MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous)

NEW PANVEL

PROJECT REPORT ON

"Predictive Analysis and Insights on Car Attributes Using Machine Learning"

IN PARTIAL FULFILLMENT OF

BACHELOR OF COMPUTER SCIENCE

SEMESTER VI -- 2024-45

PROJECT GUIDE

NAME: PROF. SANJANA BHANGALE

SUBMITTED BY: SHREEYA N. PARAB

ROLL NO: 9156

Mahatma Education Society's
Pillai College of Arts, Commerce & Science
(Autonomous)
Affiliated to University of Mumbai
NAAC Accredited 'A' grade (3 cycles)
Best College Award by University of Mumbai
ISO 9001:2015 Certified

Evaluation sheet for continuous assessment with rubrics

Course: Computer Science
Subject: Machine Learning
Details about the continuous Assessment 2/Project work : Predictive Analysis and Insights on Car Attributes
Name of the Student: Shreeya N. Parab
Roll Number: 9156
Class / Division: TYCS 'A'
Name of Evaluator: Prof. Sanjana Bhangale

Please circle appropriate score

Grading Criteria	Fair	Good	Excellent	Total
Introduction/ Description of the Case	1	2	3	/3
SWOT Analysis of the company used for case analysis pertaining to the case (strength of CA2 topic: e.g main important feature of CA1, Weakness: limitations of the project, Opportunities: in carrier in the future, Threat: obstacles that can cause failure to project CA2	3	4	5	/5
Learnings from the case	2	3	4	/4
Delivery/presentation skills	1	2	3	/3
Total				/15

Co-ordinator,
Shubhangi Pawar

Mahatma Education Society's
Pillai College of Arts, Commerce & Science
(Autonomous)
Affiliated to University of Mumbai
NAAC Accredited 'A' grade (3 cycles)
Best College Award by University of Mumbai
ISO 9001:2015 Certified

CERTIFICATE

This is to certify that **Ms. Shreeya N. Parab** of T.Y B.Sc. C.S. **Semester VI** has completed the Project work of the Subject Of Machine Learning during the academic year **2024-25** under the guidance of **Prof. Sanjana Bhangale** is the partial requirement for the fulfilment of the curriculum of the Degree of Bachelor of Science in Computer Science, University of Mumbai.

Place:

Date:

Name & Signature of faculty

Name & Signature of external

Name & Signature of Co-ordinator

Dr. K.M. Vasudevan Pillai Campus, Sector 16, New Panvel - 410206. Tel : 27456100 / 1700 |
Fax : 27483208 | Website : www.pcacs.ac.in

Predictive Analysis and Insights on Car Attributes Using Machine Learning: A Study with CarDekho Dataset

1. INTRODUCTION

The used car market has become an essential part of the global automotive industry, growing rapidly due to factors such as affordability, increased environmental awareness, and the shift towards more sustainable practices. In many countries, consumers increasingly prefer used cars over new ones because of their lower prices, yet they still offer a wide variety of options in terms of make, model, and features. For businesses operating in the used car market, having a deep understanding of what drives the pricing of used cars is crucial. This knowledge allows businesses to set competitive prices, optimize sales strategies, and increase customer satisfaction.

This project leverages data from CarDekho, one of the largest online platforms for buying and selling used cars in India, to explore the factors that influence the selling price of used vehicles. By analyzing key features such as the car's brand, year of manufacture, mileage, fuel type, transmission type, engine capacity, and number of previous owners, this project aims to uncover patterns that can help businesses predict the price of used cars more accurately. The results of this analysis will not only offer insights into pricing strategies but also assist consumers in making more informed purchasing decisions.

Through machine learning techniques, the project will develop a predictive model that can estimate the selling price of a used car based on its characteristics. By examining the correlation between various features and the car's selling price, the model will offer actionable insights to sellers and buyers, optimizing the pricing process and contributing to a more efficient used car market. Ultimately, this project aims to demonstrate the value of data-driven insights in transforming the way used cars are bought and sold, benefiting both businesses and consumers.

2. BUSINESS UNDERSTANDING

The used car market is a vital and expanding segment within the automotive industry. With factors such as affordability, increasing environmental awareness, and a shift toward sustainability, the demand for pre-owned vehicles has surged. Consumers prefer used cars due to their more accessible pricing compared to new vehicles, while still offering diverse options in terms of brand, model, and features. For businesses involved in the used car market, understanding the various factors that influence car pricing is crucial for setting competitive prices, improving sales strategies, and enhancing customer satisfaction.

3. DOMAIN OF THE DATASET

The dataset used in this project is sourced from CarDekho, a leading online platform for buying and selling used cars. It focuses on the automotive industry, specifically the used car market. The data includes key features such as the car's brand, year of manufacture, mileage, fuel type, transmission type, number of owners, engine capacity, and selling price. These variables are crucial for determining the value and pricing of a used car and provide insight into how they interact to influence market trends.

4. WHY THIS DATASET WAS SELECTED

This dataset was chosen because it contains a variety of attributes that directly impact pricing and purchasing decisions in the used car market. Analyzing variables like car age, mileage, fuel type, transmission, and brand can help businesses understand market behavior and refine their pricing strategies.

Additionally, the dataset's diversity in terms of car brands, fuel types, and other factors provides a comprehensive overview of the market, making it ideal for conducting in-depth analysis and developing predictive models for price forecasting.

5. IMPORTANCE OF THE DATASET

The CarDekho dataset holds significant importance for several reasons:

1. Market Insights: The dataset offers a clear picture of the used car market, enabling businesses and sellers to gain valuable insights into pricing trends and consumer preferences.
2. Price Optimization: By examining the factors influencing selling price, businesses can optimize their pricing strategies to stay competitive in the market.
3. Consumer Decision Making: For buyers, the dataset offers insights into how variables like vehicle age, mileage, and fuel type affect pricing, allowing them to make more informed purchasing decisions.
4. Predictive Analytics: The dataset supports the development of machine learning models capable of predicting car prices based on key features, helping both buyers and sellers assess car values more accurately.

6. IMPACT ON THE FUTURE AND SOCIETY

The impact of this dataset on the future and society can be seen in various ways:

1. Sustainability: By understanding and optimizing the used car market, businesses can encourage the resale of vehicles, contributing to a more sustainable economy. This reduces waste and lowers the demand for new vehicle manufacturing, helping to reduce the environmental impact of the automotive industry.
2. Access to Affordable Vehicles: The availability of competitively priced used cars enhances mobility for individuals, especially in developing economies where access to new cars might be limited.
3. Enhanced Consumer Awareness: With more data-driven insights available, consumers can make better-informed decisions when purchasing used cars, leading to greater market transparency and more fair pricing.
4. Innovation in Business Models: As businesses gain deeper insights into the used car market, they can create innovative business models such as online car auctions, car subscription services, or leasing, which could transform how people access vehicles.
5. Economic Growth: The used car market contributes significantly to the economy by generating sales, creating jobs within the automotive sector, and boosting industries related to car maintenance, insurance, and financing.

1. Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
from six.moves import urllib

warnings.filterwarnings("ignore")

%matplotlib inline
```

```
[2] df = pd.read_csv("/content/9156Shreeya_Cardekhodataset.csv")
```

2. Understanding the data

```
df.head()
```

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_type	transmission_type	mileage	engine	max_power
0	0	Maruti Alto	Maruti	Alto	9	120000	Individual	Petrol	Manual	19.70	796	46.3
1	1	Hyundai Grand	Hyundai	Grand	5	20000	Individual	Petrol	Manual	18.90	1197	82.0
2	2	Hyundai i20	Hyundai	i20	11	60000	Individual	Petrol	Manual	17.00	1197	80.0
3	3	Maruti Alto	Maruti	Alto	9	37000	Individual	Petrol	Manual	20.92	998	67.1
4	4	Ford Ecosport	Ford	Ecosport	6	30000	Dealer	Diesel	Manual	22.77	1498	98.5

```
print('The size of Dataframe is: ', df.shape)
print('-'*100)
print('The Column Name, Record Count and Data Types are as follows: ')
df.info()
print('-'*100)
```

```
The size of Dataframe is:  (15411, 14)
----------------------------------------------------------------------------------------------------
The Column Name, Record Count and Data Types are as follows:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15411 entries, 0 to 15410
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         15411 non-null  int64
 1   car_name           15411 non-null  object
 2   brand              15411 non-null  object
 3   model              15411 non-null  object
 4   vehicle_age        15411 non-null  int64
 5   km_driven          15411 non-null  int64
 6   seller_type        15411 non-null  object
 7   fuel_type          15411 non-null  object
 8   transmission_type  15411 non-null  object
 9   mileage            15411 non-null  float64
 10  engine             15411 non-null  int64
 11  max_power          15411 non-null  float64
 12  seats              15411 non-null  int64
 13  selling_price      15411 non-null  int64
dtypes: float64(2), int64(6), object(6)
memory usage: 1.6+ MB
----------------------------------------------------------------------------------------------------
```

```
[5] # Defining numerical & categorical columns
    numeric_features = [feature for feature in df.columns if df[feature].dtype != 'O']
    categorical_features = [feature for feature in df.columns if df[feature].dtype == 'O']

    # print columns
    print('We have {} numerical features : {}'.format(len(numeric_features), numeric_features))
    print('\nWe have {} categorical features : {}'.format(len(categorical_features), categorical_features))
```

```
We have 8 numerical features : ['Unnamed: 0', 'vehicle_age', 'km_driven', 'mileage', 'engine', 'max_power', 'seats', 'selling_price']

We have 6 categorical features : ['car_name', 'brand', 'model', 'seller_type', 'fuel_type', 'transmission_type']
```


```
[5] # Defining numerical & categorical columns
    numeric_features = [feature for feature in df.columns if df[feature].dtype != 'O']
    categorical_features = [feature for feature in df.columns if df[feature].dtype == 'O']

    # print columns
    print('We have {} numerical features : {}'.format(len(numeric_features), numeric_features))
    print('\nWe have {} categorical features : {}'.format(len(categorical_features), categorical_features))
```

