



USED CAR PRICE PREDICTION USING PYTHON

Data Science Project

This is a personal project aimed at demonstrating proficiency in Python and machine learning through a comprehensive analysis of used car data. The project involves developing a predictive model to estimate the prices of used cars based on various features such as make, model, year, and mileage. The process includes data collection, cleaning, exploration, and the application of machine learning algorithms to predict car prices. The goal is to showcase the ability to handle data, build predictive models, and present the findings effectively using Python

Jefferson Alexander Romero Márquez
romerojefferson1999@gmail.com

Table of Contents

Introduction.....	1
1. Project Overview	1
2. Objectives	1
3. Scope.....	1
Methodology	2
1. Data Collection.....	2
1.1 Data Acquisition	2
1.2 Basic Insight of Dataset.....	4
2. Data Wrangling.....	8
2.1 Identify and Handle Missing Values.....	8
2.2 Data Standardization and Normalization.....	13
2.3 Data Binning and Encoding	14
3. Data Analysis	19
3.1 Continuous Numerical Variables	19
3.2 Categorical Variables	23
3.3 Descriptive Statistical Analysis	26
3.4 Correlation and Causation	31
3.5 Analysis of Variance (ANOVA)	33
4. Model Development	36
4.1 Linear Regression and Multiple Linear Regression.....	36
4.2 Model Evaluation Using Visualization	38
4.3 Polynomial Regression	42
4.4 Model Measurement	44
4.5 Decision Making	45
5. Model Evaluation.....	47
5.1 Training and Testing	47
5.2 Overfitting, Underfitting and Model Selection	49
5.3 Ridge Regression and Grid Search	56
Conclusions	58

Introduction

1. Project Overview

This project is a personal initiative designed to showcase my skills in data science using Python. The primary focus is on predicting the prices of used cars based on various features such as make, model, year, and mileage. By developing and training a machine learning model, this project aims to demonstrate the ability to manage data, perform predictive analysis, and present findings effectively using Python.

2. Objectives

The main objectives of this project are:

- To collect and prepare a comprehensive dataset of used car listings for analysis.
- To explore and clean the data, handling missing values and outliers.
- To develop a predictive model using machine learning techniques in Python that accurately estimates car prices based on relevant features.
- To evaluate the model's performance using appropriate metrics and refine it for better accuracy.
- To present the findings in a clear and actionable manner, demonstrating the practical application of the model.

3. Scope

The scope of this project includes:

- Data collection from available sources of used car listings.
- Data cleaning and preparation, including handling missing values, outliers, and feature engineering.
- Model development and training using Python's machine learning libraries, with a focus on regression techniques.
- Model evaluation and refinement, using metrics such as RMSE and MAE to assess performance.
- Documentation of the entire process, including data preparation, model development, evaluation, and key findings.

Methodology

1. Data Collection

1.1 Data Acquisition

In this section, we outline the methodology used for data acquisition and preprocessing. Only the outputs of the Python code executed in Jupyter Notebook are presented here. The full code is available in the project repository.

- **Dataset Download:** The dataset was initially obtained from the UCI Machine Learning Repository. The URL for the dataset is [this link](#). A script was used to download the dataset as a CSV file named `auto.csv` using the `requests` library in Python:

```
1 3,?,alfa-romero,gas,std,two,convertible,rwd,front,88.60,168.80,64.10,48.80,2548,dohc,four,130,mpfi,3.47,2.68,9.00,111,5000,21,27,13495
2 3,?,alfa-romero,gas,std,two,convertible,rwd,front,88.60,168.80,64.10,48.80,2548,dohc,four,130,mpfi,3.47,2.68,9.00,111,5000,21,27,16500
3 1,?,alfa-romero,gas,std,two,hatchback,rwd,front,94.50,171.20,65.50,52.40,2823,ohc,six,152,mpfi,2.68,3.47,9.00,154,5000,19,26,16500
4 2,164,audi,gas,std,four,edan,fwd,front,99.80,176.60,66.20,54.30,2337,ohc,four,109,mpfi,3.19,3.40,10.00,102,5500,24,30,13950
5 2,164,audi,gas,std,four,edan,4wd,front,99.40,176.60,66.40,54.30,2824,ohc,five,136,mpfi,3.19,3.40,8.00,115,5500,18,22,17450
6 2,?,audi,gas,std,two,edan,fwd,front,99.80,177.30,66.30,53.10,2507,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,15250
7 1,158,audi,gas,std,four,edan,fwd,front,105.80,192.70,71.40,55.70,2844,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,17710
8 1,?,audi,gas,std,four,wagon,fwd,front,105.80,192.70,71.40,55.70,2954,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,18920
9 1,158,audi,gas,turbo,four,edan,fwd,front,105.80,192.70,71.40,55.90,3086,ohc,five,131,mpfi,3.13,3.40,8.30,140,5500,17,20,23875
10 0,?,audi,gas,turbo,two,hatchback,4wd,front,99.50,178.20,67.90,52.00,3053,ohc,five,131,mpfi,3.13,3.40,7.00,160,5500,16,22,?
11 2,192,bmw,gas,std,two,edan,rwd,front,101.20,176.80,64.80,54.30,2395,ohc,four,108,mpfi,3.50,2.80,8.80,101,5800,23,29,16430
12 0,192,bmw,gas,std,four,edan,rwd,front,101.20,176.80,64.80,54.30,2395,ohc,four,108,mpfi,3.50,2.80,8.80,101,5800,23,29,16925
13 0,188,bmw,gas,std,two,edan,rwd,front,101.20,176.80,64.80,54.30,2710,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,21,28,20970
14 0,188,bmw,gas,std,four,edan,rwd,front,101.20,176.80,64.80,54.30,2765,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,21,28,21105
15 1,?,bmw,gas,std,four,edan,rwd,front,103.50,189.00,66.90,55.70,3055,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,20,25,24565
```

Figure 1.1: Downloaded CSV file “auto.csv”

- **Loading Data:** The downloaded CSV file was loaded into a pandas DataFrame. Since the dataset did not include a header row, the data was initially read without specifying headers:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	edan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	edan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

Figure 1.2: First few rows of the dataset

- **Initial Inspection:** To understand the structure of the dataset, the first few rows and the last few rows were displayed:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

5 rows × 26 columns

Figure 1.3: Last few rows of the dataset

- **Adding Descriptive Headers:** The dataset initially lacked headers. Descriptive headers were added based on the dataset documentation:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115

5 rows × 26 columns

Figure 1.4: Dataset with descriptive headers

- **Handling Missing Values:** Missing values represented by `?"` were replaced with `NaN` to facilitate further data cleaning:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115

5 rows × 26 columns

Figure 1.5: Dataset with Missing Values Replaced by NaN

```

1  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
2  3,,alfa-romero,gas,std,two,convertible,rwd,front,88.6,168.8,64.1,48.8,2548,dohc,four,130,mpfi,3.47,2.68,9.0,111,5000,21,27,13495
3  3,,alfa-romero,gas,std,two,convertible,rwd,front,88.6,168.8,64.1,48.8,2548,dohc,four,130,mpfi,3.47,2.68,9.0,111,5000,21,27,16500
4  1,,alfa-romero,gas,std,two,hatchback,rwd,front,94.5,171.2,65.5,52.4,2823,ohcv,six,152,mpfi,2.68,3.47,9.0,154,5000,19,26,16500
5  2,164,audi,gas,std,four,sedan,fwd,front,99.8,176.6,66.2,54.3,2337,ohc,four,109,mpfi,3.19,3.40,10.0,102,5500,24,30,13950
6  2,164,audi,gas,std,four,sedan,4wd,front,99.4,176.6,66.4,54.3,2824,ohc,five,136,mpfi,3.19,3.40,8.0,115,5500,18,22,17450
7  2,,audi,gas,std,two,sedan,fwd,front,99.8,177.3,66.3,53.1,2507,ohc,five,136,mpfi,3.19,3.40,8.5,110,5500,19,25,15250
8  1,158,audi,gas,std,four,sedan,fwd,front,105.8,192.7,71.4,55.7,2844,ohc,five,136,mpfi,3.19,3.40,8.5,110,5500,19,25,17710
9  1,,audi,gas,std,four,wagon,fwd,front,105.8,192.7,71.4,55.7,2954,ohc,five,136,mpfi,3.19,3.40,8.5,110,5500,19,25,18920
10 1,158,audi,gas,turbo,four,sedan,fwd,front,105.8,192.7,71.4,55.9,3086,ohc,five,131,mpfi,3.13,3.40,8.3,140,5500,17,20,23875
11 0,,audi,gas,turbo,two,hatchback,4wd,front,99.5,178.2,67.9,52.0,3053,ohc,five,131,mpfi,3.13,3.40,7.0,160,5500,16,22,
12 2,192,bmw,gas,std,two,sedan,rwd,front,101.2,176.8,64.8,54.3,2395,ohc,four,108,mpfi,3.50,2.80,8.8,101,5800,23,29,16430
13 0,192,bmw,gas,std,four,sedan,rwd,front,101.2,176.8,64.8,54.3,2395,ohc,four,108,mpfi,3.50,2.80,8.8,101,5800,23,29,16925
14 0,188,bmw,gas,std,two,sedan,rwd,front,101.2,176.8,64.8,54.3,2710,ohc,six,164,mpfi,3.31,3.19,9.0,121,4250,21,28,20970
15 0,188,bmw,gas,std,four,sedan,rwd,front,101.2,176.8,64.8,54.3,2765,ohc,six,164,mpfi,3.31,3.19,9.0,121,4250,21,28,21105

```

Figure 1.6: “automobile.csv” Saved to CSV File

By following these steps, the dataset was successfully acquired, cleaned, and prepared for analysis. This methodology ensures that the data is in a suitable format for further exploration and modeling.

1.2 Basic Insight of Dataset

In this section, we provide a basic overview of the dataset, including data types, summary statistics, and general information. The focus is on understanding the structure and characteristics of the data to guide further analysis.

- Data Types of Each Column: To gain insights into the types of data stored in each column, the data types of each column in the DataFrame were displayed:

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	

Figure 1.7: Data types of each column

This output shows the data type for each column, which helps in understanding how the data is formatted and guides preprocessing steps.

- **Summary Statistics for Numerical Columns:** Statistics for the numerical columns were generated to provide an overview of the central tendencies and dispersion of the data:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

Figure 1.8: Summary statistics for numerical columns

This includes metrics such as count, mean, standard deviation, minimum, maximum, and quartiles, which are essential for understanding the distribution and range of numerical values.

- **Summary Statistics for All Columns:** A comprehensive summary of all columns, including non-numerical (categorical) columns, was generated:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower
count	205.000000	164	205	205	205	203	205	205	205	205.000000	...	205.000000	205	201	201	205.000000	
unique	NaN	51	22	2	2	2	5	3	2	NaN	...	NaN	8	38	36	NaN	
top	NaN	161	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40	NaN	
freq	NaN	11	32	185	168	114	96	120	202	NaN	...	NaN	94	23	20	NaN	
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317	NaN	NaN	NaN	10.142537	
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693	NaN	NaN	NaN	3.972040	
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN	NaN	NaN	7.000000	
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000	NaN	NaN	NaN	8.600000	
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000	NaN	NaN	NaN	9.000000	
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	141.000000	NaN	NaN	NaN	9.400000	
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000	NaN	NaN	NaN	23.000000	

11 rows × 26 columns

Figure 1.9: Summary statistics for all columns

This output includes statistics for both numerical and categorical columns, providing insights into unique values, most frequent values, and their frequencies.

- **Concise Summary of the DataFrame:** A concise summary of the DataFrame was obtained, which includes information on data types, non-null counts, and memory usage:


```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
#   Column                Non-Null Count  Dtype
---  -
0   symboling              205 non-null    int64
1   normalized-losses      164 non-null    object
2   make                   205 non-null    object
3   fuel-type              205 non-null    object
4   aspiration              205 non-null    object
5   num-of-doors            203 non-null    object
6   body-style              205 non-null    object
7   drive-wheels            205 non-null    object
8   engine-location         205 non-null    object
9   wheel-base              205 non-null    float64
10  length                  205 non-null    float64
11  width                    205 non-null    float64
12  height                   205 non-null    float64
13  curb-weight             205 non-null    int64
14  engine-type             205 non-null    object
15  num-of-cylinders        205 non-null    object
16  engine-size             205 non-null    int64
17  fuel-system             205 non-null    object
18  bore                    201 non-null    object
19  stroke                  201 non-null    object
20  compression-ratio       205 non-null    float64
21  horsepower              203 non-null    object
22  peak-rpm                203 non-null    object
23  city-mpg                205 non-null    int64
24  highway-mpg             205 non-null    int64
25  price                   201 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB

```

Figure 1.10: Concise summary of the DataFrame

This summary helps in assessing the completeness of the dataset and identifying any columns with missing values.

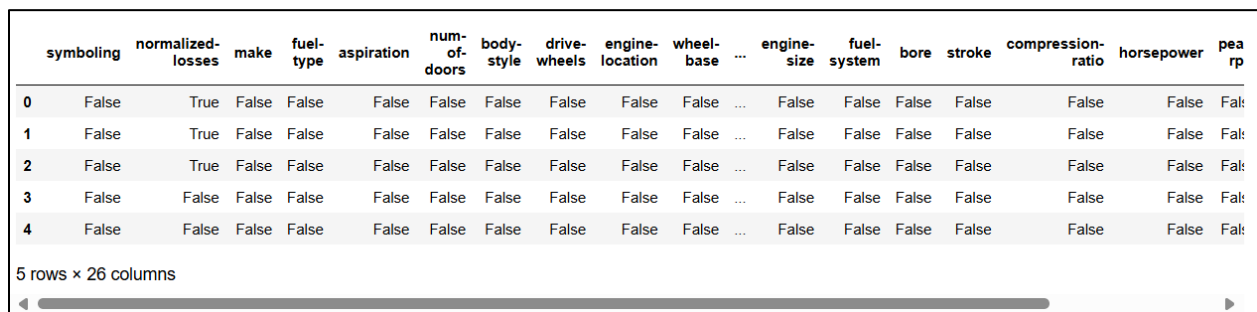
2. Data Wrangling

2.1 Identify and Handle Missing Values

- **Evaluating for Missing Data:** To evaluate and identify missing data in our dataset, we use specific functions that convert the data into boolean values. The two primary methods for detecting missing data are:

- “.isnull()”
- “.notnull()”

These methods return a boolean DataFrame where “True” indicates a missing value, and “False” indicates a non-missing value.



	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rp
0	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
1	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
2	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False

5 rows × 26 columns

Figure 2.1: Display of missing data in the first five rows of the dataset

In the output, a value of “True” signifies a missing value, while “False” indicates that the value is present.

- **Count Missing Values in Each Column:** To quantify the missing values in each column, we utilize a “for” loop in Python. This approach allows us to quickly determine the number of missing values per column. As noted earlier, “True” represents a missing value, and “False” indicates that the value is present. Within the loop, the “.value_counts()” method counts the occurrences of “True” and “False” values:

```

symboling
symboling
False      205
Name: count, dtype: int64

normalized-losses
normalized-losses
False      164
True        41
Name: count, dtype: int64

make
make
False      205
Name: count, dtype: int64

fuel-type
fuel-type
False      205

```

Figure 2.2: Count of missing and non-missing values in each column

Based on the output, we observe that the dataset contains 205 rows of data, and the following seven columns have missing values:

- "normalized-losses": 41 missing values
- "num-of-doors": 2 missing values
- "bore": 4 missing values
- "stroke": 4 missing values
- "horsepower": 2 missing values
- "peak-rpm": 2 missing values
- "price": 4 missing values

- **Handling Missing Value:** To ensure the accuracy and completeness of the dataset, several steps were undertaken to address missing values:

- **Replacing Missing Values with Calculated Averages:** To handle missing values in numerical columns, we first calculated the average of each column with missing data. This allows us to replace NaN values with meaningful statistical values. The following code was used to compute and replace missing values:

```
Average of normalized-losses: 122.0
Average of bore: 3.3297512437810943
Average of stroke: 3.255422885572139
Average horsepower: 104.25615763546799
Average peak rpm: 5125.369458128079
```

Figure 2.3: Average values calculated for columns with missing data

Then the calculated averages were used to replace missing values in the respective columns.

- **Handling Categorical Missing Values:** For the "num-of-doors" column, which contains categorical data, we identified the most frequent value to replace missing entries:

```
num-of-doors
four      114
two       89
Name: count, dtype: int64
four
```

Figure 2.4: Most frequent value in the "num-of-doors" column used to replace missing values

It was observed that "four" is the most common option in the "num-of-doors" column, so missing values were replaced with this value.

- **Final Data Cleaning:** Finally, any rows with missing values in the "price" column were removed. Given that the goal of the project is to predict the price of used cars, it is crucial to have complete data in this column. Replacing missing values with the mean is not suitable for this purpose. After removing these rows, the DataFrame index was reset to ensure continuity.

