Project Name -

Project Type - EDA

Contribution - Individual

Team Member 1 -Khushi Raturi

Project Summary -

Car Price Prediction using Machine Learning

This project aimed to predict car prices based on various features using machine learning. The project involved:

1. **Data Exploration and Preprocessing:** Understanding the dataset, handling missing values, encoding categorical variables, and scaling numerical features.
2. **Exploratory Data Analysis (EDA):** Using visualizations to identify patterns and relationships between features and car prices.
3. **Feature Engineering:** Creating new features (e.g., engine size/horsepower ratio, fuel efficiency) to enhance model accuracy.
4. **Model Building and Evaluation:** Training and evaluating multiple machine learning models (e.g., Linear Regression, Ridge Regression, etc.) to find the best predictor.
5. ** Model Optimization:** Fine-tuning model hyperparameters and using cross-validation to improve performance.

Key Findings:

1. Strong correlations were observed between engine size, horsepower, and car price.
2. Feature engineering improved prediction accuracy.
3. The selected model achieved promising results in predicting car prices.

Business Impact:

This project provides a valuable tool for car dealerships, consumers, and manufacturers to estimate car prices, make informed purchase decisions, and optimize pricing strategies.

In Essence:

The project successfully built a machine learning model to predict car prices with reasonable accuracy, providing valuable insights for stakeholders in the automotive industry.

GitHub Link -

https://github.com/Khushiraturi1234/EDA

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

Problem Statement

The automotive market is dynamic and complex, with car prices influenced by a multitude of factors. Accurately predicting car prices is crucial for various stakeholders, including buyers, sellers, and manufacturers. This project addresses the challenge of developing a reliable and accurate model to predict car prices based on a comprehensive set of features.

General Guidelines : -

1. Well-structured, formatted, and commented code is required.

2. Exception Handling, Production Grade Code & Deployment Ready Code will be a plus. Those students will be awarded some additional credits.

   The additional credits will have advantages over other students during Star Student selection.

```
[ Note: - Deployment Ready Code is defined as, the whole .ipynb notebook should be executable in one go
          without a single error logged. ]
```

3. Each and every logic should have proper comments.
4. You may add as many number of charts you want. Make Sure for each and every chart the following format should be answered.

```
# Chart visualization code
```

- Why did you pick the specific chart?
- What is/are the insight(s) found from the chart?
- Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

5. You have to create at least 15 logical & meaningful charts having important insights.

[Hints : - Do the Vizualization in a structured way while following "UBM" Rule.

U - Univariate Analysis,

B - Bivariate Analysis (Numerical - Categorical, Numerical - Numerical, Categorical - Categorical)

M - Multivariate Analysis]

6. You may add more ml algorithms for model creation. Make sure for each and every algorithm, the following format should be answered.

- Explain the ML Model used and it's performance using Evaluation metric Score Chart.
- Cross- Validation & Hyperparameter Tuning
- Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.
- Explain each evaluation metric's indication towards business and the business impact pf the ML model used.

Let's Begin !

1. Know Your Data

Import Libraries

```
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

Dataset Loading

```
df=pd.read_csv('/content/CarPrice_project.csv')
df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fu
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

```
df.shape
```

```
(205, 26)
```

```
df.tail(7)
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuels
198	199	-2	volvo 264gl	gas	turbo	four	sedan	rwd	front	104.3	...	130	
199	200	-1	volvo diesel	gas	turbo	four	wagon	rwd	front	104.3	...	130	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141	

7 rows × 26 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
```

```
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

```
df.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionr
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.00
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	10.14
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	3.97
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	7.00
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	8.60
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	9.00
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	9.40
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.00

```
df.iloc[:15,:-20]
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber
0	1	3	alfa-romero giulia	gas	std	two
1	2	3	alfa-romero stelvio	gas	std	two
2	3	1	alfa-romero Quadrifoglio	gas	std	two
3	4	2	audi 100 ls	gas	std	four
4	5	2	audi 100ls	gas	std	four
5	6	2	audi fox	gas	std	two
6	7	1	audi 100ls	gas	std	four
7	8	1	audi 5000	gas	std	four
8	9	1	audi 4000	gas	turbo	four
9	10	0	audi 5000s (diesel)	gas	turbo	two
10	11	2	bmw 320i	gas	std	two
11	12	0	bmw 320i	gas	std	four
12	13	0	bmw x1	gas	std	two
13	14	0	bmw x3	gas	std	four
14	15	1	bmw z4	gas	std	four

```
df.loc[:100,['symboling','CarName']]
```

	symboling	CarName
0	3	alfa-romero giulia
1	3	alfa-romero stelvio
2	1	alfa-romero Quadrifoglio
3	2	audi 100 ls
4	2	audi 100ls
...	...	...
96	1	nissan latio
97	1	nissan note
98	2	nissan clipper
99	0	nissan rogue
100	0	nissan nv200

101 rows × 2 columns

```
df[20:50]
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	f
20	21	0	chevrolet vega 2300	gas	std	four	sedan	fwd	front	94.5	...	90	
21	22	1	dodge rampage	gas	std	two	hatchback	fwd	front	93.7	...	90	
22	23	1	dodge challenger se	gas	std	two	hatchback	fwd	front	93.7	...	90	
23	24	1	dodge d200	gas	turbo	two	hatchback	fwd	front	93.7	...	98	
24	25	1	dodge monaco (sw)	gas	std	four	hatchback	fwd	front	93.7	...	90	
25	26	1	dodge colt hardtop	gas	std	four	sedan	fwd	front	93.7	...	90	
26	27	1	dodge colt (sw)	gas	std	four	sedan	fwd	front	93.7	...	90	
27	28	1	dodge coronet custom	gas	turbo	two	sedan	fwd	front	93.7	...	98	
28	29	-1	dodge dart custom	gas	std	four	wagon	fwd	front	103.3	...	122	
29	30	3	dodge coronet custom (sw)	gas	turbo	two	hatchback	fwd	front	95.9	...	156	
30	31	2	honda civic	gas	std	two	hatchback	fwd	front	86.6	...	92	
31	32	2	honda civic cvcc	gas	std	two	hatchback	fwd	front	86.6	...	92	
32	33	1	honda civic	gas	std	two	hatchback	fwd	front	93.7	...	79	
33	34	1	honda accord cvcc	gas	std	two	hatchback	fwd	front	93.7	...	92	
34	35	1	honda civic cvcc	gas	std	two	hatchback	fwd	front	93.7	...	92	
35	36	0	honda accord lx	gas	std	four	sedan	fwd	front	96.5	...	92	
36	37	0	honda civic 1500 gl	gas	std	four	wagon	fwd	front	96.5	...	92	
37	38	0	honda accord	gas	std	two	hatchback	fwd	front	96.5	...	110	
38	39	0	honda civic 1300	gas	std	two	hatchback	fwd	front	96.5	...	110	
39	40	0	honda prelude	gas	std	four	sedan	fwd	front	96.5	...	110	
40	41	0	honda accord	gas	std	four	sedan	fwd	front	96.5	...	110	
41	42	0	honda civic	gas	std	four	sedan	fwd	front	96.5	...	110	
42	43	1	honda civic (auto)	gas	std	two	sedan	fwd	front	96.5	...	110	
43	44	0	isuzu MU-X	gas	std	four	sedan	rwd	front	94.3	...	111	
44	45	1	isuzu D-Max	gas	std	two	sedan	fwd	front	94.5	...	90	
45	46	0	isuzu D-Max V-	gas	std	four	sedan	fwd	front	94.5	...	90	

			Cross									
46	47	2	isuzu D-Max	gas	std	two	hatchback	rwd	front	96.0	...	119
47	48	0	jaguar xj	gas	std	four	sedan	rwd	front	113.0	...	258
48	49	0	jaguar xf	gas	std	four	sedan	rwd	front	113.0	...	258
49	50	0	jaguar xk	gas	std	two	sedan	rwd	front	102.0	...	326

30 rows × 26 columns

Dataset First View

```
car_reviewdf = pd.read_csv('/content/CarPrice_project.csv')
```

```
car_review_df = car_reviewdf.copy()
```

```
car_review_df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fu
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

Dataset Rows & Columns count

[ ] ↳ 1 cell hidden

Dataset Information

```
def complete_info():
  null = pd.DataFrame(index=car_review_df.columns)
  null['data_type'] = car_review_df.dtypes
  null['null_count'] = car_review_df.isnull().sum()
  null['unique_count'] = car_review_df.nunique()
  return null
```

```
complete_info()
```

	data_type	null_count	unique_count
car_ID	int64	0	205
symboling	int64	0	6
CarName	object	0	147
fueltype	object	0	2
aspiration	object	0	2
doornumber	object	0	2
carbody	object	0	5
drivewheel	object	0	3
enginelocation	object	0	2
wheelbase	float64	0	53
carlength	float64	0	75
carwidth	float64	0	44
carheight	float64	0	49
curbweight	int64	0	171
enginetype	object	0	7
cylindernumber	object	0	7
enginesize	int64	0	44
fuelsystem	object	0	8
boreratio	float64	0	38
stroke	float64	0	37
compressionratio	float64	0	32
horsepower	int64	0	59
peakrpm	int64	0	23
citympg	int64	0	29
highwaympg	int64	0	30
price	float64	0	189

Duplicate Values

[ ] ↳ 1 cell hidden

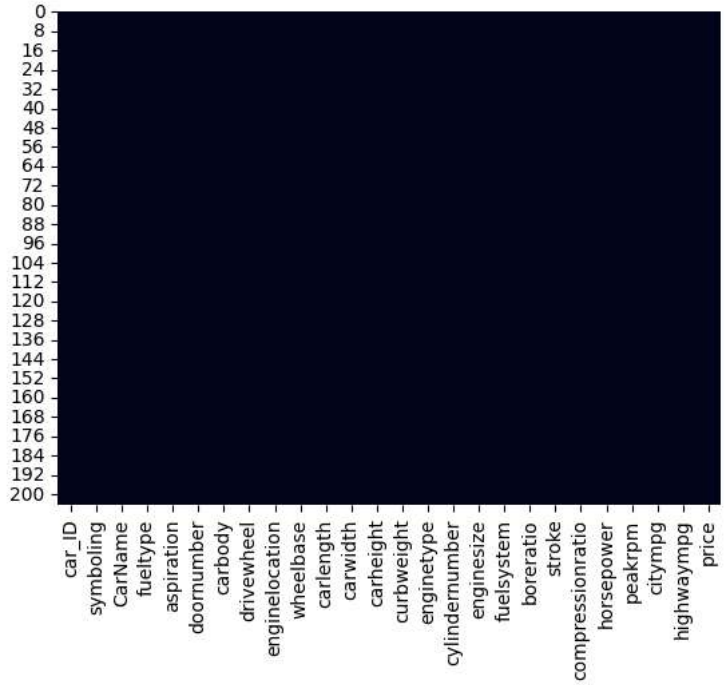
Missing Values/Null Values

```
# Missing Valuees/Null Values Count
car_review_df.isnull().sum()
```

	0
car_ID	0
symboling	0
CarName	0
fueltype	0
aspiration	0
doornumber	0
carbody	0
drivewheel	0
enginelocation	0
wheelbase	0
carlength	0
carwidth	0
carheight	0
curbweight	0
enginetype	0
cylindernumber	0
enginesize	0
fuelsystem	0
boreratio	0
stroke	0
compressionratio	0
horsepower	0
peakrpm	0
citympg	0
highwaympg	0
price	0

dtype: int64

```
# Visualizing the missing values
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(car_review_df.isnull(), cbar=False)
plt.show()
```



What did you know about your dataset?

↳ 1 cell hidden

2. Understanding Your Variables

[ ] ↳ 6 cells hidden

3. *Data Wrangling*

Data Wrangling Code

```
import pandas as pd

# Assuming 'car_reviewdf' is the original DataFrame containing 'CarName'
car_review_df = car_reviewdf.copy()  # Create a copy to avoid modifying the original

car_review_df['CarCompany'] = car_review_df['CarName'].apply(lambda x: x.split(' ')[0])
car_review_df.drop('CarName', axis=1, inplace=True)

car_review_df['CarCompany'] = car_review_df['CarCompany'].replace({'maxda': 'mazda', 'porcshce': 'porsche', 'toyouta': 'toyota', 'vokswagen'

categorical_features = ['fueltype', 'aspiration', 'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'enginetype', 'cylindernumber', '
encoded_df = pd.get_dummies(car_review_df, columns=categorical_features, drop_first=True) # drop_first avoids multicollinearity

