

HeshamMasterProjects

November 14, 2025

Mount Google Drive

```
[1]: from google.colab import drive
drive.mount('/content/gdrive')
```

Mounted at /content/gdrive

DATASET URL <https://www.kaggle.com/datasets/sumaya23abdul/automobile-database>

Import libraries

```
[17]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.impute import KNNImputer
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('display.max_columns', None)
pd.set_option('display.width', 220)
```

Explore Dataset

```
[3]: import pandas as pd
df = pd.read_csv('/content/gdrive/MyDrive/Master-Project/ExtractedData/
↳Automobile_data.csv')
df.head()
```

```
[3]:      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  \
0          3          ?  alfa-romero      gas      std          two
convertible          rwd          front      88.6  168.8   64.1   48.8
2548          dohc          four      130
1          3          ?  alfa-romero      gas      std          two
convertible          rwd          front      88.6  168.8   64.1   48.8
2548          dohc          four      130
2          1          ?  alfa-romero      gas      std          two
hatchback          rwd          front      94.5  171.2   65.5   52.4
```

2823	ohcv	six	152				
3	2	164	audi	gas	std	four	
sedan	fwd	front	99.8	176.6	66.2	54.3	
2337	ohc	four	109				
4	2	164	audi	gas	std	four	
sedan	4wd	front	99.4	176.6	66.4	54.3	
2824	ohc	five	136				

		fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg
highway-mpg	price							
0	mpfi	3.47	2.68		9.0	111	5000	21
27	13495							
1	mpfi	3.47	2.68		9.0	111	5000	21
27	16500							
2	mpfi	2.68	3.47		9.0	154	5000	19
26	16500							
3	mpfi	3.19	3.4		10.0	102	5500	24
30	13950							
4	mpfi	3.19	3.4		8.0	115	5500	18
22	17450							

Explore Data Types

```
[4]: df = pd.read_csv('/content/gdrive/MyDrive/Master-Project/ExtractedData/
Automobile_data.csv')
df.dtypes
```

```
[4]: symboling          int64
normalized-losses      object
make                   object
fuel-type               object
aspiration              object
num-of-doors            object
body-style              object
drive-wheels            object
engine-location         object
wheel-base             float64
length                 float64
width                  float64
height                 float64
curb-weight             int64
engine-type             object
num-of-cylinders        object
engine-size             int64
fuel-system             object
bore                   object
stroke                 object
```

```

compression-ratio    float64
horsepower           object
peak-rpm             object
city-mpg             int64
highway-mpg          int64
price               object
dtype: object

```

Explore Dataset Columns

```

[5]: df = pd.read_csv('/content/gdrive/MyDrive/Master-Project/ExtractedData/
↳Automobile_data.csv')
df.columns

[5]: Index(['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'],
dtype='object')

```

check missing data

```

[6]: df = pd.read_csv('/content/gdrive/MyDrive/Master-Project/ExtractedData/
↳Automobile_data.csv')
df.isnull().sum()

```

```

[6]: symboling           0
normalized-losses      0
make                   0
fuel-type              0
aspiration             0
num-of-doors           0
body-style             0
drive-wheels           0
engine-location        0
wheel-base            0
length                0
width                 0
height                0
curb-weight            0
engine-type            0
num-of-cylinders       0
engine-size            0
fuel-system            0
bore                   0
stroke                0

```

```

compression-ratio    0
horsepower            0
peak-rpm              0
city-mpg              0
highway-mpg          0
price                0
dtype: int64

```

handle missing data

```

[7]: print("\nPercentage of missing data:")
      missing_percent = df.isnull().mean()*100
      print(missing_percent)

      threshold = 30
      cols_to_drop = missing_percent[missing_percent > threshold].index
      df.drop(columns=cols_to_drop)
      print("\nDropped columns (>30% missing):", cols_to_drop)
      print("Remaining columns:", df.columns.tolist())

```

Percentage of missing data:

```

symboling            0.0
normalized-losses    0.0
make                0.0
fuel-type            0.0
aspiration            0.0
num-of-doors         0.0
body-style           0.0
drive-wheels         0.0
engine-location      0.0
wheel-base           0.0
length              0.0
width               0.0
height              0.0
curb-weight          0.0
engine-type          0.0
num-of-cylinders     0.0
engine-size          0.0
fuel-system          0.0
bore                 0.0
stroke              0.0
compression-ratio    0.0
horsepower           0.0
peak-rpm             0.0
city-mpg             0.0
highway-mpg          0.0
price               0.0

```

dtype: float64

Dropped columns (>30% missing): Index([], dtype='object')

Remaining columns: ['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']

