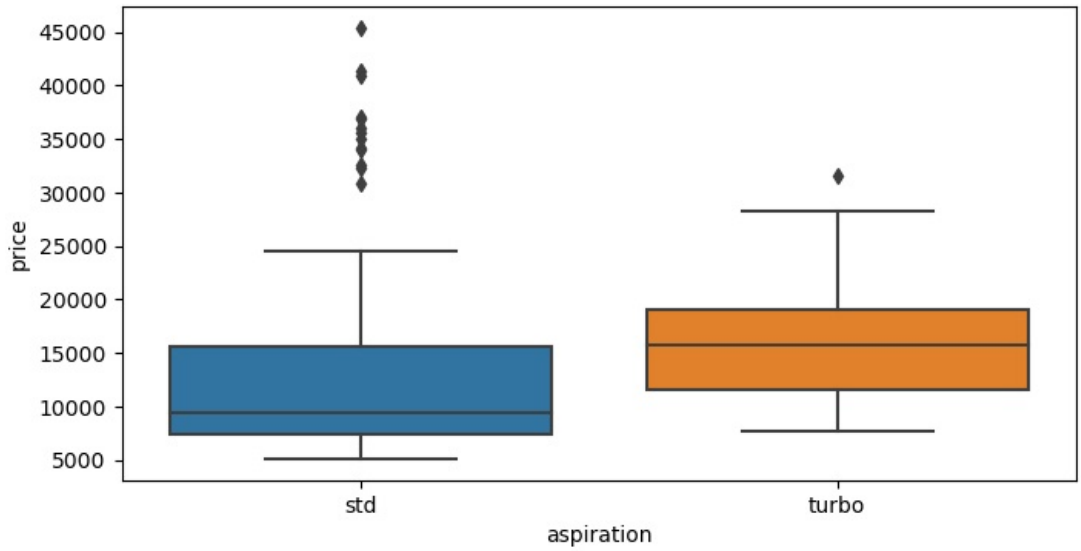


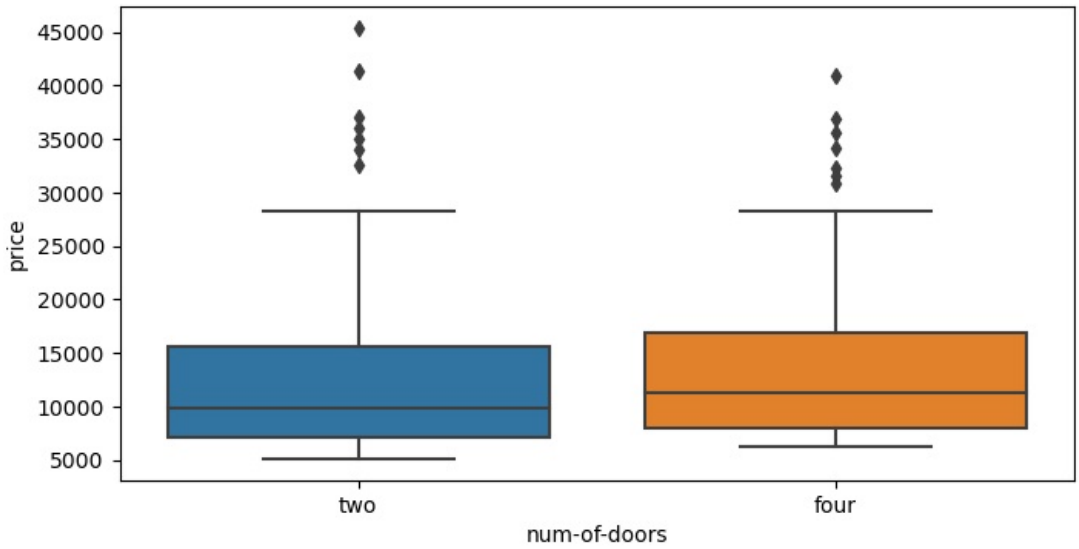


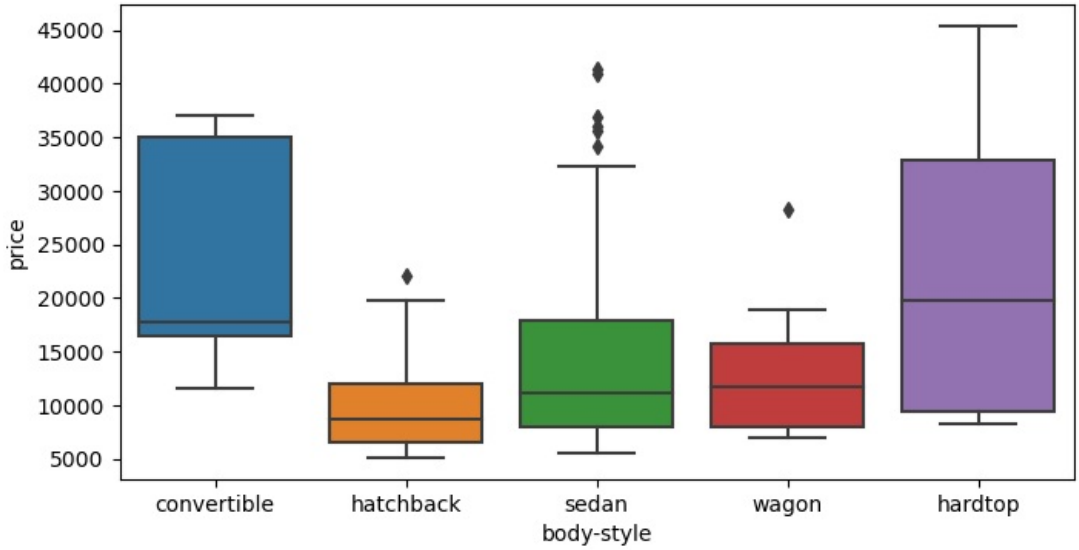
In [13]:
```
for col in categorical_features:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=df[col], y=df['price'])
```