```
We have 8 numerical features : ['Unnamed: 0', 'vehicle_age', 'km_driven', 'mileage', 'engine', 'max_power', 'seats', 'selling_price']

We have 6 categorical features : ['car_name', 'brand', 'model', 'seller_type', 'fuel_type', 'transmission_type']
```


```
[7] print('Summary Statistics of numerical features for DataFrame are as follows:')
    print('-'*100)
    df.describe()
```

```
Summary Statistics of numerical features for DataFrame are as follows:
----------------------------------------------------------------------------------------------------
```

	Unnamed: 0	vehicle_age	km_driven	mileage	engine	max_power	seats	selling_price
count	15411.000000	15411.000000	1.541100e+04	15411.000000	15411.000000	15411.000000	15411.000000	1.541100e+04
mean	9811.857699	6.036338	5.561648e+04	19.701151	1486.057751	100.588254	5.325482	7.749711e+05
std	5643.418542	3.013291	5.161855e+04	4.171265	521.106696	42.972979	0.807628	8.941284e+05
min	0.000000	0.000000	1.000000e+02	4.000000	793.000000	38.400000	0.000000	4.000000e+04
25%	4906.500000	4.000000	3.000000e+04	17.000000	1197.000000	74.000000	5.000000	3.850000e+05
50%	9872.000000	6.000000	5.000000e+04	19.670000	1248.000000	88.500000	5.000000	5.560000e+05
75%	14668.500000	8.000000	7.000000e+04	22.700000	1582.000000	117.300000	5.000000	8.250000e+05
max	19543.000000	29.000000	3.800000e+06	33.540000	6592.000000	626.000000	9.000000	3.950000e+07

```
[8] print('Summary Statistics of categorical features for DataFrame are as follows:')
    print('-'*100)
    df.describe(include= 'object')
```

```
Summary Statistics of categorical features for DataFrame are as follows:
----------------------------------------------------------------------------------------------------
```

	car_name	brand	model	seller_type	fuel_type	transmission_type
count	15411	15411	15411	15411	15411	15411
unique	121	32	120	3	5	2
top	Hyundai i20	Maruti	i20	Dealer	Petrol	Manual
freq	906	4992	906	9539	7643	12225

3. Exploratory data Analysis

```
# Set the "Road Ahead" theme for visualizations
sns.set_theme(style="whitegrid", palette="muted")
custom_palette = sns.color_palette(["#004AAD", "#008000", "#56B4E9", "#FF8C00", "#C70039"])
```

Analytical Questions

1. What is the correlation between vehicle_age and selling_price?
"*Analyze if older cars tend to have lower prices and how the vehicle age impacts the selling price.*"

```
[41] # Calculate the correlation between vehicle_age and selling_price
correlation = df['vehicle_age'].corr(df['selling_price'])

print(f"The correlation between vehicle_age and selling_price is: {correlation}")

# Visualize the relationship using a scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='vehicle_age', y='selling_price', data=df, color=custom_palette[0])
plt.title('Correlation between Vehicle Age and Selling Price')
plt.xlabel('Vehicle Age (Years)')
plt.ylabel('Selling Price')
plt.show()
```

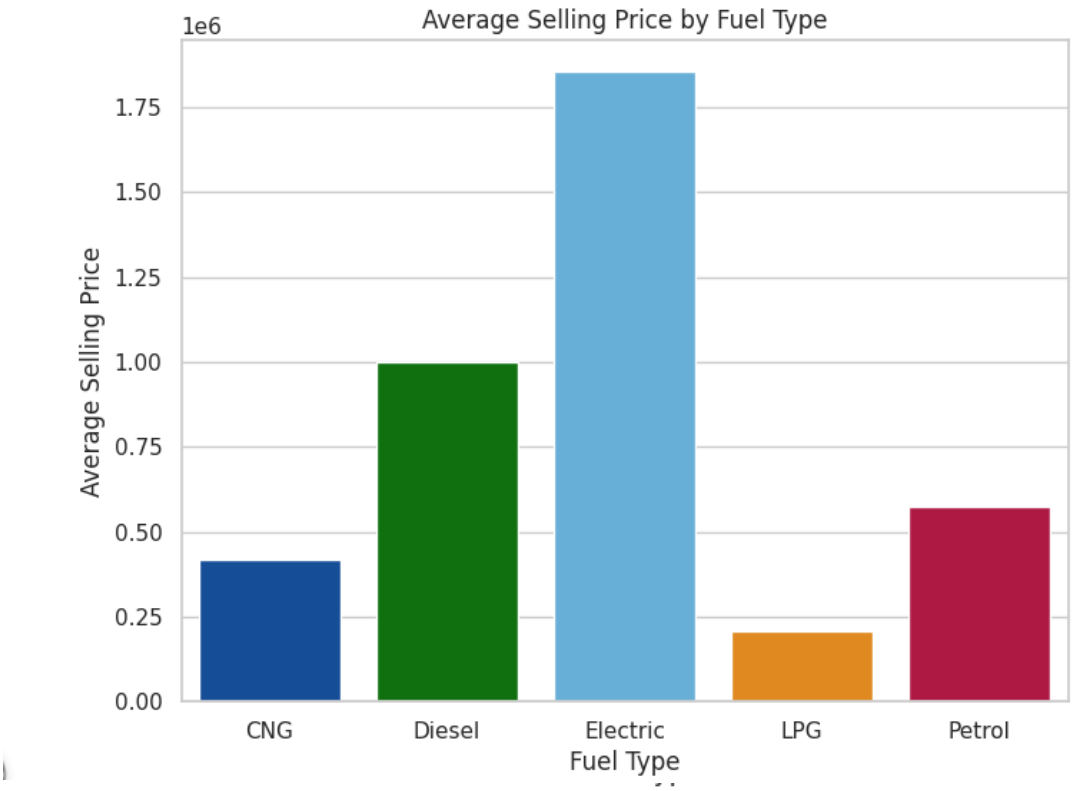
```
The correlation between vehicle_age and selling_price is: -0.2418514586696868
```

2. How does fuel_type (Petrol, Diesel) affect the selling_price of cars?

"Compare the average selling price of cars based on fuel type and identify if there is a significant price difference."

```
[24] # Calculate the average selling price for each fuel type
     average_price_by_fuel = df.groupby('fuel_type')['selling_price'].mean()
     print(average_price_by_fuel)
     # Visualize the average selling price by fuel type
     plt.figure(figsize=(8, 6))
     sns.barplot(x=average_price_by_fuel.index, y=average_price_by_fuel.values, palette=custom_palette)
     plt.title('Average Selling Price by Fuel Type')
     plt.xlabel('Fuel Type')
     plt.ylabel('Average Selling Price')
     plt.show()
     # Perform a statistical test (e.g., t-test) to check for significant difference
     from scipy.stats import ttest_ind
     petrol_prices = df[df['fuel_type'] == 'Petrol']['selling_price']
     diesel_prices = df[df['fuel_type'] == 'Diesel']['selling_price']
     t_statistic, p_value = ttest_ind(petrol_prices, diesel_prices)
     print(f"T-statistic: {t_statistic}")
     print(f"P-value: {p_value}")
     alpha = 0.05  # Significance level
     if p_value < alpha:
         print("There is a statistically significant difference in selling prices between petrol and diesel cars.")
     else:
         print("There is no statistically significant difference in selling prices between petrol and diesel cars.")
```

```
fuel_type
CNG         4.176877e+05
Diesel      1.000469e+06
Electric    1.853500e+06
LPG         2.062727e+05
Petrol      5.728619e+05
Name: selling_price, dtype: float64
```




```
T-statistic: -29.942342756530383
P-value: 2.139739197558993e-191
There is a statistically significant difference in selling prices between petrol and diesel cars.
```

.

3. What is the relationship between km_driven and selling_price?

" Investigate if higher mileage (km driven) leads to a decrease in selling price."

```
correlation = df['km_driven'].corr(df['selling_price'])
print(f"Correlation Coefficient between KM Driven and Selling Price: {correlation:.4f}")
km_stats = df['km_driven'].describe()
km_stats = km_stats.apply(lambda x: f"{x:,.0f}" if isinstance(x, (int, float)) else x)
price_stats = df['selling_price'].describe()
price_stats = price_stats.apply(lambda x: f"₹{x:,.0f}" if isinstance(x, (int, float)) else x)
print("\n--- KM Driven Statistics ---")
print(km_stats)
print("\n--- Selling Price Statistics ---")
print(price_stats)
# Interpretation based on correlation
if correlation < 0:
    print("\nThere is a negative correlation, suggesting that cars with higher mileage tend to have lower selling prices.")
elif correlation > 0:
    print("\nThere is a positive correlation, suggesting that cars with higher mileage tend to have higher selling prices.")
else:
    print("\nThere is no correlation between KM Driven and Selling Price.")
```

```
Correlation Coefficient between KM Driven and Selling Price: -0.0800

--- KM Driven Statistics ---
count       15,411
mean        55,616
std         51,619
min            100
25%         30,000
50%         50,000
75%         70,000
max      3,800,000
Name: km_driven, dtype: object

--- Selling Price Statistics ---
count        ₹15,411
mean        ₹774,971
std         ₹894,128
min          ₹40,000
25%         ₹385,000
50%         ₹556,000
75%         ₹825,000
max      ₹39,500,000
Name: selling_price, dtype: object

There is a negative correlation, suggesting that cars with higher mileage tend to have lower selling prices.
```

4. How does transmission_type (Manual, Automatic) affect the selling_price?
"Compare the average selling price between cars with manual and automatic transmission types."

```
[ ]  avg_price_transmission = df.groupby('transmission_type')['selling_price'].mean()
     avg_price_transmission = avg_price_transmission.apply(lambda x: f"₹{x:,.0f}")
     print("Average Selling Price by Transmission Type:")
     print(avg_price_transmission)
```

```
Average Selling Price by Transmission Type:
transmission_type
Automatic    ₹1,579,557
Manual         ₹565,285
Name: selling_price, dtype: object
```

5. What is the average selling_price for cars by brand?
"Calculate and compare the average price for each car brand (e.g., Maruti, Hyundai, Renault)."