Preprocessing

```
[11]: label_encoder = LabelEncoder()
num_cols = df.select_dtypes(include=[np.number]).columns
for col in num_cols:
    df[col] = label_encoder.fit_transform(df[col])

cat_cols = df.select_dtypes(include=['object']).columns
for col in cat_cols:
    df[col] = label_encoder.fit_transform(df[col])
```

Filling missing values

```
[35]: num_cols = df.select_dtypes(include=[np.number]).columns
for col in num_cols:
    mean_value = df[col].mean()
    df[col] = df[col].fillna(mean_value)

cat_cols = df.select_dtypes(include=['object']).columns
for col in cat_cols:
    mode_value = df[col].mode()[0]
    df[col] = df[col].fillna(mode_value)

print("\nMissing values after handling:")
print(df.isnull().sum())

clean_path = '/content/gdrive/MyDrive/Master-Project/ExtractedData/'
            ↪automobile_missing_data_handeled.csv'
df.to_csv(clean_path, index=False)
print(f"\n Cleaned dataset saved to: {clean_path}")
```

Missing values after handling:

symboling	0
normalized-losses	0
make	0

```

fuel-type          0
aspiration         0
num-of-doors       0
body-style         0
drive-wheels       0
engine-location    0
wheel-base        0
length            0
width             0
height            0
curb-weight        0
engine-type        0
num-of-cylinders   0
engine-size        0
fuel-system        0
bore              0
stroke            0
compression-ratio  0
horsepower         0
peak-rpm          0
city-mpg           0
highway-mpg        0
price             0
dtype: int64

```

Cleaned dataset saved to: /content/gdrive/MyDrive/Master-Project/ExtractedData/automobile_missing_data_handeled.csv

Handle Outliers

```

[52]: df = pd.read_csv('/content/gdrive/MyDrive/Master-Project/ExtractedData/
↳automobile_missing_data_handeled.csv')
# Select only numeric columns
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
print("\nNumeric columns for outlier detection:", num_cols)

# --- 1. IQR Method ---
def detect_outliers_iqr(data, col):
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    return data[(data[col] < lower) | (data[col] > upper)]

# Detect and count outliers for each numeric column
print("\n--- IQR Method ---")
for col in num_cols:

```

```

outliers = detect_outliers_iqr(df, col)
print(f"{col}: {len(outliers)} outliers")

```

Numeric columns for outlier detection: ['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']

```

--- IQR Method ---
symboling: 0 outliers
normalized-losses: 0 outliers
make: 0 outliers
fuel-type: 0 outliers
aspiration: 0 outliers
num-of-doors: 0 outliers
body-style: 0 outliers
drive-wheels: 0 outliers
engine-location: 0 outliers
wheel-base: 0 outliers
length: 0 outliers
width: 0 outliers
height: 0 outliers
curb-weight: 0 outliers
engine-type: 0 outliers
num-of-cylinders: 0 outliers
engine-size: 0 outliers
fuel-system: 0 outliers
bore: 0 outliers
stroke: 0 outliers
compression-ratio: 0 outliers
horsepower: 0 outliers
peak-rpm: 0 outliers
city-mpg: 0 outliers
highway-mpg: 0 outliers
price: 0 outliers

```

```

[53]: # --- 2. Z-score Method ---
def detect_outliers_zscore(data, col):
    mean = data[col].mean()
    std = data[col].std()
    z_score = (data[col] - mean) / std
    return data[(z_score > 3) | (z_score < -3)]

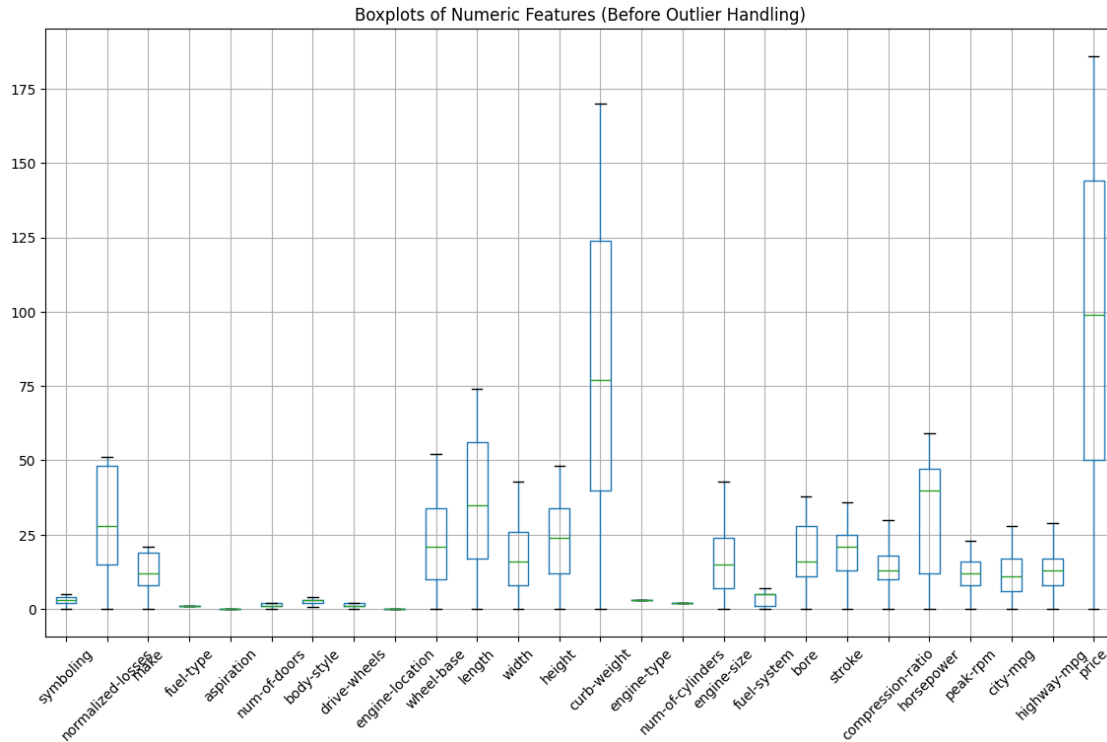
print("\n--- Z-score Outliers ---")