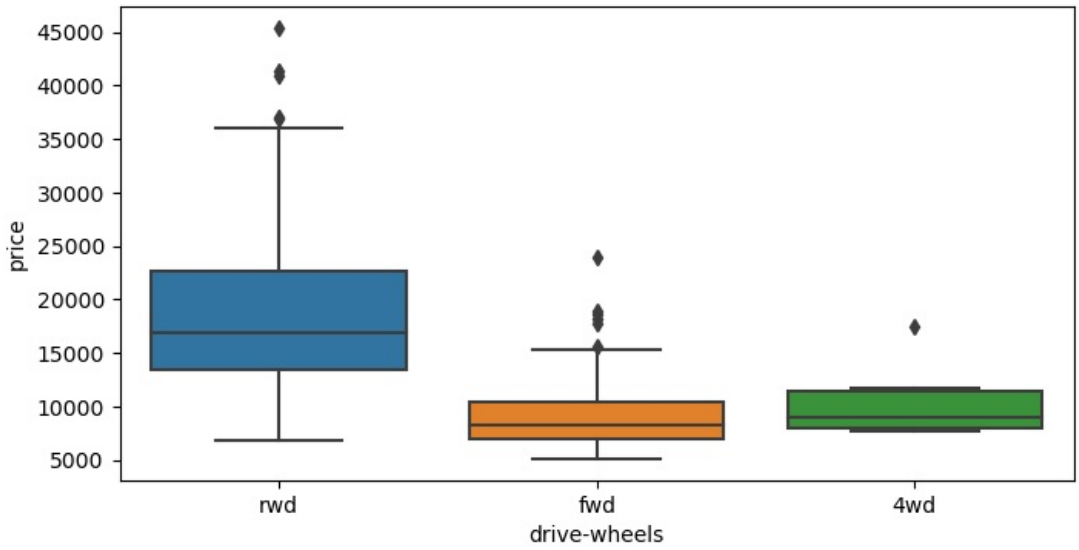

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
alfa-romero
audi
bmw
chevrolet
dodge
honda
isuzu
jaguar
mazda
mercedes-benz
mercury
mitsubishi
nissan
peugot
plymouth
porsche
renault
saab
subaru
toyota
volkswagen
volvo
make