```
print(avg_price_brand)
print("Average Selling Price by Brand:")
avg_price_brand = avg_price_brand.apply(lambda x: f"₹{x:,.0f}")
avg_price_brand = df.groupby('brand')['selling_price'].mean().sort_values(ascending=False)
```

5. Average Selling Price by Brand

```
Average Selling Price by Brand:
brand
Ferrari          ₹39,500,000
Rolls-Royce      ₹24,200,000
Bentley           ₹9,266,667
Maserati          ₹6,100,000
Porsche           ₹5,161,190
Lexus             ₹5,146,500
Mercedes-AMG      ₹5,100,000
Land Rover        ₹3,823,902
Volvo             ₹3,729,700
BMW               ₹2,693,827
Jaguar            ₹2,643,034
Mercedes-Benz     ₹2,480,742
Mini              ₹2,182,647
Audi              ₹1,966,865
ISUZU             ₹1,897,500
Jeep              ₹1,795,805
MG                ₹1,752,947
Kia               ₹1,735,250
Toyota            ₹1,371,317
Isuzu             ₹1,355,000
Nissan              ₹955,364
Mahindra            ₹787,455
Skoda               ₹784,090
Force               ₹700,000
Tata                ₹683,535
Ford                ₹645,224
Honda               ₹617,757
Hyundai             ₹576,154
Volkswagen          ₹516,547
Maruti              ₹487,089
Renault             ₹440,985
Datsun              ₹320,518
Name: selling_price, dtype: object
```

6. What is the distribution of selling_price based on vehicle_age?

" Analyze how the selling price varies with different vehicle ages to see if older cars generally cost less."

```
# Analyze how selling_price varies with vehicle age
avg_price_age = df.groupby('vehicle_age')['selling_price'].mean()
avg_price_age = avg_price_age.apply(lambda x: f"₹{x:,.0f}")
print("Average Selling Price by Vehicle Age:")
print(avg_price_age)
```

```
Average Selling Price by Vehicle Age:
vehicle_age
0     ₹2,230,000
1     ₹1,096,977
2     ₹1,143,023
3       ₹977,095
4       ₹950,943
5       ₹818,893
6       ₹728,515
7       ₹699,076
8       ₹625,945
9       ₹592,466
10      ₹496,148
11      ₹426,238
12      ₹318,290
13      ₹278,485
14      ₹260,682
15      ₹188,376
16      ₹126,240
17      ₹156,471
18      ₹125,000
19       ₹74,000
21       ₹73,333
22       ₹70,000
25    ₹1,000,000
29       ₹60,000
Name: selling_price, dtype: object
```

7. How does the engine_capacity (engine) affect the selling_price of cars?
" Examine whether cars with larger engine capacities are priced higher than cars with smaller engine capacities."

7. Engine capacity vs Selling Price

```
[ ] # Compute average selling price by engine capacity (grouped)
    df['engine_bins'] = pd.cut(df['engine'], bins=[0, 1000, 1500, 2000, 2500, 3000, 4000], labels=[
        '0-1000', '1000-1500', '1500-2000', '2000-2500', '2500-3000', '3000-4000'])
    avg_price_engine = df.groupby('engine_bins')['selling_price'].mean()
    avg_price_engine = avg_price_engine.apply(lambda x: f"₹{x:,.0f}")
    print("Average Selling Price by Engine Capacity:")
    print(avg_price_engine)
```

```
Average Selling Price by Engine Capacity:
engine_bins
0-1000         ₹328,774
1000-1500      ₹561,974
1500-2000    ₹1,278,281
2000-2500    ₹1,203,003
2500-3000    ₹2,100,115
3000-4000    ₹3,762,451
Name: selling_price, dtype: object
```

8. Is there a relationship between mileage (km per liter) and selling_price?
"Investigate whether cars with better mileage are priced higher than those with lower mileage."

8. Mileage vs Selling Price

```
[ ] # Compute average selling price by mileage (grouped)
    df['mileage_bins'] = pd.cut(df['mileage'], bins=[0, 10, 15, 20, 25, 30, 50], labels=[
        '0-10', '10-15', '15-20', '20-25', '25-30', '30+'])
    avg_price_mileage = df.groupby('mileage_bins')['selling_price'].mean()
    avg_price_mileage = avg_price_mileage.apply(lambda x: f"₹{x:,.0f}")
    print("Average Selling Price by Mileage Range:")
    print(avg_price_mileage)
```

```
Average Selling Price by Mileage Range:
mileage_bins
0-10     ₹4,329,519
10-15    ₹1,431,519
15-20      ₹775,138
20-25      ₹573,799
25-30      ₹565,935
30+        ₹356,340
Name: selling_price, dtype: object
```

9. How does the seller_type (Individual, Dealer) affect the selling_price of cars?
" Compare the selling prices of cars sold by individual sellers and dealers."

9. Seller type vs Selling Price

```
# Compute average selling price by seller type
avg_price_seller = df.groupby('seller_type')['selling_price'].mean()
avg_price_seller = avg_price_seller.apply(lambda x: f"₹{x:,.0f}")
print("Average Selling Price by Seller Type:")
print(avg_price_seller)
```

```
Average Selling Price by Seller Type:
seller_type
Dealer              ₹872,506
Individual          ₹617,880
Trustmark Dealer    ₹571,960
Name: selling_price, dtype: object
```

10. What is the distribution of selling_price for different seats configurations (e.g., 5 seats)?
"Analyze if the number of seats in a car impacts its selling price by comparing cars with different seating capacities."

10. Selling Price for Different Seat Configurations

```
# Compute average selling price by seating capacity
avg_price_seats = df.groupby('seats')['selling_price'].mean()
avg_price_seats = avg_price_seats.apply(lambda x: f"₹{x:,.0f}")
print("Average Selling Price by Seat Configuration:")
print(avg_price_seats)
```

```
Average Selling Price by Seat Configuration:
seats
0      ₹634,500
2    ₹4,124,286
4    ₹3,974,597
5      ₹694,254
6      ₹646,787
7    ₹1,164,031
8      ₹929,360
9      ₹648,000
Name: selling_price, dtype: object
```

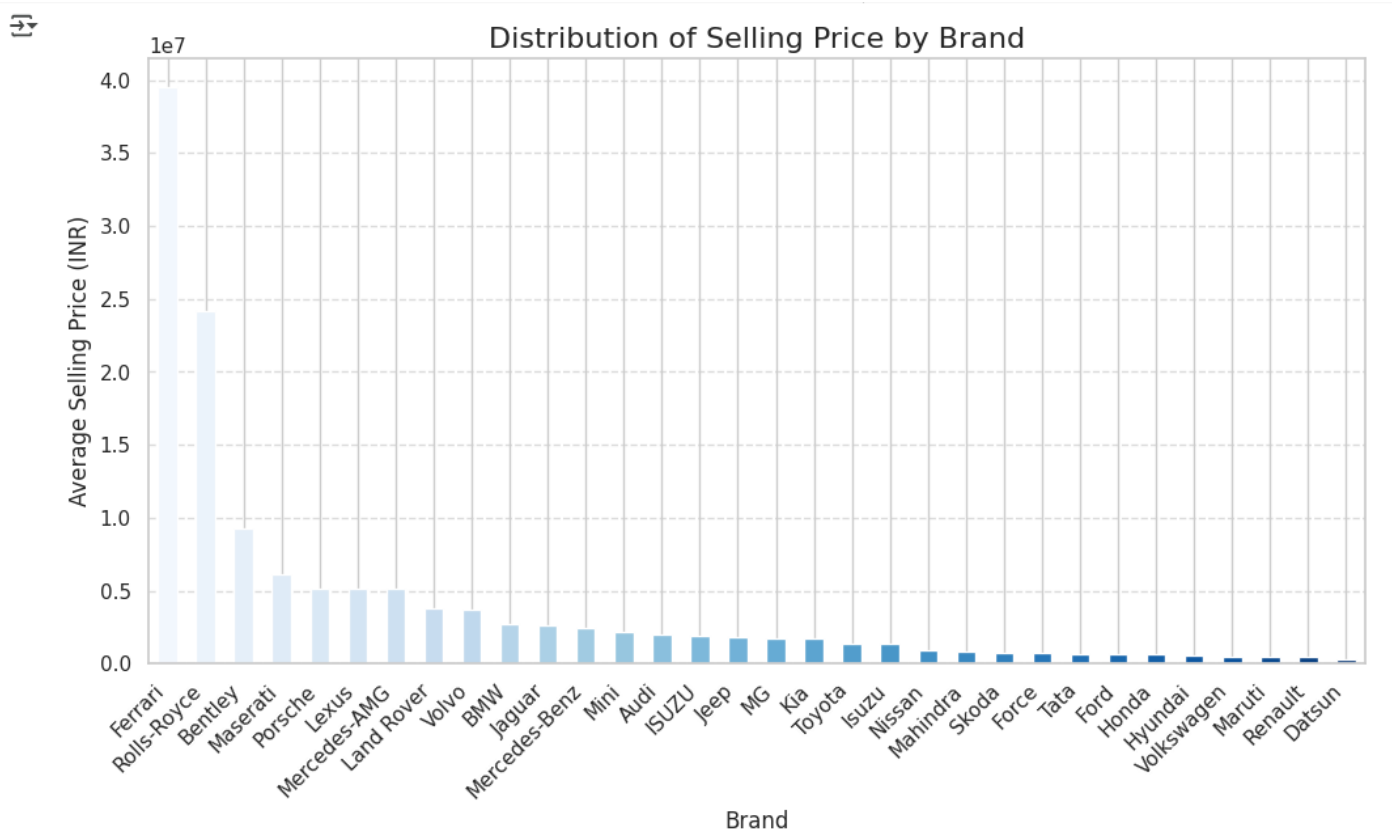
4. Visualization

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Customizing the "Road Ahead" Theme
custom_palette = sns.color_palette("muted")
```