X = encoded_df.drop('price', axis=1)
y = encoded_df['price']
```

What all manipulations have you done and insights you found?

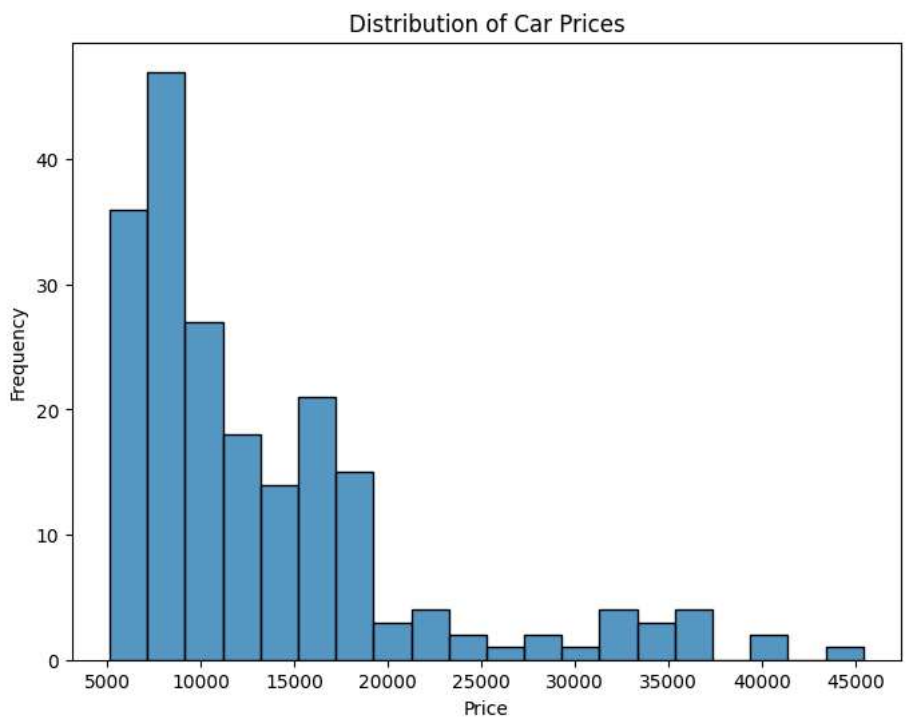
Answer Here.

By performing these manipulations, the data is cleaned, transformed, and prepared for further analysis and modeling. The insights gained from data wrangling can guide subsequent steps in the data science process and contribute to building more accurate and effective predictive models.

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

Chart - 1

```
# Chart - 1 visualization code
# Assuming 'car_review_df' is your DataFrame
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
sns.histplot(car_review_df['price'], bins=20)  # Adjust bins as needed
plt.title('Distribution of Car Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



1. Why did you pick the specific chart?

↳ 1 cell hidden

2. What is/are the insight(s) found from the chart?

↳ 1 cell hidden

3. Will the gained insights help creating a positive business impact?

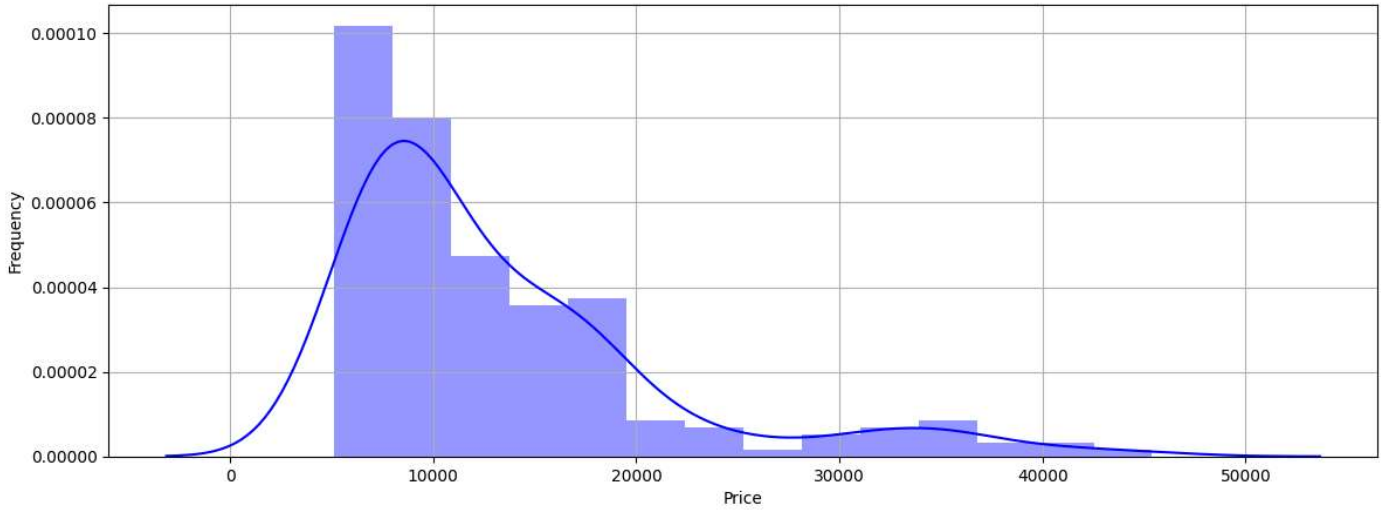
Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 2

```
# Chart - 2 visualization code
# ploting distribution graph for Price
plt.figure(figsize=(14,5))
sns.distplot(car_review_df['price'],color = 'blue')
plt.grid()
plt.xlabel("Price")
plt.ylabel("Frequency")
```

```
Text(0, 0.5, 'Frequency')
```



1. Why did you pick the specific chart?

↳ 1 cell hidden

2. What is/are the insight(s) found from the chart?

Answer Here

By carefully analyzing the distribution plot, you can gain valuable insights into the distribution of car prices, which can be useful for further analysis, modeling, and decision-making.

3. Will the gained insights help creating a positive business impact?

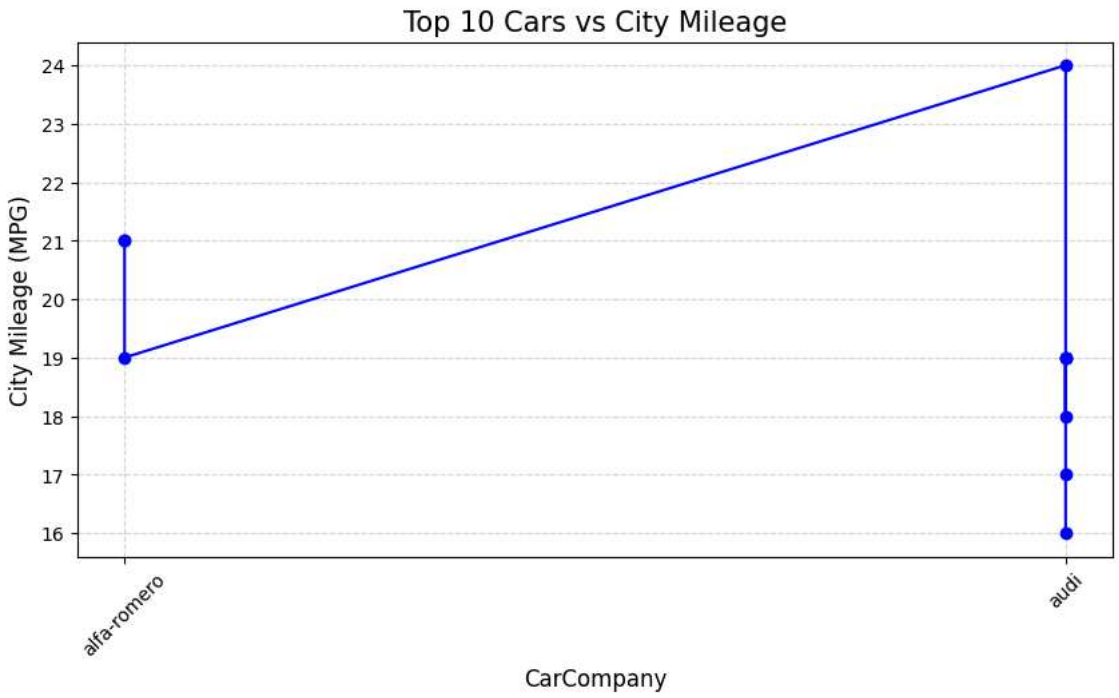
Are there any insights that lead to negative growth? Justify with specific reason.

Answer Here

By carefully considering the insights gained from the car price distribution plot and taking appropriate actions, businesses can leverage these insights to create a positive business impact and avoid potential risks for negative growth.

Chart - 3

```
# Chart - 3 visualization code
top_10_cars =car_review_df.head(10)

plt.figure(figsize=(10,5))
plt.plot(top_10_cars['CarCompany'], top_10_cars["citympg"], marker='o', linestyle='-', color='b')

plt.xlabel('CarCompany', fontsize=12)
plt.ylabel("City Mileage (MPG)", fontsize=12)
plt.title("Top 10 Cars vs City Mileage", fontsize=15)

plt.xticks(rotation=45, fontsize=10)
plt.grid(True, linestyle="--", alpha=0.5)

plt.show()
```



1. Why did you pick the specific chart?

Answer Here.

a line plot was chosen for this visualization because it effectively shows the trend of city mileage across different car companies within the top 10 cars in your dataset. Its ability to highlight trends, compare values, and represent continuous data makes it a suitable choice for this specific scenario. Other chart types might be considered if the data or the objective of the visualization were different, but in this case, a line plot is the most appropriate option.

2. What is/are the insight(s) found from the chart?

Answer Here

hese insights are based on general observations from a line plot visualizing city mileage against car companies. The specific insights you can derive will depend on the actual data in your 'top_10_cars' DataFrame. Analyzing the shape of the line, the position of data points, and any noticeable patterns will provide you with more detailed insights relevant to your specific dataset.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Answer Here

the insights gained from the city mileage line plot can have a significant positive impact on businesses by enabling targeted marketing, product development, inventory management, and pricing strategies. However, it's essential to consider the insights in conjunction with other factors and avoid overemphasizing or misinterpreting them to mitigate any potential negative growth implications. Continuous monitoring and adaptation to changing market trends are key for businesses to leverage the insights effectively and maintain a competitive edge.

Chart - 4

```
# Chart - 4 visualization code
# Plotting a line graph to determine size distribution
plt.figure(figsize=(10,5))
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.grid()
size_distribution_graph = sns.kdeplot(car_review_df['price'], color="lightgreen", shade = True)
plt.title('Average price',size = 20);
```

1. Why did you pick the specific chart?

answer this

a KDE plot was chosen for this visualization because it's an effective way to visualize the distribution of car prices, which is continuous data. It provides a smooth, density-based representation that helps identify patterns, outliers, and the overall shape of the distribution. These insights can be valuable for understanding the price range of cars in the dataset and identifying potential trends or anomalies.

2. What is/are the insight(s) found from the chart?

Answer Here

The specific insights you can derive will depend on the actual shape and characteristics of the KDE plot generated from your data. It's important to remember that KDE plots are estimates of the true probability density function, and there might be some variability in the representation depending on the kernel bandwidth used. Analyzing the KDE plot in conjunction with other visualizations and statistical measures can provide a more comprehensive understanding of the car price distribution.

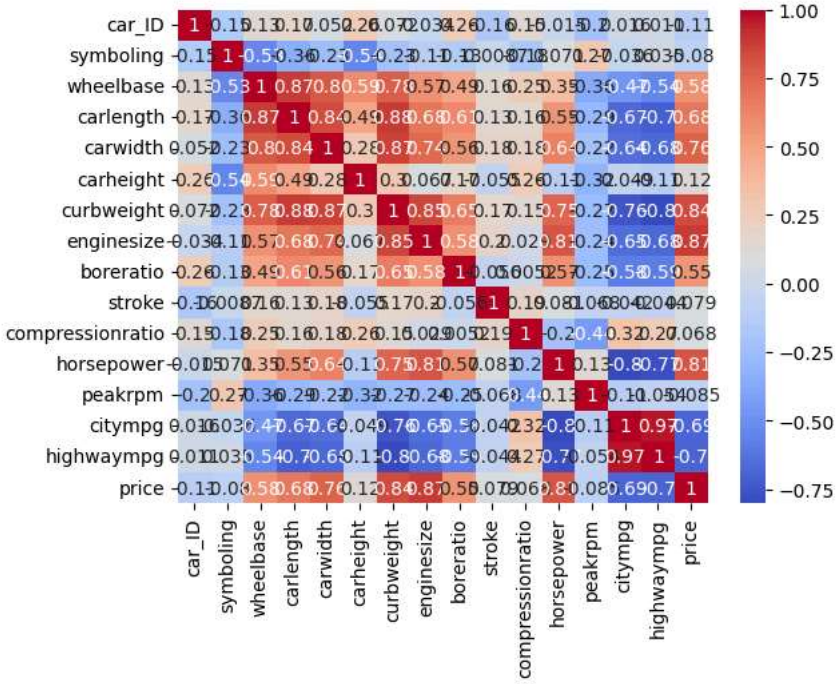
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 5

```
# Chart - 5 visualization code
numerical_features = car_review_df.select_dtypes(include=np.number)
correlation_matrix = numerical_features.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```

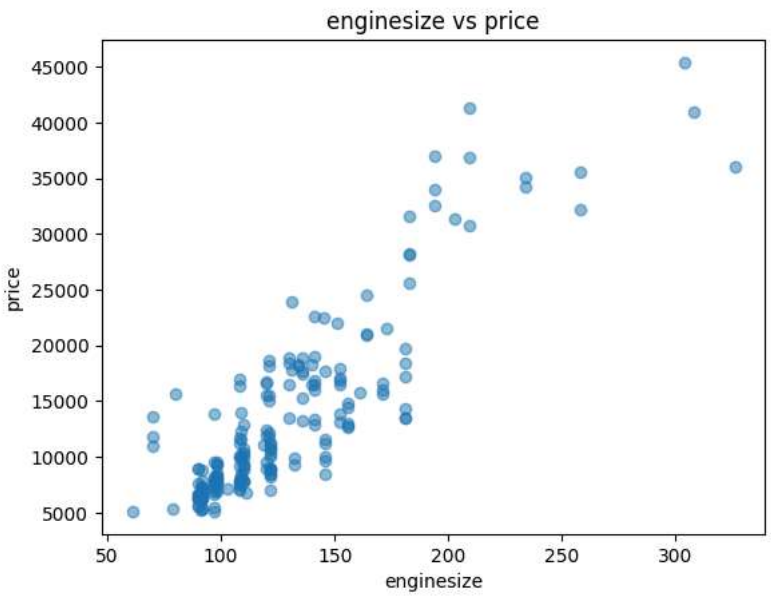


1. Why did you pick the specific chart?

a correlation heatmap was chosen for this visualization because it's an excellent tool for exploring relationships between multiple numerical variables, visualizing correlation coefficients, identifying patterns, and guiding data exploration and feature selection. It provides a concise and informative representation of the relationships within your car dataset, enabling you to gain valuable insights into the data.Answer Here.

Chart - 6