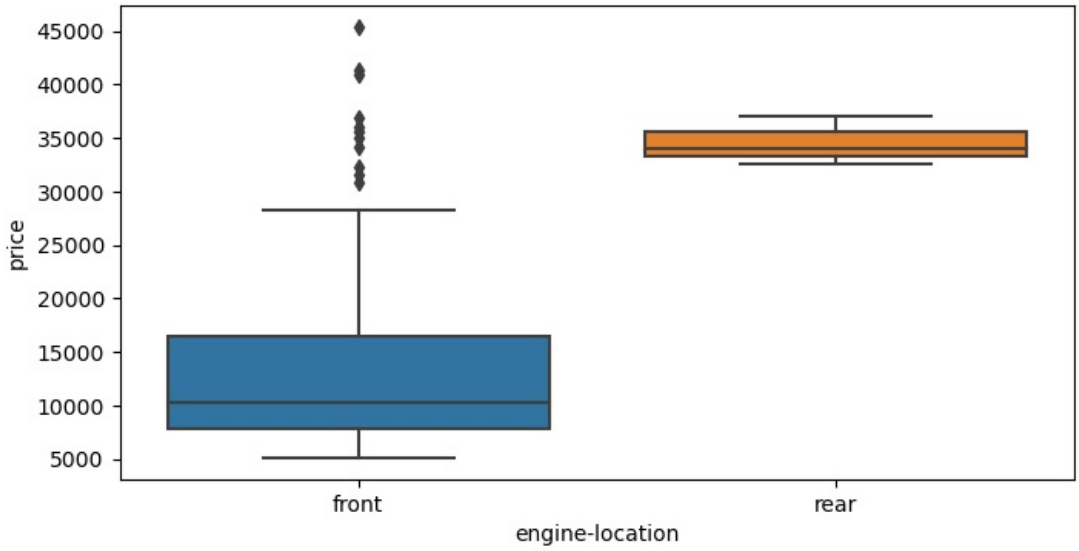

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
gas
diesel
fuel-type

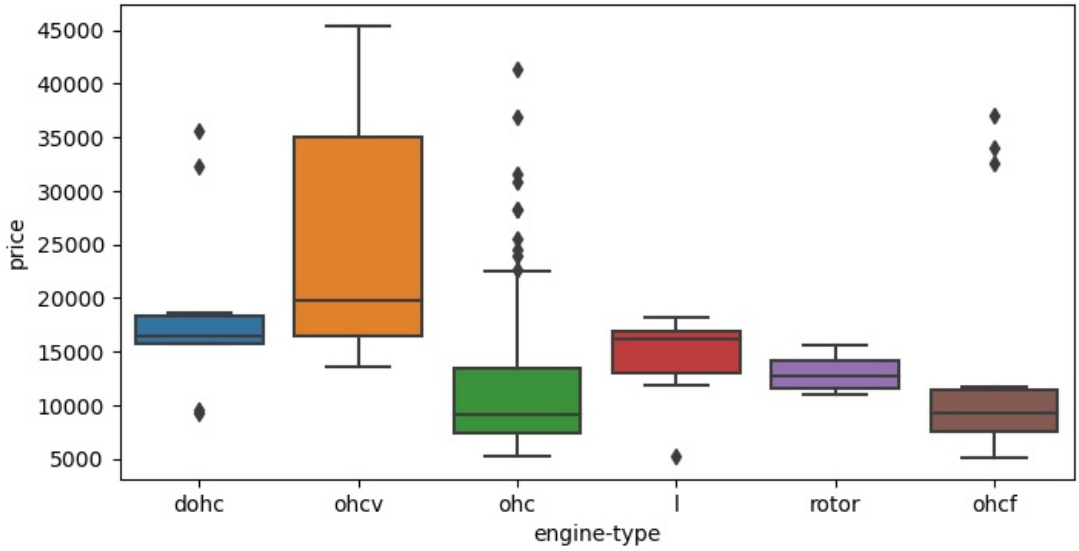

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
std
turbo
aspiration

