Student Name: Tushar Panchal

En. No. : 21162101014

Branch: CBA

Batch: 71

Subject: ML (Machine Learning)

EXPERIMENT 2

Feature extraction using visualization and statistical test

Import dataset available at following url:

https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data

In [31]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

# Load the dataset
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.
df = pd.read_csv(path, na_values="?", header=None)

# Define the column headers
headers = [
    "symboling",
    "normalized-losses",
    "make",
    "fuel-type",
    "aspiration",
    "num-of-doors",
    "body-style",
    "drive-wheels",
    "engine-location",
    "wheel-base",
    "length",
    "width",          # Added missing comma here
```

```
    "height",
    "curb-weight",
    "engine-type",
    "num-of-cylinders",
    "engine-size",
    "fuel-system",
    "bore",
    "stroke",
    "compression-ratio",
    "horsepower",
    "peak-rpm",
    "city-mpg",
    "highway-mpg",
    "price",
]

# Assign column names to the DataFrame
df.columns = headers

# Display the first 5 rows of the DataFrame
print("The first 5 rows of the dataframe:")
print(df.head())
```

```
The first 5 rows of the dataframe:
   symboling  normalized-losses         make fuel-type aspiration  \
0          3                NaN  alfa-romero       gas        std
1          3                NaN  alfa-romero       gas        std
2          1                NaN  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheel-base  ...  \
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  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

Data Preprocessing:

In [32]:

```
# Replace missing values with the mean for numerical columns and the mode for categ
df['normalized-losses'].replace('?', np.nan, inplace=True)
df['bore'].replace('?', np.nan, inplace=True)
df['stroke'].replace('?', np.nan, inplace=True)
df ['horsepower'].replace('?', np.nan, inplace=True)
df['peak-rpm'].replace('?', np.nan, inplace=True)
df['num-of-doors'].replace('?', np.nan, inplace=True)
```

```python
# Fill missing values
df ['normalized-losses'].fillna(df['normalized-losses'].astype('float').mean(), inp
df['bore'].fillna (df ['bore'].astype('float').mean(), inplace=True)
df['stroke'].fillna (df ['stroke'].astype('float').mean(), inplace=True)
df ['horsepower'].fillna(df['horsepower'].astype('float').mean(), inplace=True)
df['peak-rpm'].fillna (df ['peak-rpm'].astype('float').mean(), inplace=True)
df ['num-of-doors'].fillna (df['num-of-doors'].mode() [0], inplace=True)

#Convert data types to appropriate formats
df['price'] = df ['price'].replace('?', np.nan).astype('float')
df.dropna (subset=['price'], inplace=True) # Remove rows with NaN values in 'price'
df ['price'] = df['price'].astype('float')
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')
```

1.List down all the continuous attributes in the dataset:

In [33]:

```python
# Select continuous attributes (numerical columns)
continuous_attributes = df.select_dtypes(include=['float64', 'int64']).columns.toli

print("\nContinuous Attributes: \n")
for attribute in continuous_attributes:
    print(f'⭐{attribute}')
```

```
Continuous Attributes:

⭐symboling
⭐normalized-losses
⭐wheel-base
⭐length
⭐width
⭐height
⭐curb-weight
⭐engine-size
⭐bore
⭐stroke
⭐compression-ratio
⭐horsepower
⭐peak-rpm
⭐city-mpg
⭐highway-mpg
⭐price
```

2. List down all the categorical attributes in the dataset:

In [34]:

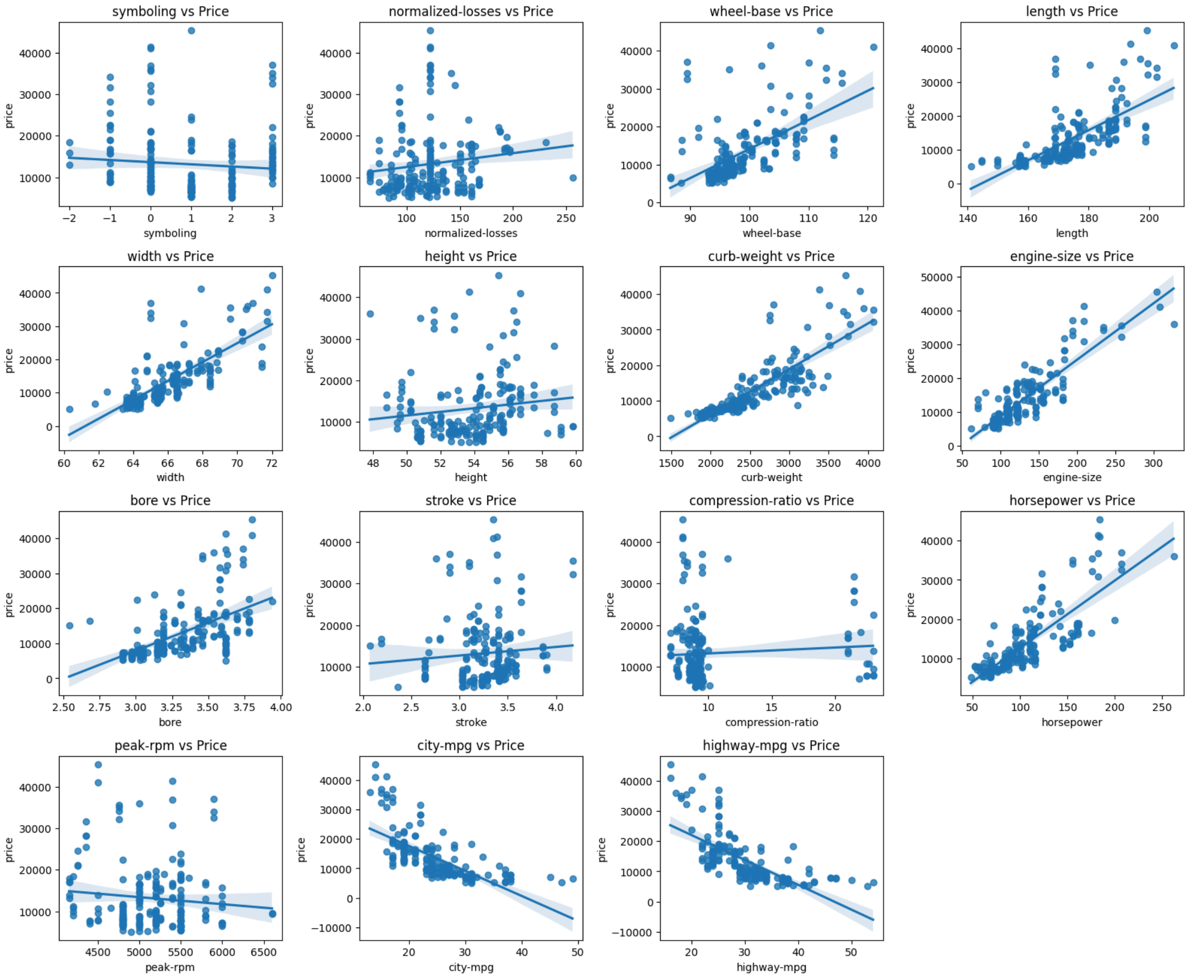
```python
# Select categorical attributes (columns with object type)
categorical_attributes = df.select_dtypes(include=['object']).columns.tolist()

print("\nCategorical Attributes: \n")
for attribute in categorical_attributes:
    print(f'🌟{attribute}')
```

```
Categorical Attributes:

🌟 make
🌟 fuel-type
🌟 aspiration
🌟 num-of-doors
🌟 body-style
🌟 drive-wheels
🌟 engine-location
🌟 engine-type
🌟 num-of-cylinders
🌟 fuel-system
```

3. Draw regplot between each continuous attribute and price and write down whether that attribute is related to price or not.