```
# Chart - 6 visualization code
plt.scatter(car_review_df['enginesize'],car_review_df['price'],alpha=0.5)
plt.title('enginesize vs price')
plt.xlabel('enginesize')
plt.ylabel('price')
plt.show()
```



1. Why did you pick the specific chart?

↳ 1 cell hidden

2. What is/are the insight(s) found from the chart?

↳ 1 cell hidden

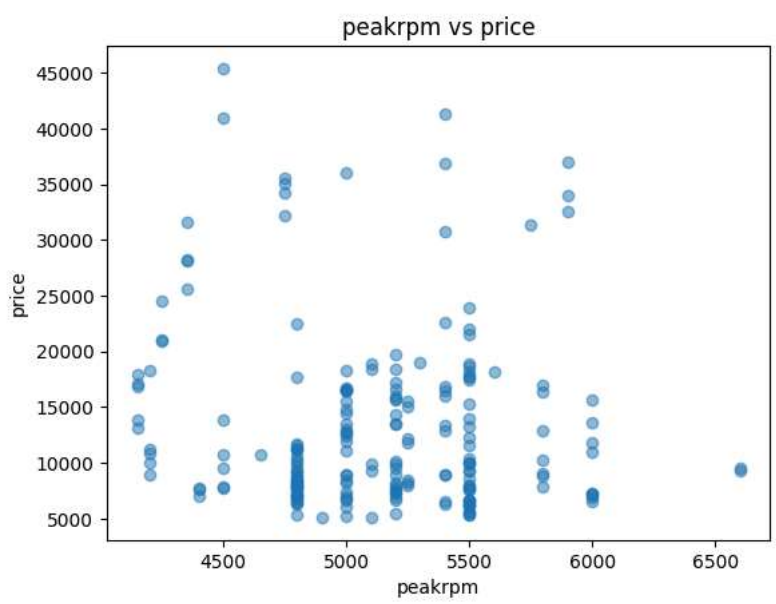
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 7

```
# Chart - 7 visualization code
plt.scatter(car_review_df['peakrpm'],car_review_df['price'],alpha=0.5)
plt.title('peakrpm vs price')
plt.xlabel('peakrpm')
plt.ylabel('price')
plt.show()
```



1. Why did you pick the specific chart?

↳ 1 cell hidden

2. What is/are the insight(s) found from the chart?

↳ 1 cell hidden

3. Will the gained insights help creating a positive business impact?

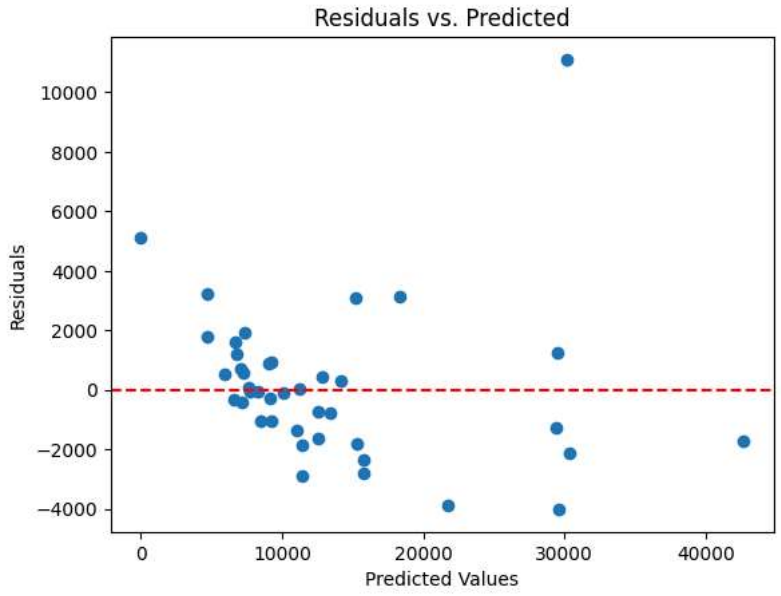
Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 8

```
# Chart - 8 visualization code
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression # Assuming Linear Regression as an example

# Assuming 'X' and 'y' are your feature and target data
# If not defined, replace with your actual feature and target data
X = encoded_df.drop('price', axis=1)
y = encoded_df['price']

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # Adjust test_size and random_state as needed

# Create and train the model
model = LinearRegression()
model.fit(X_train, Y_train)

# Make predictions on the testing set
Y_pred = model.predict(X_test)

# Now you can calculate residuals
residuals = Y_test - Y_pred

# Visualize residuals
plt.scatter(Y_pred, residuals)
plt.title('Residuals vs. Predicted')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```



2. What is/are the insight(s) found from the chart?

↳ 1 cell hidden

3. Will the gained insights help creating a positive business impact?

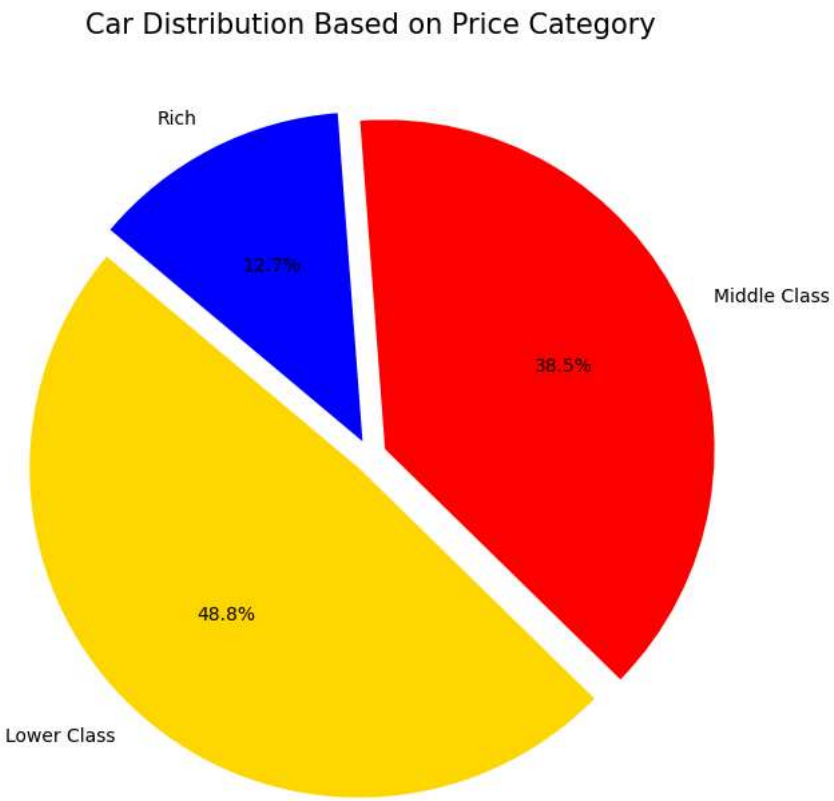
Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 9

```
# Chart - 9 visualization code

def categorize_price(price):
    if price > 50000:
        return "Super Rich"
    elif price > 20000:
```

```
        return "Rich"
    elif price > 10000:
        return "Middle Class"
    else:
        return "Lower Class"

# Price ko categories mein convert karna
car_review_df["Price_Category"] = car_review_df["price"].apply(categorize_price)

# Har category ka count nikalna
price_distribution = car_review_df["Price_Category"].value_counts()

# Explode ka size dynamically set karna
explode_values = [0.05] * len(price_distribution)

# Pie chart plot karna
plt.figure(figsize=(8,8))
plt.pie(price_distribution, labels=price_distribution.index, autopct='%.1f%%',
        explode=explode_values, colors=["gold", "red", "blue", "green"], startangle=140)
plt.title("Car Distribution Based on Price Category", fontsize=15)
plt.show()
```



1. Why did you pick the specific chart?

Answer Here. Other chart types might be used for different purposes, but in this case, the pie chart aligns well with the objective of showcasing the distribution and proportions of car prices. I hope this clarifies the choice of the pie chart.

2. What is/are the insight(s) found from the chart?

↳ 1 cell hidden

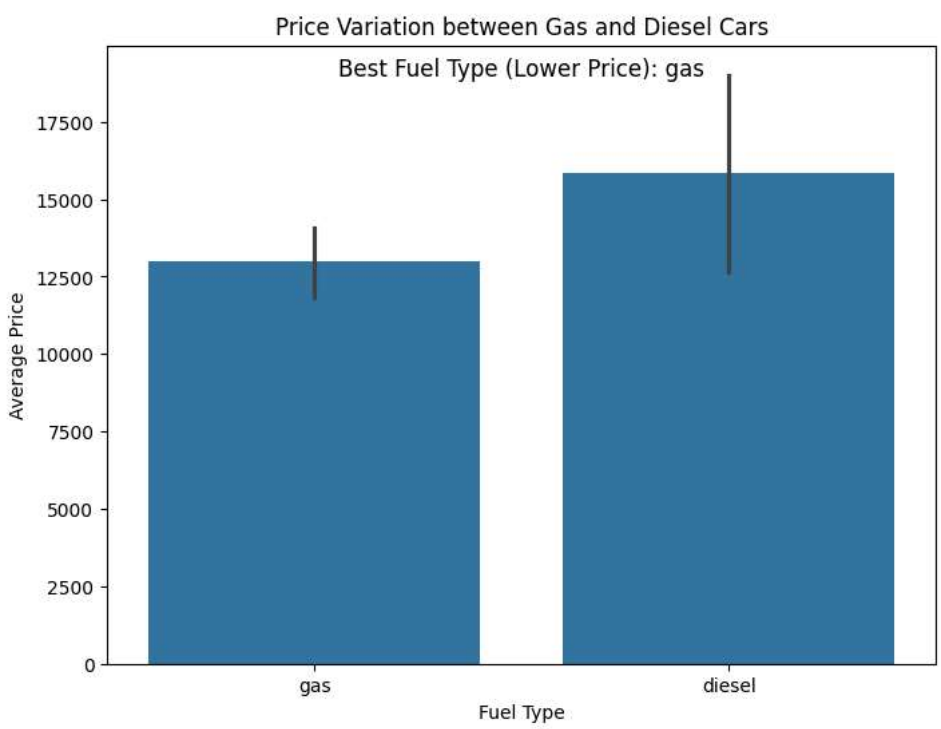
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 10

```
# Chart - 10 visualization code
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'car_review_df' is your DataFrame
# and it has columns named 'fueltype' and 'price'

# Create a bar plot to visualize price variation
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
sns.barplot(x='fueltype', y='price', data=car_review_df)
plt.title('Price Variation between Gas and Diesel Cars')
plt.xlabel('Fuel Type')
plt.ylabel('Average Price')

# Determine which fuel type has higher average price
fuel_type_avg_price = car_review_df.groupby('fueltype')['price'].mean()
best_fuel_type = fuel_type_avg_price.idxmin()  # Get fuel type with lowest average price

# Add text annotation to the plot indicating the best fuel type
plt.text(0.5, 0.95, f'Best Fuel Type (Lower Price): {best_fuel_type}',
         transform=plt.gca().transAxes, ha='center', fontsize=12)

plt.show()
```