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115

5 rows × 26 columns

Figure 2.5: DataFrame head showing data after all missing values have been addressed

- **Data Summary and Integrity Check:** To ensure the integrity of the dataset after handling missing values, we performed a final check on the data. Initially, the dataset contained 205 entries. However, after removing 4 rows with missing values in the "price" column—an essential feature for the project, given that the goal is to predict car prices—we are left with 201 entries. The output of the following code confirms that all columns now have 201 non-null entries:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
#   Column                Non-Null Count  Dtype
---  -
0   symboling              201 non-null   int64
1   normalized-losses      201 non-null   object
2   make                   201 non-null   object
3   fuel-type              201 non-null   object
4   aspiration              201 non-null   object
5   num-of-doors           201 non-null   object
6   body-style             201 non-null   object
7   drive-wheels           201 non-null   object
8   engine-location         201 non-null   object
9   wheel-base             201 non-null   float64
10  length                 201 non-null   float64
11  width                  201 non-null   float64
12  height                 201 non-null   float64
13  curb-weight            201 non-null   int64
14  engine-type            201 non-null   object
15  num-of-cylinders       201 non-null   object
16  engine-size            201 non-null   int64
17  fuel-system            201 non-null   object
18  bore                   201 non-null   object
19  stroke                 201 non-null   object
20  compression-ratio      201 non-null   float64
21  horsepower             201 non-null   object
22  peak-rpm               201 non-null   object
23  city-mpg               201 non-null   int64
24  highway-mpg            201 non-null   int64
25  price                  201 non-null   object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.0+ KB
```

Figure 2.6: DataFrame summary after addressing missing values, showing 201 non-null entries for each column

This final step ensures that the dataset is complete and ready for further analysis, with all missing values properly addressed and the DataFrame's index reset.

- **Data Type Conversion:** In reviewing the dataset, it was evident that some columns were not in the correct data type, as shown in Figure 3.6. Numerical variables like “bore” and “stroke” which describe engine measurements, were mistakenly classified as “object” types instead of “float” or “int”. To ensure accurate analysis, these columns were converted to their appropriate data types.

symboling	int64
normalized-losses	int32
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	float64
stroke	float64
compression-ratio	float64
horsepower	object
peak-rpm	float64
city-mpg	int64
highway-mpg	int64
price	float64
dtype: object	

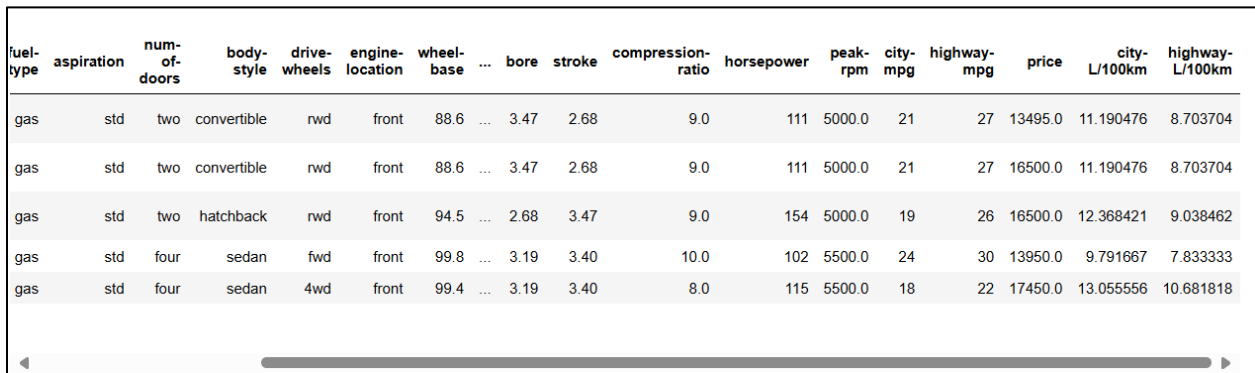
Figure 2.7: Data type summary after corrections, showing all columns with the appropriate types

2.2 Data Standardization and Normalization

- **Data Standardization:** Data collected from various sources may be presented in different formats. Standardization transforms this data into a common format, facilitating meaningful comparisons.

- **Standardization Process:** Standardization involves converting data into a unified format. For example, fuel consumption data is often reported in miles per gallon (mpg), but some regions use liters per 100 kilometers (L/100km).
- **Conversion and Renaming:** The dataset includes columns for "city-mpg" and "highway-mpg." To standardize these measurements, the following transformations were applied:
 - o The "city-mpg" column was converted to L/100km and saved as "city-L/100km."
 - o The "highway-mpg" column was converted to L/100km and saved as "highway-L/100km."

The original columns, "city-mpg" and "highway-mpg," were retained for reference, while the new columns reflect the standardized units.



fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	city-L/100km	highway-L/100km
gas	std	two	convertible	rwd	front	88.6	...	3.47	2.68	9.0	111	5000.0	21	27	13495.0	11.190476	8.703704
gas	std	two	convertible	rwd	front	88.6	...	3.47	2.68	9.0	111	5000.0	21	27	16500.0	11.190476	8.703704
gas	std	two	hatchback	rwd	front	94.5	...	2.68	3.47	9.0	154	5000.0	19	26	16500.0	12.368421	9.038462
gas	std	four	sedan	fwd	front	99.8	...	3.19	3.40	10.0	102	5500.0	24	30	13950.0	9.791667	7.833333
gas	std	four	sedan	4wd	front	99.4	...	3.19	3.40	8.0	115	5500.0	18	22	17450.0	13.055556	10.681818

Figure 2.8: DataFrame showing columns after standardization

- **Data Normalization:** Normalization transforms variables to a common scale, facilitating comparison and analysis. This process is crucial when variables have different ranges or units. Normalization techniques include scaling variables to a range of 0 to 1, adjusting so the mean is 0 and the variance is 1, or other similar methods. Normalization ensures that variables contribute equally to the analysis, especially in machine learning models where

differing scales can bias the results. For instance, when variables like "length," "width," and "height" have different ranges, normalization standardizes these values, allowing for more accurate comparisons. The goal is to normalize the columns "length," "width," and "height" so that their values range from 0 to 1. This is achieved by dividing each value by the maximum value in its column. Each value in the columns "length," "width," and "height" is divided by the maximum value of its respective column.

	length	width	height
0	0.811148	0.890278	0.816054
1	0.811148	0.890278	0.816054
2	0.822681	0.909722	0.876254
3	0.848630	0.919444	0.908027
4	0.848630	0.922222	0.908027

Figure 2.9: DataFrame showing columns after normalization

2.3 Data Binning and Encoding

- **Data Binning:** In the analysis, "horsepower" is a continuous variable ranging from 48 to 288 with 59 unique values. To simplify the analysis and focus on the general trends, we can categorize the horsepower into three distinct bins: low, medium, and high. This approach helps to analyze the impact of horsepower on car prices more effectively.
- **Histogram of Horsepower:** The histogram below shows the distribution of the "horsepower" values. This visualization helps to understand the spread of the horsepower values across the dataset.

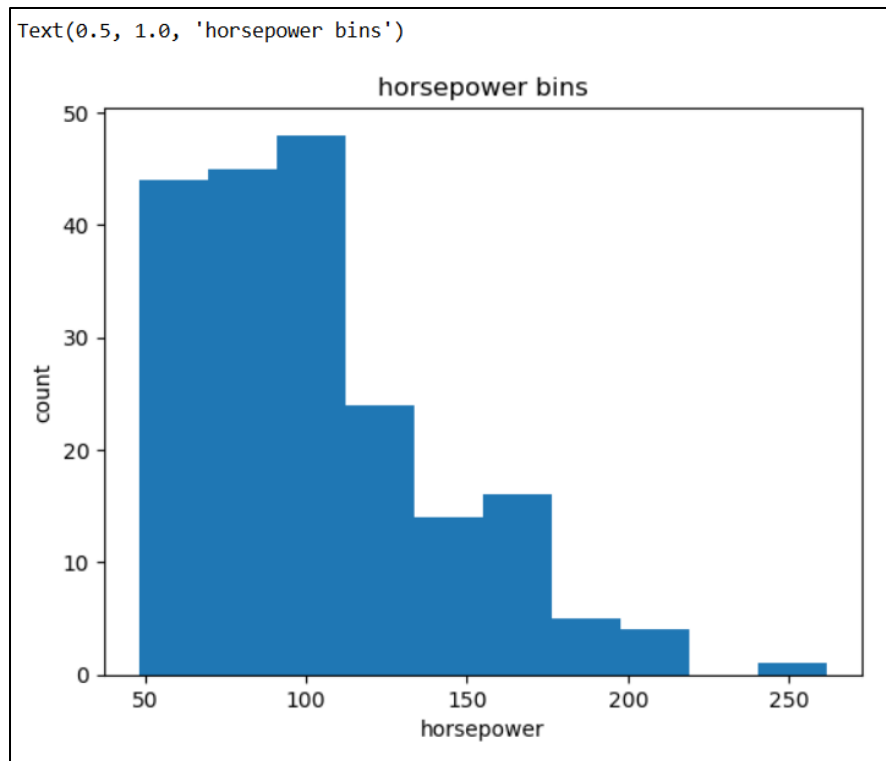


Figure 2.10: Histogram of horsepower values

- **Bins Calculation and Assignment:** To segment the horsepower values into three bins, we use NumPy's "linspace" function to create equally spaced bin edges. The bins are defined to cover the entire range of horsepower values from minimum to maximum.

```
[ 48.          119.33333333 190.66666667 262.         ]
horsepower horsepower-binned
0          111                Low
1          111                Low
2          154                Medium
3          102                Low
4          115                Low

horsepower-binned
Low          153
Medium       43
High         5
Name: count, dtype: int64
```

Figure 2.11: Bin edges for horsepower values

- **Bar Chart of Horsepower Bins:** The bar chart below illustrates the count of cars in each horsepower bin. This visualization provides a clear view of how the horsepower is distributed among the defined categories.

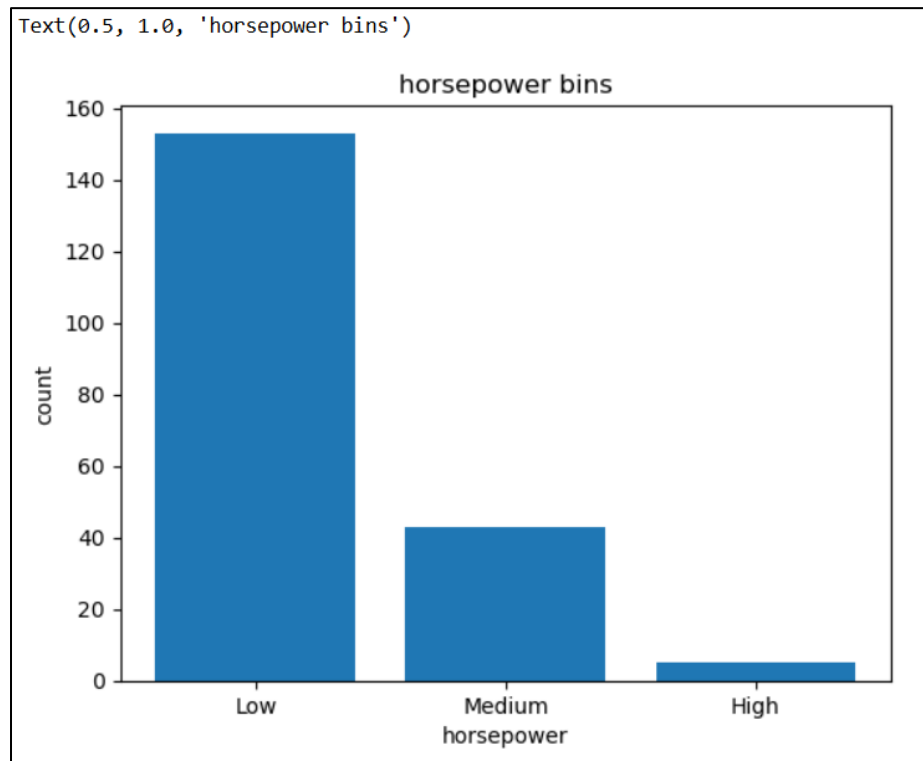


Figure 2.12: Bar chart of horsepower bins

- **Histogram of Horsepower Distribution:** To simplify the analysis of the 'horsepower' variable, which ranges from 48 to 288 and has 59 unique values, the dataset was divided into three distinct bins representing low, medium, and high horsepower ranges.

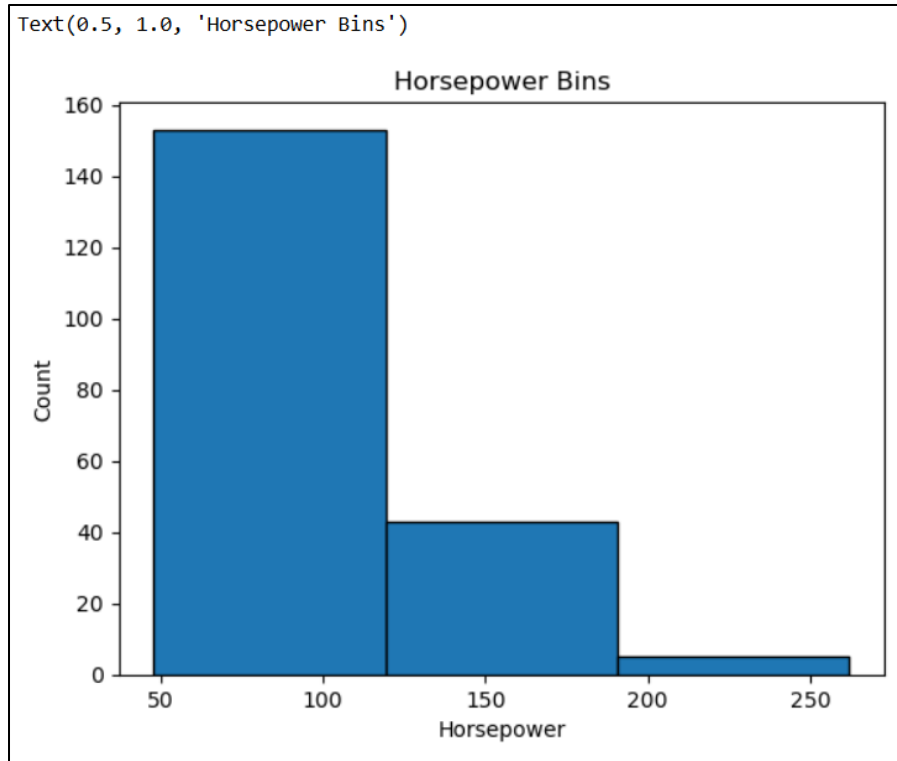


Figure 2.13: Histogram of Horsepower Distribution

- **Data Encoding:** The dataset includes categorical variables that need to be converted into a numerical format for regression analysis. Specifically, the columns "fuel-type" and "aspiration" contain categorical data:

- "fuel-type" has two unique values: "gas" and "diesel".
- "aspiration" has two unique values: "std" and "turbo".

Regression models require numerical inputs, so we will need to convert these categorical columns into numerical format. To achieve this, we will use pandas' "get_dummies" method, which creates indicator variables for each category. This method will transform the categorical variables into a format suitable for regression analysis.

	fuel-type-diesel	fuel-type-gas
0	False	True
1	False	True
2	False	True
3	False	True
4	False	True

	aspiration-std	aspiration-turbo
0	True	False
1	True	False
2	True	False
3	True	False
4	True	False

Figure 2.14: Dummy Variables for "fuel-type" and "aspiration"

- **Integration of Dummy Variables:** To incorporate the newly created dummy variables into the dataset, both sets of dummy variables for the "fuel-type" and "aspiration" columns are merged with the original DataFrame. This step transforms the categorical data into a numerical format suitable for regression analysis.

aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	city-mpg	highway-mpg	price	city-L/100km	highway-L/100km	horsepower-binned	fuel-type-diesel	fuel-type-gas	aspiration-std	aspiration-turbo
std	two	convertible	rwd	front	88.6	21	27	13495.0	11.190476	8.703704	Low	False	True	True	False
std	two	convertible	rwd	front	88.6	21	27	16500.0	11.190476	8.703704	Low	False	True	True	False
std	two	hatchback	rwd	front	94.5	19	26	16500.0	12.368421	9.038462	Medium	False	True	True	False
std	four	sedan	fwd	front	99.8	24	30	13950.0	9.791667	7.833333	Low	False	True	True	False
std	four	sedan	4wd	front	99.4	18	22	17450.0	13.055556	10.681818	Low	False	True	True	False

Figure 2.15: Updated Dataset with Integrated Dummy Variables