```

```
for col in num_cols:
    outliers = detect_outliers_zscore(df, col)
    print(f"{col}: {len(outliers)} outliers")
```

```
--- Z-score Outliers ---
symboling: 0 outliers
normalized-losses: 0 outliers
make: 0 outliers
fuel-type: 0 outliers
aspiration: 0 outliers
num-of-doors: 0 outliers
body-style: 0 outliers
drive-wheels: 0 outliers
engine-location: 0 outliers
wheel-base: 0 outliers
length: 0 outliers
width: 0 outliers
height: 0 outliers
curb-weight: 0 outliers
engine-type: 0 outliers
num-of-cylinders: 0 outliers
engine-size: 0 outliers
fuel-system: 0 outliers
bore: 0 outliers
stroke: 0 outliers
compression-ratio: 0 outliers
horsepower: 0 outliers
peak-rpm: 0 outliers
city-mpg: 0 outliers
highway-mpg: 0 outliers
price: 0 outliers
```

```
[54]: # --- 3. Visualize with Boxplots ---
plt.figure(figsize=(14, 8))
df[num_cols].boxplot()
plt.title('Boxplots of Numeric Features (Before Outlier Handling)')
plt.xticks(rotation=45)
plt.show()
```



```
[56]: # --- 4. Handle outliers by capping (winsorization) ---
for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df[col] = np.where(df[col] < lower, lower, np.where(df[col] > upper, upper,
    ↪df[col]))

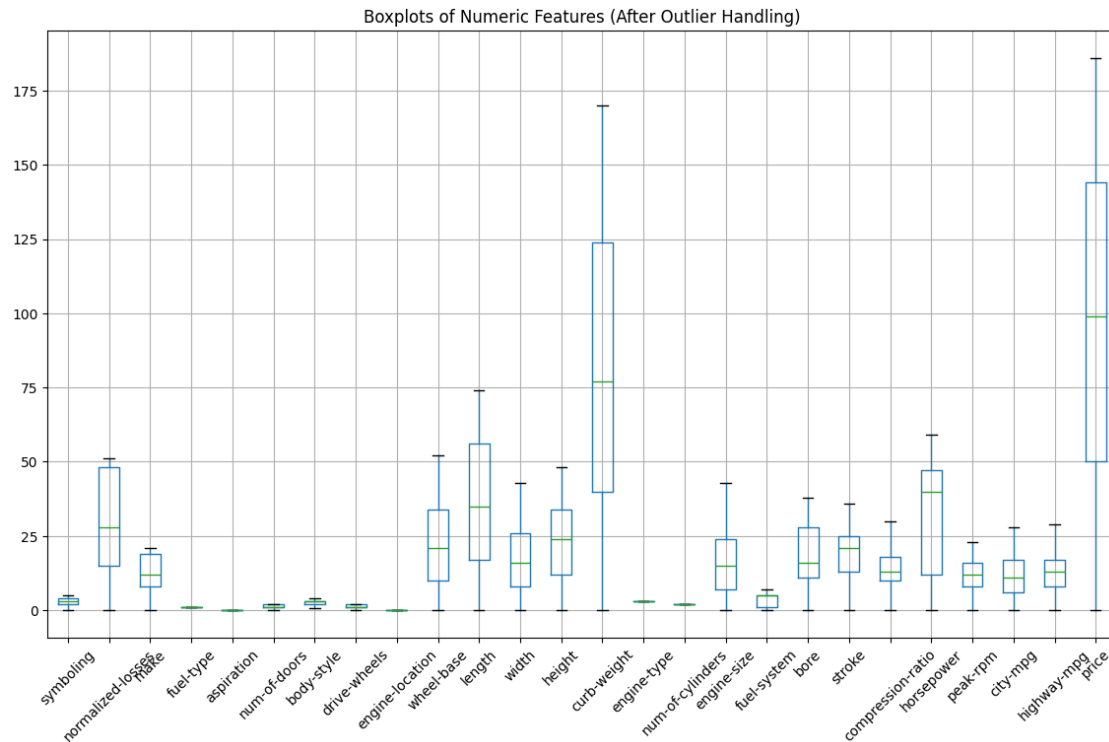
print("\nOutliers have been capped using IQR bounds.")

# --- 5. Verify after handling ---
plt.figure(figsize=(14, 8))
df[num_cols].boxplot()
plt.title('Boxplots of Numeric Features (After Outlier Handling)')
plt.xticks(rotation=45)
plt.show()

# --- Save cleaned dataset for visualization step ---
clean_outlier_path = '/content/gdrive/MyDrive/Master-Project/ExtractedData/
    ↪automobile_outliers_cleaned.csv'
df.to_csv(clean_outlier_path, index=False)
```

```
print(f"\n Outlier-handled dataset saved to: {clean_outlier_path}")
```