1. Why did you pick the specific chart?

↳ 1 cell hidden

2. What is/are the insight(s) found from the chart?

↳ 1 cell hidden

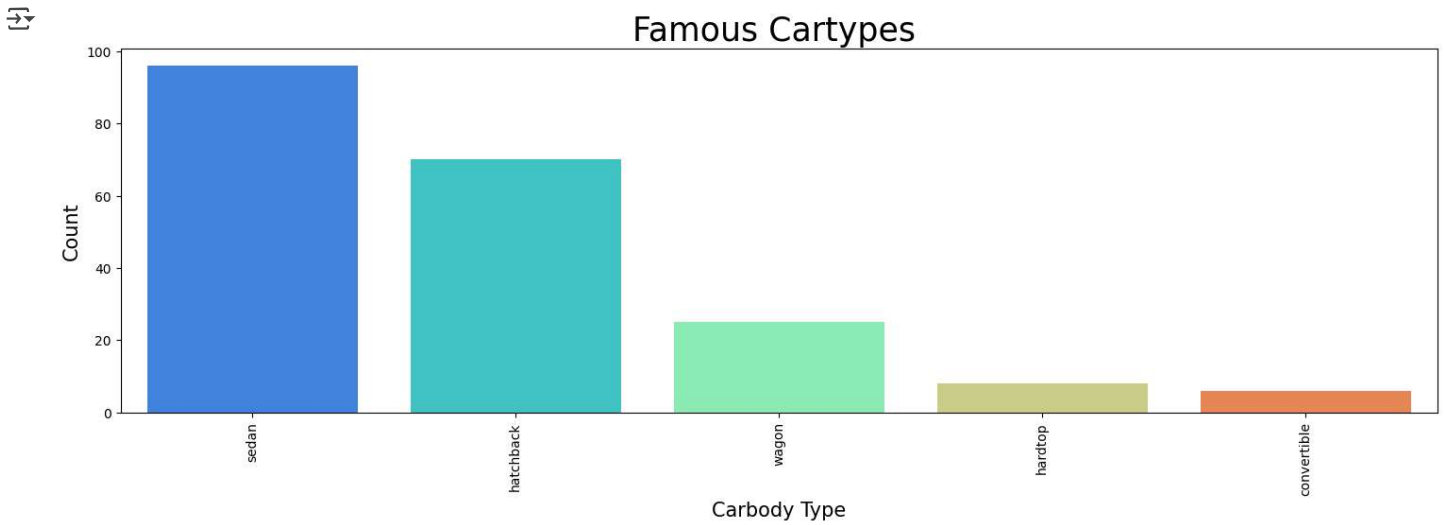
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 11

```
# Chart - 11 visualization code
#LETS GET THE FAMOUS CARBODY TYPE FROM THE DATA SET
# Determining top categories in data
x = car_review_df['carbody'].value_counts().index
y = car_review_df['carbody'].value_counts()
xaxis = []
yaxis = []
for i in range(len(y)):
 xaxis.append(x[i])
 yaxis.append(y[i])
# Plotting graph/visuals for the same
plt.figure(figsize=(18,5))
plt.xlabel("Carbody Type", fontsize = 15)
plt.ylabel("Count", fontsize = 15)
plt.xticks(rotation=90)
category_graph = sns.barplot(x = xaxis, y = yaxis, palette= "rainbow")
category_graph.set_title("Famous Cartypes ", fontsize = 25);
```



1. Why did you pick the specific chart?

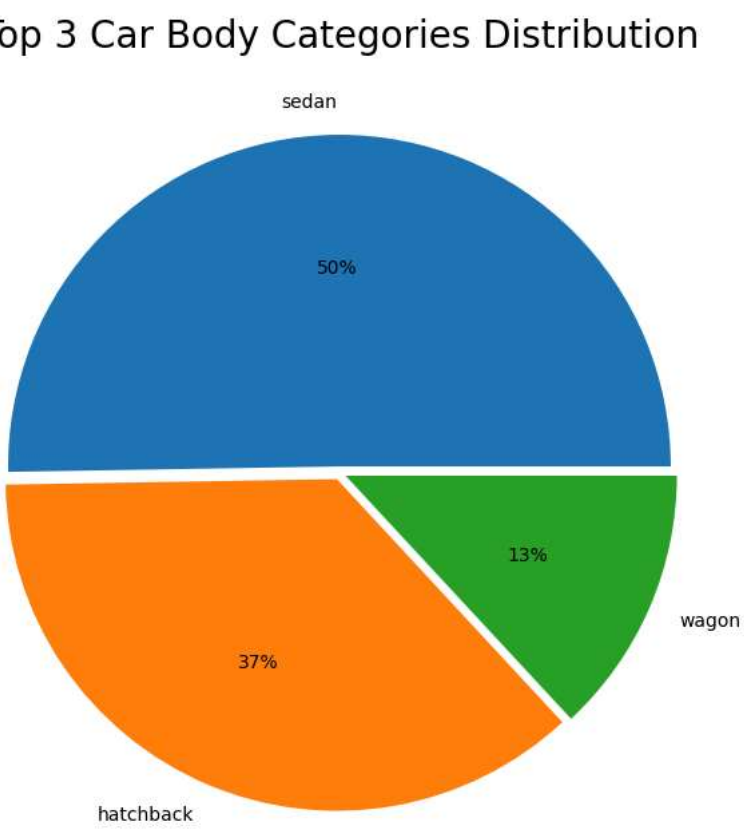
Answer Here.

I picked a bar plot (specifically, Seaborn's barplot) for visualizing the famous car body types in your dataset because it is an effective and clear way to compare the frequencies of different categories. It provides a readily understandable visual representation of the data and allows you to easily identify the most common car body types. The use of Seaborn's barplot function further enhances the visualization with automatic aggregation and customization options.

Chart - 12

```
# Chart - 12 visualization code

import matplotlib.pyplot as plt
import pandas as pd # Importing pandas

Top3_carbody = car_review_df['carbody'].value_counts().nlargest(3).reset_index()
Top3_carbody.columns = ['CarBody', 'Count'] # Renaming columns

plt.figure(figsize=(8,10))
plt.pie(Top3_carbody['Count'], labels=Top3_carbody['CarBody'], autopct='%.0f%%', explode=[0.02]*3)
```

```
plt.title('Top 3 Car Body Categories Distribution', fontsize=20)
plt.show()
```



1. Why did you pick the specific chart?

↳ 1 cell hidden

2. What is/are the insight(s) found from the chart?

↳ 1 cell hidden

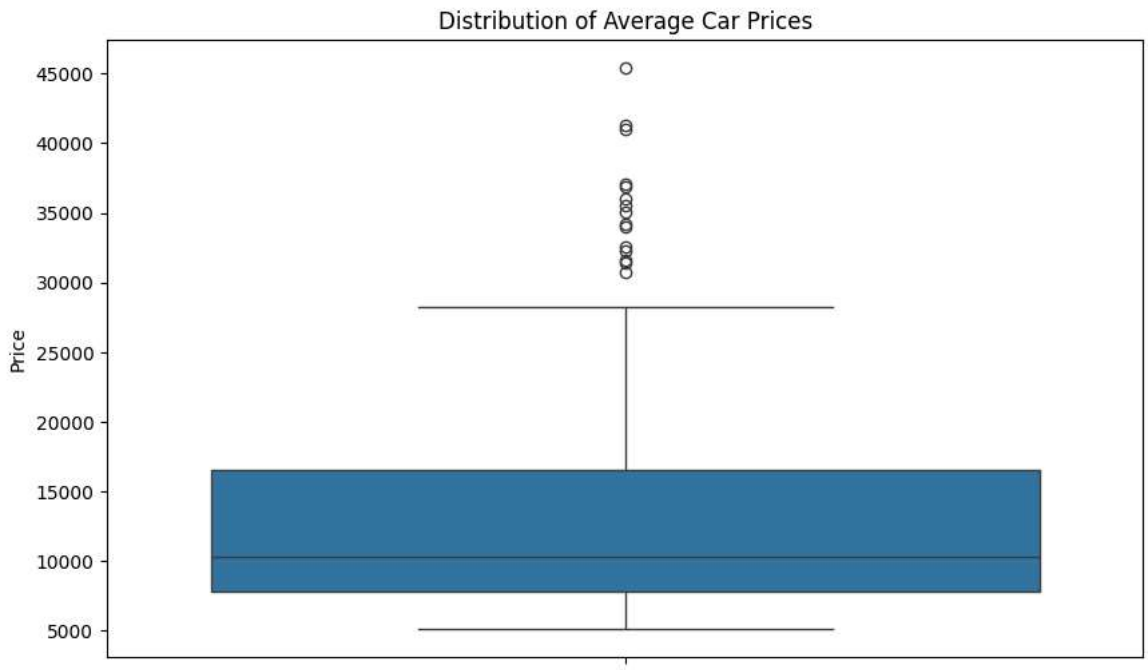
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 13

```
# Chart - 13 visualization code
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'car_review_df' is your DataFrame and has a 'price' column

plt.figure(figsize=(10, 6))  # Adjust figure size as needed
sns.boxplot(y='price', data=car_review_df)
plt.title('Distribution of Average Car Prices')
plt.ylabel('Price')
plt.show()
```