```

1 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,city-L/100km,highway-L/100km,horsepower- binned,fuel-type- diesel,fuel-type- gas,aspiration- std,aspiration- turbo
2 3,122,alfa-
romero,gas,std,two,convertible,rwd,front,88.6,0.8111484863046613,0.8902777777777777,0.8160535117056856,2548,dohc,four,130,mpfi,3.47,2.68,9.0
,111,5000.0,21,27,13495.0,11.19047619047619,8.703703703703704,Low,False,True,True,False
3 3,122,alfa-
romero,gas,std,two,convertible,rwd,front,88.6,0.8111484863046613,0.8902777777777777,0.8160535117056856,2548,dohc,four,130,mpfi,3.47,2.68,9.0
,111,5000.0,21,27,16500.0,11.19047619047619,8.703703703703704,Low,False,True,True,False
4 1,122,alfa-
romero,gas,std,two,hatchback,rwd,front,94.5,0.8226814031715521,0.9097222222222222,0.8762541806020067,2823,ohcv,six,152,mpfi,2.68,3.47,9.0,15
4,5000.0,19,26,16500.0,12.368421052631579,9.038461538461538,Medium,False,True,True,False
5 2,164,audi,gas,std,four, sedan, fwd, front, 99.8, 0.8486304661220567, 0.9194444444444445, 0.9080267558528428, 2337, ohc, four, 109, mpfi, 3.19, 3.4, 10.0, 1
02, 5500.0, 24, 30, 13950.0, 9.791666666666666, 7.833333333333333, Low, False, True, True, False
6 2,164,audi,gas,std,four, sedan, fwd, front, 99.4, 0.8486304661220567, 0.9222222222222223, 0.9080267558528428, 2824, ohc, five, 136, mpfi, 3.19, 3.4, 8.0, 11
5, 5500.0, 18, 22, 17450.0, 13.055555555555555, 10.681818181818182, Low, False, True, True, False
7 2,122,audi,gas,std,two, sedan, fwd, front, 99.8, 0.8519942335415667, 0.9208333333333333, 0.8879598662207359, 2507, ohc, five, 136, mpfi, 3.19, 3.4, 8.5, 110
, 5500.0, 19, 25, 15250.0, 12.368421052631579, 9.4, Low, False, True, True, False
8 1,158,audi,gas,std,four, sedan, fwd, front, 105.8, 0.9259971167707832, 0.9916666666666667, 0.931438127090301, 2844, ohc, five, 136, mpfi, 3.19, 3.4, 8.5, 11
0, 5500.0, 19, 25, 17710.0, 12.368421052631579, 9.4, Low, False, True, True, False
9 1,122,audi,gas,std,four, wagon, fwd, front, 105.8, 0.9259971167707832, 0.9916666666666667, 0.931438127090301, 2954, ohc, five, 136, mpfi, 3.19, 3.4, 8.5, 11
0, 5500.0, 19, 25, 18920.0, 12.368421052631579, 9.4, Low, False, True, True, False
10 1,158,audi,gas,turbo,four, sedan, fwd, front, 105.8, 0.9259971167707832, 0.9916666666666667, 0.9347826086956522, 3086, ohc, five, 131, mpfi, 3.13, 3.4, 8.3
, 140, 5500.0, 17, 20, 23875.0, 13.823529411764707, 11.75, Medium, False, True, False, True
11 2,192,bmw,gas,std,two, sedan, rwd, front, 101.2, 0.849591542527631, 0.8999999999999999, 0.9080267558528428, 2395, ohc, four, 108, mpfi, 3.5, 2.8, 8.8, 101, 5
800.0, 23, 29, 16430.0, 10.217391304347826, 8.10344827586207, Low, False, True, True, False
12 0,192,bmw,gas,std,four, sedan, rwd, front, 101.2, 0.849591542527631, 0.8999999999999999, 0.9080267558528428, 2395, ohc, four, 108, mpfi, 3.5, 2.8, 8.8, 101, 5
800.0, 23, 29, 16925.0, 10.217391304347826, 8.10344827586207, Low, False, True, True, False
13 0,188,bmw,gas,std,two, sedan, rwd, front, 101.2, 0.849591542527631, 0.8999999999999999, 0.9080267558528428, 2710, ohc, six, 164, mpfi, 3.31, 3.19, 9.0, 121,
4250.0, 21, 28, 20970.0, 11.19047619047619, 8.392857142857142, Medium, False, True, True, False
14 0,188,bmw,gas,std,four, sedan, rwd, front, 101.2, 0.849591542527631, 0.8999999999999999, 0.9080267558528428, 2765, ohc, six, 164, mpfi, 3.31, 3.19, 9.0, 121
, 4250.0, 21, 28, 21105.0, 11.19047619047619, 8.392857142857142, Medium, False, True, True, False
15 1,122,bmw,gas,std,four, sedan, rwd, front, 103.5, 0.9082172032676598, 0.9291666666666667, 0.931438127090301, 3055, ohc, six, 164, mpfi, 3.31, 3.19, 9.0, 121
, 4250.0, 20, 25, 24565.0, 11.75, 9.4, Medium, False, True, True, False

```

Figure 2.16: “Clean_df.csv” Saved to CSV File

3. Data Analysis

Understanding the factors that significantly influence car prices is crucial in predictive modeling. To uncover these relationships, it's essential to conduct a thorough analysis of the data, starting with a clear identification of the variables and their respective types.

3.1 Continuous Numerical Variables

Continuous numerical variables can take any value within a range and are typically represented as "int64" or "float64". They are crucial for analyzing relationships within the dataset. Scatterplots with fitted lines are ideal for visualizing these variables, as they help reveal linear relationships with the target variable, such as price. For example, using the “regplot” function allows us to plot engine size against price and observe how changes in engine size impact car price. This method provides valuable insights into the correlations and predictive power of continuous variables.

- Positive Linear Relationship:

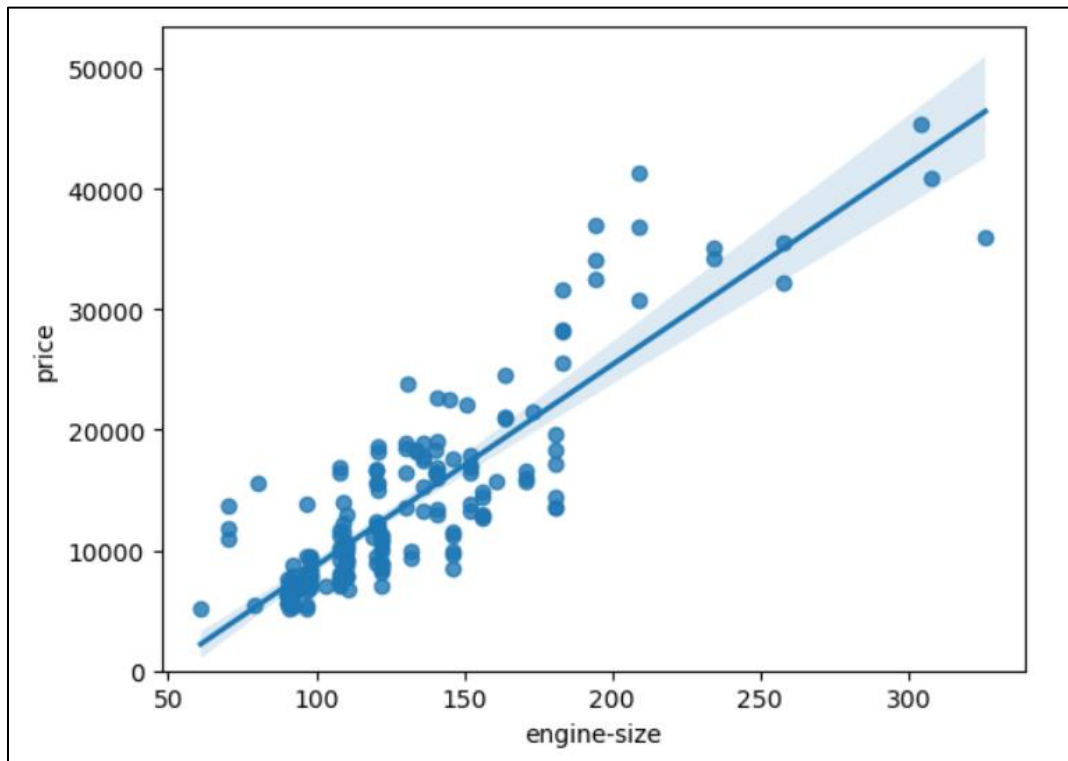


Figure 3.1: Regression plot showing the positive linear relationship between "engine-size" and "price"

	engine-size	price
engine-size	1.000000	0.872335
price	0.872335	1.000000

Figure 3.2: Correlation matrix output for "engine-size" and "price"

The plot above shows that as engine size increases, so does the price, indicating a positive correlation between these variables. The regression line approximates a perfect diagonal, suggesting that engine size is a strong predictor of price. The correlation coefficient is approximately 0.87, reflecting a strong positive relationship.

- Negative Linear Relationship:

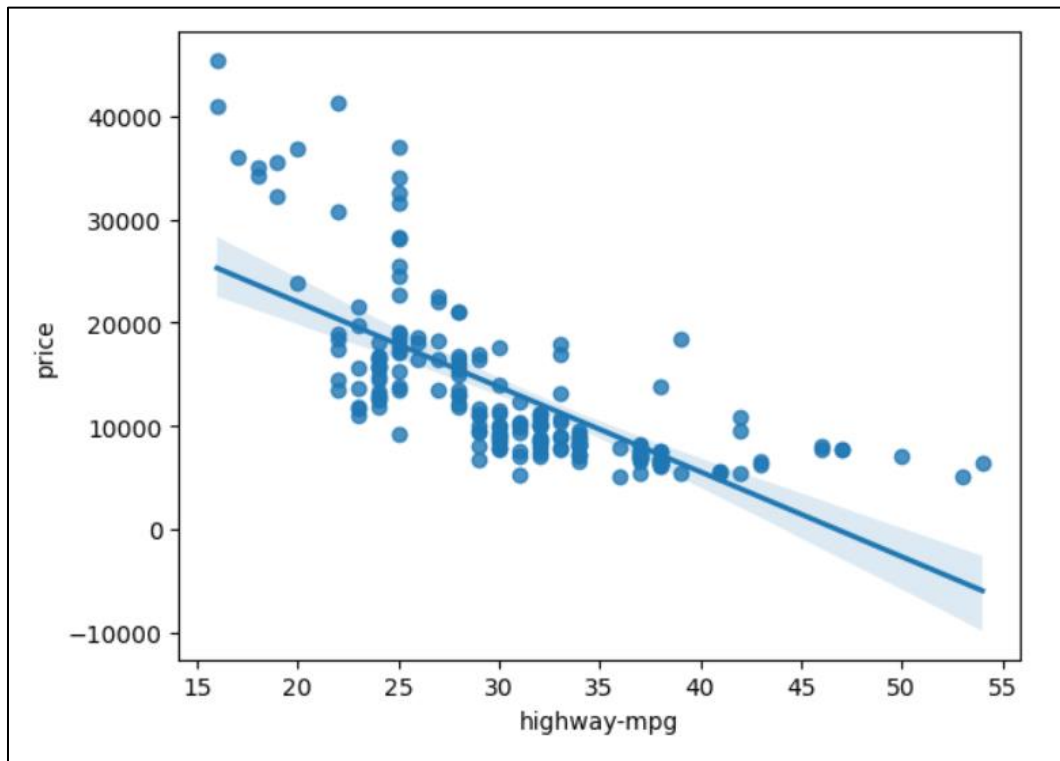


Figure 3.3: Regression plot showing the negative linear relationship between "highway-mpg" and "price"

	highway-mpg	price
highway-mpg	1.000000	-0.704692
price	-0.704692	1.000000

Figure 3.4: Correlation matrix output for "highway-mpg" and "price"

In contrast, the plot for highway-mpg versus price shows an inverse relationship: as highway-mpg increases, the price tends to decrease. This negative correlation suggests that highway-mpg could be a predictor of price, with a correlation coefficient of approximately -0.704.

- Weak Linear Relationship:

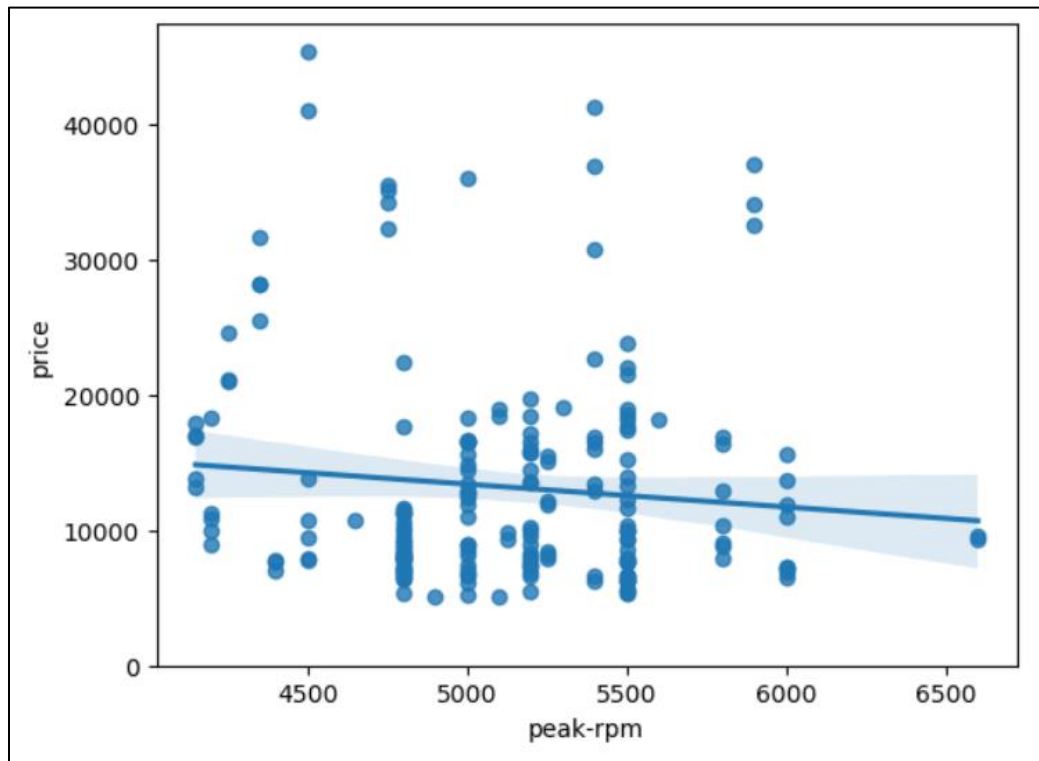


Figure 3.5: Regression plot showing the relationship between peak RPM and price.

	peak-rpm	price
peak-rpm	1.000000	-0.101616
price	-0.101616	1.000000

Figure 3.6: Correlation matrix between peak RPM and price.

The regression line in the plot appears nearly horizontal, indicating that "peak-rpm" is not a good predictor of car price. The data points are widely scattered, suggesting a high level of variability and a weak linear relationship. The correlation coefficient between "peak-rpm" and "price" is approximately -0.101616, reinforcing the conclusion that "peak-rpm" is not a reliable predictor.

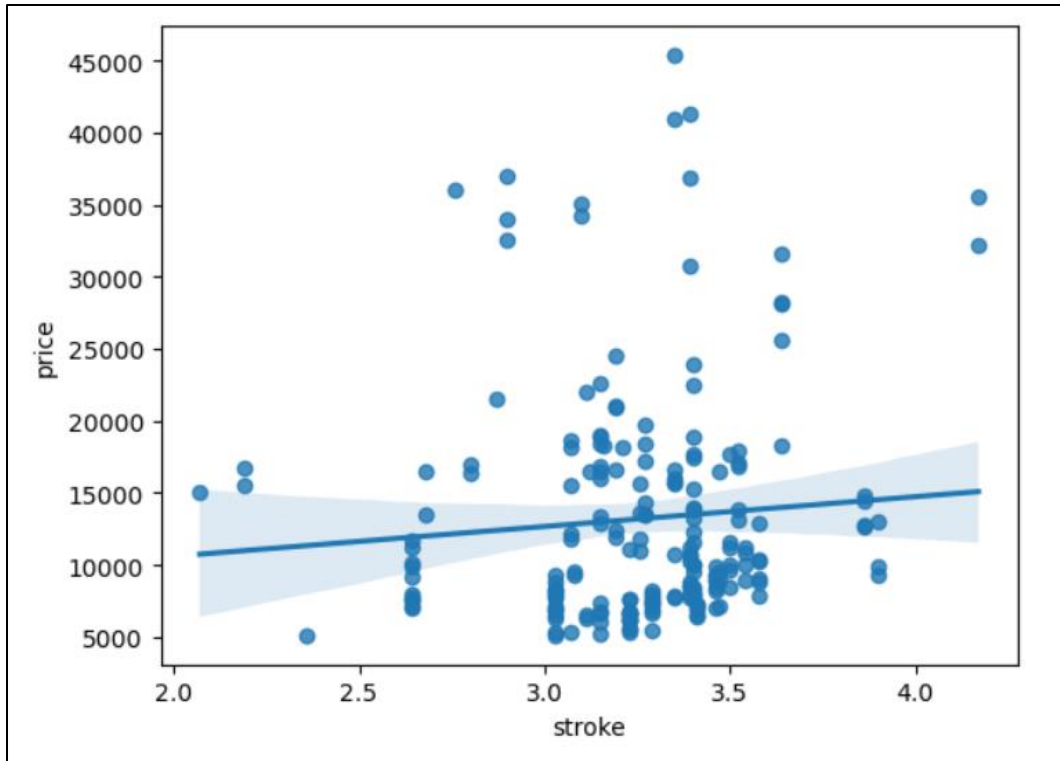


Figure 3.7: Regression plot showing the relationship between stroke and price.

	stroke	price
stroke	1.000000	0.082269
price	0.082269	1.000000

Figure 3.8: Correlation matrix between stroke and price.

The correlation coefficient for "stroke" and "price" is approximately 0.082, which is very close to zero, indicating a very weak linear relationship. Thus, "stroke" is not a significant predictor of car price.

3.2 Categorical Variables

Categorical variables represent characteristics of data units and are typically drawn from a small set of categories. These variables can be of type "object" or "int64". Boxplots are an effective way to visualize the distribution of categorical variables in relation to other variables, such as price.

To illustrate this, consider the distribution of price across different body-style categories:

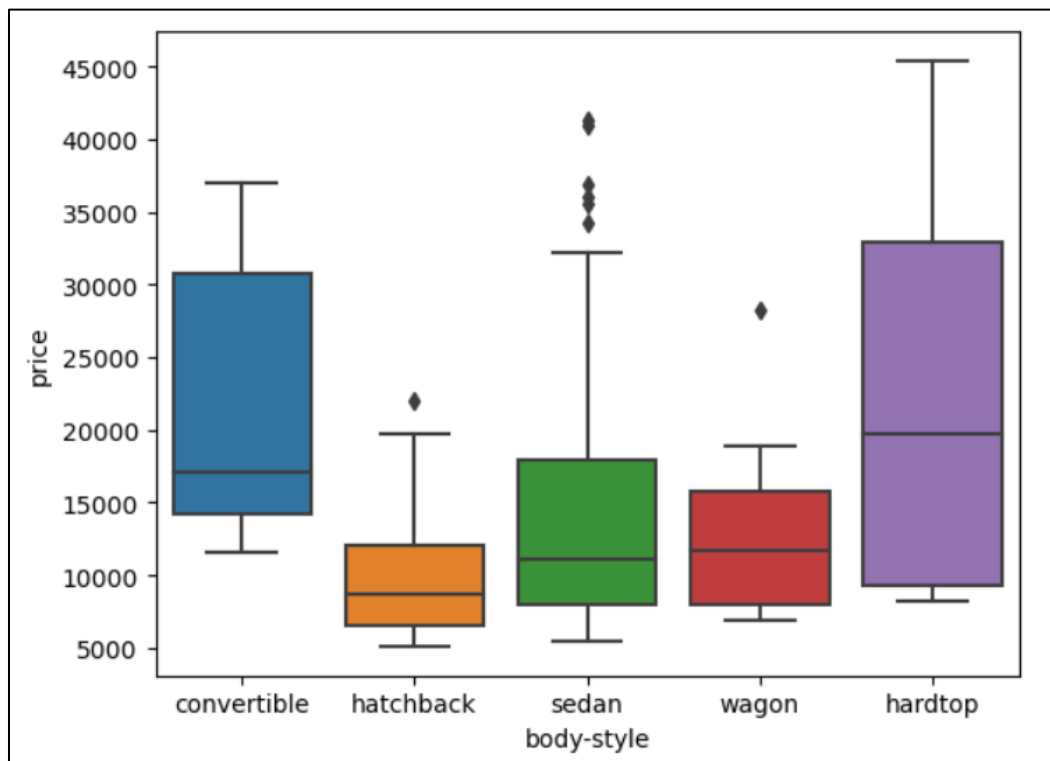


Figure 3.9: Price Distribution by Body Style

The boxplot reveals that the distributions of price among the various body-style categories have significant overlap, suggesting that body-style may not be a strong predictor of price.

Next, we examine the relationship between engine location and price:

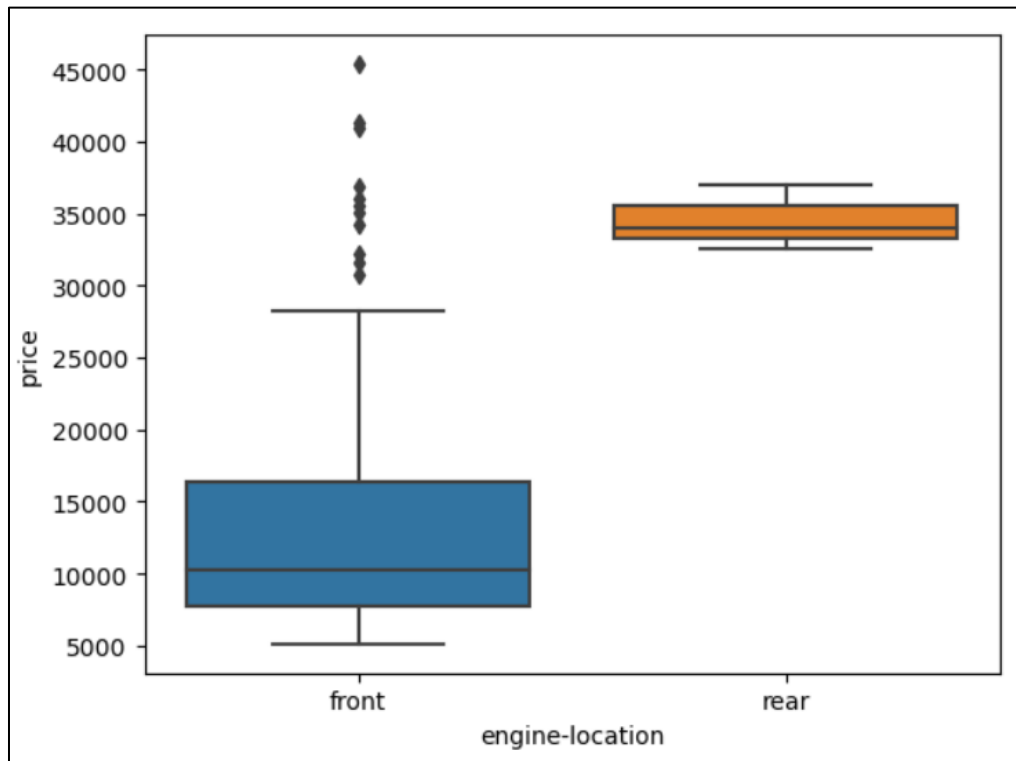


Figure 3.10: Price Distribution by Engine Location

In this case, the price distribution between the two engine-location categories (front and rear) is distinct enough to consider engine location as a potential predictor of price.

Finally, we analyze the distribution of price across different drive-wheels categories:

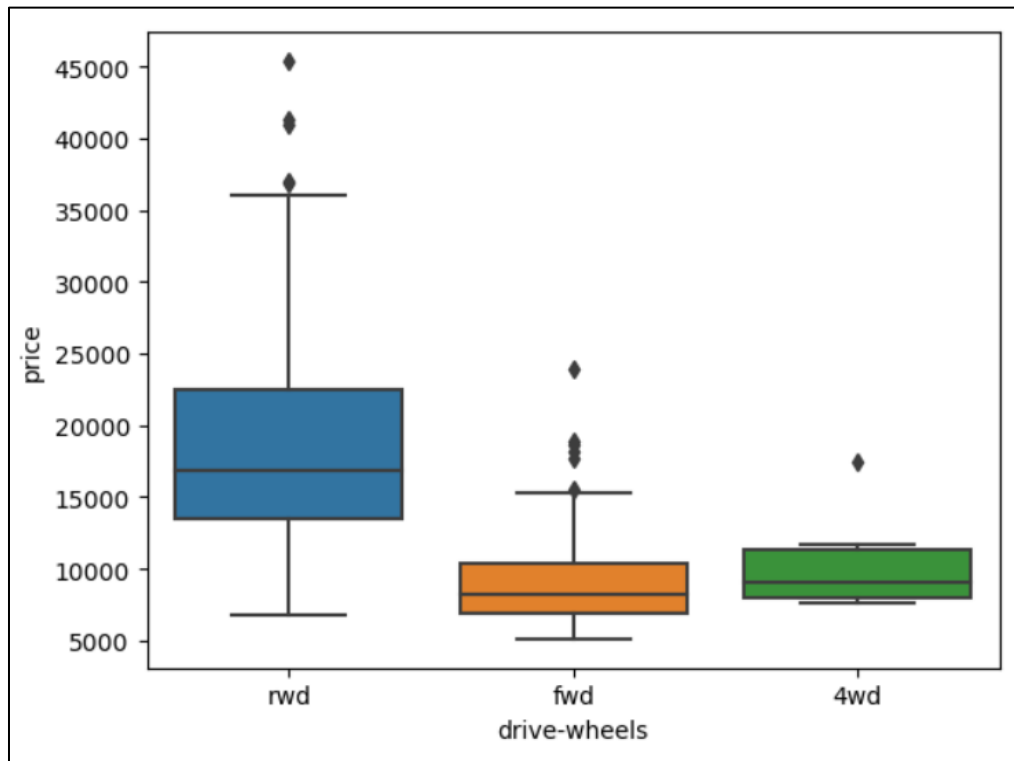


Figure 3.11: Price Distribution by Drive Wheels

The boxplot shows that the price distribution varies among the different drive-wheels categories, indicating that drive-wheels could potentially serve as a predictor of price.

3.3 Descriptive Statistical Analysis

After performing data wrangling, we generate descriptive statistics for the dataset to observe the changes and improvements compared to the initial data. The differences between the initial dataset, as shown in Figure 2.2, and this new dataset highlight the impact of data wrangling on the overall data quality.

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower
count	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000
mean	0.840796	122.000000	98.797015	0.837102	0.915126	0.899108	2555.666667	126.875622	3.330692	3.256874	10.164279	103.402985
std	1.254802	31.99625	6.066366	0.059213	0.029187	0.040933	517.296727	41.546834	0.268072	0.316048	4.004965	37.365650
min	-2.000000	65.000000	86.600000	0.678039	0.837500	0.799331	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000
25%	0.000000	101.000000	94.500000	0.801538	0.890278	0.869565	2169.000000	98.000000	3.150000	3.110000	8.600000	70.000000
50%	1.000000	122.000000	97.000000	0.832292	0.909722	0.904682	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000
75%	2.000000	137.000000	102.400000	0.881788	0.925000	0.928094	2926.000000	141.000000	3.580000	3.410000	9.400000	116.000000
max	3.000000	256.000000	120.900000	1.000000	1.000000	1.000000	4066.000000	326.000000	3.940000	4.170000	23.000000	262.000000