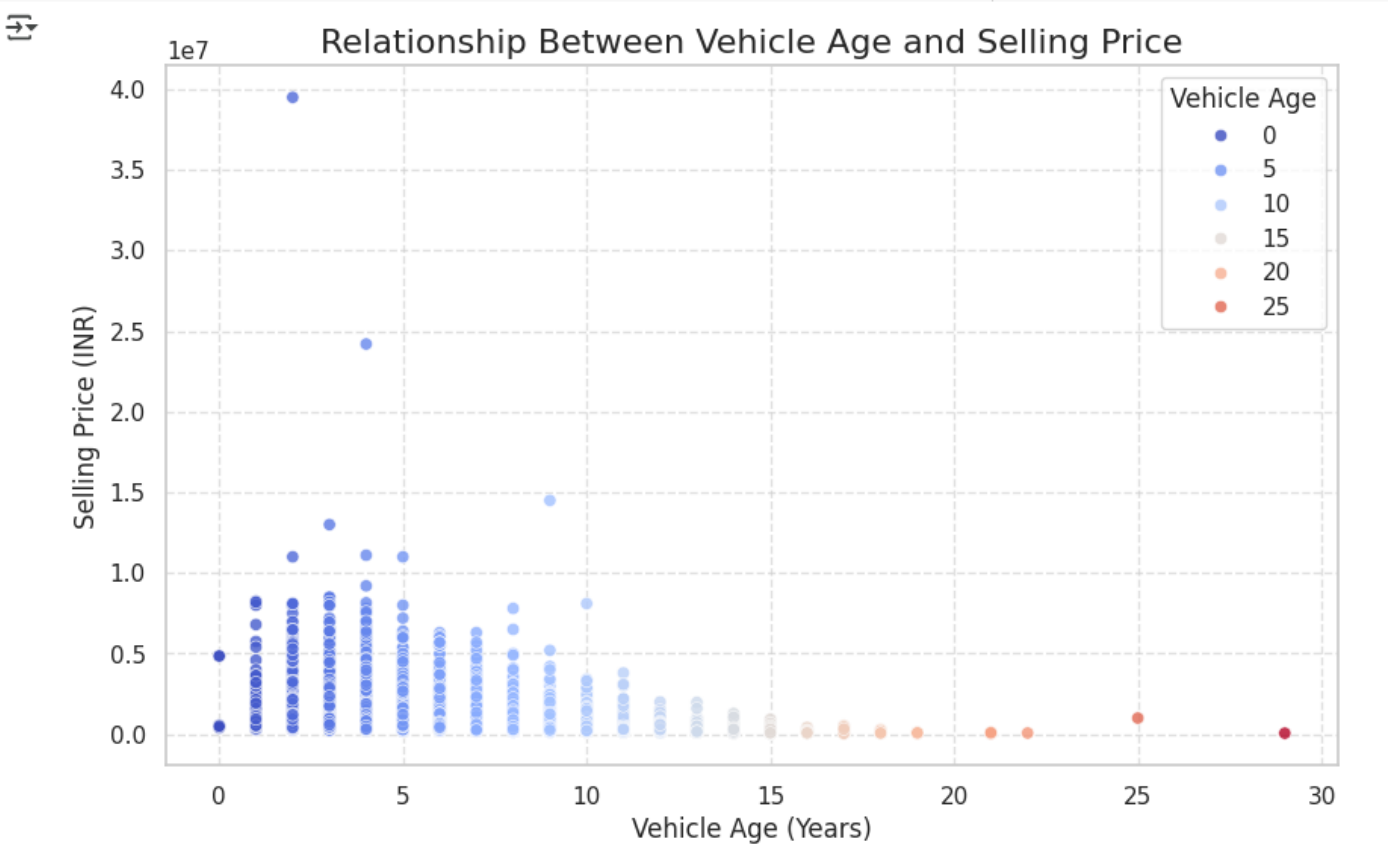
1. Create a bar plot to show the distribution of selling_price by brand.
"This will visualize how prices are distributed across different brands like Maruti, Hyundai, and Renault."

```
plt.figure(figsize=(12, 6))
avg_price_by_brand = df.groupby('brand')['selling_price'].mean().sort_values(ascending=False)
avg_price_by_brand.plot(kind='bar', color=sns.color_palette("Blues", len(avg_price_by_brand)))
plt.title("Distribution of Selling Price by Brand", fontsize=16)
plt.xlabel("Brand", fontsize=12)
plt.ylabel("Average Selling Price (INR)", fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



2. Generate a scatter plot to visualize the relationship between vehicle_age and selling_price.
“ This scatter plot will help understand if there's an inverse relationship between age and price.”

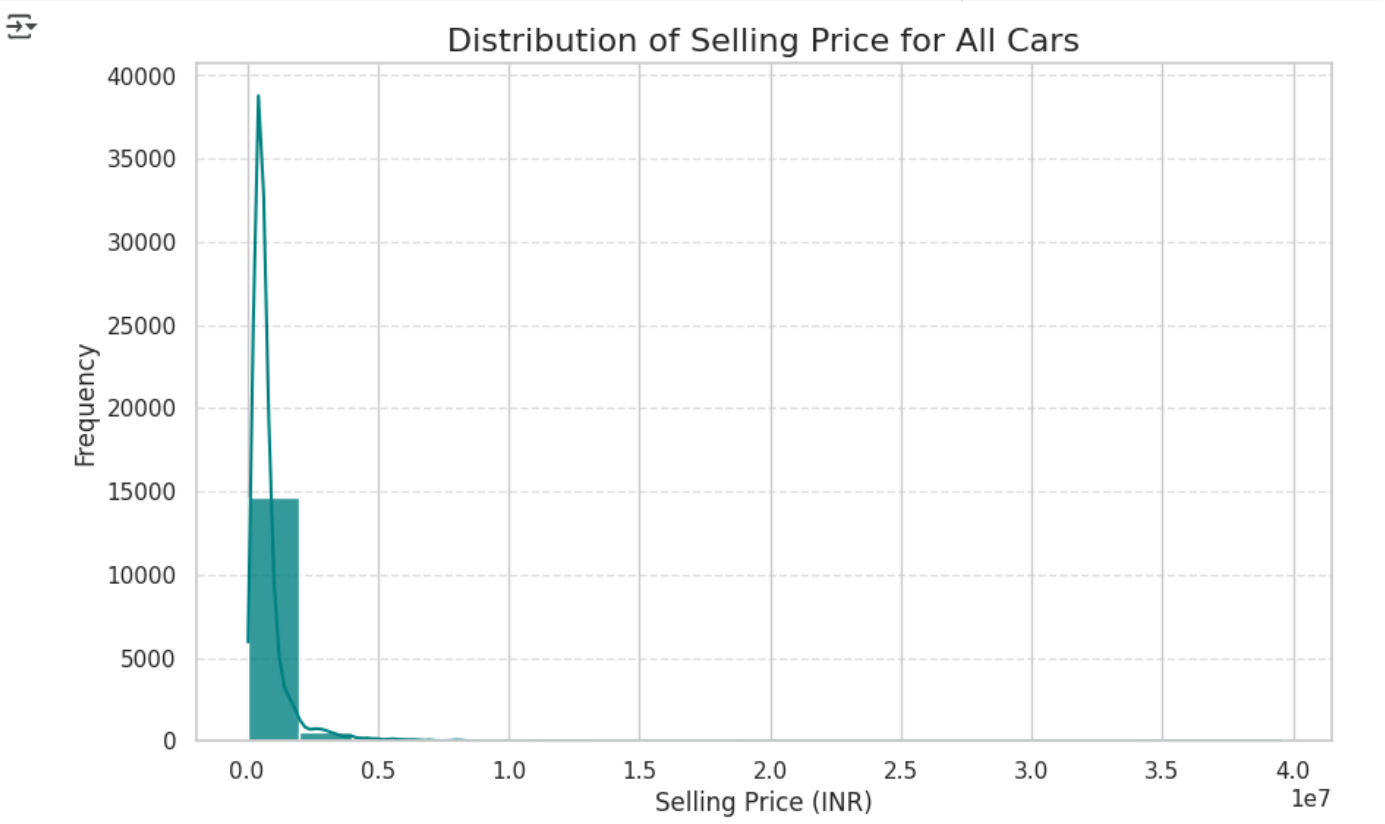
```
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=df, x='vehicle_age', y='selling_price',
    hue='vehicle_age', palette="coolwarm", alpha=0.8
)
plt.title("Relationship Between Vehicle Age and Selling Price", fontsize=16)
plt.xlabel("Vehicle Age (Years)", fontsize=12)
plt.ylabel("Selling Price (INR)", fontsize=12)
plt.legend(title="Vehicle Age", loc='upper right')
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```



3. Create a histogram to visualize the distribution of selling_price for all cars.
" A histogram will help understand the price range and its frequency across all cars in the dataset."