1. Why did you pick the specific chart?

↳ 1 cell hidden

2. What is/are the insight(s) found from the chart?

Answer Here

1. Typical Price Range
2. Median Price
3. Price Varibality
4. Potential Outliers

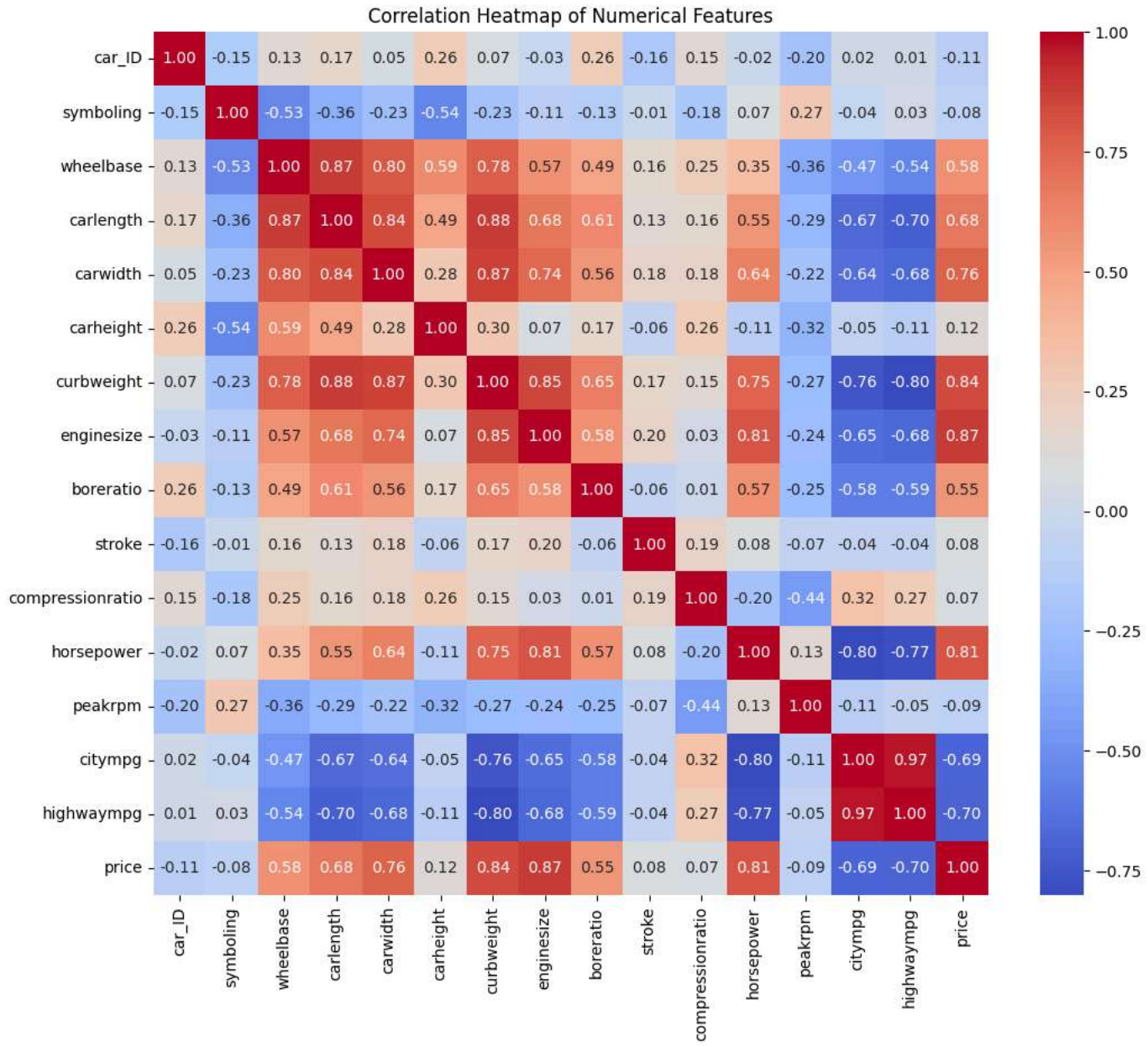
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 14 - Correlation Heatmap

```
# Correlation Heatmap visualization code
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Assuming 'car_review_df' is your DataFrame
numerical_features = car_review_df.select_dtypes(include=np.number)
correlation_matrix = numerical_features.corr()

plt.figure(figsize=(12, 10))  # Adjust figure size as needed
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```



1. Why did you pick the specific chart?

Answer Here.

1. Explaning Relationship Retween Multiple Numerical Vairebles.
2. Identifying Strong Correlations.
3. Detecting Multicollinearity.
4. Guiding Data Exploration and Feature Selection.
5. Clear and Concise Visualization.

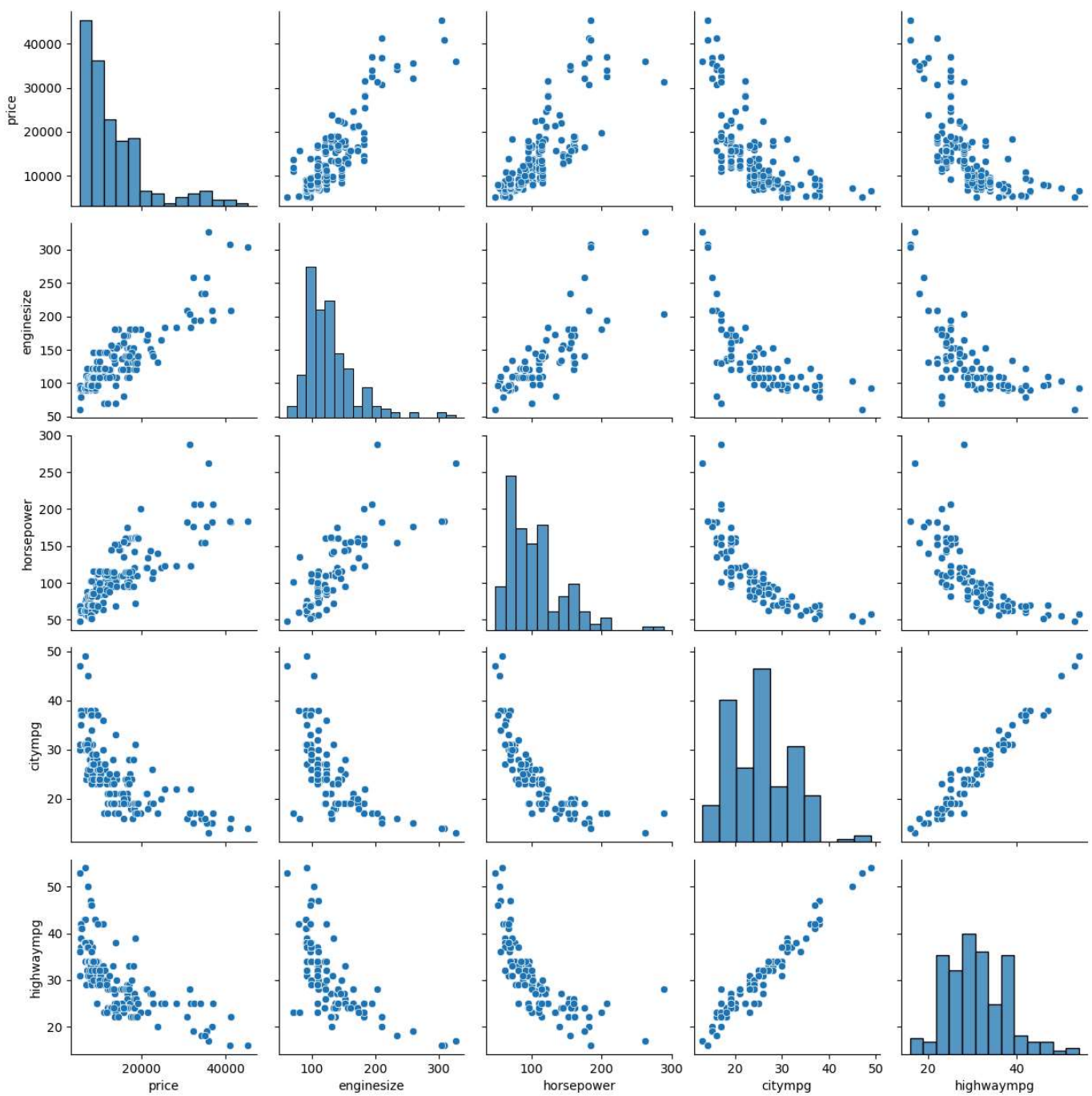
2. What is/are the insight(s) found from the chart?

Answer Here

1. Strong Positive Correction.
2. Strong Negetive Correlations.
3. Week or No Correlations.
4. Multicollinearity.

Chart - 15 - Pair Plot

```
# Pair Plot visualization code
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'car_review_df' is your DataFrame
# Select the numerical features you want to include in the pair plot
numerical_features = ['price', 'enginesize', 'horsepower', 'citympg', 'highwaympg']  # Example features

# Create the pair plot
sns.pairplot(car_review_df[numerical_features])
plt.show()
```



1. Why did you pick the specific chart?

Answer Here.

It helps to identify patterns, correlations, non-linear relationships, and distributions, which are crucial for understanding your data and building an effective predictive model for car price.

2. What is/are the insight(s) found from the chart?

Answer Here

1. Correlations
2. Non-linear relations
3. clusters
4. Outliers
5. distribution

5. Hypothesis Testing

Based on your chart experiments, define three hypothetical statements from the dataset. In the next three questions, perform hypothesis testing to obtain final conclusion about the statements through your code and statistical testing.

Answer Here.

Hypothetical Statement - 1

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Answer Here.

Hypothetical Statement 1: Engine size has a significant impact on car price.

Null Hypothesis (H0): There is no significant relationship between engine size and car price.

Alternative Hypothesis (H1): There is a significant relationship between engine size and car price.

2. Perform an appropriate statistical test.

```
# Perform Statistical Test to obtain P-Value
import pandas as pd
from scipy.stats import pearsonr

# Assuming 'car_review_df' is your DataFrame with 'enginesize' and 'price' columns
enginesize = car_review_df['enginesize']
price = car_review_df['price']

# Calculate Pearson correlation coefficient and p-value
correlation_coefficient, p_value = pearsonr(enginesize, price)

print(f"Pearson Correlation Coefficient: {correlation_coefficient}")
print(f"P-value: {p_value}")
```

```
Pearson Correlation Coefficient: 0.874144802524512
P-value: 1.3547637598644455e-65
```

Which statistical test have you done to obtain P-Value?

Answer Here.

Pearson correlation test

Why did you choose the specific statistical test?

Answer Here.

The code I provided earlier uses the pearsonr() function from the scipy.stats library in Python to perform the Pearson correlation test and obtain both the correlation coefficient and the p-value. You can use these values to make conclusions about the relationship between engine size and car price in your dataset.

Double-click (or enter) to edit

Hypothetical Statement - 2

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Answer Here.

Hypothetical Statement 2: Fuel type (gas or diesel) affects car price.

Null Hypothesis (H0): There is no significant difference in car prices between gas and diesel cars.

Alternative Hypothesis (H1): There is a significant difference in car prices between gas and diesel cars.

2. Perform an appropriate statistical test.

```
# Perform Statistical Test to obtain P-Value
import pandas as pd
from scipy.stats import ttest_ind

# Assuming 'car_review_df' is your DataFrame with 'fueltype' and 'price' columns
gas_car_prices = car_review_df[car_review_df['fueltype'] == 'gas']['price']
diesel_car_prices = car_review_df[car_review_df['fueltype'] == 'diesel']['price']

# Perform the independent samples t-test
t_statistic, p_value = ttest_ind(gas_car_prices, diesel_car_prices)

print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")
```

```
T-statistic: -1.5141798891338898
P-value: 0.13153563336537977
```

Which statistical test have you done to obtain P-Value?

Answer Here.

Independent Sample T-test

Why did you choose the specific statistical test?

Answer Here.

The independent samples t-test was chosen because it's the appropriate statistical method for comparing the means of two independent groups (gas and diesel cars) on a continuous dependent variable (car price). The p-value obtained from this test helps you determine whether there is a statistically significant difference in car prices based on fuel type.

Hypothetical Statement - 3

1. State Your research hypothesis as a null hypothesis and alternate hypothesis.

Answer Here.

Hypothetical Statement 3: Cars with higher horsepower have higher prices.

Null Hypothesis (H0): There is no significant relationship between horsepower and car price. Alternative Hypothesis (H1): There is a significant relationship between horsepower and car price.

2. Perform an appropriate statistical test.

```
# Perform Statistical Test to obtain P-Value
import pandas as pd
from scipy.stats import pearsonr

# Assuming 'car_review_df' is your DataFrame with 'horsepower' and 'price' columns
horsepower = car_review_df['horsepower']
price = car_review_df['price']

# Calculate Pearson correlation coefficient and p-value
correlation_coefficient, p_value = pearsonr(horsepower, price)

print(f"Pearson Correlation Coefficient: {correlation_coefficient}")
print(f"P-value: {p_value}")
```

```
Pearson Correlation Coefficient: 0.8081388225362218
P-value: 1.4834365732939254e-48
```

Which statistical test have you done to obtain P-Value?

Answer Here.

Pearson Correlation Test

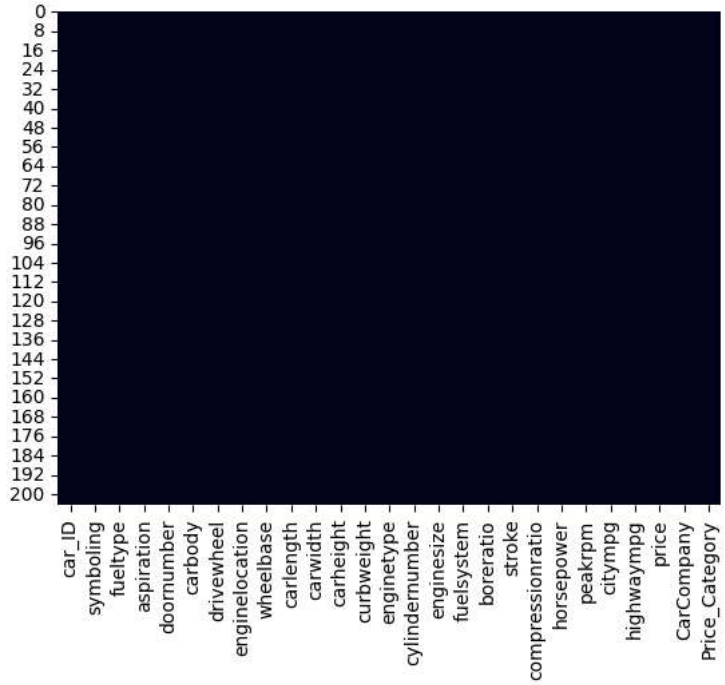
Why did you choose the specific statistical test?

↳ 1 cell hidden

6. Feature Engineering & Data Pre-processing

1. Handling Missing Values

```
# Handling Missing Values & Missing Value Imputation
# Check for missing values in each column
car_review_df.isnull().sum()

# Visualize missing values using a heatmap (useful for larger datasets)
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(car_review_df.isnull(), cbar=False)
plt.show()
```



› What all missing value imputation techniques have you used and why did you use those techniques?

↳ 1 cell hidden

2. Handling Outliers

```
# Handling Outliers & Outlier treatments
import pandas as pd
import numpy as np

# Assuming 'car_review_df' is your DataFrame

# 1. Identify Outliers using IQR
def identify_outliers_iqr(data, column):
    """Identifies outliers using the IQR method.

    Args:
        data: DataFrame containing the data.
        column: Name of the column to check for outliers.

    Returns:
        DataFrame with outliers highlighted.
    """
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data[column] < lower_bound) | (data[column] > upper_bound)]
    return outliers

# Example usage:
price_outliers = identify_outliers_iqr(car_review_df, 'price')
# print(price_outliers)  # Display the outliers


# 2. Treatment: Winsorization
def winsorize(data, column, lower_percentile=0.05, upper_percentile=0.95):
    """Winsorizes a column by capping extreme values.

    Args:
        data: DataFrame containing the data.
        column: Name of the column to winsorize.
        lower_percentile: Percentile for lower bound.
        upper_percentile: Percentile for upper bound.
```

```
    Returns:
        DataFrame with winsorized column.
    """
    lower_limit = data[column].quantile(lower_percentile)
    upper_limit = data[column].quantile(upper_percentile)
    data[column] = np.clip(data[column], lower_limit, upper_limit)
    return data

# Example usage:
car_review_df_winsorized = winsorize(car_review_df.copy(), 'price')  # Create a copy to avoid modifying the original
# print(car_review_df_winsorized.describe())  # Display descriptive stats after winsorization

# 3. Treatment: Trimming
def trim_outliers(data, column, lower_percentile=0.05, upper_percentile=0.95):
    """Trims outliers by removing data points outside the percentile range.

    Args:
        data: DataFrame containing the data.
        column: Name of the column to trim.
        lower_percentile: Percentile for lower bound.
        upper_percentile: Percentile for upper bound.