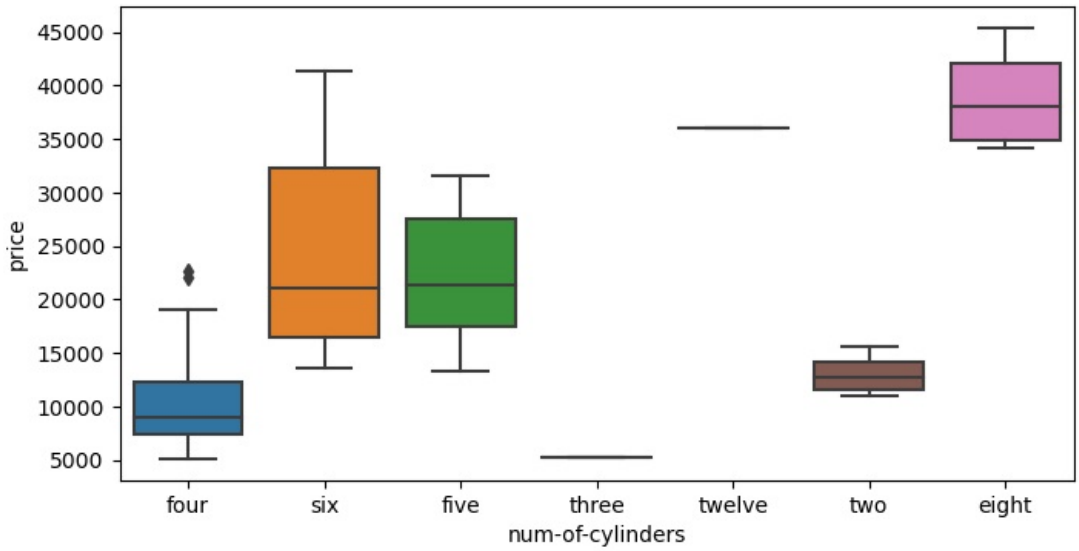

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
two
four
num-of-doors

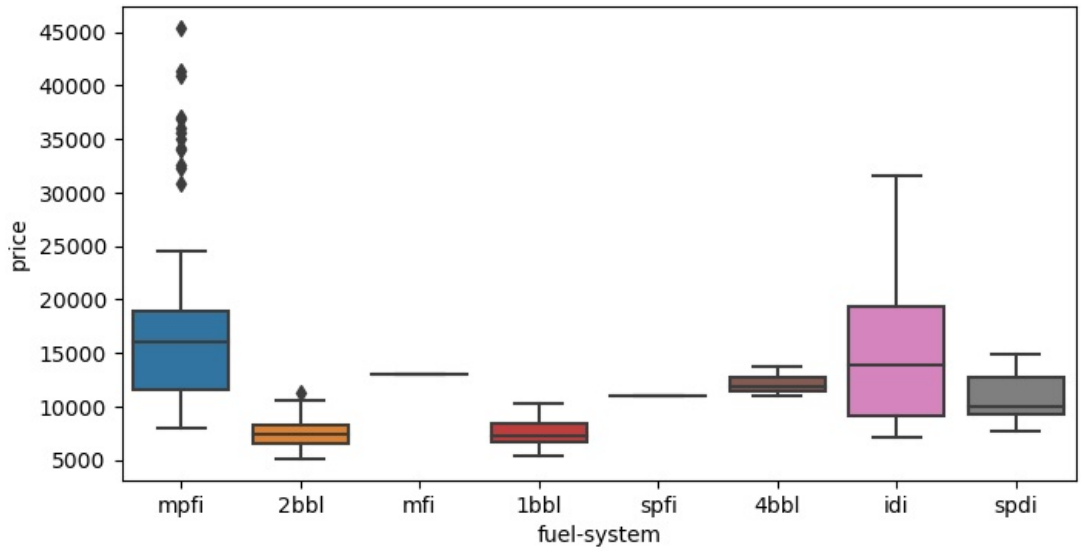

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
convertible
hatchback
sedan
wagon
hardtop
body-style