3. Histogram: Distribution of selling_price

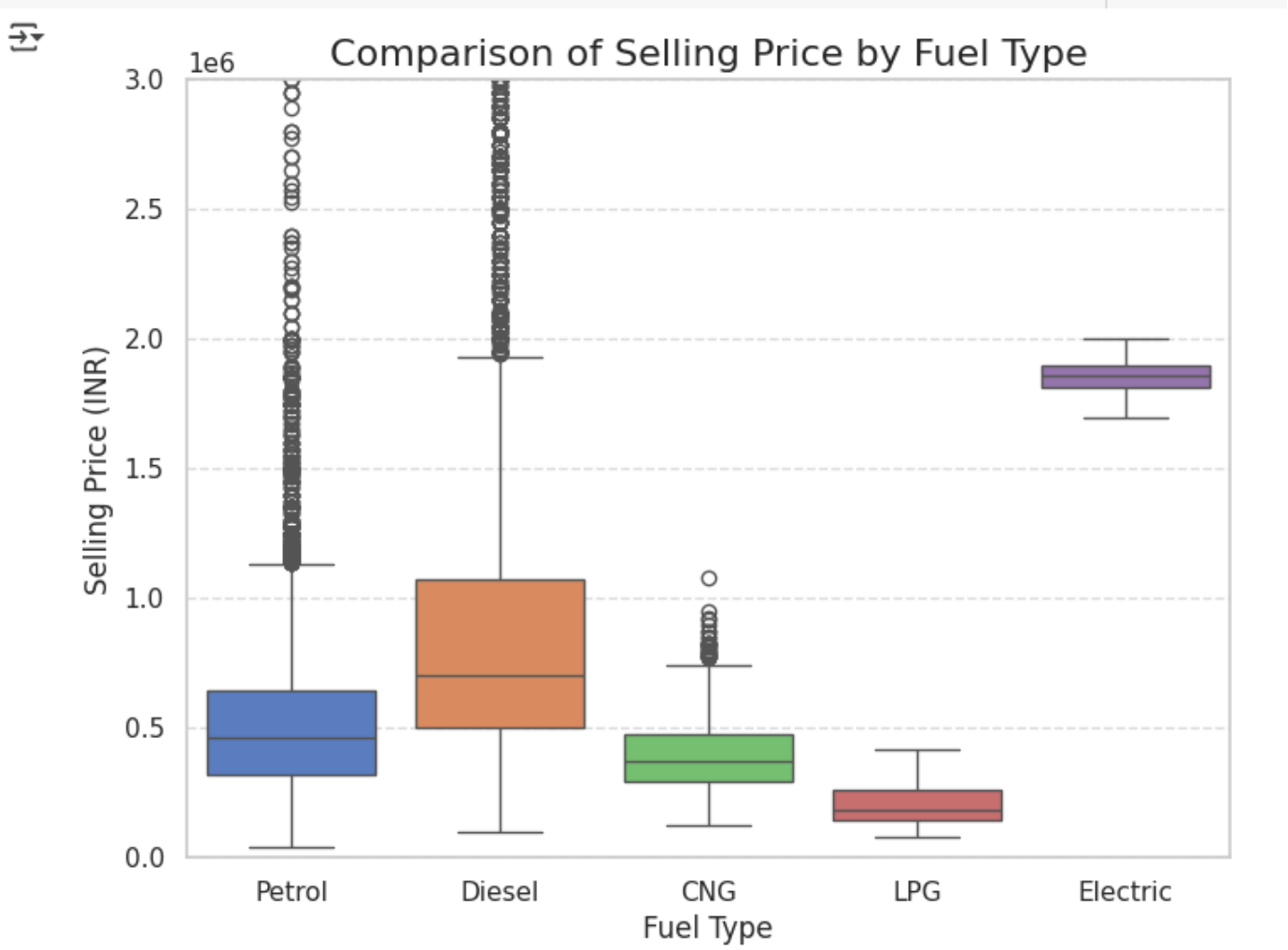
```
[ ]  plt.figure(figsize=(10, 6))
     sns.histplot(df['selling_price'], bins=20, kde=True, color="teal", alpha=0.8)
     plt.title("Distribution of Selling Price for All Cars", fontsize=16)
     plt.xlabel("Selling Price (INR)", fontsize=12)
     plt.ylabel("Frequency", fontsize=12)
     plt.grid(axis='y', linestyle='--', alpha=0.6)
     plt.show()
```



4. Generate a box plot to compare selling_price by fuel_type (Petrol, Diesel).
" This will help visualize how the selling price differs for cars with petrol and diesel engines."

4. Box Plot: Compare selling_price by fuel_type

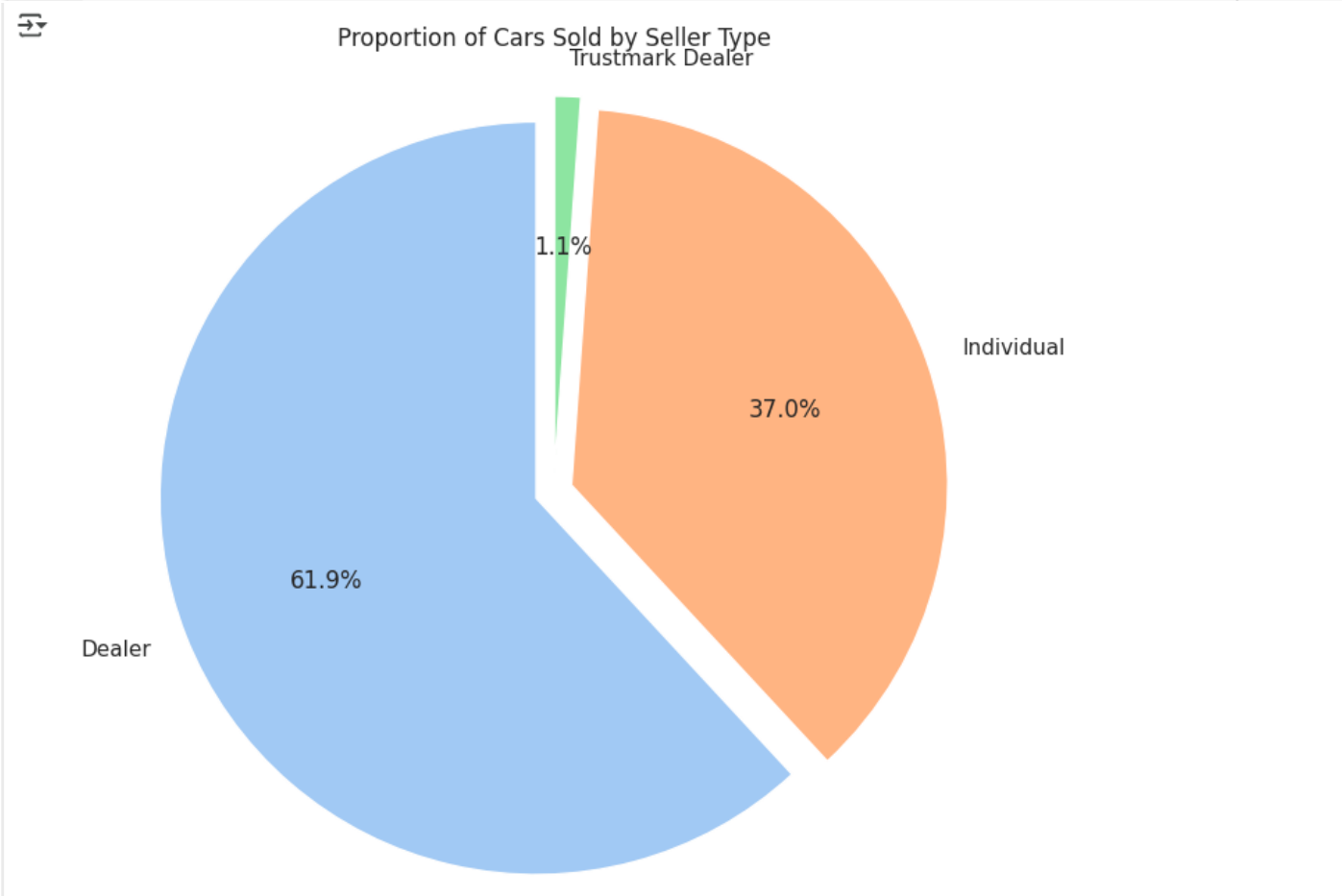
```
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='fuel_type', y='selling_price', palette=custom_palette)
plt.title("Comparison of Selling Price by Fuel Type", fontsize=16)
plt.xlabel("Fuel Type", fontsize=12)
plt.ylabel("Selling Price (INR)", fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.ylim(0, 3_000_000)
plt.show()
```



5. Create a pie chart showing the proportion of cars sold by seller_type (Individual vs Dealer).
" This pie chart will show the proportion of cars listed by individual sellers versus dealers."

5. Pie Chart: Proportion of cars by seller_type

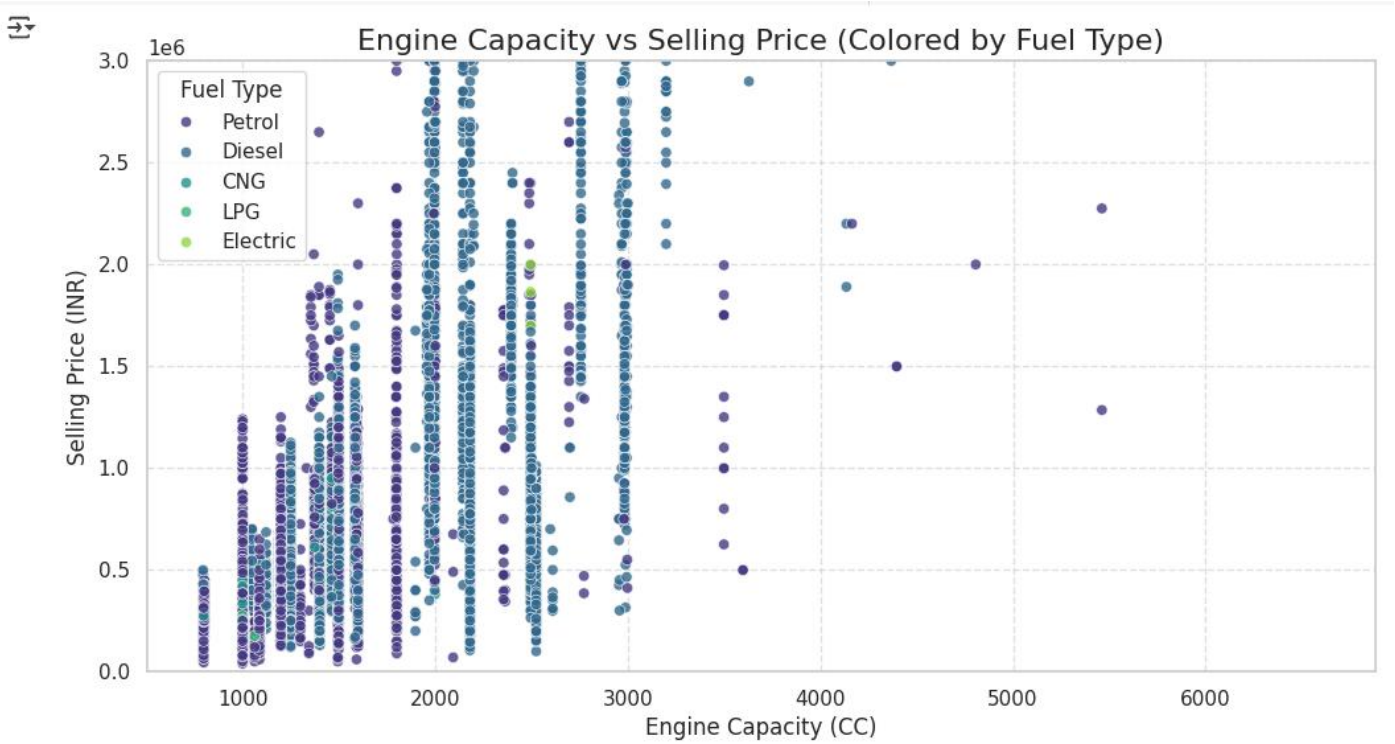
```
plt.figure(figsize=(8, 8))
seller_counts = df['seller_type'].value_counts()
explode_values = [0.05] * len(seller_counts)  # Explode all slices slightly
plt.pie(
    seller_counts,
    labels=seller_counts.index,
    autopct='%1.1f%%',
    startangle=90,
    colors=sns.color_palette("pastel"),
    explode=explode_values  # Ensure the explode array matches the number of categories
)
plt.title("Proportion of Cars Sold by Seller Type", fontsize=12)
plt.axis('equal')  # Ensure the pie chart is a perfect circle
plt.show()
```