Figure 3.12: Descriptive Statistics After Data Wrangling

To further explore the categorical variables, we generate descriptive statistics specifically for those variables. This helps in understanding the distribution and frequency of categories in the cleaned dataset.

	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	fuel-system
count	201	201	201	201	201	201	201	201	201	201
unique	22	2	2	2	5	3	2	6	7	8
top	toyota	gas	std	four	sedan	fwd	front	ohc	four	mpfi
freq	32	181	165	115	94	118	198	145	157	92

Figure 3.13: Descriptive Statistics for Categorical Variables

- **Value Counts:** To better understand the distribution of categorical variables, the frequency of each unique value in the 'drive-wheels' and 'engine-location' columns was analyzed.

First, the count of each category in the “drive-wheels” column was computed and displayed as follows:

	count
drive-wheels	
fwd	118
rwd	75
4wd	8

Figure 3.14: Value Counts for “drive-wheels”

The results show that the dataset contains 118 cars with front-wheel drive (fwd), 75 with rear-wheel drive (rwd), and 8 with four-wheel drive (4wd). This indicates that front-wheel drive is the most common type in the dataset, while four-wheel drive is relatively rare.

Next, the frequency of each unique value in the 'engine-location' column was analyzed:

count	
engine-location	
front	198
rear	3

Figure 3.15: Value Counts for “engine-location”

The results indicate that there are 198 cars with a front engine and only 3 with a rear engine. Due to the significant imbalance in the number of cars with each engine location, this variable is not suitable as a predictor for price in this dataset.

- **Grouping:** To better understand the relationships between categorical variables like “drive-wheels”, “body-style”, and their impact on “price”, a series of grouping operations were performed. These operations help in summarizing the data by calculating the mean “price” for each group, making it easier to identify patterns and trends.

First, the dataset was grouped by “drive-wheels” to calculate the average “price” for each type of drive-wheel configuration.

drive-wheels		price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19757.613333

Figure 3.16: Mean Price by Drive-Wheels

Next, the data was grouped by “body-style” to determine the average “price” for each body style. This helps in understanding how different body styles are priced on average.

	body-style	price
0	convertible	21890.500000
1	hardtop	22208.500000
2	hatchback	9957.441176
3	sedan	14459.755319
4	wagon	12371.960000

Figure 3.17: Mean Price by Body-Style

To explore the interaction between “drive-wheels” and “body-style” in relation to “price”, the data was grouped by both variables, and the mean “price” was calculated for each combination.

	drive-wheels	body-style	price
0	4wd	convertible	NaN
1	4wd	hardtop	NaN
2	4wd	hatchback	7603.000000
3	4wd	sedan	12647.333333
4	4wd	wagon	9095.750000
5	fwd	convertible	11595.000000
6	fwd	hardtop	8249.000000
7	fwd	hatchback	8396.387755
8	fwd	sedan	9811.800000
9	fwd	wagon	9997.333333
10	rwd	convertible	23949.600000
11	rwd	hardtop	24202.714286
12	rwd	hatchback	14337.777778
13	rwd	sedan	21711.833333
14	rwd	wagon	16994.222222

Figure 3.18: Mean Price by Drive-Wheels and Body-Style

The grouped data was then pivoted to create a matrix format, which allows for easier comparison between the different categories. Missing values were filled with zeros to handle

cases where certain combinations of “drive-wheels” and “body-style” do not exist in the dataset.

	price				
	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	0.0	0.000000	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222

Figure 3.19: Pivoted Mean Price by Drive-Wheels and Body-Style

To visualize the grouped data, a heatmap was created using a pseudocolor plot, where the color intensity represents the mean “price”. This plot provides a clear visual representation of how the “price” varies across different combinations of “drive-wheels” and “body-style”.

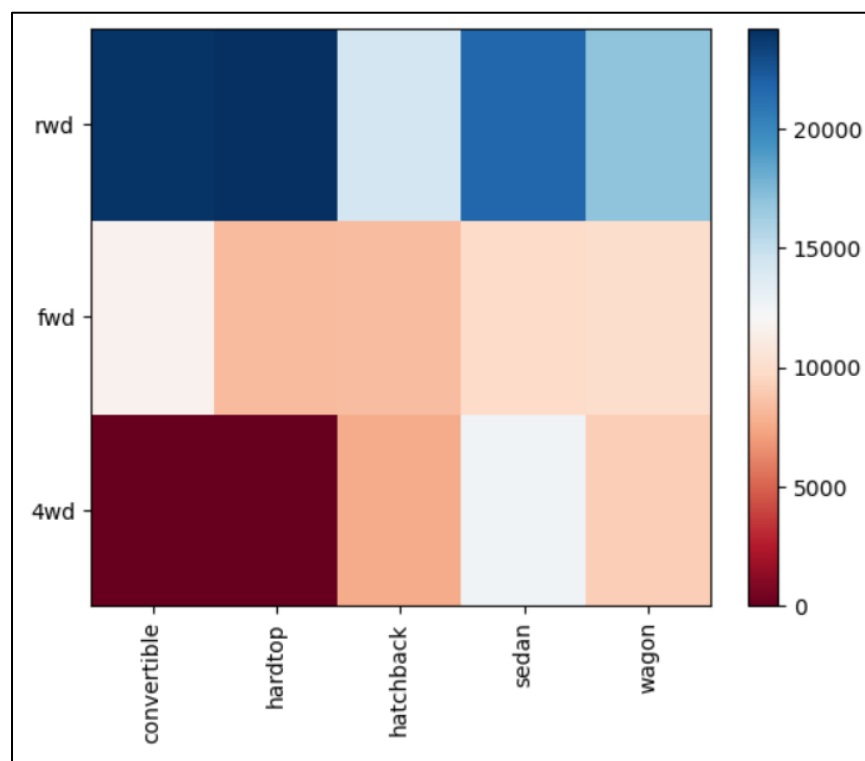


Figure 3.20: Heatmap of Mean Price by Drive-Wheels and Body-Style

3.4 Correlation and Causation

Understanding the relationship between variables is crucial for effective data analysis. One way to quantify the strength and direction of a relationship between two continuous variables is through the **Pearson correlation coefficient**. This coefficient, ranging from -1 to 1, measures the linear relationship between two variables.

- A Pearson correlation close to 1 implies a strong positive linear relationship.
- A Pearson correlation close to -1 implies a strong negative linear relationship.
- A Pearson correlation close to 0 implies no linear relationship.

We can calculate the Pearson correlation for our dataset using the “`df.corr()`” function, which provides the correlation matrix for all continuous variables in the DataFrame.

Sometimes, we also need to know the significance of these correlations. This is where the **P-value** comes into play. The P-value indicates the probability that the observed correlation is statistically significant.

The P-value helps us determine whether the correlation observed in the data occurred by chance. Typically, a significance level of 0.05 is chosen, meaning there is a 95% confidence level that the correlation is significant. Based on the P-value:

- **P-value < 0.001**: Strong evidence that the correlation is significant.
- **P-value < 0.05**: Moderate evidence that the correlation is significant.
- **P-value < 0.1**: Weak evidence that the correlation is significant.
- **P-value > 0.1**: No evidence that the correlation is significant.

We can obtain both the Pearson correlation coefficient and the P-value using the “stats” module from the “scipy” library, providing a more comprehensive understanding of the relationship between variables. This allows us to make informed decisions about the predictive power of different variables in our dataset.

- **Wheel-Base vs. Price**: The Pearson Correlation Coefficient for Wheel-Base vs. Price is **0.585**, with a P-value of <0.001 . This result indicates a statistically significant correlation between the wheelbase of a vehicle and its price. The positive coefficient suggests a moderately strong

relationship, meaning that as the wheelbase increases, the price of the vehicle tends to increase as well.

- **Horsepower vs. Price:** The Pearson Correlation Coefficient for Horsepower vs. Price is **0.809**, with a P-value of <0.001 . This strong positive correlation indicates a significant relationship between horsepower and price, suggesting that vehicles with higher horsepower generally command higher prices. The strength of this relationship is one of the highest observed among the variables analyzed.

- **Length vs. Price:** The Pearson Correlation Coefficient for Length vs. Price is **0.691**, with a P-value of <0.001 . This indicates a statistically significant and moderately strong positive correlation between the length of a vehicle and its price. Longer vehicles tend to be priced higher, which may reflect the association of vehicle size with luxury or higher market segments.

- **Width vs. Price:** The Pearson Correlation Coefficient for Width vs. Price is **0.751**, with a P-value of <0.001 . The positive correlation is both significant and strong, indicating that wider vehicles are typically priced higher. This trend suggests that vehicle width may be associated with additional features or enhanced performance, contributing to higher prices.

- **Curb-Weight vs. Price:** The Pearson Correlation Coefficient for Curb-Weight vs. Price is **0.834**, with a P-value of <0.001 . This very strong positive correlation suggests a significant relationship between the curb weight of a vehicle and its price. Heavier vehicles, which often include additional safety features, luxury materials, or larger engines, tend to have higher prices.

- **Engine-Size vs. Price:** The Pearson Correlation Coefficient for Engine-Size vs. Price is **0.872**, with a P-value of <0.001 . The correlation between engine size and price is not only statistically significant but also very strong, indicating that vehicles with larger engines are generally more expensive. This result reflects the common association of larger engines with more powerful, and therefore higher-priced, vehicles.

- **Bore vs. Price:** The Pearson Correlation Coefficient for Bore vs. Price is **0.544**, with a P-value of <0.001 . Although statistically significant, the relationship between bore size and price is

moderately strong and less pronounced than other variables. This suggests that while bore size does influence price, it is likely one of many factors contributing to a vehicle's overall cost.

- City-mpg vs. Price: The Pearson Correlation Coefficient for City-mpg vs. Price is **-0.687**, with a P-value of <0.001 . This statistically significant negative correlation indicates that vehicles with higher city fuel efficiency (measured in miles per gallon) tend to have lower prices. This may reflect a market preference where fuel-efficient cars are positioned as more economical options, leading to lower pricing.

- Highway-mpg vs. Price: The Pearson Correlation Coefficient for Highway-mpg vs. Price is **-0.705**, with a P-value of <0.001 . Similar to city-mpg, this strong and statistically significant negative correlation shows that vehicles with better highway fuel efficiency are generally less expensive. This result aligns with market trends that value higher fuel efficiency in more budget-friendly vehicle segments.

