Car Dataset Analysis Project Report

Prepared by: Muhammad Talha

Date: April 24, 2025

1. Introduction

This report presents a comprehensive analysis of a car dataset, focusing on key steps such as data cleaning, exploratory data analysis (EDA), and data visualization to uncover trends and insights. The goal is to gain a better understanding of the dataset and support informed decision-making using Python-based data analysis techniques.

# 2. Data Cleaning & Preprocessing

This section includes the steps taken to clean the dataset, handle missing values, and prepare the data for further analysis. The process involves checking for null values, encoding categorical variables, and converting data types as necessary.

# 3. Exploratory Data Analysis (EDA)

Exploratory data analysis helps identify patterns, relationships, and anomalies in the dataset. Summary statistics, distribution analysis, and correlation matrices are typically used to understand variable behavior.

# 4. Data Visualization

Visualization techniques are applied to graphically represent the dataset and highlight important features. This includes bar plots, histograms, scatter plots, and heatmaps, which are effective in revealing trends and patterns.

# 5. Conclusions & Recommendations

Based on the analysis and visual insights, this section summarizes key findings from the car dataset and provides actionable recommendations. This may include observations on pricing trends, performance characteristics, and other valuable factors.

# Appendix: Code and Output

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv('./car\_data.csv')

# Data Analysis Methods for Car Dataset

# 1. \*\*Highest Price\*\*: Finds the highest price in the dataset.

highest\_price = df['price'].max()

print(highest\_price)

45400.0

## 2. \*\*Lowest Price\*\*: Finds the lowest price in the dataset.

lowest\_price = df['price'].min()

print(lowest\_price)

5118.0

### 3. \*\*Average Price\*\*: Calculates the average price of all cars.

average\_price = df['price'].mean()

print(average\_price)

13276.710570731706

### 4. \*\*Car with Highest Horsepower\*\*: Finds the car with the highest horsepower.

car\_highest\_hp = df.loc[df['horsepower'].idxmax()]

print(car\_highest\_hp)

car\_ID 130

symboling 1

CarName porsche cayenne

fueltype gas

aspiration std

doornumber two

carbody hatchback

drivewheel rwd

enginelocation front

wheelbase 98.4

carlength 175.7

carwidth 72.3

carheight 50.5

curbweight 3366

enginetype dohcv

cylindernumber eight

enginesize 203

fuelsystem mpfi

boreratio 3.94

stroke 3.11

compressionratio 10.0

horsepower 288

peakrpm 5750

citympg 17

highwaympg 28

price 31400.5

Name: 129, dtype: object

### 5. \*\*Car with Lowest Horsepower\*\*: Finds the car with the lowest horsepower.

car\_lowest\_hp = df.loc[df['horsepower'].idxmin()]

print(car\_lowest\_hp)

car\_ID 19

symboling 2

CarName chevrolet impala

fueltype gas

aspiration std

doornumber two

carbody hatchback

drivewheel fwd

enginelocation front

wheelbase 88.4

carlength 141.1

carwidth 60.3

carheight 53.2

curbweight 1488

enginetype l

cylindernumber three

enginesize 61

fuelsystem 2bbl

boreratio 2.91

stroke 3.03

compressionratio 9.5

horsepower 48

peakrpm 5100

citympg 47

highwaympg 53

price 5151.0

Name: 18, dtype: object

### 6. \*\*Average Horsepower\*\*: Calculates the average horsepower across all cars.

average\_hp = df['horsepower'].mean()

print(average\_hp)

104.1170731707317

### 7. \*\*Number of Unique `carbody` Types\*\*: Counts the number of unique `carbody` types in the dataset.

unique\_carbody\_types = df['carbody'].nunique()

print(unique\_carbody\_types)

5

### 8. \*\*Number of Cars by `fueltype`\*\*: Counts how many cars there are for each fuel type (e.g., gas, diesel).

cars\_by\_fueltype = df['fueltype'].value\_counts()

print(cars\_by\_fueltype)

fueltype

gas 185

diesel 20

Name: count, dtype: int64

### 9. \*\*Median Price\*\*: Finds the median value of `price` (middle value when sorted).

median\_price = df['price'].median()

print(median\_price)