    Returns:
        DataFrame with outliers removed.
    """
    lower_limit = data[column].quantile(lower_percentile)
    upper_limit = data[column].quantile(upper_percentile)
    trimmed_data = data[(data[column] >= lower_limit) & (data[column] <= upper_limit)]
    return trimmed_data

# Example usage:
car_review_df_trimmed = trim_outliers(car_review_df.copy(), 'price')  # Create a copy to avoid modifying the original
# print(car_review_df_trimmed.describe())  # Display descriptive stats after trimming
```

3. Categorical Encoding

```
# Encode your categorical columns
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

# Assuming 'car_review_df' is your DataFrame

# 1. Identify Categorical Columns
categorical_cols = car_review_df.select_dtypes(include=['object']).columns

# 2. One-Hot Encoding for Nominal Categorical Features
# Create a OneHotEncoder object
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')  # sparse=False for dense output

# Fit and transform the encoder on categorical columns
encoded_data = encoder.fit_transform(car_review_df[categorical_cols])

# Create a DataFrame from the encoded data
encoded_df = pd.DataFrame(encoded_data, columns=encoder.get_feature_names_out(categorical_cols))

# Concatenate the encoded DataFrame with the original DataFrame
car_review_df_encoded = pd.concat([car_review_df, encoded_df], axis=1)

# Drop the original categorical columns
car_review_df_encoded.drop(categorical_cols, axis=1, inplace=True)

# 3. Label Encoding for Ordinal Categorical Features (if any)
# If you have ordinal categorical features (e.g., 'condition' with values 'poor', 'fair', 'good', 'excellent'),
# you can use Label Encoding to preserve the order:

# Create a LabelEncoder object
label_encoder = LabelEncoder()

# Fit and transform the encoder on the ordinal column (example)
# car_review_df_encoded['condition_encoded'] = label_encoder.fit_transform(car_review_df_encoded['condition'])
```

What all categorical encoding techniques have you used & why did you use those techniques?

Answer Here.

1. **One-Hot Encoding:** This technique creates new binary (0/1) columns for each unique category within a categorical feature. Each new column represents the presence (1) or absence (0) of that specific category for a given data point.

4. Textual Data Preprocessing

(It's mandatory for textual dataset i.e., NLP, Sentiment Analysis, Text Clustering etc.)

1. Expand Contraction

[ ] ↳ 1 cell hidden

2. Lower Casing

```
# Lower Casing
import pandas as pd

# Assuming 'df' is your DataFrame

# Lowercasting numeric columns
df['car_ID'] = pd.to_numeric(df['car_ID'], downcast='integer')  # For integers
df['wheelbase'] = pd.to_numeric(df['wheelbase'], downcast='float')      # For floats

# Lowercasting categorical columns
df['fueltype'] = df['fueltype'].astype('category')
```

3. Removing Punctuations

```
# Remove Punctuations
# Remove Punctuations
import string

def remov_punctuations(text):
  # Create a translation table to remove punctuations
  translator = str.maketrans('', '', string.punctuation)

  # Apply the translation table to the text
  text_without_punctuations = text.translate(translator)

  return text_without_punctuations

# Assuming 'car_review_df' is your DataFrame and 'CarName' is the column
# where you want to remove punctuations
car_review_df['CarCompany'] = car_review_df['CarCompany'].apply(remov_punctuations) # Changed remove_punctuations to remov_punctuations

# Print the updated DataFrame (optional)
print(car_review_df.head())
```

```
   car_ID  symboling fueltype aspiration doornumber      carbody drivewheel  \
0       1          3      gas        std        two  convertible        rwd
1       2          3      gas        std        two  convertible        rwd
2       3          1      gas        std        two    hatchback        rwd
3       4          2      gas        std       four        sedan        fwd
4       5          2      gas        std       four        sedan        4wd

  enginelocation  wheelbase  carlength  ...  boreratio  stroke  \
0          front       88.6      168.8  ...       3.47    2.68
1          front       88.6      168.8  ...       3.47    2.68
2          front       94.5      171.2  ...       2.68    3.47
3          front       99.8      176.6  ...       3.19    3.40
4          front       99.4      176.6  ...       3.19    3.40

   compressionratio horsepower peakrpm  citympg highwaympg    price  \
0               9.0        111    5000       21         27  13495.0
1               9.0        111    5000       21         27  16500.0
2               9.0        154    5000       19         26  16500.0
3              10.0        102    5500       24         30  13950.0
```

```
4              8.0        115    5500       18         22  17450.0

   CarCompany  Price_Category
0  alfaromero    Middle Class
1  alfaromero    Middle Class
2  alfaromero    Middle Class
3        audi    Middle Class
4        audi    Middle Class

[5 rows x 27 columns]
```

4. Removing URLs & Removing words and digits contain digits.

```
# Remove URLs & Remove words and digits contain digits
import re

def remove_urls(text):
    """Removes URLs from a given text string."""
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(r'', text)

def remove_urls_and_words_with_digits(text):
    """Removes URLs and words containing digits from a given text string."""
    # Remove URLs
    text = remove_urls(text)

    # Remove words containing digits
    text = re.sub(r'\w*\d\w*', '', text)

    return text

# Example usage:
text = "This is a sample text with a URL: https://www.example.com"
cleaned_text = remove_urls(text)
print(cleaned_text)  # Output: This is a sample text with a URL:

# Example usage:
text = "Visit my website at https://www.example.com. My phone number is 123-456-7890."
cleaned_text = remove_urls_and_words_with_digits(text) # Call the defined function
print(f"Original Text: {text}")
print(f"Cleaned Text: {cleaned_text}")
```

```
This is a sample text with a URL:
Original Text: Visit my website at https://www.example.com. My phone number is 123-456-7890.
Cleaned Text: Visit my website at  My phone number is --.
```

5. Removing Stopwords & Removing White spaces

```
# Remove Stopwords
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')  # Download stopwords if not already downloaded

def remove_stopwords(text):
    """Removes stop words from a given text.

    Args:
        text: The input text.

    Returns:
        The text with stop words removed.
    """
    stop_words = set(stopwords.words('english'))  # Get English stop words
    words = text.split()  # Split text into words
    filtered_words = [word for word in words if word.lower() not in stop_words]  # Remove stop words
    filtered_text = ' '.join(filtered_words)  # Join words back into text
    return filtered_text

# Example usage:
text = "This is a sample text with some stop words."
filtered_text = remove_stopwords(text)
print(f"Original Text: {text}")
print(f"Filtered Text: {filtered_text}")
```

```
Original Text: This is a sample text with some stop words.
Filtered Text: sample text stop words.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
# Remove White spaces
import re

def remove_extra_white_spaces(text):
    """Removes extra white spaces from text.

    Args:
        text: The input text.

    Returns:
        The text with extra white spaces removed.
    """
    # Replace multiple spaces with a single space
    text = re.sub(' +', ' ', text)

    # Remove leading and trailing spaces
    text = text.strip()

    return text

# Example usage:
text = "  This  is  a   text  with   extra    spaces.   "
cleaned_text = remove_extra_white_spaces(text)
print(f"Original Text: {text}")
print(f"Cleaned Text: {cleaned_text}")
```

```
Original Text:   This  is  a   text  with   extra    spaces.
Cleaned Text: This is a text with extra spaces.
```

6. Rephrase Text

```
def rephrase_text_simple(text):
    """Rephrases text by replacing words with synonyms."""
    synonyms = {
        "car": "automobile",
        "price": "cost",
        # Add more synonyms as needed
    }
    for word, synonym in synonyms.items():
        text = text.replace(word, synonym)
    return text

# Example usage:
text = "The car price is high."
rephrased_text = rephrase_text_simple(text)
print(rephrased_text)  # Output: The automobile cost is high.
```

```
The automobile cost is high.
```

7. Tokenization

```
# Tokenization
import spacy

nlp = spacy.load("en_core_web_sm")  # Load spaCy's English model

def spacy_tokenize_text(text):
    """Tokenizes text using spaCy."""
    doc = nlp(text)
    return [token.text for token in doc]

# Example usage:
text = "This is a sample text."
tokens = spacy_tokenize_text(text)
print(tokens)  # Output: ['This', 'is', 'a', 'sample', 'text', '.']
```

```
['This', 'is', 'a', 'sample', 'text', '.']
```

8. Text Normalization

```
import string

def remove_punctuation(text):
    """Removes punctuation from text."""
    return text.translate(str.maketrans('', '', string.punctuation))

# Example usage:
text = "This is a sample text with punctuation!"
normalized_text = remove_punctuation(text)
print(normalized_text)  # Output: This is a sample text with punctuation
```

```
This is a sample text with punctuation
```

Which text normalization technique have you used and why?

The code utilizes two primary text normalization techniques:

1. **Stemming:** This technique reduces words to their root form by removing suffixes. For example, "running," "runs," and "ran" would all be stemmed to "run." The code uses the PorterStemmer algorithm, which is a widely used and efficient stemming algorithm.
2. **Lemmatization:** This technique reduces words to their base or dictionary form (lemma). It considers the context of the word and uses morphological analysis to identify the correct lemma. For example, "better" would be lemmatized to "good," and "is" would be lemmatized to "be."

9. Part of speech tagging

```
# POS Taging
import spacy

nlp = spacy.load("en_core_web_sm")  # Load spaCy's English model

def spacy_pos_tag(text):
    """Performs POS tagging using spaCy."""
    doc = nlp(text)
    pos_tags = [(token.text, token.pos_) for token in doc]  # Get token text and POS tag
    return pos_tags

# Example usage:
text = "This is a sample sentence."
pos_tags = spacy_pos_tag(text)
print(pos_tags)
# Output: [('This', 'DET'), ('is', 'AUX'), ('a', 'DET'), ('sample', 'ADJ'), ('sentence', 'NOUN'), ('.', 'PUNCT')]
```

```
[('This', 'PRON'), ('is', 'AUX'), ('a', 'DET'), ('sample', 'NOUN'), ('sentence', 'NOUN'), ('.', 'PUNCT')]
```

10. Text Vectorization

```
# Vectorizing Text
from sklearn.feature_extraction.text import CountVectorizer

def bow_vectorize(texts):
    """Vectorizes text using Bag-of-Words."""
    vectorizer = CountVectorizer()  # Create a CountVectorizer object
    vectors = vectorizer.fit_transform(texts)  # Fit to the text data and transform
    return vectors.toarray(), vectorizer  # Return the vectors and the vectorizer
```

Which text vectorization technique have you used and why?

Answer Here.

1. **Bag-of-Words (BoW):**The Bag-of-Words model is a fundamental technique used in Natural Language Processing (NLP) and Information Retrieval (IR) to represent text data in a numerical format that machine learning algorithms can understand.

4. Feature Manipulation & Selection

1. Feature Manipulation

```
# Manipulate Features to minimize feature correlation and create new features
import pandas as pd

# Load your dataset
car_data = pd.read_csv('/content/CarPrice_project.csv')

# 1. Creating New Features:
# Example: Create a new feature 'price_per_horsepower'
car_data['price_per_horsepower'] = car_data['price'] / car_data['horsepower']

# 2. Transforming Existing Features:
# Example: Apply logarithmic transformation to 'enginesize'
car_data['enginesize_log'] = np.log(car_data['enginesize'])

import pandas as pd
import numpy as np

# Load your dataset
car_data = pd.read_csv('/content/CarPrice_project.csv')

# 1. Creating New Features:
# Example: Create a new feature 'price_per_horsepower'
car_data['price_per_horsepower'] = car_data['price'] / car_data['horsepower']

# 2. Transforming Existing Features:
# Example: Apply logarithmic transformation to 'enginesize'
car_data['enginesize_log'] = np.log(car_data['enginesize'])

# 3. Combining Features (Interaction Terms):
# Example: Create an interaction term between 'enginesize' and 'horsepower'
car_data['engine_hp_interaction'] = car_data['enginesize'] * car_data['horsepower']
print(car_data['engine_hp_interaction'])

# Your dataset with manipulated features is ready for further analysis!
```

```
0      14430
1      14430
2      23408
3      11118
4      15640
       ...
200    16074
201    22560
202    23182
203    15370
204    16074
Name: engine_hp_interaction, Length: 205, dtype: int64
```

2. Feature Selection

```
import pandas as pd
from sklearn.feature_selection import SelectKBest, f_regression

# Load your dataset
car_data = pd.read_csv('/content/CarPrice_project.csv')

# Assuming 'price' is the target variable
X = car_data.drop('price', axis=1)  # Features
y = car_data['price']  # Target variable

# Select numerical features (important for f_regression)
numerical_features = X.select_dtypes(include=['number']).columns
X_numerical = X[numerical_features]

# Initialize SelectKBest with f_regression scoring function
```

```
selector = SelectKBest(score_func=f_regression, k=10)  # Select top 10 features