Outliers have been capped using IQR bounds.



Outlier-handled dataset saved to: /content/gdrive/MyDrive/Master-Project/ExtractedData/automobile_outliers_cleaned.csv

DATA VISUALIZATION

```
[61]: # Load the outlier-handled dataset
df = pd.read_csv('/content/gdrive/MyDrive/Master-Project/ExtractedData/
↳automobile_outliers_cleaned.csv')
print("Data shape:", df.shape)
df.head()

# Separate numeric and categorical columns
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
cat_cols = df.select_dtypes(include=['object']).columns.tolist()
print("\nNumeric Columns:", num_cols)
print("\nCategorical Columns:", cat_cols)
```

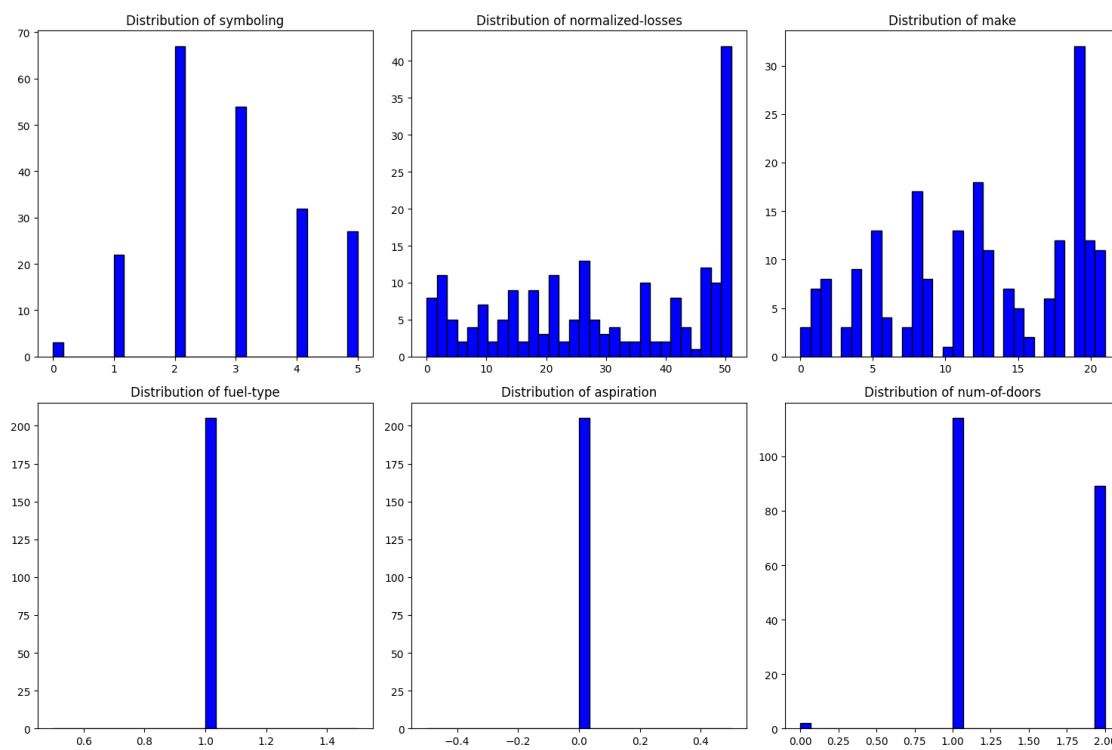
Data shape: (205, 26)