In [35]:

```python
# Data visualization for continuous attributes
plt.figure(figsize=(16, 16))

for i, attr in enumerate(continuous_attributes):
    if attr != 'price':
        plt.subplot(5, 4, i + 1)
        sns.regplot(x=attr, y='price', data=df)
        plt.title(f'{attr} vs Price')

plt.tight_layout()
plt.show()

# Check if continuous attributes are related to price
print("\nRelationship of Continuous Attributes with Price:\n")

for attr in continuous_attributes:
    if attr != 'price':
        correlation = df[[attr, 'price']].corr().iloc[0, 1]
        print(f'{attr}: {"✅" if abs(correlation) > 0.5 else "❌"}')
```



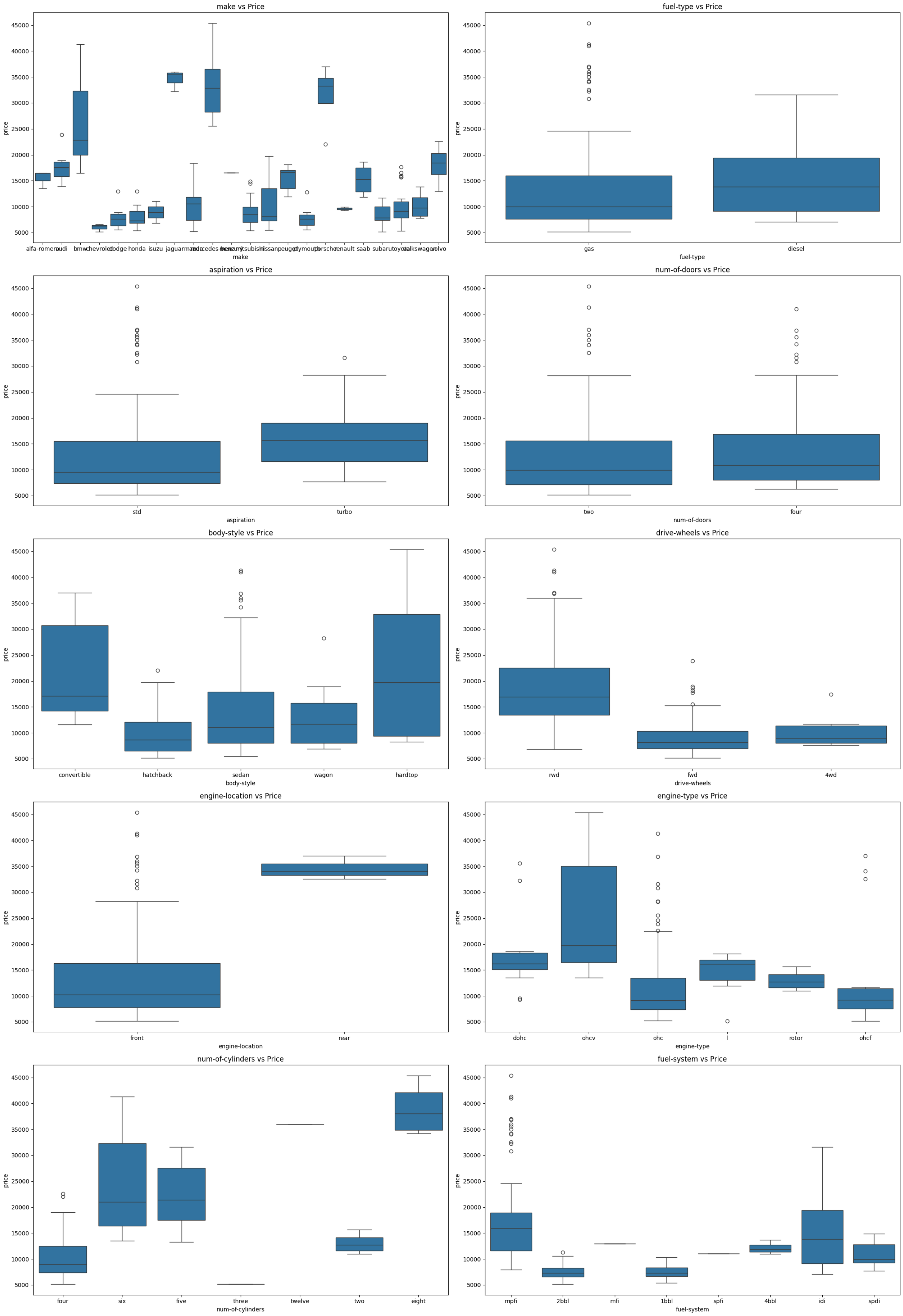

```
Relationship of Continuous Attributes with Price:

symboling: ❌
normalized-losses: ❌
wheel-base: ✅
length: ✅
width: ✅
height: ❌
curb-weight: ✅
engine-size: ✅
bore: ✅
stroke: ❌
compression-ratio: ❌
horsepower: ✅
peak-rpm: ❌
city-mpg: ✅
highway-mpg: ✅
```