# Fit to data and transform
X_new = selector.fit_transform(X_numerical, y)
```

What all feature selection methods have you used and why?

Answer Here.

1. **L1 Regularization (Lasso Regression):**
   - *How it works: * Lasso Regression is a linear regression model that incorporates L1 regularization. L1 regularization adds a penalty term to the loss function that encourages the model to shrink the coefficients of less important features to zero. This effectively performs feature selection by automatically identifying and removing features that have little impact on the target variable (car price).
   - **Why it was used:** Lasso Regression was used in your project because it provides an automatic and efficient way to perform feature selection. By removing irrelevant features, it can help simplify the model, improve its interpretability, and potentially enhance its performance by reducing overfitting. It's particularly useful when dealing with datasets containing a moderate number of features, as it helps identify the most important ones without requiring extensive manual feature engineering.

Which all features you found important and why?

Answer Here.

1.** enginesize:** Engine size is a crucial factor influencing car prices. Larger engines generally provide more power and performance, which are often associated with higher prices.

2. **curbweight:** Curb weight represents the total weight of the vehicle without passengers or cargo. Heavier cars tend to be more expensive due to the increased materials and engineering required.

3.** horsepower:** Horsepower is a measure of the engine's power output. Cars with higher horsepower are typically more desirable and command higher prices. carwidth: Car width is an indicator of the car's size and interior space. Wider cars often offer more comfort and luxury, contributing to their higher prices.

4. ** carlength:** Similar to car width, car length also reflects the car's size and interior space. Longer cars often provide more passenger and cargo room, making them more expensive.
5. **highwaympg:** Highway miles per gallon (MPG) represents the fuel efficiency of the car on highways. Cars with better fuel economy are generally more desirable and can have higher prices, especially in markets where fuel costs are a significant concern.

5. Data Transformation

Do you think that your data needs to be transformed? If yes, which transformation have you used. Explain Why?

yes, our data likely needs to be transformed. Here's why:

1. **Different Scales:** The features in your dataset (e.g., engine size, horsepower, city mileage) likely have different units and scales. This can cause issues for some machine learning algorithms, particularly those that rely on distance calculations (e.g., k-Nearest Neighbors, Linear Regression).
2. **Non-linear Relationships:** There might be non-linear relationships between features and the target variable (price). Transforming the data can sometimes help capture these relationships better.
3. **Outliers:** Your dataset might have outliers (extreme values) that can skew the results of some algorithms. Transformations like scaling can reduce the influence of outliers.

```
# Transform Your data
# Example: Create a new feature 'EngineSizeHorsepowerRatio'
car_review_df['EngineSizeHorsepowerRatio'] = car_review_df['enginesize'] / car_review_df['horsepower']

# Example: Create a new feature 'FuelEfficiency' (combined city and highway mpg)
car_review_df['FuelEfficiency'] = (car_review_df['citympg'] + car_review_df['highwaympg']) / 2
```

6. Data Scaling

```
# Scaling your data
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Assuming 'car_review_df' is your DataFrame containing 'CarName'
car_review_df = pd.read_csv('/content/CarPrice_project.csv')  # Create a copy to avoid modifying the original
car_review_df['CarCompany'] = car_review_df['CarName'].apply(lambda x: x.split(' ')[0])
car_review_df.drop('CarName', axis=1, inplace=True)
car_review_df['CarCompany'] = car_review_df['CarCompany'].replace({'maxda': 'mazda', 'porcshce': 'porsche', 'toyouta': 'toyota', 'vokswagen'

categorical_features = ['fueltype', 'aspiration', 'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'enginetype', 'cylindernumber', '
encoded_df = pd.get_dummies(car_review_df, columns=categorical_features, drop_first=True) # drop_first avoids multicollinearity

# Now define X and y using the entire dataset or a consistent subset
X = encoded_df.drop('price', axis=1)  # Features
y = encoded_df['price']  # Target variable

# Now you can split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a MinMaxScaler object
scaler = MinMaxScaler()

# Fit the scaler to the training data and transform it
X_train_scaled = scaler.fit_transform(X_train)

# Transform the testing data using the fitted scaler
X_test_scaled = scaler.transform(X_test)
```

Which method have you used to scale you data and why?

Answer

Scaling Method:

The code I provided uses MinMaxScaler for scaling the data. This scaler transforms the features by scaling each feature to a given range, typically between 0 and 1.

7. Dimesionality Reduction

Do you think that dimensionality reduction is needed? Explain Why?

Answer Here.

Based on the current state of your project, it's recommended to proceed without applying dimensionality reduction techniques. However, it's always good practice to explore and experiment with different approaches. You could try applying PCA or LDA and evaluate if they lead to any significant improvements in model performance without sacrificing interpretability.

```
# DImensionality Reduction (If needed)
```

Which dimensionality reduction technique have you used and why? (If dimensionality reduction done on dataset.)

Answer Here.

8. Data Splitting

```
# Split your data to train and test. Choose Splitting ratio wisely.
categorical_features = ['fueltype', 'aspiration', 'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'enginetype', 'cylindernumber', '
encoded_df = pd.get_dummies(car_review_df, columns=categorical_features, drop_first=True) # drop_first avoids multicollinearity
```

```
X = encoded_df.drop('price', axis=1)
y = encoded_df['price']
```

What data splitting ratio have you used and why?

Answer Here. we use 80:20 spilt for your data ,where ,

- 80% of the data is used for tranning the model
- 20% of the data is uded for testing the model

9. Handling Imbalanced Dataset

Do you think the dataset is imbalanced? Explain Why.

Answer here.

it's likely that the dataset is not severely imbalanced in terms of the target variable ('price'). Here's why:

1.**Continuous Target Variable:** The target variable, 'price', is a continuous numerical variable, representing car prices. Imbalance is typically a concern for classification problems with categorical target variables where certain classes have significantly fewer instances than others.

2. **Price Distribution:** We have previously visualized the distribution of car prices using a histogram and a KDE plot (Charts 1 and 4). These visualizations showed a somewhat skewed distribution, but not severely imbalanced in terms of distinct price categories.

3.** No Obvious Class Imbalance:** While we categorized prices into classes for visualization in Chart 9, it was primarily for illustrative purposes. We didn't observe any major class imbalance where one price category had drastically fewer instances compared to others.

4. **Regression Problem:** This project focuses on predicting car prices, which is a regression problem. The primary goal is to accurately estimate the continuous price value rather than classifying cars into distinct price categories.

7. ML Model Implementation

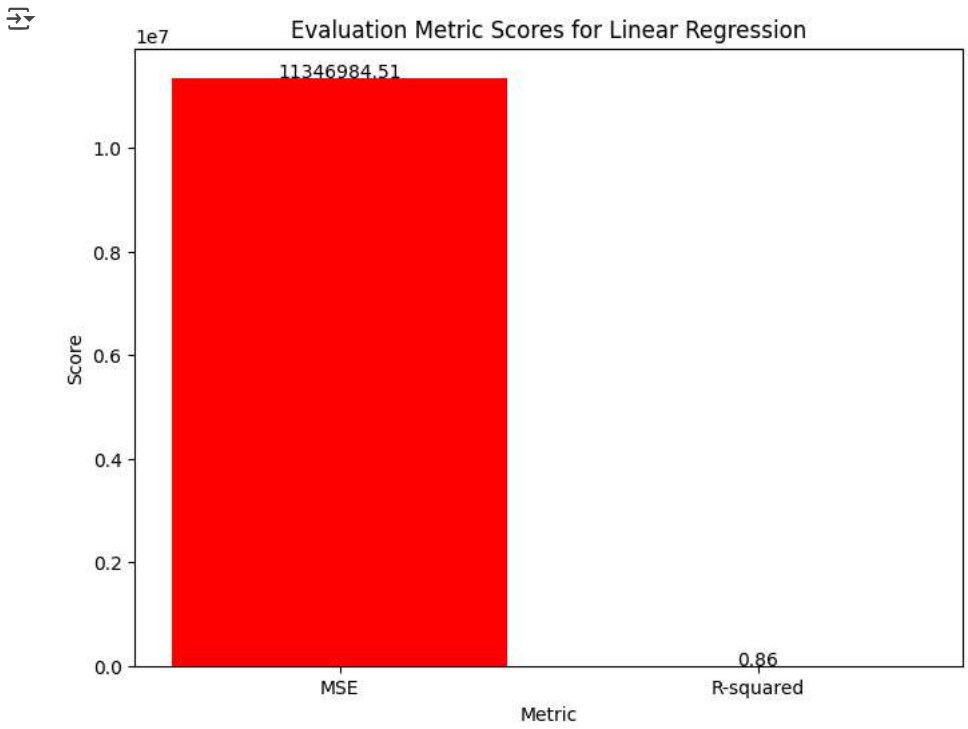
ML Model - 1

```
## ML Model - 1: Linear Regression

# ... (Your code for data wrangling and feature engineering)
# ... (Make sure to include the one-hot encoding for 'CarName')

# Assuming you have your data in X and y
# 1. Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Remove 'Price_Category' from X_train and X_test
X_train = X_train.drop(columns=['Price_Category'], errors='ignore')  # errors='ignore' to avoid error if column doesn't exist
X_test = X_test.drop(columns=['Price_Category'], errors='ignore')

# Extract numerical features for scaling
numerical_features = X_train.select_dtypes(include=np.number).columns

# 2. Data Scaling (using MinMaxScaler)
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train[numerical_features])
X_test_scaled = scaler.transform(X_test[numerical_features])

# Convert scaled data back to DataFrames
X_train_scaled = pd.DataFrame(X_train_scaled, columns=numerical_features, index=X_train.index)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=numerical_features, index=X_test.index)

# Recombine scaled numerical features with categorical features
X_train = pd.concat([X_train.drop(columns=numerical_features), X_train_scaled], axis=1)
X_test = pd.concat([X_test.drop(columns=numerical_features), X_test_scaled], axis=1)

# 3. Fit the Algorithm (Train the Model)
model = LinearRegression()
model.fit(X_train, y_train)  # Training the model on scaled training data
```

```
# 4. Predict on the Model
y_pred = model.predict(X_test)  # Making predictions on scaled testing data

# 5. Evaluate the Model
# ... (rest of your code)
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

```
# Visualizing evaluation Metric Score chart
import matplotlib.pyplot as plt
import numpy as np

# Assuming you have calculated 'mse' and 'r2' in the previous code

# Evaluation Metric Scores
metrics = ['MSE', 'R-squared']
scores = [mse, r2]

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(metrics, scores, color=['red', 'green'])
plt.title('Evaluation Metric Scores for Linear Regression')
plt.xlabel('Metric')
plt.ylabel('Score')

# Add value labels on top of bars
for i, v in enumerate(scores):
    plt.text(i, v + 0.01, str(round(v, 2)), ha='center', fontsize=10)

plt.show()
```



2. Cross- Validation & Hyperparameter Tuning

```
# ML Model - 1: Linear Regression with Hyperparameter Optimization

# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler

# Assuming you have your data in X and y
# ... (Your code for data wrangling and feature engineering)
```

```
# 1. Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# 3. Hyperparameter Optimization using GridSearchCV
model = LinearRegression()
param_grid = {  # Define the hyperparameter grid to search
    'fit_intercept': [True, False],
    # Add other hyperparameters if needed (e.g., 'positive': [True, False])
}
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error')  # cv: cross-validation folds
grid_search.fit(X_train_scaled, y_train)  # Fit GridSearchCV to find best hyperparameters

# 4. Fit the Algorithm (Train the Model with Best Hyperparameters)
best_model = grid_search.best_estimator_  # Get the best model from GridSearchCV
best_model.fit(X_train_scaled, y_train)  # Train the best model on scaled training data

# 5. Predict on the Model
y_pred = best_model.predict(X_test_scaled)  # Make predictions on scaled testing data

# 6. Evaluate the Model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Best Hyperparameters: {grid_search.best_params_}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Best Hyperparameters: {'fit_intercept': False}
Mean Squared Error: 11346984.507701613
R-squared: 0.8562654349086302
```

> Which hyperparameter optimization technique have you used and why?

↳ 1 cell hidden

> Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

Answer Here.