Numeric Columns: ['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']

Categorical Columns: []

[62]: # 1. Histogram

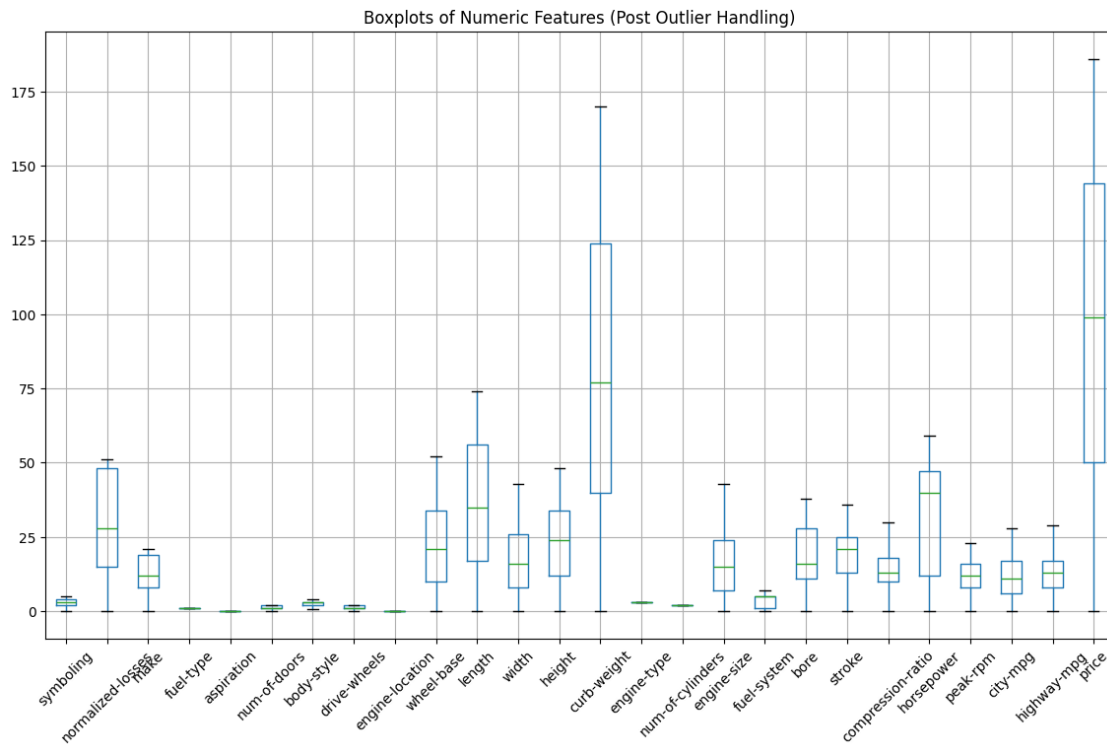
```
plt.figure(figsize=(15, 10))
for i, col in enumerate(num_cols[:6]): # first 6 columns
    plt.subplot(2, 3, i + 1)
    plt.hist(df[col], bins=30, color='blue', edgecolor='black')
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
```



[59]: # 2. Boxplots

```
plt.figure(figsize=(14, 8))
df[num_cols].boxplot()
plt.title('Boxplots of Numeric Features (Post Outlier Handling)')
```