10295.0

### 10. \*\*Standard Deviation of Price\*\*: Measures the spread of `price` values.

price\_std\_dev = df['price'].std()

print(price\_std\_dev)

7988.85233174315

### 11. \*\*Correlation Between `price` and `horsepower`\*\*: Shows the correlation between price and horsepower (how closely they are related).

price\_hp\_correlation = df[['price', 'horsepower']].corr().iloc[0, 1]

print(price\_hp\_correlation)

0.8081388225362214

### 12. \*\*Count of Cars by `carbody`\*\*: Counts how many cars there are of each `carbody` type.

count\_by\_carbody = df['carbody'].value\_counts()

print(count\_by\_carbody)

carbody

sedan 96

hatchback 70

wagon 25

hardtop 8

convertible 6

Name: count, dtype: int64

### 13. \*\*Sum of All Engine Sizes\*\*: Calculates the sum of the `enginesize` column.

sum\_enginesize = df['enginesize'].sum()

print(sum\_enginesize)

26016

### 14. \*\*Unique Values in `aspiration`\*\*: Finds the unique values in the `aspiration` column (e.g., std, turbo).

unique\_aspiration = df['aspiration'].unique()

print(unique\_aspiration)

['std' 'turbo']

### 15. \*\*Count of Cars with Turbo Aspiration\*\*: Counts how many cars have a `turbo` aspiration.

count\_turbo = df[df['aspiration'] == 'turbo'].shape[0]

print(count\_turbo)

37

### 16. \*\*Car with Highest Engine Size\*\*: Finds the car with the highest engine size.

car\_highest\_enginesize = df.loc[df['enginesize'].idxmax()]

print(car\_highest\_enginesize)

car\_ID 50

symboling 0

CarName jaguar xk

fueltype gas

aspiration std

doornumber two

carbody sedan

drivewheel rwd

enginelocation front

wheelbase 102.0

carlength 191.7

carwidth 70.6

carheight 47.8

curbweight 3950

enginetype ohcv

cylindernumber twelve

enginesize 326

fuelsystem mpfi

boreratio 3.54

stroke 2.76

compressionratio 11.5

horsepower 262

peakrpm 5000

citympg 13

highwaympg 17

price 36000.0

Name: 49, dtype: object

### 17. \*\*Car with Lowest Engine Size\*\*: Finds the car with the lowest engine size.

car\_lowest\_enginesize = df.loc[df['enginesize'].idxmin()]

print(car\_lowest\_enginesize)

car\_ID 19

symboling 2

CarName chevrolet impala

fueltype gas

aspiration std

doornumber two

carbody hatchback

drivewheel fwd

enginelocation front

wheelbase 88.4

carlength 141.1

carwidth 60.3

carheight 53.2

curbweight 1488

enginetype l

cylindernumber three

enginesize 61

fuelsystem 2bbl

boreratio 2.91

stroke 3.03

compressionratio 9.5

horsepower 48

peakrpm 5100

citympg 47

highwaympg 53

price 5151.0

Name: 18, dtype: object

### 18. \*\*Number of Cars with High Mileage\*\*: Counts the number of cars with high `citympg` (e.g., more than 30 miles per gallon).

high\_mileage\_cars = df[df['citympg'] > 30].shape[0]

print(high\_mileage\_cars)

49

### 19. \*\*Car with Maximum Wheelbase\*\*: Finds the car with the maximum `wheelbase`.

car\_max\_wheelbase = df.loc[df['wheelbase'].idxmax()]

print(car\_max\_wheelbase)

car\_ID 74

symboling 0

CarName buick century special

fueltype gas

aspiration std

doornumber four

carbody sedan

drivewheel rwd

enginelocation front

wheelbase 120.9

carlength 208.1

carwidth 71.7

carheight 56.7

curbweight 3900

enginetype ohcv

cylindernumber eight

enginesize 308

fuelsystem mpfi

boreratio 3.8

stroke 3.35

compressionratio 8.0

horsepower 184

peakrpm 4500

citympg 14

highwaympg 16

price 40960.0

Name: 73, dtype: object

### 20. \*\*Range of `price` (Max - Min)\*\*: Calculates the range of car prices (i.e., difference between max and min price).