4. Draw boxplot between each categorical attribute and price and write down whether that attribute is related to price or not.

In [36]:

```python
# Data visualization for categorical attributes
plt.figure(figsize=(22, 32))

for i, attr in enumerate(categorical_attributes):
    plt.subplot(5, 2, i + 1)
    sns.boxplot(x=attr, y='price', data=df)
    plt.title(f'{attr} vs Price')

plt.tight_layout()
```

```python
plt.show()

# Check if categorical attributes are related to price
print("\nRelationship of Categorical Attributes with Price:\n")

for attr in categorical_attributes:
    grouped_test = df[[attr, 'price']].groupby([attr])
    unique_values = df[attr].unique()

    if len(unique_values) > 1:
        f_val, p_val = stats.f_oneway(*[grouped_test.get_group(val)['price'] for va
        print(f'{attr}: {"✅" if p_val < 0.05 else "❌"}')
```


make vs Price
price
make
fuel-type vs Price
gas
diesel
fuel-type
aspiration vs Price
std
turbo
aspiration
num-of-doors vs Price
two
four
num-of-doors
body-style vs Price
convertible
hatchback
sedan
wagon
hardtop
body-style
drive-wheels vs Price
rwd
fwd
4wd
drive-wheels
engine-location vs Price
front
rear
engine-location
engine-type vs Price
dohc
ohcv
ohc
l
rotor
ohcf
engine-type
num-of-cylinders vs Price
four
six
five
three
twelve
two
eight
num-of-cylinders
fuel-system vs Price
mpfi
2bbl
mfi
1bbl
spfi
4bbl
idi
spdi
fuel-system

```
Relationship of Categorical Attributes with Price:

make: ✅
fuel-type: ❌
aspiration: ✅
num-of-doors: ❌
body-style: ✅
drive-wheels: ✅
engine-location: ✅
engine-type: ✅
num-of-cylinders: ✅
fuel-system: ✅
```

5. Calculate pearson correlation between each continuous attribute and price and write down whether that attribute is related to price or not.

In [37]:

```
# Print Pearson Correlation Coefficients with Price (and if related)
print("\nPearson Correlation Coefficients with Price (and if related):\n")

for attr in continuous_attributes:
    if attr != 'price':
        correlation = df[[attr, 'price']].corr().iloc[0, 1]
        print(f'{attr}: {correlation:.2f} {"(✅)" if abs(correlation) > 0.5 else "
```

```
Pearson Correlation Coefficients with Price (and if related):

symboling: -0.08 (❌)
normalized-losses: 0.13 (❌)
wheel-base: 0.58 (✅)
length: 0.69 (✅)
width: 0.75 (✅)
height: 0.14 (❌)
curb-weight: 0.83 (✅)
engine-size: 0.87 (✅)
bore: 0.54 (✅)
stroke: 0.08 (❌)
compression-ratio: 0.07 (❌)
horsepower: 0.81 (✅)
peak-rpm: -0.10 (❌)
city-mpg: -0.69 (✅)
highway-mpg: -0.70 (✅)
```

6. Calculate ANOVA between each categorical attribute and price and write down whether that attribute is related to price or not.