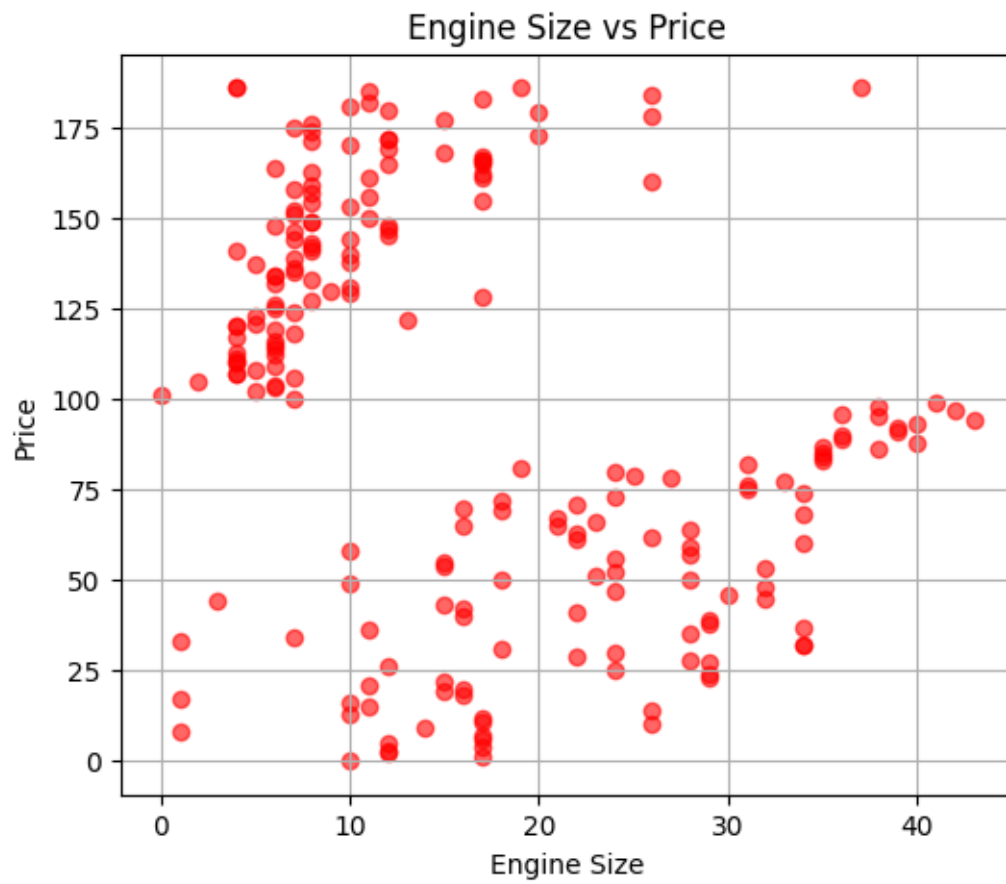
```
plt.xticks(rotation=45)
plt.show()
```