6. Generate a scatter plot to visualize engine_capacity versus selling_price, colored by fuel_type.
"This scatter plot will help compare how engine size and fuel type influence the price."

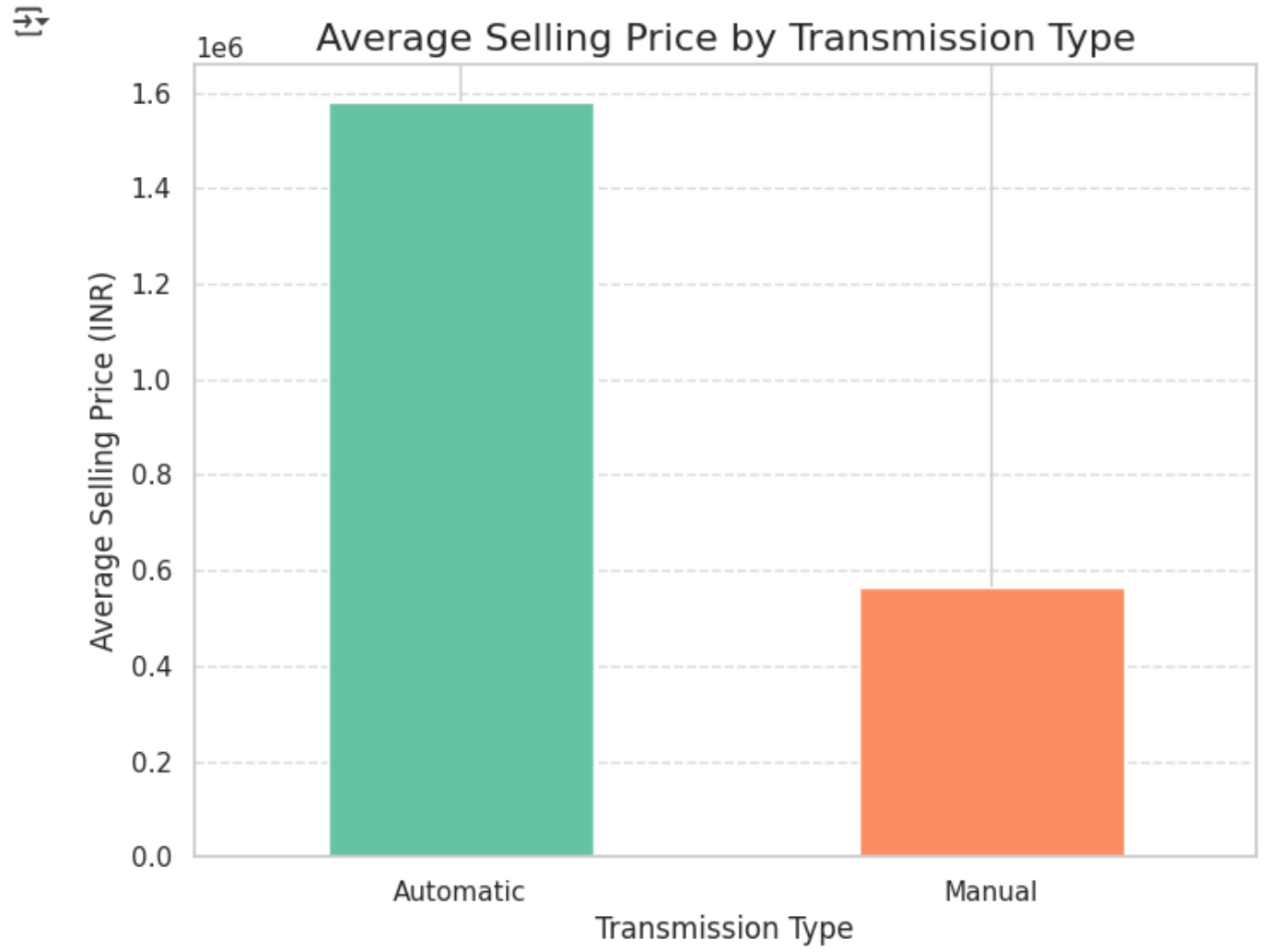
6. Scatter Plot: Engine Capacity vs Selling Price, colored by Fuel Type

```
plt.figure(figsize=(12, 6))
sns.scatterplot(
    data=df, x='engine', y='selling_price', hue='fuel_type',
    palette="viridis", alpha=0.8
)
plt.title("Engine Capacity vs Selling Price (Colored by Fuel Type)", fontsize=16)
plt.xlabel("Engine Capacity (CC)", fontsize=12)
plt.ylabel("Selling Price (INR)", fontsize=12)
plt.legend(title="Fuel Type", loc='upper left')
plt.grid(True, linestyle='--', alpha=0.6)
plt.ylim(0, 3_000_000)
plt.show()
```



7. Create a bar plot to show the average selling_price by transmission_type (Manual vs Automatic).
"This bar plot will highlight the difference in pricing between manual and automatic transmission vehicles."