```
The Pearson Correlation Coefficient for Wheel-Base vs. Price is 0.5846418222655083 with a P-value of P = 8.076488270732552e-20
The Pearson Correlation Coefficient for Horsepower vs. Price is 0.8096068016571052 with a P-value of P = 6.273536270651023e-48
The Pearson Correlation Coefficient for Length vs. Price is 0.6906283804483644 with a P-value of P = 8.016477466158383e-30
The Pearson Correlation Coefficient for Width vs. Price is 0.7512653440522674 with a P-value of P = 9.20033551048144e-38
The Pearson Correlation Coefficient for Curb-Weight vs. Price is 0.8344145257702849 with a P-value of P = 2.189577238893391e-53
The Pearson Correlation Coefficient for Engine-Size vs. Price is 0.8723351674455185 with a P-value of P = 9.265491622198793e-64
The Pearson Correlation Coefficient for Bore vs. Price is 0.5431553832626606 with a P-value of P = 8.049189483935034e-17
The Pearson Correlation Coefficient for City-mpg vs. Price is -0.6865710067844681 with a P-value of P = 2.3211320655673725e-29
The Pearson Correlation Coefficient for Highway-mpg vs. Price is -0.7046922650589533 with a P-value of P = 1.7495471144474792e-31
```

Figure 3.21: Correlation Analysis between Various Features and Price

3.5 Analysis of Variance (ANOVA)

The Analysis of Variance (ANOVA) is a statistical technique employed to determine if there are significant differences between the means of two or more groups. It evaluates whether the variation in data is due to the differences between groups or due to random variation within groups.

ANOVA produces two key parameters:

- **F-test Score:** This score assesses the extent to which the actual means of the groups deviate from the means assumed to be equal. A higher F-test score indicates a greater difference between the means of the groups.
- **P-value:** This value helps determine the statistical significance of the F-test score. A smaller p-value suggests that the observed differences between group means are statistically significant.

In our analysis, we will use ANOVA to assess whether the 'drive-wheels' category impacts the 'price' variable. Given that ANOVA is designed to analyze differences between groups, we will leverage the groupby function to segment the data by 'drive-wheels'. Since ANOVA computes group averages automatically, there is no need for preliminary averaging.

By applying ANOVA to the “drive-wheels” and “price” variables, we can evaluate whether different drive-wheel configurations significantly affect vehicle prices. A significant F-test score and a low p-value would indicate that the type of 'drive-wheels' has a considerable impact on the price, aligning with the expectation that such a correlation would result in a

- **Drive Wheels Analysis:** To analyze the impact of different drive-wheel types on vehicle price, ANOVA was conducted on the “drive-wheels” and “price” data. The overall ANOVA results yielded an F-test score of **67.954** and a p-value of **<0.001**. This indicates a significant difference in mean prices across different drive-wheel types, suggesting that the type of drive-wheel does have a substantial effect on vehicle price.

This result is compelling, with a substantial F-test score indicating a strong correlation, and a p-value close to 0, suggesting near-certain statistical significance. However, does this imply that all three drive-wheel types are equally influential on vehicle price? We need to examine the pairwise comparisons to understand if each individual drive-wheel type significantly differs from the others.

- **fwd and rwd:** The ANOVA results for the comparison of “fwd” and “rwd” drive-wheels with respect to vehicle price yielded an F-test score of **130.55** and a p-value of **<0.001**. This high F-test score and extremely low p-value indicate a significant difference in mean prices between these two drive-wheel types, confirming a strong correlation.

- **4wd and rwd:** When comparing “4wd” and “rwd” drive-wheels, the ANOVA test produced an F-test score of **8.58** and a p-value of **0.0044**. This result shows a moderate but statistically significant difference in prices between these groups, suggesting that drive-wheel type does impact vehicle price, though less strongly than between 'fwd' and 'rwd'.

- **4wd and fwd:** Finally, the comparison of '4wd' and 'fwd' drive-wheels resulted in an F-test score of **0.67** and a p-value of **0.4162**. The low F-test score and relatively high p-value indicate that there is no significant difference in prices between these two types of drive-wheels.