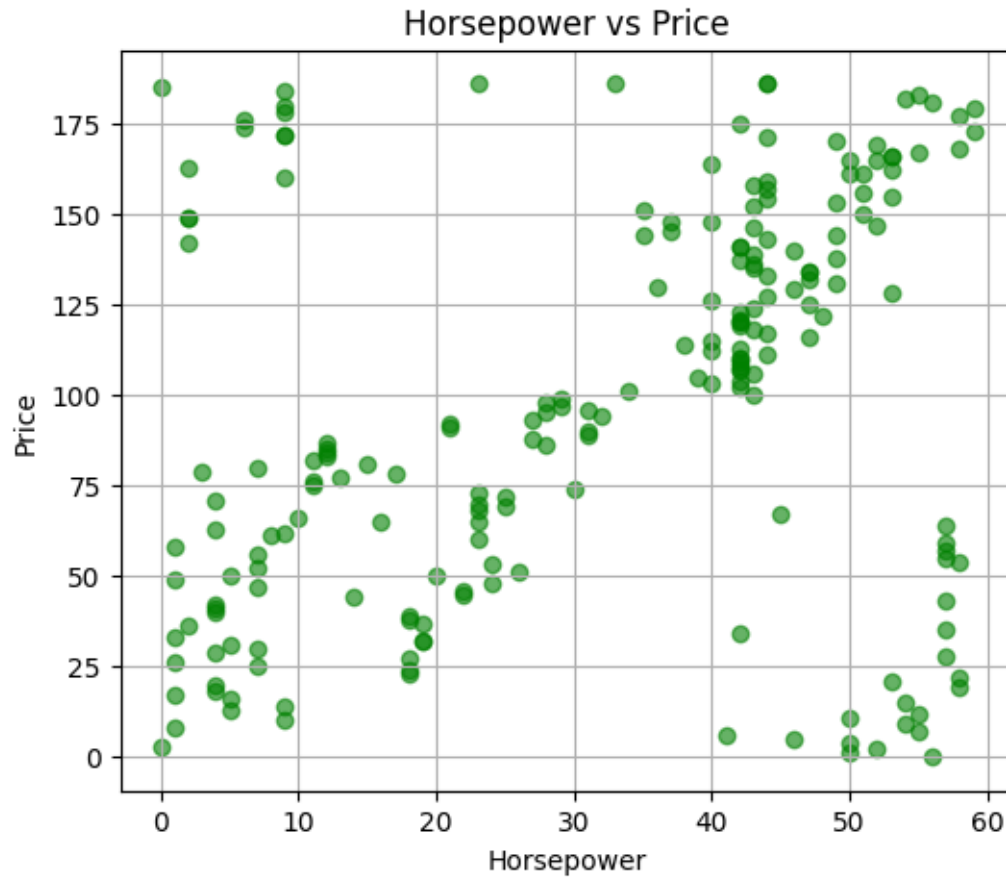


[63]: # 3. Scatterplots

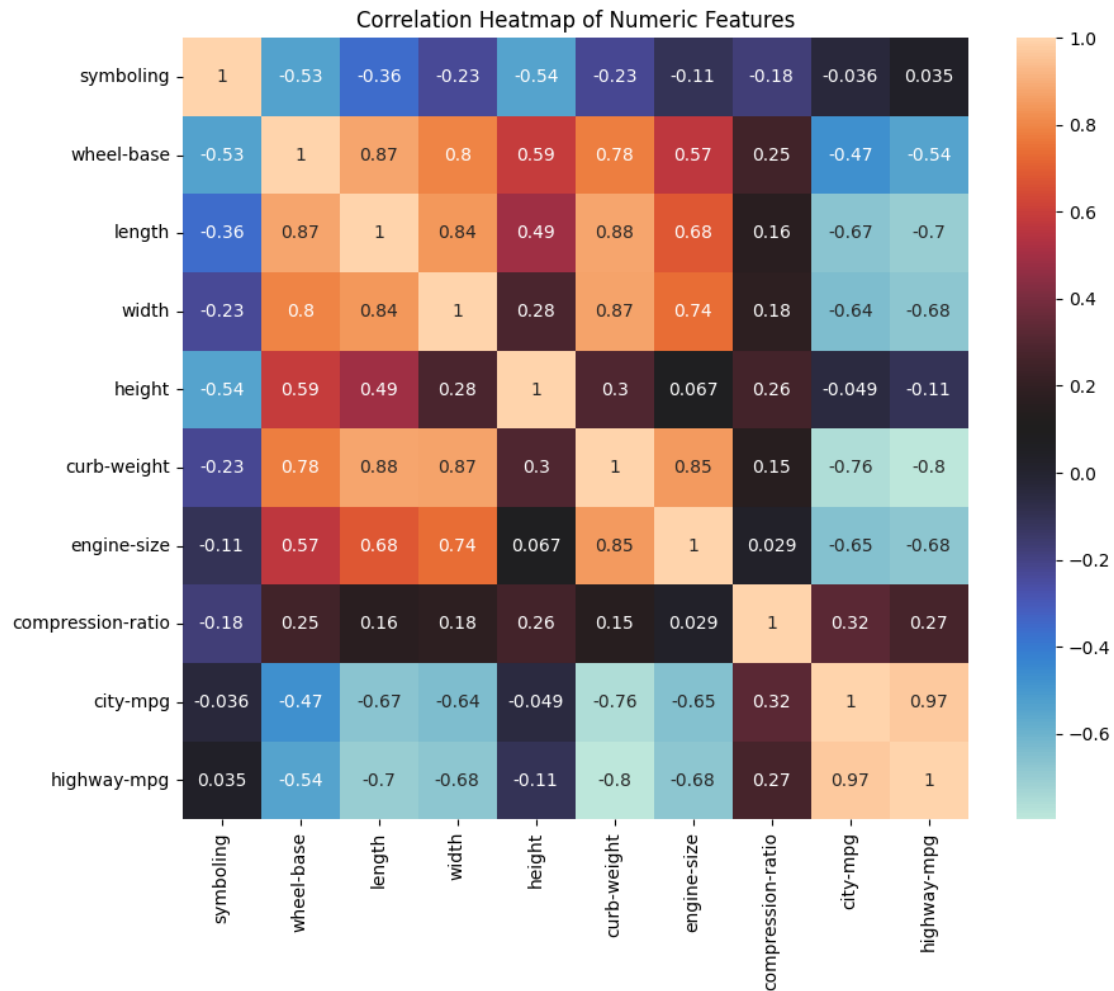
```
if 'engine-size' in df.columns and 'price' in df.columns:
    plt.figure(figsize=(6, 5))
    plt.scatter(df['engine-size'], df['price'], color='red', alpha=0.6)
    plt.title('Engine Size vs Price')
    plt.xlabel('Engine Size')
    plt.ylabel('Price')
    plt.grid(True)
    plt.show()

if 'horsepower' in df.columns and 'price' in df.columns:
    plt.figure(figsize=(6, 5))
    plt.scatter(df['horsepower'], df['price'], color='green', alpha=0.6)
    plt.title('Horsepower vs Price')
    plt.xlabel('Horsepower')
    plt.ylabel('Price')
    plt.grid(True)
    plt.show()
```