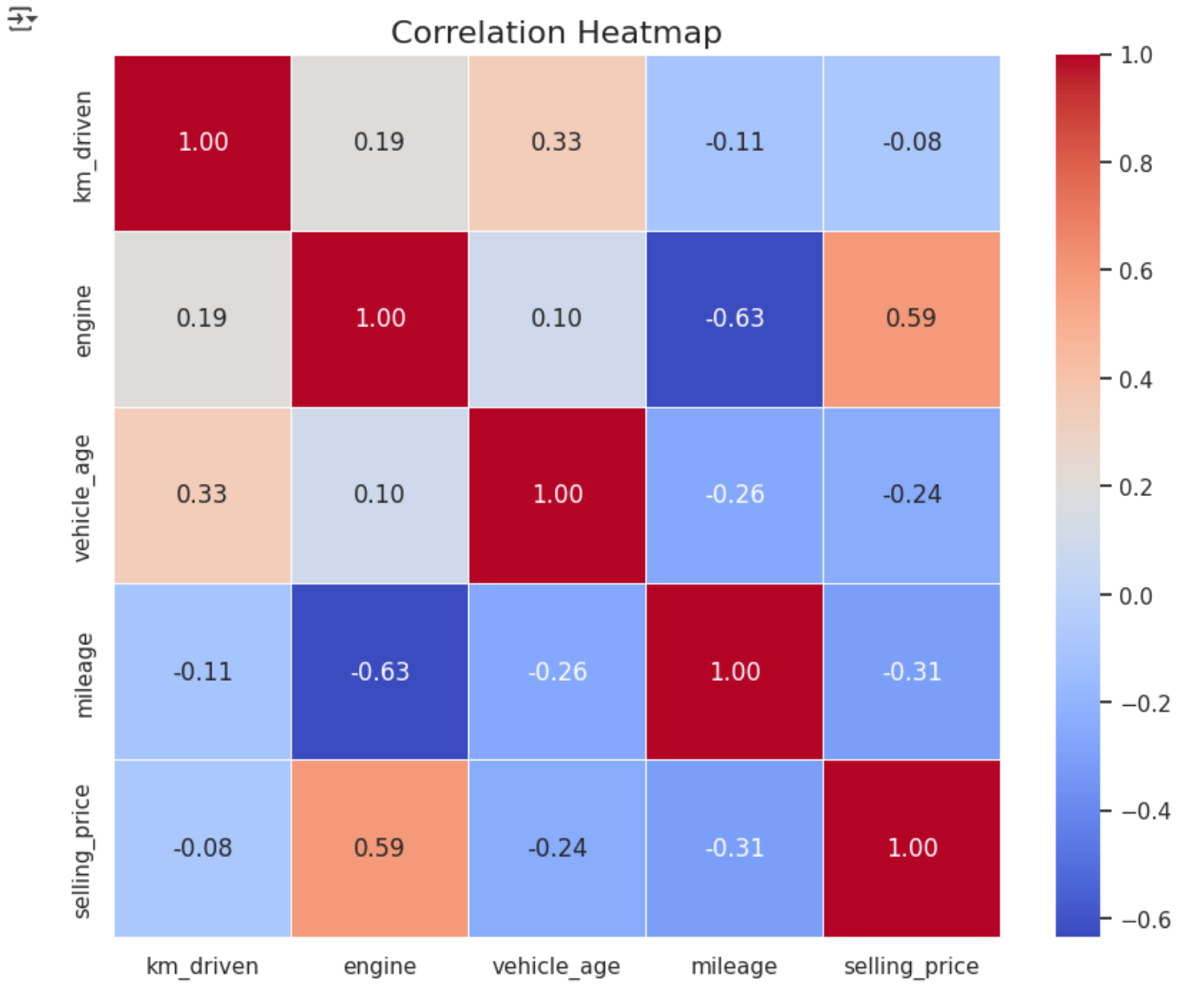
```
plt.figure(figsize=(8, 6))
avg_price_by_transmission = df.groupby('transmission_type')['selling_price'].mean()
avg_price_by_transmission.plot(kind='bar', color=sns.color_palette("Set2"))
plt.title("Average Selling Price by Transmission Type", fontsize=16)
plt.xlabel("Transmission Type", fontsize=12)
plt.ylabel("Average Selling Price (INR)", fontsize=12)
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



8. Generate a heatmap to visualize correlations between km_driven, engine, vehicle_age, mileage, and selling_price.

" A heatmap will provide insights into how these variables correlate with each other and affect the selling price. "

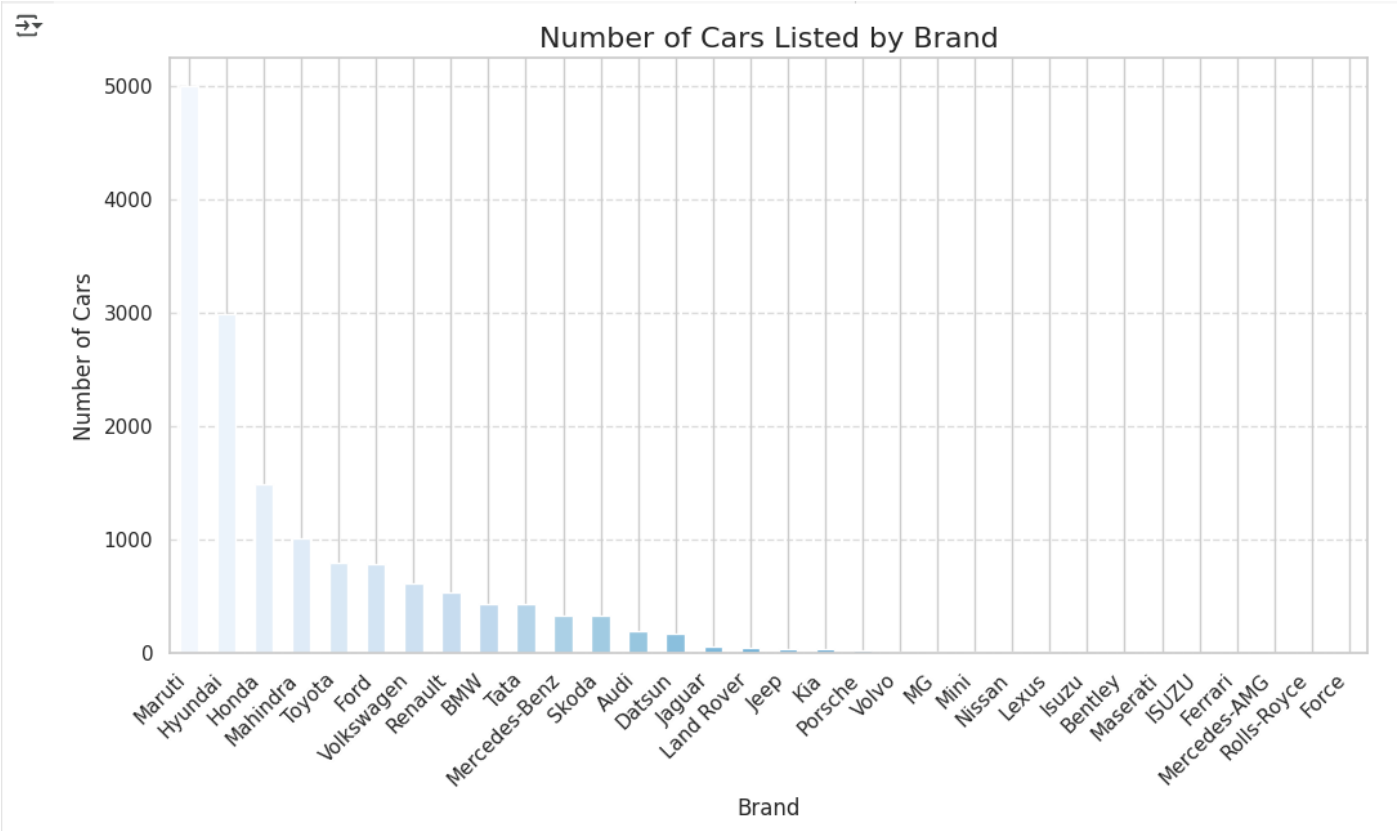
```
plt.figure(figsize=(10, 8))
correlation_matrix = df[['km_driven', 'engine', 'vehicle_age', 'mileage', 'selling_price']].corr()
sns.heatmap(
    correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5
)
plt.title("Correlation Heatmap", fontsize=16)
plt.show()
```



9. Create a bar plot to show the number of cars listed by brand.

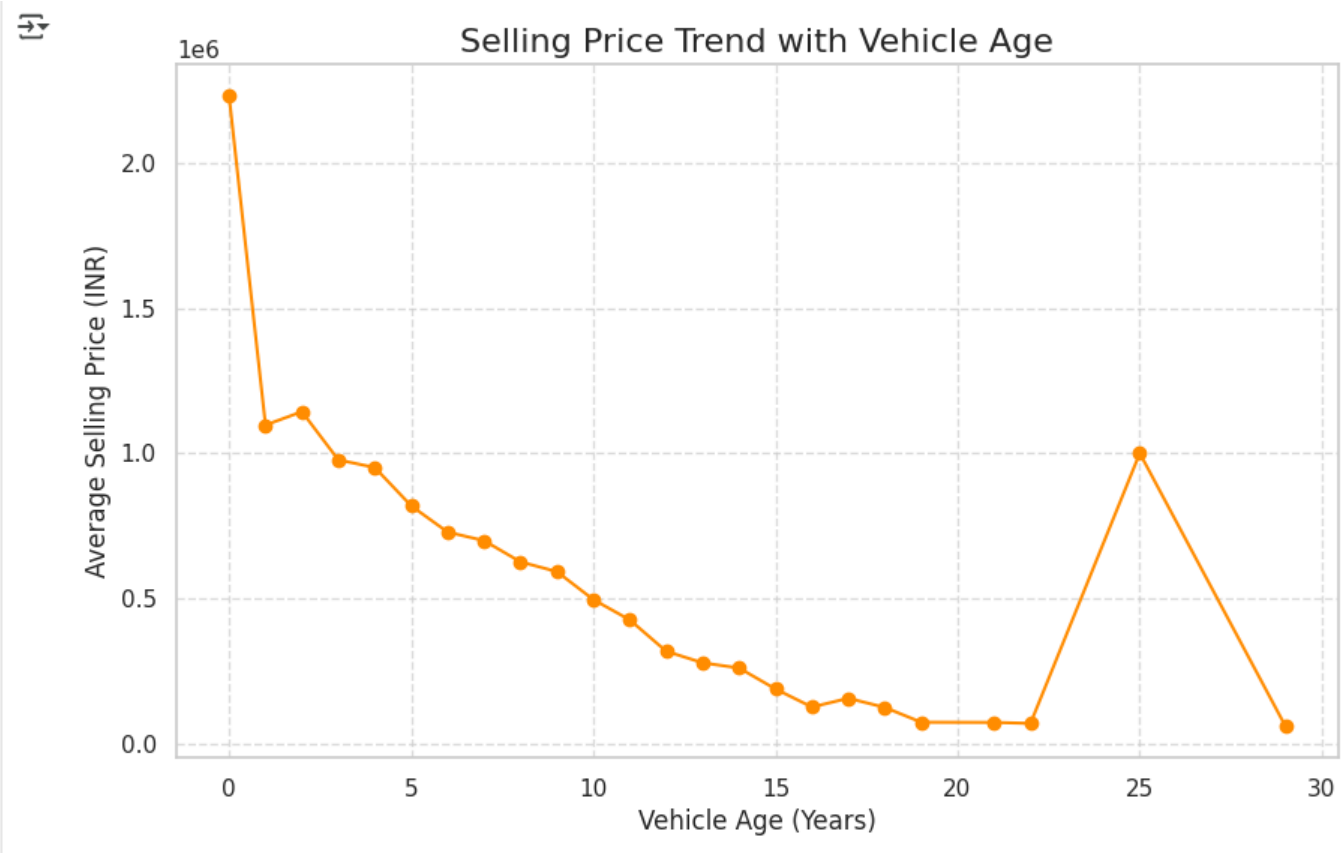
" This bar plot will show the distribution of car listings across different brands."