```
ANOVA results for 'drive-wheels' and 'price': F= 67.95406500780399 , P = 3.3945443577151245e-23
ANOVA results for 'fwd' vs 'rwd' drive-wheels and 'price': F = 130.5533160959111 , P = 2.2355306355677845e-23
ANOVA results for '4wd' vs 'rwd' drive-wheels and 'price': F = 8.580681368924756 , P = 0.004411492211225333
ANOVA results for '4wd' vs 'fwd' drive-wheels and 'price': F = 0.665465750252303 , P = 0.41620116697845666
```

Figure 3.22: ANOVA Results for Drive Wheels and Vehicle Price

- **Important Variables:** Having analyzed our data, we now have a clearer understanding of the key variables that significantly impact car prices. Our analysis has highlighted the following crucial variables:

- **Continuous Numerical Variables:**

- Length
- Width
- Curb-weight
- Engine-size
- Horsepower
- City-mpg
- Highway-mpg
- Wheel-base
- Bore

- **Categorical Variables:**

- Drive-wheel

As we advance to building machine learning models for automating our analysis, incorporating these variables that meaningfully influence our target variable will enhance the performance of our predictions.

4. Model Development

Model development allows us to estimate car prices using selected variables, offering an objective measure of a car's potential market value. This analysis is crucial for determining whether the price set is reasonable and aligns with the intrinsic value derived from the car's features.

By creating models, we can explore the relationships between different variables and assess their predictive power. These models will guide us in setting prices that accurately reflect the car's worth, ensuring that the pricing decisions are well-informed and data-driven.

4.1 Linear Regression and Multiple Linear Regression

In our previous analysis, we identified the following variables as significant predictors of car price:

- | | | |
|---------------|---------------|---------------|
| - Length | - Horsepower | - Bore |
| - Width | - City-mpg | - Drive-wheel |
| - Curb-weight | - Highway-mpg | |
| - Engine-size | - Wheel-base | |

For the Simple Linear Regression, we will use “highway-mpg” as the predictor variable. This choice allows us to assess how `highway-mpg` alone influences car price.

In the Multiple Linear Regression model, we will include “highway-mpg”, “engine-size”, “horsepower”, and “curb-weight2 as predictor variables. This approach enables us to analyze the combined effect of these variables on car price, providing a more comprehensive understanding of the factors influencing the price.

- **Simple Linear Regression:** Simple Linear Regression is a statistical method used to understand the relationship between two variables: the predictor or independent variable (X) and the response or dependent variable (Y), which we aim to predict. The outcome of this analysis is a linear function that models the dependent variable as a function of the independent variable. In this linear function, "a" represents the intercept of the regression

line, indicating the value of Y when X is zero, while "b" denotes the slope of the line, reflecting the change in Y for every one-unit increase in X. The Linear Function

$$Y = a + bX$$

The first predicted values are: [16236.50464347 16236.50464347 17058.23802179 13771.3045085 20345.17153508]
 The intercept is: a = 38423.305858157386
 The coefficient is: b = -821.733378321925
 The linear equation is: Y = 38423.31 + -821.73X

Figure 4.1: Linear Regression for Predicting Price using Highway-mpg

The linear regression analysis performed to predict car prices based on the "highway-mpg" variable resulted in the equation:

$$\text{Price} = 38423.31 - 821.73(\text{Highway} - \text{mpg})$$

- Multiple Linear Regression: To enhance the prediction of car prices, we can employ Multiple Linear Regression, which extends the concept of Simple Linear Regression to accommodate more than one predictor variable. Unlike Simple Linear Regression, which examines the relationship between a single predictor variable and a response variable, Multiple Linear Regression models the relationship between one continuous response variable and two or more predictor variables. This approach is useful in real-world scenarios where multiple factors influence the outcome. In our case, we will use four predictor variables to illustrate this method, although it can be generalized to any number of predictors. The general equation for Multiple Linear Regression is given by:

$$Y = a + b_1X_1 + b_2X_2 + b_3X_3 + \dots$$

The first predicted values are: [13699.07700462 13699.07700462 19052.71346719 10620.61524404 15520.90025344]
 The coefficients for each predictor are: [36.1593925 81.51280006 53.53022809 4.70805253]
 The intercept is: a = -15811.86376772925
 The linear equation is: Y = -15811.86 + 36.16*highway-mpg + 81.51*engine-size + 53.53*horsepower + 4.71*curb-weight

Figure 4.2: Multiple Linear Regression with Highway-mpg, Engine-size, Horsepower, and Curb-weight

The Multiple Linear Regression analysis conducted to predict car prices using the variables “highway-mpg”, “engine-size”, “horsepower”, and “curb-weight” resulted in the following equation:

$$\text{Price} = -15811.86 + 36.16(\text{Highway - mpg}) + 81.51(\text{Engine - Size}) + 53.53(\text{Horsepower}) + 4.71(\text{Curb - weight})$$

4.2 Model Evaluation Using Visualization

- **Regression Plot:** Visualizing the relationship between two variables in simple linear regression can be effectively achieved using regression plots. These plots combine scatterplots with a fitted linear regression line, offering a clear visual representation of the data's relationship. This visualization helps estimate the correlation's strength and direction, indicating whether it is positive or negative. For this analysis, we'll examine how 'highway-mpg' can serve as a predictor for car price.

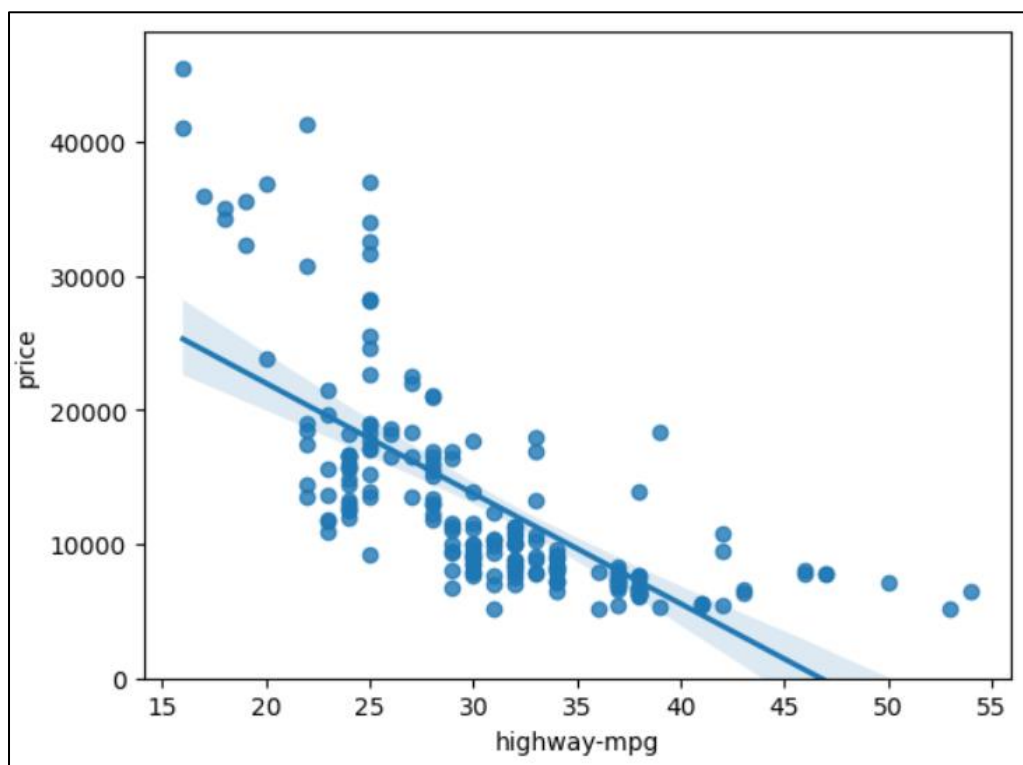


Figure 4.3: Regression Plot of Highway-mpg as a Predictor of Price

The regression plot reveals a negative correlation between price and highway-mpg, as indicated by the downward slope of the regression line. It is important to observe the spread

of data points around this line, as it reflects the variance in the data. If the points are widely scattered, it suggests that a linear model may not be the most appropriate fit for this relationship. For comparison, let's examine the regression plot for "peak-rpm."

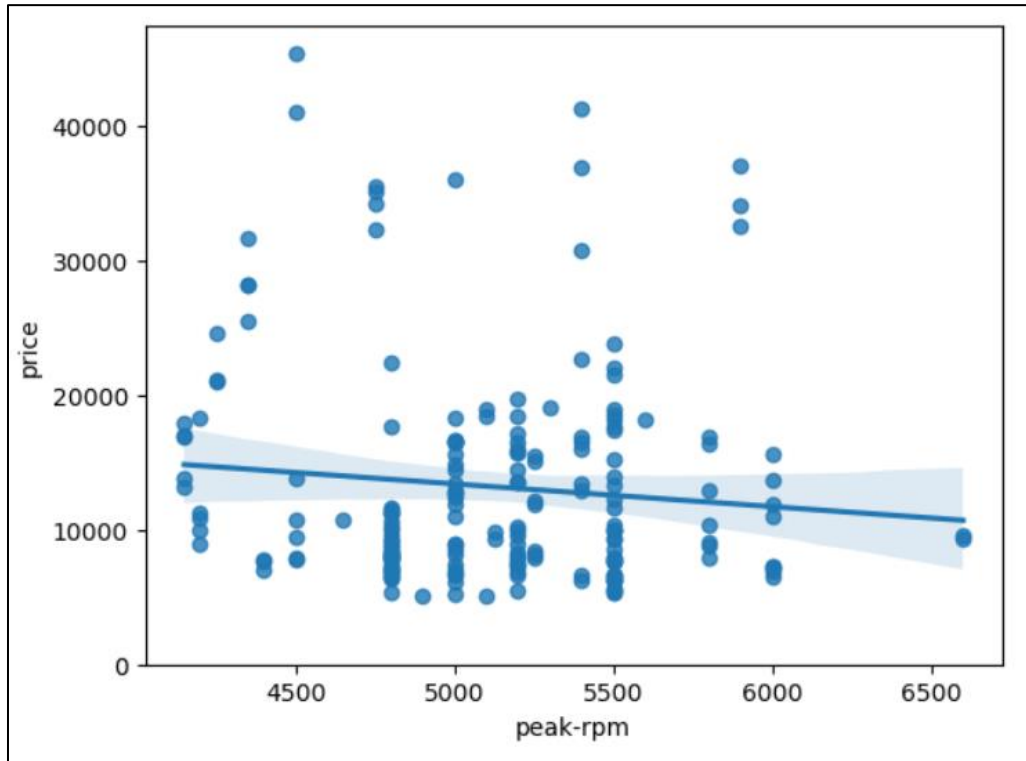


Figure 4.4: Regression Plot of Peak-rpm as a Predictor of Price

The comparison between the regression plots of "peak-rpm" and "highway-mpg" reveals distinct differences in the data's fit to the linear models. The "highway-mpg" plot shows data points closely clustered around the regression line, indicating a clearer and more consistent negative correlation. In contrast, the "peak-rpm" plot exhibits a wider spread of data points, making it challenging to discern a clear trend, which suggests that "peak-rpm" may not be as reliable a predictor of price as "highway-mpg".

The correlation reveals the relationships between "peak-rpm", "highway-mpg", and "price". A notable negative correlation exists between 'highway-mpg' and 'price' with a coefficient of -0.704692, indicating that as "highway-mpg" increases, the price tends to decrease. On the other hand, "peak-rpm" shows a weak negative correlation with "price" at -0.101616,

suggesting a minimal inverse relationship. Additionally, the correlation between “peak-rpm” and “highway-mpg” is almost negligible, with a value of -0.058598.

	peak-rpm	highway-mpg	price
peak-rpm	1.000000	-0.058598	-0.101616
highway-mpg	-0.058598	1.000000	-0.704692
price	-0.101616	-0.704692	1.000000

Figure 4.5: Correlation Matrix for “peak-rpm”, “highway-mpg”, and “price”

- **Residual Plot:** A residual plot is an effective tool for visualizing the variance in the data. The residual, defined as the difference between the observed value (y) and the predicted value (\hat{y}), represents how far off our model's predictions are from the actual data points. The residual plot graphs these residuals on the y-axis against the independent variable on the x-axis.

In a well-fitting linear model, the residuals should be randomly scattered around the x-axis, indicating constant variance and suggesting that a linear model is appropriate. However, the residual plot generated here shows that the residuals are not randomly distributed, which implies that the variance may not be constant and suggests that a non-linear model might be more suitable for this data.

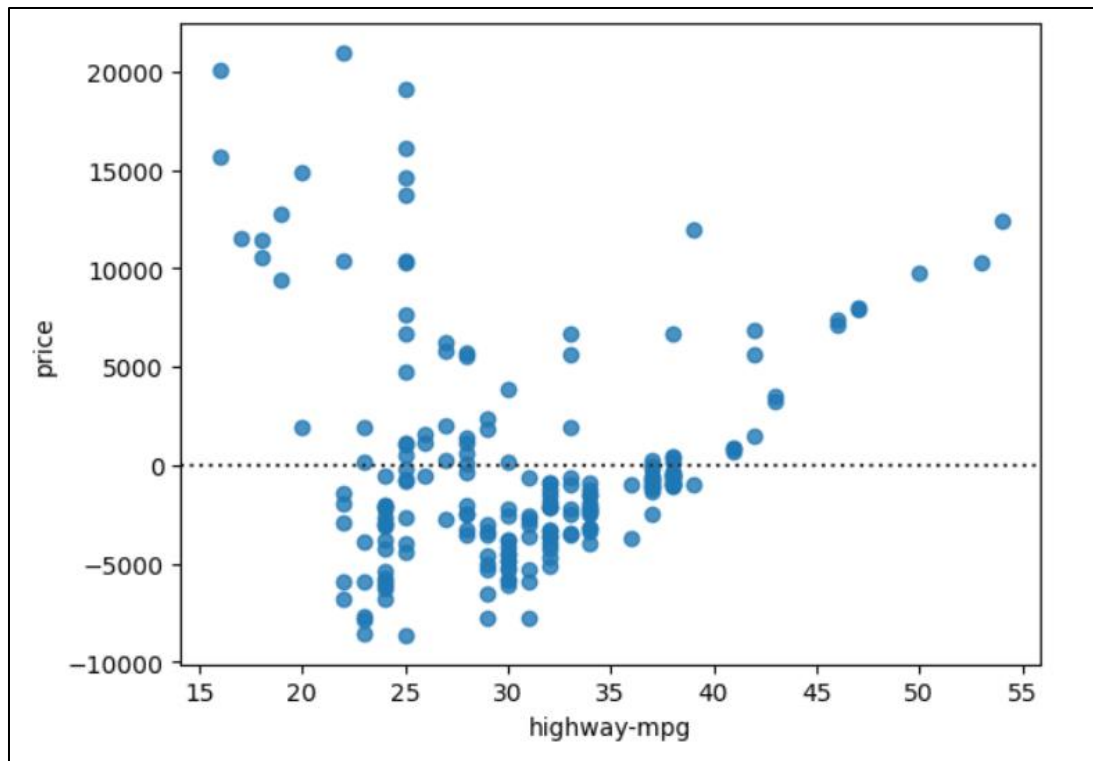


Figure 4.6: Residual Plot of Highway-mpg as a Predictor of Price

- **Visualizing Multiple Linear Regression Models:** Visualizing the fit of a Multiple Linear Regression model can be challenging, as traditional regression or residual plots might not fully capture the model's performance. One effective method is to use distribution plots to compare the fitted values with the actual values.

A distribution plot helps us assess how well the model's predictions match the true data. By comparing the distribution of the fitted values (from the model) to the distribution of the actual values, we can gauge the model's performance.

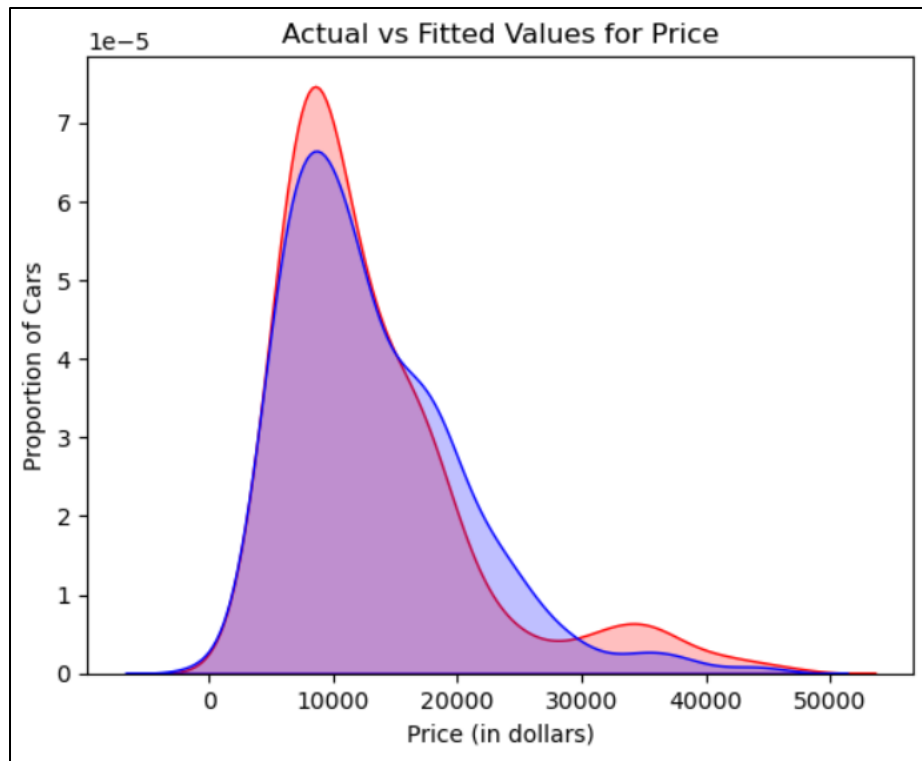


Figure 4.7: Distribution Comparison of Actual vs Fitted Values

In this plot, the distribution of actual car prices is shown in red, while the fitted values predicted by the model are depicted in blue. The plot is titled "Actual vs Fitted Values for Price," with the x-axis labeled "Price (in dollars)" and the y-axis labeled "Proportion of Cars."

The overlap between the distributions of the actual and fitted values suggests that the model's predictions are reasonably close to the actual values. However, there is still some room for improvement in the model's accuracy.

4.3 Polynomial Regression

- **Polynomial Regression:** Polynomial regression extends the general linear regression model by incorporating polynomial terms, allowing us to model non-linear relationships between predictor variables and the target variable. By including squared or higher-order terms of the predictor variables, we can capture more complex patterns in the data. Polynomial regression models can be classified based on their degree: a quadratic regression represents a 2nd-order polynomial, a cubic regression represents a 3rd-order polynomial, and higher-

order polynomials can capture even more intricate relationships. A higher order polynomial equation is:

$$Y = a + b_1X + b_2X^2 + b_3X^3 + \dots$$

Previously, we observed that a linear model did not provide the best fit when using "highway-mpg" as the predictor variable. Therefore, we will explore fitting a polynomial regression model to the data to potentially improve the fit and capture more nuanced relationships.

Polynomial equation:

$$\text{Price} = 137923.59 + -8965.43 * (\text{highway-mpg}) + 204.75 * (\text{highway-mpg})^2 + -1.56 * (\text{highway-mpg})^3$$

Figure 4.8: Polynomial Equation for Predicting Price using Highway-mpg

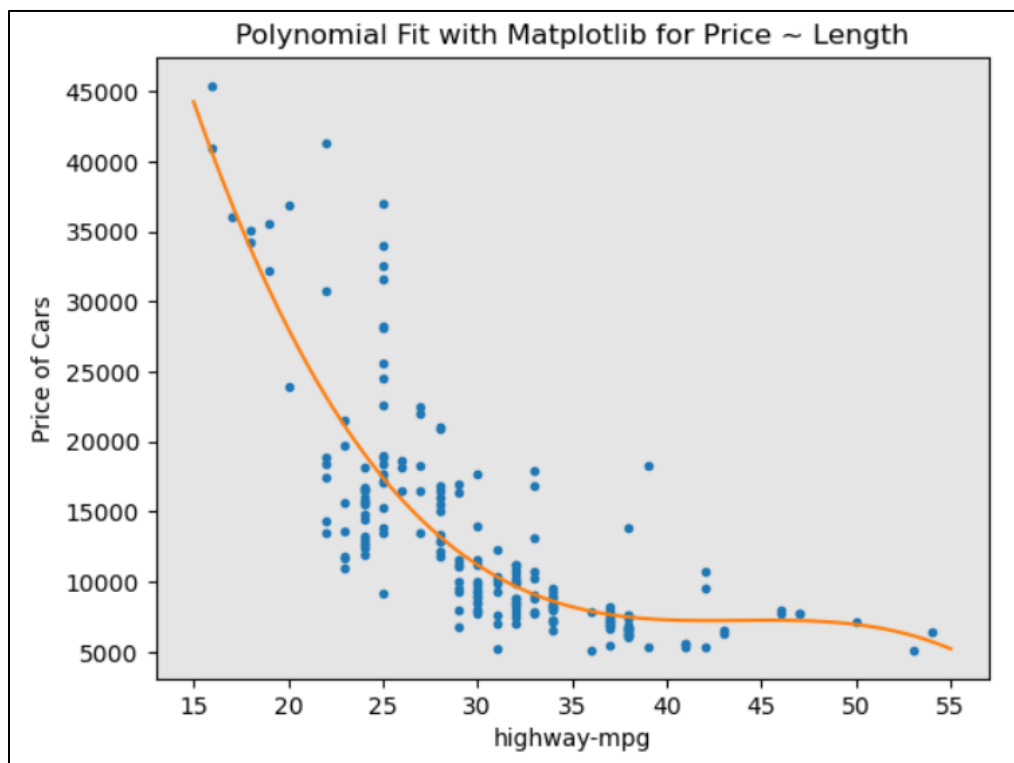


Figure 4.9: Polynomial Regression Fit for Car Prices Based on Highway-mpg

The polynomial regression model shows an improved fit compared to the linear model, as evidenced by the plotted results. The polynomial function more effectively captures the variability in the data, providing a closer alignment with the observed data points. This

enhanced performance is attributed to the polynomial's ability to accommodate non-linear relationships, thereby "hitting" more of the data points than the simpler linear model.

4.4 Model Measurement

- **Simple Linear Regression:** To evaluate our linear regression model, we first calculate the R-squared value, which indicates how well our model explains the variation in car prices. By fitting the model with the "X" (independent variable) and "Y" (dependent variable) data, we find an R-squared value of approximately 49.659%, meaning that this simple linear model accounts for about 49.659% of the variation in prices based on "highway-mpg". Next, we assess the accuracy of our model using the Mean Squared Error (MSE). By predicting the values with the `predict` method and comparing them to the actual values using the "mean_squared_error" function, we compute the MSE. This metric measures the average squared difference between the predicted and actual prices, providing insight into the model's prediction accuracy.