price\_range = df['price'].max() - df['price'].min()

print(price\_range)

40282.0

### 21. \*\*Count of Cars by `drivewheel` Type\*\*: Counts how many cars there are of each `drivewheel` type (e.g., fwd, rwd).

count\_by\_drivewheel = df['drivewheel'].value\_counts()

print(count\_by\_drivewheel)

drivewheel

fwd 120

rwd 76

4wd 9

Name: count, dtype: int64

### 22. \*\*Top 5 Expensive Cars\*\*: Lists the top 5 cars with the highest prices.

top\_5\_expensive\_cars = df.nlargest(5, 'price')[['CarName', 'price']]

print(top\_5\_expensive\_cars)

CarName price

74 buick regal sport coupe (turbo) 45400.0

16 bmw x5 41315.0

73 buick century special 40960.0

128 porsche boxter 37028.0

17 bmw x3 36880.0

### 23. \*\*Top 5 Cheapest Cars\*\*: Lists the top 5 cars with the lowest prices.

top\_5\_cheapest\_cars = df.nsmallest(5, 'price')[['CarName', 'price']]

print(top\_5\_cheapest\_cars)

CarName price

138 subaru 5118.0

18 chevrolet impala 5151.0

50 maxda rx3 5195.0

150 toyota corona mark ii 5348.0

76 mitsubishi mirage 5389.0

### 24. \*\*Count of Cars with Missing Values\*\*: Counts how many rows have missing values in any column.

missing\_values = df.isnull().sum().sum()

print(missing\_values)

0

# Data Visualization Methods for Car Dataset

### 1. \*\*Box Plot for Price by `carbody`\*\*:

plt.figure(figsize=(10, 6))

sns.boxplot(x='carbody', y='price', data=df)

plt.title('Price Distribution by Car Body Type')

plt.show()

![png](output\_54\_0.png)

### 2. \*\*Histogram for `horsepower` Distribution\*\*:

plt.figure(figsize=(10, 6))

sns.histplot(df['horsepower'], bins=20, kde=True)

plt.title('Horsepower Distribution')

plt.show()

![png](output\_56\_0.png)

### 3. \*\*Scatter Plot for `horsepower` vs `price`\*\*:

plt.figure(figsize=(10, 6))

sns.scatterplot(x='horsepower', y='price', data=df)

plt.title('Horsepower vs Price')

plt.show()

![png](output\_58\_0.png)

### 4 \*\*Pairplot for Multiple Variables\*\*:

sns.pairplot(df[['price', 'horsepower', 'enginesize', 'curbweight']])

plt.title('Pairplot of Price, Horsepower, Engine Size, and Curb Weight')

plt.show()

![png](output\_60\_0.png)

### 5. \*\*Correlation Heatmap for Numerical Columns\*\*:

plt.figure(figsize=(10, 8))

sns.heatmap(df[['price', 'horsepower', 'enginesize', 'curbweight']].corr(), annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap of Numerical Features')

plt.show()

![png](output\_62\_0.png)

### 6 . \*\*Bar Chart for Number of Cars by `fueltype`\*\*:

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

sns.countplot(x='fueltype', data=df)

plt.title('Number of Cars by Fuel Type')

plt.show()

![png](output\_64\_0.png)

### 7 . \*\*Count Plot for `carbody`\*\*:

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

sns.countplot(x='carbody', data=df)

plt.title('Number of Cars by Car Body Type')

plt.show()

![png](output\_66\_0.png)

### 8 \*\*Scatter Plot for `curbweight` vs `enginesize`\*\*:

plt.figure(figsize=(10, 6))

sns.scatterplot(x='curbweight', y='enginesize', data=df)

plt.title('Curb Weight vs Engine Size')

plt.show()

![png](output\_68\_0.png)