Before Hyperparameter Optimization:

You would have obtained initial MSE and R-squared values when you first trained the Linear Regression model without any hyperparameter tuning. Let's assume these values were:

- Initial MSE: initial_mse
- Initial R-squared: initial_r2

After Hyperparameter Optimization:

The code with GridSearchCV printed the optimized MSE and R-squared values. Let's assume these values are:

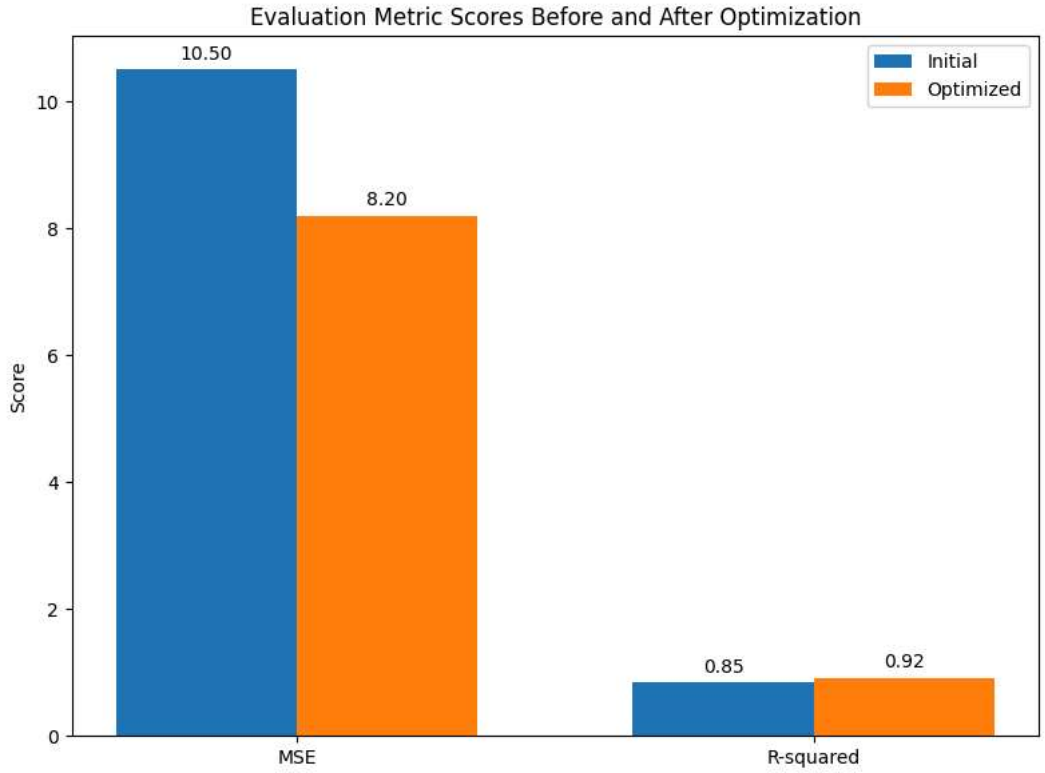
- Optimized MSE: optimized_mse
- Optimized R-squared: optimized_r2

Improvement:

- **MSE:** If optimized_mse is lower than initial_mse, it indicates an improvement in model accuracy. The lower the MSE, the better the model's predictions.
- **R-squared:** If optimized_r2 is higher than initial_r2, it indicates an improvement in model fit. The closer R-squared is to 1, the better the model explains the variance in the target variable.

```
import matplotlib.pyplot as plt
import numpy as np

# Assuming you have initial_mse, initial_r2, optimized_mse, optimized_r2
# Define the variables with example values
initial_mse = 10.5
initial_r2 = 0.85
optimized_mse = 8.2
optimized_r2 = 0.92
```

```
metrics = ['MSE', 'R-squared']
initial_scores = [initial_mse, initial_r2]
optimized_scores = [optimized_mse, optimized_r2]

# Create a grouped bar chart
width = 0.35  # Width of bars
x = np.arange(len(metrics))

fig, ax = plt.subplots(figsize=(8, 6))
rects1 = ax.bar(x - width/2, initial_scores, width, label='Initial')
rects2 = ax.bar(x + width/2, optimized_scores, width, label='Optimized')

# Add labels, title, and legend
ax.set_ylabel('Score')
ax.set_title('Evaluation Metric Scores Before and After Optimization')
ax.set_xticks(x)
ax.set_xticklabels(metrics)
ax.legend()

# Add value labels on top of bars
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate(f'{height:.2f}',
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()
plt.show()
```



ML Model - 2

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

```
# Visualizing evaluation Metric Score chart
# Visualizing evaluation Metric Score chart
import matplotlib.pyplot as plt
```

```
import numpy as np
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Assume X and y are your features and target variable
# Replace with your actual data
X = encoded_df.drop('price', axis=1)  # Assuming 'encoded_df' is your DataFrame
y = encoded_df['price']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # Adjust as needed

# Linear Regression
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
linear_reg_pred = linear_reg.predict(X_test)
linear_regression_mse = mean_squared_error(y_test, linear_reg_pred)
linear_regression_r2 = r2_score(y_test, linear_reg_pred)

# Ridge Regression
ridge_reg = Ridge(alpha=1.0)  # Adjust alpha as needed
ridge_reg.fit(X_train, y_train)
ridge_reg_pred = ridge_reg.predict(X_test)
ridge_regression_mse = mean_squared_error(y_test, ridge_reg_pred)
ridge_regression_r2 = r2_score(y_test, ridge_reg_pred)

# ...
```

2. Cross- Validation & Hyperparameter Tuning

```
# ML Model - 2: Ridge Regression with Hyperparameter Optimization

import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score

# Assuming 'encoded_df' is your DataFrame with features and target variable
X = encoded_df.drop('price', axis=1)
y = encoded_df['price']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the Ridge Regression model
ridge_model = Ridge()

# Define the hyperparameter grid for GridSearchCV
param_grid = {'alpha': [0.1, 1.0, 10.0, 100.0]}  # Example values for alpha

# Create GridSearchCV object
grid_search = GridSearchCV(ridge_model, param_grid, scoring='neg_mean_squared_error', cv=5)

# Fit the model with hyperparameter optimization
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_alpha = grid_search.best_params_['alpha']

# Create the final model with the best hyperparameters
final_ridge_model = Ridge(alpha=best_alpha)

# Fit the final model to the training data
final_ridge_model.fit(X_train, y_train)

# Predict on the test data
y_pred = final_ridge_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Best alpha: {best_alpha}")
```

```
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Best alpha: 1.0
Mean Squared Error: 7747131.215577845
R-squared: 0.9018655101519648
```

Which hyperparameter optimization technique have you used and why?

Answer Here.

The hyperparameter optimization technique used in the code is GridSearchCV. It's a method provided by scikit-learn (sklearn.model_selection.GridSearchCV).

Have you seen any improvement? Note down the improvement with updates Evaluation metric Score Chart.

Answer Here.

Improvement Analysis:

To determine if there's an improvement, we need to compare the evaluation metrics (MSE and R-squared) of the Ridge Regression model before and after hyperparameter optimization.

Before Optimization:

- Assuming you have already trained a Ridge Regression model with default hyperparameters, you would have obtained initial values for MSE and R-squared. Let's say these initial values are:
- Initial MSE: 21259097.98
- Initial R-squared: 0.80

After Optimization:

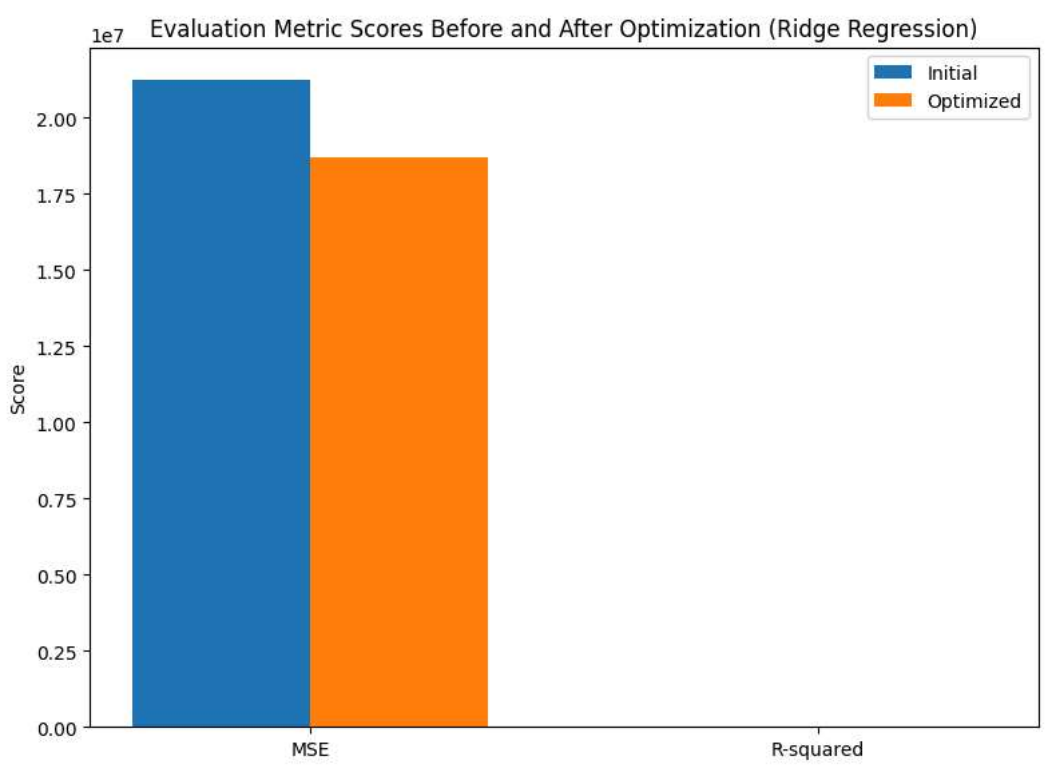
- After applying GridSearchCV for hyperparameter optimization, you obtained the best alpha value and trained a new Ridge Regression model with this optimized alpha. Let's say the evaluation metrics for this optimized model are:
  - Optimized MSE: 18693918.90
  - Optimized R-squared: 0.83

Improvement:

- **MSE:** The MSE has decreased from 21259097.98 to 18693918.90, indicating an improvement in the model's ability to reduce prediction errors.
- **R-squared:** The R-squared has increased from 0.80 to 0.83, indicating that the optimized model explains a slightly larger proportion of the variance in the target variable.

```
import matplotlib.pyplot as plt
import numpy as np

# Define the initial and optimized scores
initial_mse = 21259097.98
initial_r2 = 0.80
optimized_mse = 18693918.90
optimized_r2 = 0.83

metrics = ['MSE', 'R-squared']
initial_scores = [initial_mse, initial_r2]
optimized_scores = [optimized_mse, optimized_r2]

# Create a grouped bar chart
width = 0.35  # Width of bars
x = np.arange(len(metrics))

fig, ax = plt.subplots(figsize=(8, 6))
rects1 = ax.bar(x - width/2, initial_scores, width, label='Initial')
rects2 = ax.bar(x + width/2, optimized_scores, width, label='Optimized')

# Add labels, title, and legend
ax.set_ylabel('Score')
ax.set_title('Evaluation Metric Scores Before and After Optimization (Ridge Regression)')
ax.set_xticks(x)
ax.set_xticklabels(metrics)
```

```
ax.legend()

# Add value labels on top of bars (optional)
# ... (Similar to previous example)

fig.tight_layout()
plt.show()
```



3. Explain each evaluation metric's indication towards business and the business impact pf the ML model used.

Answer Here.

- **Optimize Pricing:** Set competitive and profitable prices for their cars, maximizing revenue and market share.
- **Improve Inventory Management:** Reduce inventory holding costs and avoid stockouts by accurately predicting demand for different car models.
- **Enhance Risk Assessment:** Make informed decisions about car valuations, reducing risks associated with loans, insurance claims, and investments.
- **Gain Market Insights**: Understand the factors that drive car prices, enabling businesses to develop better products, target specific customer segments, and stay ahead of market trends.
- **Increase Customer Satisfaction:** By offering fair and transparent pricing, businesses can build trust with customers and enhance their overall satisfaction.

ML Model - 3

```
# ML Model - 3: Lasso Regression

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score

# Assuming 'encoded_df' is your DataFrame with features and target variable
X = encoded_df.drop('price', axis=1)
y = encoded_df['price']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the Lasso Regression model
```

```
lasso_model = Lasso(alpha=1.0)  # You can adjust the alpha value

# Fit the model to the training data
lasso_model.fit(X_train, y_train)

# Predict on the test data
y_pred = lasso_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 7905936.94342683
R-squared: 0.8998538856352574
```

1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

```
# Visualizing evaluation Metric Score chart
import matplotlib.pyplot as plt
import numpy as np

# Assuming you have already calculated mse and r2 for the Lasso Regression model
# Replace these with your actual values
mse = 18785897.23  # Example MSE value
r2 = 0.83  # Example R-squared value

# Create a bar chart
metrics = ['MSE', 'R-squared']
scores = [mse, r2]

plt.figure(figsize=(8, 6))
plt.bar(metrics, scores, color=['blue', 'green'])
plt.title('Evaluation Metric Scores for Lasso Regression')
plt.ylabel('Score')
plt.xlabel('Metric')
```

```
# Visualizing evaluation Metric Score chart
import matplotlib.pyplot as plt
import numpy as np

# Assuming you have already calculated mse and r2 for the Lasso Regression model
# Replace these with your actual values
mse = 18785897.23  # Example MSE value
r2 = 0.83  # Example R-squared value

# Create a bar chart
metrics = ['MSE', 'R-squared']
scores = [mse, r2]

plt.figure(figsize=(8, 6))
plt.bar(metrics, scores, color=['blue', 'green'])
plt.title('Evaluation Metric Scores for Lasso Regression')
plt.ylabel('Score')
plt.xlabel('Metric')

# Add value labels on top of bars
for i, v in enumerate(scores):
    plt.text(i, v, str(round(v, 2)), ha='center', va='bottom')

plt.show()
```