```
The R-square is: 0.4965911884339175
The output of the first four predicted value is: [16236.50464347 16236.50464347 17058.23802179 13771.3045085 ]
The mean square error of price and predicted value is: 31635042.944639895
```

Figure 4.10: R-squared and MSE Evaluation for Simple Linear Regression Model

- **Multiple Linear Regression:** To evaluate the performance of our multiple linear regression model, we first calculate the R^2 value to determine the proportion of variation in car prices explained by the model. The R^2 value indicates that approximately 80.896% of the variation in prices is accounted for by this model. Next, we assess the prediction accuracy by calculating the Mean Squared Error (MSE). Predictions are made using the fitted model, and we compare these predicted values with the actual prices to gauge the model's effectiveness.

Figure 4.11: R-squared and MSE Evaluation for Multiple Linear Regression Model

- **Polynomial Fit:** In evaluating the polynomial regression model, we find that approximately 67.419% of the variation in car prices is explained by the polynomial fit. This R^2 value indicates that the polynomial model provides a relatively good explanation of the variability in the data, capturing more of the data's structure compared to simpler models. To further assess the model's performance, we also calculate the Mean Squared Error (MSE), which

quantifies the average squared difference between the actual prices and the predicted values from the polynomial model.

The R-square value is: 0.674194666390652
The Mean Squared Error is: 20474146.426361207

Figure 4.12: R-squared and MSE Evaluation for Polynomial Fit Model

4.5 Decision Making

After visualizing the different models and generating the R-squared and Mean Squared Error (MSE) values for each fit, the next step is to determine which model offers the best fit for the data.

- Understanding R-squared and MSE:

- R-squared is a statistical measure that indicates how well the independent variables explain the variability of the dependent variable. A higher R-squared value suggests that the model explains a larger portion of the variability, making it a better fit.
- MSE measures the average of the squares of the errors, which are the differences between the actual and predicted values. A smaller MSE indicates a model that has better predictive accuracy.

Let's examine the performance metrics for three different models:

1. Simple Linear Regression (SLR): Using "Highway-mpg" as the predictor variable for "Price."

- R-squared: 0.4966
- MSE: 3.16×10^7

2. Multiple Linear Regression (MLR): Using "Horsepower," "Curb-weight," "Engine-size," and "Highway-mpg" as predictor variables for "Price."

- R-squared: 0.8089
- MSE: 1.2×10^7

3. Polynomial Fit: Using "Highway-mpg" as the predictor variable for "Price."

- R-squared: 0.6742
- MSE: 2.05×10^7

- Comparison:

1. Simple Linear Regression (SLR) vs. Multiple Linear Regression (MLR):

- **MSE Comparison:** The MSE of the SLR model is 3.16×10^7 , while the MSE for the MLR model is significantly lower at 1.2×10^7 . This indicates that the MLR model has a better predictive accuracy.
- **R-squared Comparison:** The R-squared value for the SLR model is 0.4966, which is much lower than the R-squared value of 0.8089 for the MLR model. This suggests that the MLR model explains a greater portion of the variance in the price.
- **Conclusion:** The MLR model, with its lower MSE and higher R-squared value, provides a better fit compared to the SLR model.

2. Simple Linear Regression (SLR) vs. Polynomial Fit:

- **MSE Comparison:** The Polynomial Fit reduces the MSE compared to the SLR model, indicating improved accuracy.
- **R-squared Comparison:** The Polynomial Fit also increases the R-squared value, suggesting it explains more variance in the price than the SLR model.
- **Conclusion:** The Polynomial Fit is a better model than the SLR for predicting price using "Highway-mpg" as the predictor variable.

3. Multiple Linear Regression (MLR) vs. Polynomial Fit:

- **MSE Comparison:** The MSE for the MLR model is smaller than that for the Polynomial Fit, indicating better accuracy.
- **R-squared Comparison:** The R-squared value for the MLR model is also higher than that for the Polynomial Fit, further confirming the MLR model's superiority.
- **Conclusion:** Among the three models, the Multiple Linear Regression (MLR) model emerges as the best fit for predicting car prices based on the dataset. This conclusion is supported by

its significantly lower MSE and higher R-squared value compared to both the Simple Linear Regression and Polynomial Fit models. The MLR model's ability to incorporate multiple predictor variables (such as "Horsepower," "Curb-weight," "Engine-size," and "Highway-mpg") allows it to better capture the complexity of the data, leading to more accurate predictions of car prices. Given that the dataset includes 27 variables, it's logical that a model considering multiple variables would provide a more nuanced and effective prediction, making the MLR model the most suitable choice for this analysis.

5. Model Evaluation

For effective model evaluation, it's essential to assess how well the model performs on both training and test datasets. Evaluating a model typically involves calculating metrics such as R-squared and Mean Squared Error (MSE). These metrics help determine how well the model fits the data and predicts new, unseen data. Ensuring that a model has a high R-squared value and a low MSE can indicate a better fit, suggesting that the model captures the underlying patterns in the data effectively. Proper evaluation and refinement help ensure that the model generalizes well and performs accurately on new data.

5.1 Training and Testing

- Training and Testing 90/10: In the model evaluation process, the dataset was first split into training and testing subsets using a 90/10 ratio, ensuring that 10% of the data was reserved for testing the model's performance. This split resulted in 21 samples for the test set and 180 samples for the training set.

```
Number of test samples: 21
Number of training samples: 180
R-squared value for the test set with 10% split: 0.3635480624962414
R-squared value for the training set with 10% split: 0.662028747521533
```

Figure 5.1: Model Evaluation with 10% Test Split

After fitting the Linear Regression model using "horsepower" as the predictor variable, the R-squared value for the test set was approximately 0.364. This indicates that the model explains about 36.4% of the variance in car prices based on the 'horsepower' feature. This

moderate R-squared value suggests that while 'horsepower' is a useful predictor, other factors may be influencing car prices that are not captured by this model.

For the training set, the R-squared value was around 0.662, meaning the model explains approximately 66.2% of the variance in the training data. The higher R-squared value for the training set compared to the test set indicates that the model fits the training data better. However, the noticeable difference between the training and test R-squared values points to potential overfitting, suggesting that the model may not generalize well to new, unseen data. This disparity emphasizes the need for further refinement or inclusion of additional features to improve the model's performance and generalizability.

- Training and Testing 60/40: In this model evaluation process, the dataset was divided into training and testing subsets using a 60/40 ratio, reserving 40% of the data for testing the model's performance. This resulted in 81 samples for the test set and 120 samples for the training set.

```
Number of test samples: 81
Number of training samples: 120
R-squared value for the test set with 40% split: 0.7139737368233015
R-squared value for the training set with 40% split: 0.5754853866574969
```

Figure 5.2: Model Evaluation with 40% Test Split

After fitting the Linear Regression model with "horsepower" as the predictor variable, the R-squared value for the test set was approximately 0.714. This indicates that the model explains about 71.4% of the variance in car prices based on the 'horsepower' feature. This relatively high R-squared value for the test set suggests that the model performs well in predicting car prices and generalizes reasonably well to unseen data.

For the training set, the R-squared value was around 0.575, which means the model explains about 57.5% of the variance in the training data. The difference between the training and test R-squared values is less pronounced compared to the previous 90/10 split, indicating reduced overfitting and a more balanced performance across both datasets. This suggests that the model is better at generalizing to new data, though further refinement and feature inclusion could still enhance its predictive power.

- **Cross-Validation Analysis:** To assess the model's performance more robustly, cross-validation was employed using the 'horsepower' feature. Cross-validation splits the data into multiple folds, trains the model on each fold, and evaluates it on the remaining data. This technique helps in understanding how well the model generalizes to unseen data.

The mean of the folds are 0.5220592359225413 and the standard deviation is 0.29130480666118463 The mean of the negative MSEs is 23521246.47057339 and the standard deviation is 12000467.588458164 The mean R-squared score with 2-fold cross-validation is 0.516835099979672 The first five predicted values are: [14142.23793549 14142.23793549 20815.3029844 12745.549902 14762.9881726]

Figure 5.3: Cross-Validation Evaluation

The mean R-squared score across four folds was approximately 0.522, with a standard deviation of 0.291. This suggests a moderate level of model performance across different subsets of the data. Additionally, the mean of the negative Mean Squared Errors (MSEs) was around 23,521,246.47, with a standard deviation of 12,000,467.59. Since MSE values are negative in this context, a lower absolute value indicates better performance.

In a separate evaluation with 2-fold cross-validation, the mean R-squared score was 0.517, reflecting a similar level of performance but with fewer folds. The model's performance was also assessed through cross-validation predictions. These results provide insight into the model's prediction accuracy and consistency across different subsets of the data.

5.2 Overfitting, Underfitting and Model Selection

- **Model Performance:** Evaluating model performance involves assessing how well the model generalizes to unseen data, commonly referred to as the "out-of-sample data." One key aspect of this evaluation is detecting overfitting, which occurs when a model performs exceptionally well on the training data but fails to generalize to new, unseen data.

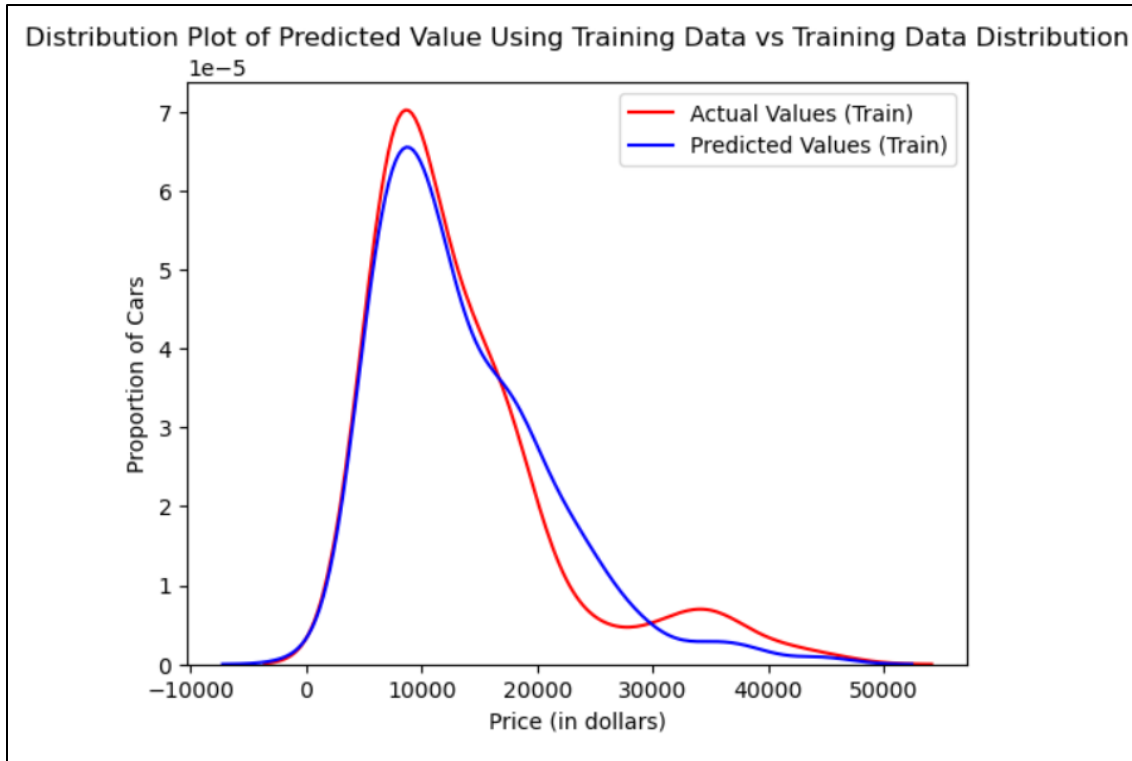


Figure 5.4: Distribution Plot of Predicted Values Using Training Data vs. Training Data Distribution

This figure illustrates the distribution of predicted values for the training dataset compared to the actual values. The model appears to perform well on the training data, as the predicted values align closely with the actual values.

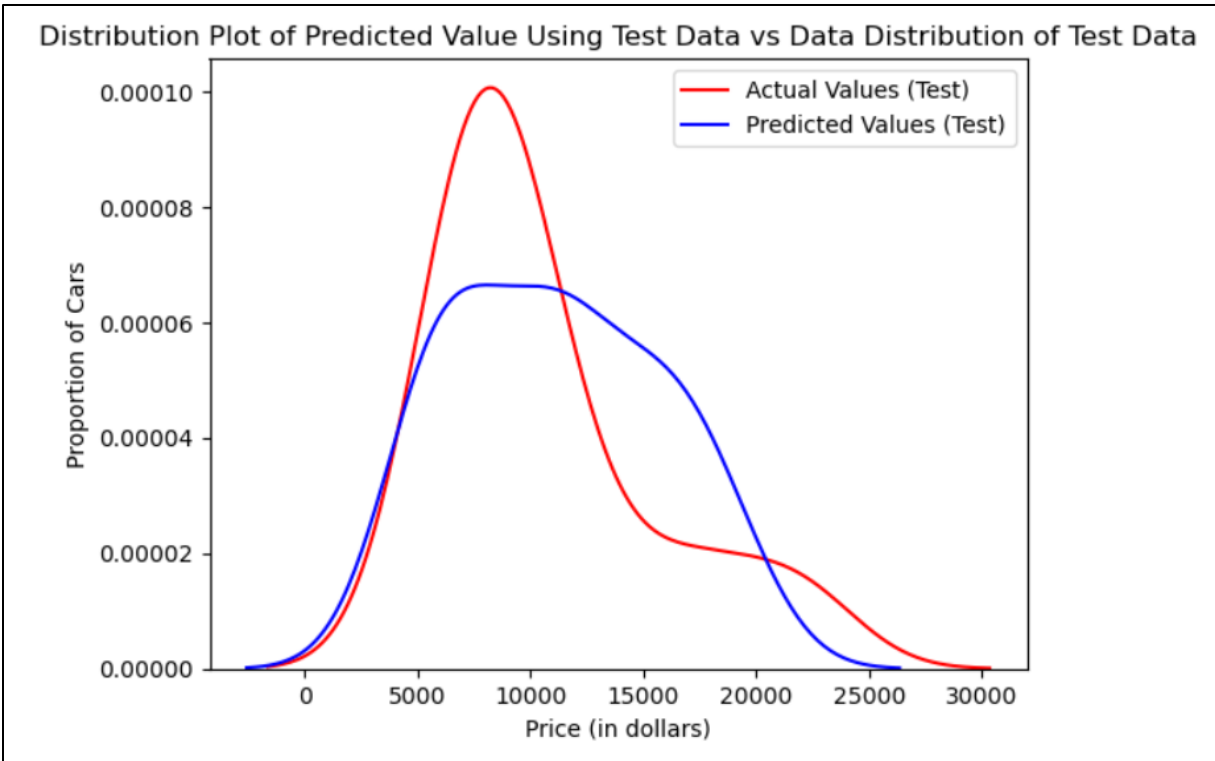


Figure 5.5: Distribution Plot of Predicted Values Using Test Data vs. Test Data Distribution

In contrast, Figure 5.5 shows the distribution of predicted values for the test dataset against the actual test values. The discrepancy between predicted and actual values is more pronounced, particularly in the range of \$5,000 to \$15,000. This indicates that the model's performance on the test data is not as robust as on the training data.

Comparing Figure 5.4 and Figure 5.5 highlights the difference in model performance between training and testing datasets. The reduced accuracy on the test data suggests potential overfitting, where the model has learned the training data too well but struggles to generalize to new data. This prompts further investigation into model refinement or the inclusion of additional features to improve generalizability.

- **Overfitting:** Overfitting happens when a model captures the noise in the data rather than the true underlying patterns. As a result, the model performs well on the training set but fails to generalize when applied to new, unseen data. This poor performance on the test set indicates that the model is not effectively capturing the underlying process but is instead

modeling random variations in the data. To illustrate this concept, we will create a degree 5 polynomial model and analyze its performance.

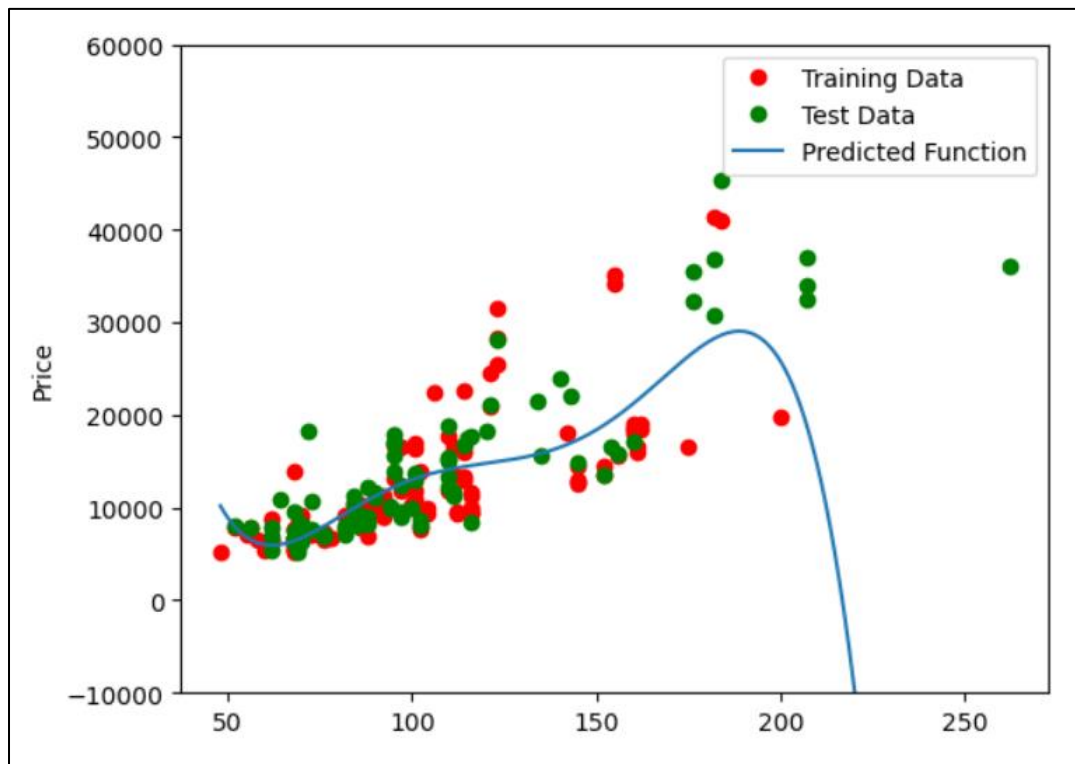


Figure 5.6: Polynomial Regression Model Visualization

This figure illustrates the polynomial regression model with red dots representing the training data, green dots showing the test data, and the blue line indicating the model's predictions. The plot helps us assess the accuracy of the model and whether it captures the underlying patterns in the data. The polynomial regression model appears to fit the training data well. However, the divergence of the blue line from the data points, particularly around 200 horsepower, suggests that the model may overfit the training data and not generalize effectively to the test data. This indicates potential overfitting issues.

- Model Performance:

- **Training Data R^2 :** The R^2 value for the training data is 0.5567. This indicates that approximately 55.67% of the variance in the training data is explained by the model.

- **Test Data R^2 :** The R^2 value for the test data is -29.87. A negative R^2 value suggests that the model performs worse than a simple mean-based prediction. This is a clear sign of overfitting, where the model captures noise rather than the underlying pattern.