In [38]:

```
# Print ANOVA results for categorical attributes
print("\nANOVA Results for Categorical Attributes:\n")

for attr in categorical_attributes:
    grouped_test = df[[attr, 'price']].groupby([attr])
    unique_values = df[attr].unique()

    if len(unique_values) > 1:
        f_val, p_val = stats.f_oneway(*[grouped_test.get_group(val)['price'] for va
        print(f'{attr}: \n\tF={f_val:.2f}, \n\tp={p_val:.2e} {"(✅)" if p_val < 0.
```

```
ANOVA Results for Categorical Attributes:

make:
        F=33.23,
        p=1.07e-50 (✅)

fuel-type:
        F=2.45,
        p=1.19e-01 (❌)

aspiration:
        F=6.63,
        p=1.07e-02 (✅)

num-of-doors:
        F=0.36,
        p=5.50e-01 (❌)

body-style:
        F=9.13,
        p=8.78e-07 (✅)

drive-wheels:
        F=67.95,
        p=3.39e-23 (✅)

engine-location:
        F=24.50,
        p=1.58e-06 (✅)

engine-type:
        F=9.85,
        p=2.09e-08 (✅)

num-of-cylinders:
        F=54.94,
        p=2.87e-39 (✅)

fuel-system:
        F=15.02,
        p=1.31e-15 (✅)
```

7. List down the attributes which has significant impact on price.

In [39]:

```
# Lists to store attributes with significant impact on price
significant_continuous = []
significant_categorical = []

# Check for significant continuous attributes
for attr in continuous_attributes:
    if attr != 'price':
        correlation = df[[attr, 'price']].corr().iloc[0, 1]
        if abs(correlation) > 0.5:
            significant_continuous.append(attr)

# Check for significant categorical attributes
for attr in categorical_attributes:
    grouped_test = df[[attr, 'price']].groupby([attr])
    unique_values = df[attr].unique()
    mean_prices = [grouped_test.get_group(val)['price'].mean() for val in unique_va
```

```
    if mean_prices:
        max_diff = max(mean_prices) - min(mean_prices)
        if max_diff / df['price'].mean() > 0.2:  # Arbitrary threshold of 20% mean
            significant_categorical.append(attr)

# Print attributes with significant impact on price
print("\nAttributes with Significant Impact on Price:\n")

print("Continuous:\n")
for attr in significant_continuous:
    print(f'💎{attr}')

print("\nCategorical:\n")
for attr in significant_categorical:
    print(f'🎇{attr}')
```

```
Attributes with Significant Impact on Price:

Continuous:

💎wheel-base
💎length
💎width
💎curb-weight
💎engine-size
💎bore
💎horsepower
💎city-mpg
💎highway-mpg

Categorical:

🎇make
🎇fuel-type
🎇aspiration
🎇body-style
🎇drive-wheels
🎇engine-location
🎇engine-type
🎇num-of-cylinders
🎇fuel-system
```

8. Clean the assigned dataset and find important features from it.

In [40]:

```
# Combine significant continuous and categorical features
important_features = significant_continuous + significant_categorical

# Print important features
print("\nImportant Features:\n")
for feature in important_features:
    print(f'⭐{feature}')
```

```
Important Features:

⭐wheel-base
⭐length
⭐width
⭐curb-weight
⭐engine-size
⭐bore
⭐horsepower
⭐city-mpg
⭐highway-mpg
⭐make
⭐fuel-type
⭐aspiration
⭐body-style
⭐drive-wheels
⭐engine-location
⭐engine-type
⭐num-of-cylinders
⭐fuel-system
```