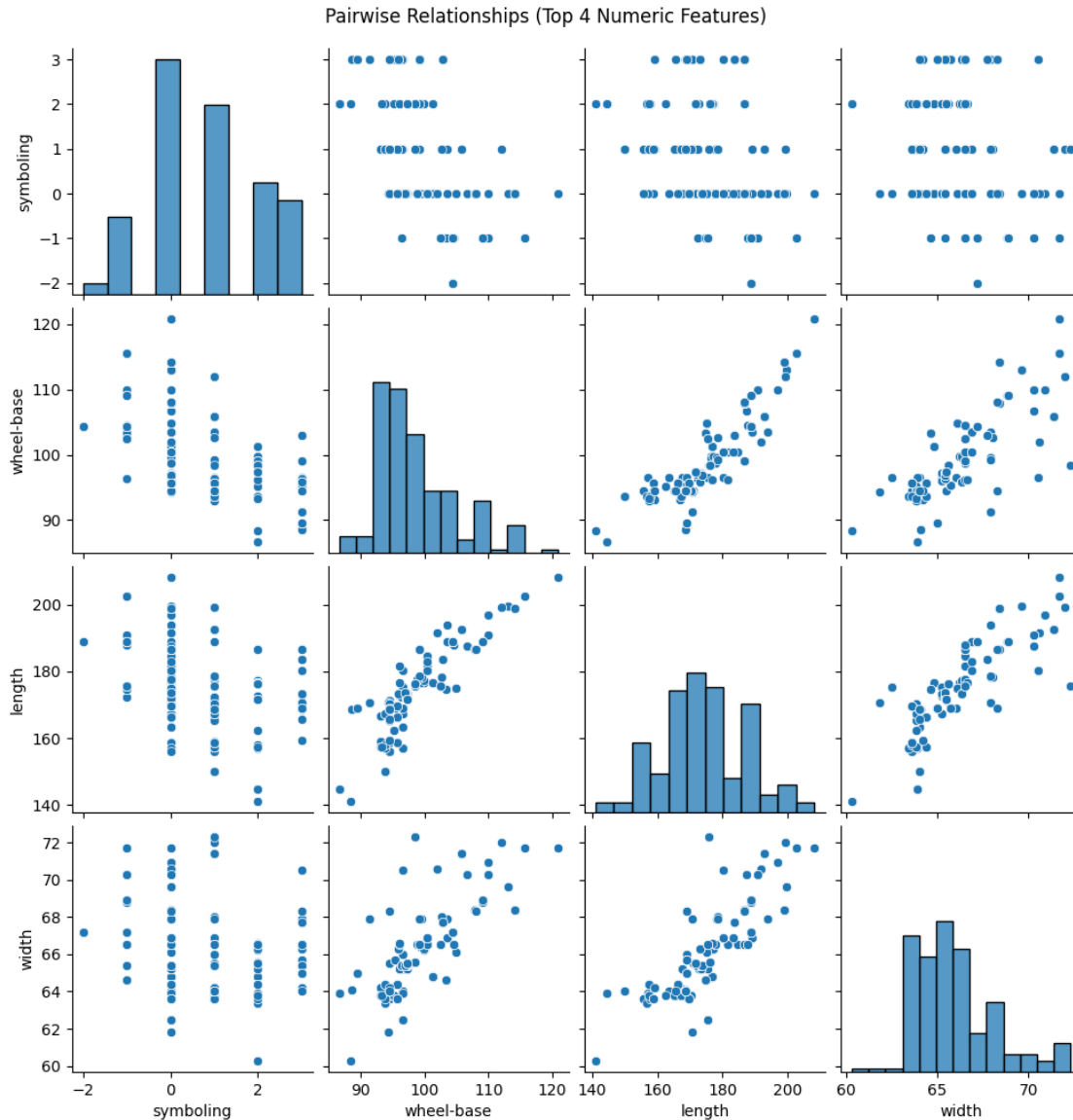
```
[64]: # 4. Correlation Heatmap
df = pd.read_csv('/content/gdrive/MyDrive/Master-Project/ExtractedData/
↳Automobile_data.csv')
num_cols = df.select_dtypes([np.number]).columns.tolist()
plt.figure(figsize=(10,8))
corr_matrix = df[num_cols].corr()
sns.heatmap(corr_matrix, annot=True, cmap= 'icefire')
plt.title('Correlation Heatmap of Numeric Features')
plt.show()
```



```
[69]: # 5. Pairplot (optional advanced)
print("\nPairplot")
plt.figure(figsize=(15, 9))
sns.pairplot(df[num_cols[:4]])
plt.suptitle('Pairwise Relationships (Top 4 Numeric Features)', y=1.02)
plt.show()
```

Pairplot

<Figure size 1500x900 with 0 Axes>



Install necessary packages to save Colab notebook to PDF

```
[ ]: !apt-get install texlive-xetex texlive-fonts-recommended texlive-plain-generic
    ↵-y
```

Change the Path to Notebook location

```
[2]: import os
    os.chdir('/content/gdrive/MyDrive/Colab Notebooks/')
    
```

Save to PDF

```
[ ]: !jupyter nbconvert --to pdf HeshamMasterProjects.ipynb
```

```
[NbConvertApp] Converting notebook HeshamMasterProjects.ipynb to pdf
[NbConvertApp] Support files will be in HeshamMasterProjects_files/
[NbConvertApp] Making directory ./HeshamMasterProjects_files
[NbConvertApp] Writing 59117 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
```