45000
40000
35000
30000
25000
20000
15000
10000
5000
price
rwd
fwd
4wd
drive-wheels


price
45000
40000
35000
30000
25000
20000
15000
10000
5000
front
rear
engine-location


price
45000
40000
35000
30000
25000
20000
15000
10000
5000
dohc
ohcv
ohc
l
rotor
ohcf
engine-type


price
45000
40000
35000
30000
25000
20000
15000
10000
5000
four
six
five
three
twelve
two
eight
num-of-cylinders



In [14]:
```
# Define features and target variable
X = df.drop(columns='price')
y = df['price']
```

In [15]:
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [16]:
```
# Identify numeric and categorical columns
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object']).columns
```

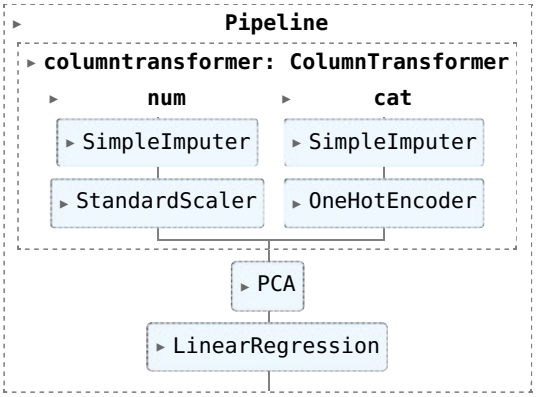
In [17]:
```
# Define preprocessing steps for numeric and categorical data
numeric_transformer = make_pipeline(
    SimpleImputer(strategy='median'),
    StandardScaler()
)
categorical_transformer = make_pipeline(
    SimpleImputer(strategy='most_frequent'),
    OneHotEncoder(handle_unknown='ignore', sparse_output=False))
```

In [18]:
```
# Combine preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)
```

In [19]:
```
# Linear Regression Model
linear_model = make_pipeline(
    preprocessor,
    PCA(n_components=0.95),
    LinearRegression()
)
```

In [20]:
```
# Fit the linear model
linear_model.fit(X_train, y_train)
```

Out[20]:



In [21]:
```
# Evaluate the linear model
y_pred_linear = linear_model.predict(X_test)
r2_linear = r2_score(y_test, y_pred_linear)
```

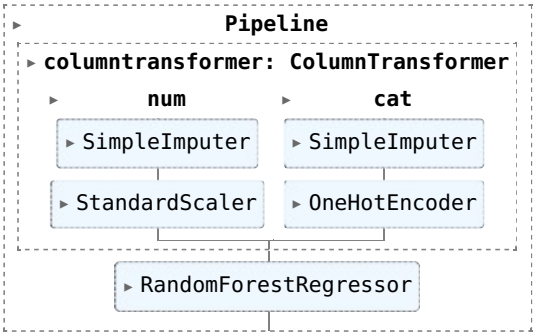
In [22]:
```
print(r2_linear)
```

```
10761872.025718259
0.9039531951316362
```

In [23]:
```
# Random Forest Model
random_forest_model = make_pipeline(
    preprocessor,
    RandomForestRegressor(n_estimators=100, random_state=42)
)
```

In [24]:
```
# Fit the random forest model
random_forest_model.fit(X_train, y_train)
```

Out[24]:



In [25]:
```
# Evaluate the random forest model
y_pred_rf = random_forest_model.predict(X_test)
```

In [26]:
```
r2_rf = r2_score(y_test, y_pred_rf)
```

In [27]:
```
print(r2_rf)
```

```
0.9685198162627805
```