```
R2 of the training data: 0.5568527852117562
R2 of the test data: -29.815108072386607
```

Figure 5.7: Polynomial Regression Model

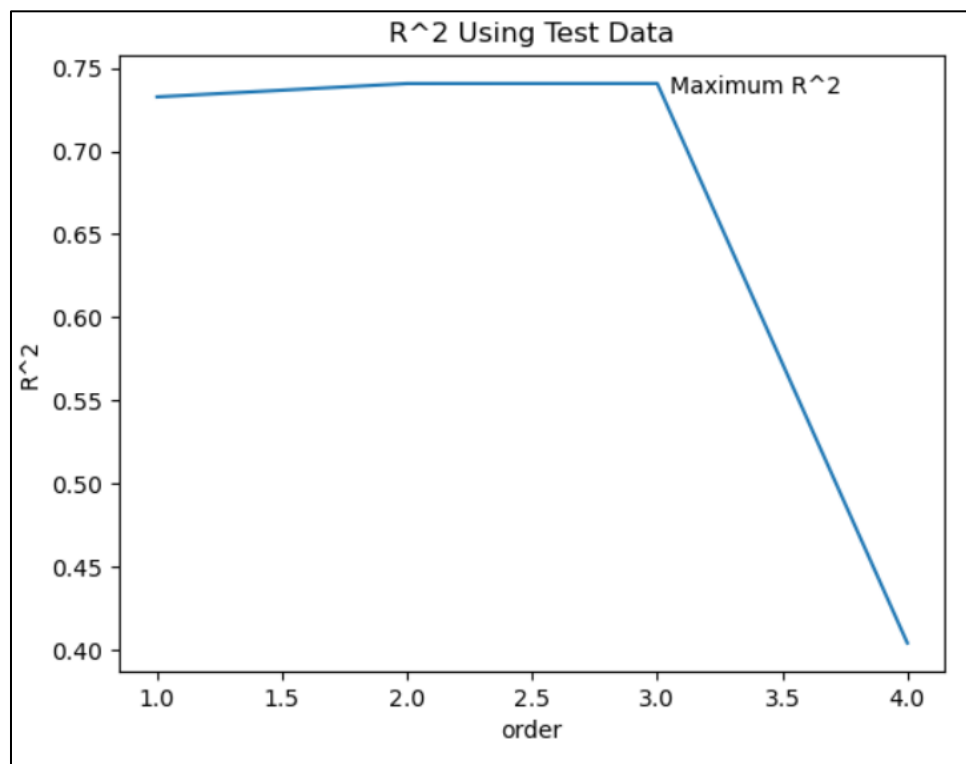


Figure 5.8: R^2 Scores for Polynomial Orders

As polynomial order increases, the R^2 score initially improves, indicating better fit to the test data. However, beyond a certain point, the R^2 score drops, suggesting potential overfitting. This drop occurs because the model starts to capture noise rather than the true underlying pattern, reducing its ability to generalize to new data.

In the following visualization, the plot displays the training data, test data, and the polynomial regression model's predictions. The training data is marked in red, the test data

in green, and the predicted polynomial fit is represented by the blue line. This setup illustrates how well the model captures the relationship between the features and the target variable. Additionally, the interactive feature in the code allows users to adjust polynomial degrees and test data sizes dynamically, offering real-time insights into how these changes affect model performance. This interactive component is available in the repository and provides a hands-on way to explore different model configurations.

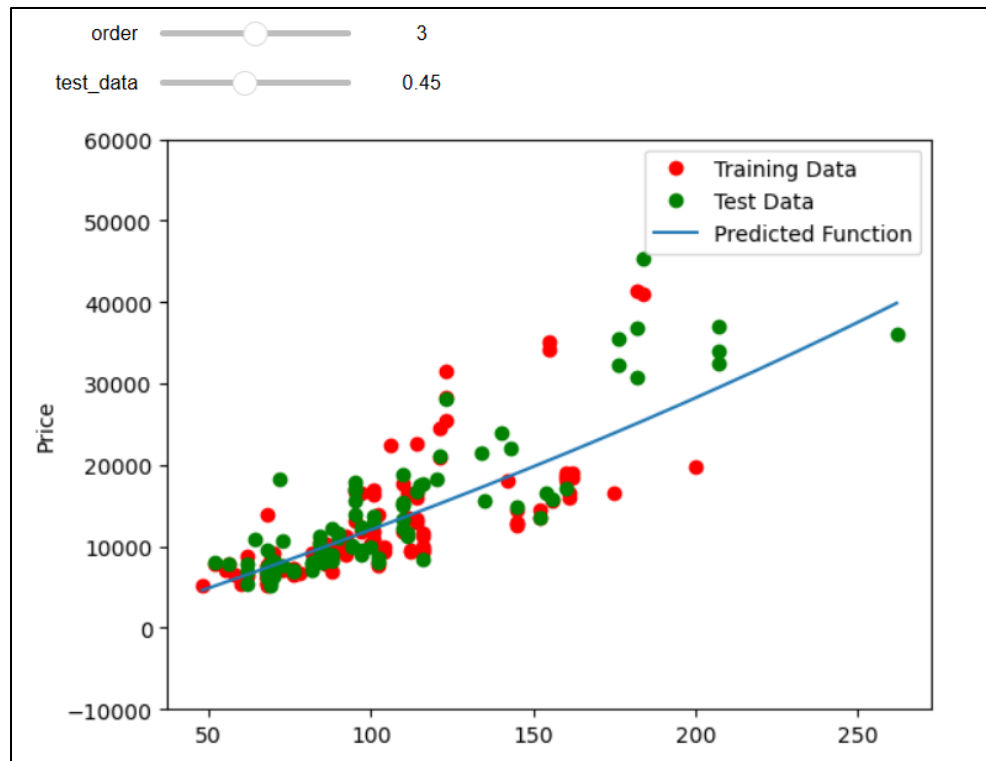


Figure 5.9: Polynomial Regression Fit Visualization

For an order of 3 and a test size of 0.45, the model demonstrates a well-fitted predicted function, as shown in the figure. The plot illustrates the relationship between horsepower and price, with the polynomial regression capturing the data points effectively. The predicted function aligns closely with the actual values, indicating that the model is performing well in predicting car prices based on horsepower. This visualization provides a clear representation of how the polynomial regression model adapts to the complexity of the data, highlighting its capability to model the non-linear relationship between the features and the target variable.

- **Polynomial Regression with Multiple Features:** The polynomial regression model was expanded to include multiple features such as 'horsepower,' 'curb-weight,' 'engine-size,' and 'highway-mpg.' After transforming the data into polynomial features of degree two, the model was trained and tested to predict car prices. The distribution plot below compares the predicted values against the actual test data. It reveals that the model's predictions are generally aligned with the actual values, though some discrepancies are observed, particularly in the lower and higher price ranges.

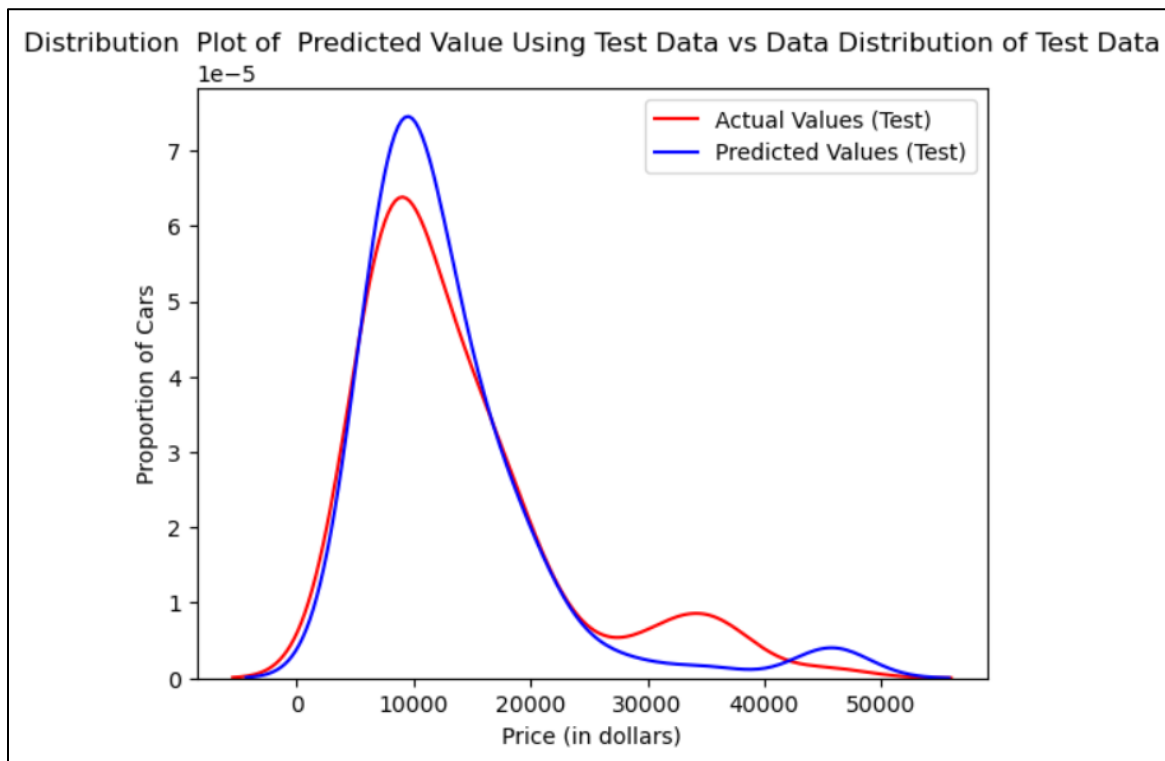


Figure 5.10: Visualization of Predicted vs Actual Car Prices

The distribution plot reveals two key regions where the model's predicted prices deviate from the actual values. In the price range of around \$10,000, the predicted values tend to be higher than the actual prices. Conversely, in the \$30,000 to \$40,000 range, the predicted values are lower than the actual prices. These discrepancies indicate that the model is less accurate in these specific price ranges.

5.3 Ridge Regression and Grid Search

- **Ridge Regression:** Ridge Regression is a technique that introduces a regularization parameter, alpha, to penalize large coefficients in a linear model. This helps prevent overfitting, especially when working with polynomial features. In this section, the effects of varying alpha on model performance are explored using a polynomial transformation of the data.

The data undergoes a degree-two polynomial transformation using the features “horsepower”, “curb-weight”, “engine-size”, “highway-mpg”, “normalized-losses”, and “symboling”. After transforming the data, a Ridge regression model is created with an initial alpha value of 1. The model is then trained and used to predict values on the test set. The first four predicted values are compared with the actual test values.

To optimize the model, a loop is used to iterate through a range of alpha values, adjusting the model's complexity and capturing the performance on both training and validation data. This process is visualized by plotting the R^2 scores for both the training and validation datasets as a function of alpha.

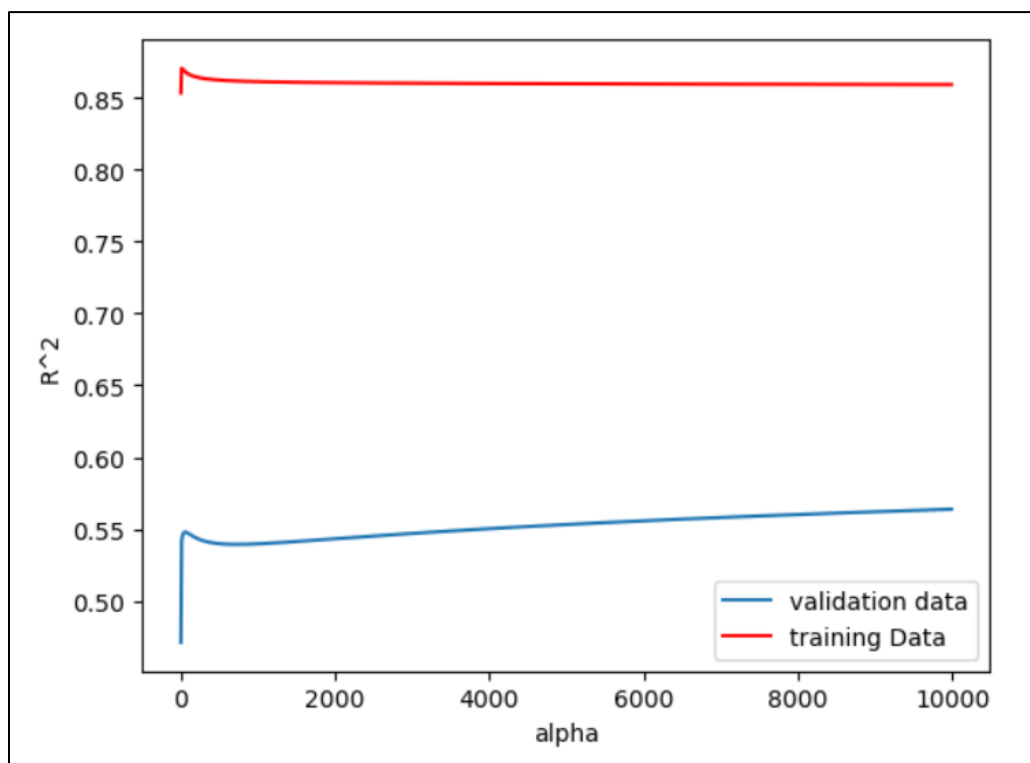


Figure 5.11: Ridge Regression R^2 Scores for Varying Alpha

The blue line represents the R^2 score for the validation data, while the red line represents the R^2 score for the training data. The x-axis indicates different alpha values. As alpha increases, the R^2 score decreases for the training data, indicating a drop in model performance. Conversely, the R^2 score for the validation data increases and eventually converges, suggesting that the model becomes more generalizable as regularization is applied. This figure illustrates how the Ridge Regression model's performance varies with different values of alpha, showcasing the trade-off between model complexity and generalization.

- **Grid Search:** In Ridge Regression, alpha is a crucial hyperparameter that needs to be fine-tuned to achieve the best model performance. To simplify this process, we use a technique called Grid Search, which systematically explores a range of alpha values to find the optimal one.

We start by defining a range of potential alpha values to test. Next, we create a Ridge Regression model and set up the Grid Search process, which will test each alpha value to determine which one results in the best performance.

Grid Search uses a method called cross-validation to ensure that the model is evaluated fairly. It divides the data into multiple subsets, trains the model on some of these subsets, and tests it on the remaining ones. This helps in avoiding overfitting and ensures that the selected alpha value generalizes well to unseen data.

After running Grid Search, we identify the best-performing alpha and create a Ridge Regression model with this optimal value. We then evaluate this model on the test data to see how well it performs. In our case, the best model achieved a score of approximately 0.77, indicating its effectiveness in predicting car prices based on the features provided.

The best model, optimized using Grid Search, achieved a R^2 score of 0.7723 on the test data. This R^2 score indicates how well the model predicts the car prices on unseen test data. A higher R^2 score signifies a better fit of the model to the test data, with 1 being a perfect fit.

Figure 12: Ridge Regression Model Performance with Optimal Alpha

Conclusions

This project on used car price prediction serves as a practical exploration of data science techniques using Python, focusing on data collection, cleaning, exploration, and model development to estimate car prices based on various features.

- Key Findings:

1. Data Quality and Preparation:

- Significant effort was required to clean the data, including handling missing values, standardizing and normalizing variables, and encoding categorical data. These steps were crucial for ensuring the accuracy and reliability of the models.

2. Impact of Continuous Variables:

- Continuous variables like engine size and curb weight showed strong positive correlations with car prices, indicating that larger, heavier vehicles generally have higher prices.
- In contrast, variables like highway-mpg had a negative correlation, suggesting that cars with better fuel efficiency on highways tend to be priced lower.

3. Role of Categorical Variables:

- Categorical variables such as drive-wheels and engine location also played a role in price prediction. The analysis showed that certain configurations, like rear-wheel drive, are associated with higher prices.

4. Model Selection and Evaluation:

- The Multiple Linear Regression model was the most effective in capturing the complexity of the data, but the exploration of Polynomial Regression highlighted the potential for capturing non-linear relationships.
- Regularization techniques like Ridge Regression were necessary to mitigate overfitting, particularly with more complex models.

5. Evaluation Metrics:

- The project employed R-squared and Mean Squared Error (MSE) to evaluate model performance. While the MLR model showed strong results, the evaluation process also revealed areas where model predictions could be further improved.

6. Challenges with Generalization:

- The project highlighted the challenges of ensuring that models generalize well to new data. Despite efforts to address overfitting, the model's performance varied between training and test datasets, indicating the need for further refinement.

- Final Thoughts:

This project provided valuable experience in handling and analyzing complex datasets, building predictive models, and refining them to achieve more accurate predictions. It also emphasized the ongoing challenges in balancing model complexity with generalization, offering lessons for future work.