```
plt.figure(figsize=(12, 6))
brand_counts = df['brand'].value_counts()
brand_counts.plot(kind='bar', color=sns.color_palette("Blues", len(brand_counts)))
plt.title("Number of Cars Listed by Brand", fontsize=16)
plt.xlabel("Brand", fontsize=12)
plt.ylabel("Number of Cars", fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



10. Generate a line graph to show the trend of selling_price with respect to vehicle_age over time.
"This line graph will help identify if there are any consistent patterns in selling price changes as vehicles age."

```
plt.figure(figsize=(10, 6))
avg_price_by_age = df.groupby('vehicle_age')['selling_price'].mean()
avg_price_by_age.plot(kind='line', color="darkorange", marker='o')
plt.title("Selling Price Trend with Vehicle Age", fontsize=16)
plt.xlabel("Vehicle Age (Years)", fontsize=12)
plt.ylabel("Average Selling Price (INR)", fontsize=12)
plt.grid(axis='both', linestyle='--', alpha=0.7)
plt.show()
```



5. Model Building

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import pandas as pd
# Load and preprocess the dataset
df = pd.read_csv('/content/9156Shreeya_Cardekhodataset.csv')
df.dropna(inplace=True)
# Separating features (X) and target (y)
X = df.drop(['selling_price'], axis=1)
y = df['selling_price']
# Convert categorical columns into numerical using one-hot encoding
X = pd.get_dummies(X, drop_first=True)
# Splitting the dataset 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)
# Initialize models
linear_model = LinearRegression()
random_forest_model = RandomForestRegressor(random_state=42)
gb_model = GradientBoostingRegressor(random_state=42)
# Train models
linear_model.fit(X_train, y_train)
random_forest_model.fit(X_train, y_train)
gb_model.fit(X_train, y_train)
# Evaluate models
models = {'Linear Regression': linear_model,
          'Random Forest': random_forest_model,
          'Gradient Boosting': gb_model}
results = {}
for name, model in models.items():
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
    results[name] = {'MSE': mse, 'RMSE': rmse, 'R2': r2}
# Print results
print("Model Evaluation Results:")
for model_name, metrics in results.items():
    print(f"\n{model_name}:")
    for metric, value in metrics.items():
        print(f"  {metric}: {value:.4f}")
# Identify the best model based on R²
best_model = max(results, key=lambda x: results[x]['R2'])
print(f"\nThe best model is: {best_model} with R²: {results[best_model]['R2']:.4f}")
```

```
Model Evaluation Results:

Linear Regression:
  MSE: 150921179588.0457
  RMSE: 388485.7521
  R2: 0.7995

Random Forest:
  MSE: 49079003941.3315
  RMSE: 221537.8161
  R2: 0.9348

Gradient Boosting:
  MSE: 58863137550.4185
  RMSE: 242617.2656
  R2: 0.9218

The best model is: Random Forest with R²: 0.9348
```

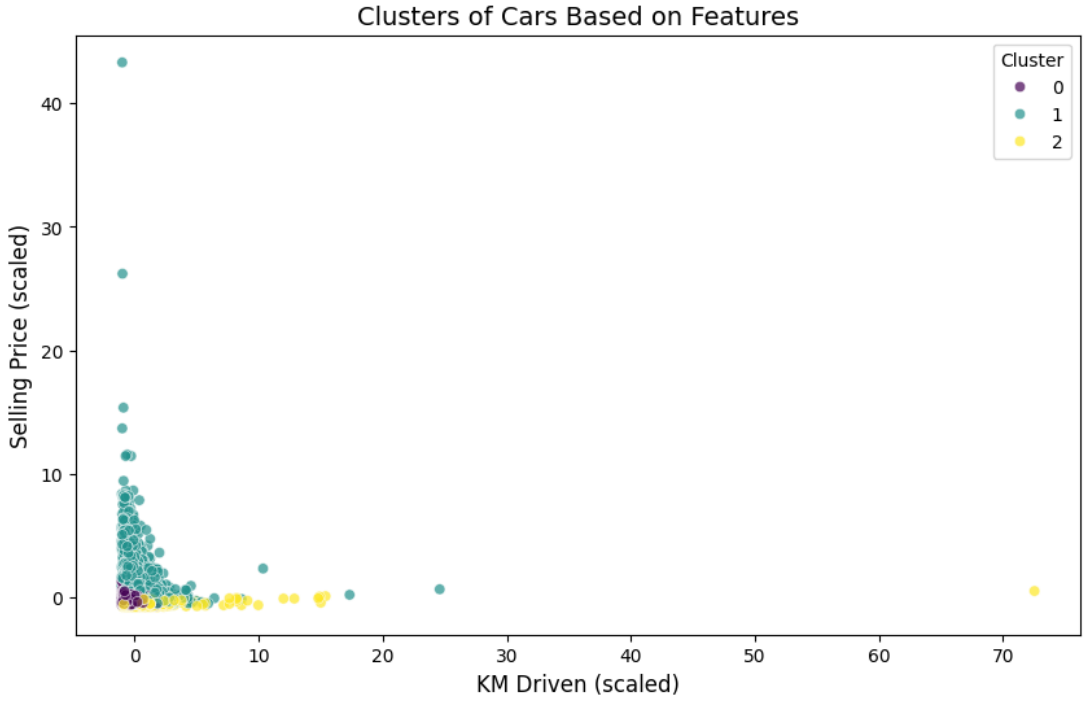
6. Machine Learning Model

Model Training

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
file_path = "/content/9156Shreeya_Cardekhodataset.csv"
df = pd.read_csv(file_path)
print("Dataset Info:")
print(df.info())
# Encode categorical variables
categorical_columns = df.select_dtypes(include=['object']).columns
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le  # Save encoders for future decoding if needed
# Feature selection
features = ['km_driven', 'selling_price', 'vehicle_age', 'engine', 'mileage']
X = df[features]
# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# 2. Training the K-Means Model
n_clusters = 3  # Define number of clusters
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(X_scaled)
# Save the trained model
joblib.dump(kmeans, "kmeans_model.pkl")
print("K-Means model saved as 'kmeans_model.pkl'.")
# Save the scaler
joblib.dump(scaler, "scaler.pkl")
print("Scaler saved as 'scaler.pkl'.")
# 3. Predict clusters for the training data
df['Cluster'] = kmeans.predict(X_scaled)
# Evaluate clustering quality
sil_score = silhouette_score(X_scaled, df['Cluster'])
print(f"Silhouette Score: {sil_score:.2f}")
# 4. Visualize Clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=X_scaled[:, 0],  # km_driven (scaled)
    y=X_scaled[:, 1],  # selling_price (scaled)
    hue=df['Cluster'],
    palette='viridis',
    alpha=0.7
)
plt.title("Clusters of Cars Based on Features", fontsize=14)
plt.xlabel("KM Driven (scaled)", fontsize=12)
plt.ylabel("Selling Price (scaled)", fontsize=12)
plt.legend(title="Cluster")
plt.show()
# 5. Use the model to predict clusters for new data points
new_data = pd.DataFrame({
    'km_driven': [40000, 60000, 120000],
    'selling_price': [500000, 300000, 200000],
    'vehicle_age': [3, 5, 8],
    'engine': [1200, 1500, 1000],
    'mileage': [20, 15, 25]
})
```

```
# Standardize new data
new_data_scaled = scaler.transform(new_data)
# Predict clusters for new data
new_clusters = kmeans.predict(new_data_scaled)
new_data['Cluster'] = new_clusters
print("New Data Predictions:")
print(new_data)
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15411 entries, 0 to 15410
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         15411 non-null  int64
 1   car_name           15411 non-null  object
 2   brand              15411 non-null  object
 3   model              15411 non-null  object
 4   vehicle_age        15411 non-null  int64
 5   km_driven          15411 non-null  int64
 6   seller_type        15411 non-null  object
 7   fuel_type          15411 non-null  object
 8   transmission_type  15411 non-null  object
 9   mileage            15411 non-null  float64
 10  engine             15411 non-null  int64
 11  max_power          15411 non-null  float64
 12  seats              15411 non-null  int64
 13  selling_price      15411 non-null  int64
dtypes: float64(2), int64(6), object(6)
memory usage: 1.6+ MB
None
K-Means model saved as 'kmeans_model.pkl'.
Scaler saved as 'scaler.pkl'.
Silhouette Score: 0.30
```



New Data Predictions:

	km_driven	selling_price	vehicle_age	engine	mileage	Cluster
0	40000	500000	3	1200	20	0
1	60000	300000	5	1500	15	0
2	120000	200000	8	1000	25	2

7. MODEL OUTPUT: Cluster Prediction

```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
import joblib
# Load the trained model and scaler
kmeans = joblib.load("kmeans_model.pkl")
scaler = joblib.load("scaler.pkl")
# Function to take user input
def get_user_input():
    print("\nEnter car details for clustering:")
    try:
        km_driven = float(input("Enter KM Driven: "))
        selling_price = float(input("Enter Selling Price (in INR): "))
        vehicle_age = int(input("Enter Vehicle Age (in years): "))
        engine = float(input("Enter Engine Capacity (in cc): "))
        mileage = float(input("Enter Mileage (in km/l): "))
        return pd.DataFrame([{
            'km_driven': km_driven,
            'selling_price': selling_price,
            'vehicle_age': vehicle_age,
            'engine': engine,
            'mileage': mileage
        }])
    except ValueError:
        print("Invalid input! Please enter numeric values.")
        return None
```

```
# Interactive loop for predictions
while True:
    # Get user input
    user_data = get_user_input()
    if user_data is None:
        continue
    # Standardize user input
    user_data_scaled = scaler.transform(user_data)
    # Predict cluster
    cluster = kmeans.predict(user_data_scaled)[0]
    # Display cluster result
    print("\n### Predicted Cluster ###")
    print(f"The car belongs to Cluster {cluster}.")
    print("\n##########################\n")
    # Option to continue or exit
    cont = input("Do you want to predict another car? (yes/no): ").strip().lower()
    if cont != 'yes':
        print("Thank you! Exiting...")
        break
```

```
Enter car details for clustering:
Enter KM Driven: 45000
Enter Selling Price (in INR): 500000
Enter Vehicle Age (in years): 5
Enter Engine Capacity (in cc): 1200
Enter Mileage (in km/l): 18

### Predicted Cluster ###
The car belongs to Cluster 0.

##########################

Do you want to predict another car? (yes/no): yes

Enter car details for clustering:
Enter KM Driven: 90000
Enter Selling Price (in INR): 300000
Enter Vehicle Age (in years): 8
Enter Engine Capacity (in cc): 1500
Enter Mileage (in km/l): 15

### Predicted Cluster ###
The car belongs to Cluster 2.

##########################

Do you want to predict another car? (yes/no): no
Thank you! Exiting...
```

8. CONCLUSION

This project effectively leverages machine learning to analyze and predict car prices, offering valuable insights for buyers and sellers. While it showcases strong real-world relevance and scalability, addressing data limitations and adapting to market dynamics are crucial for maximizing its impact.

9. SWOT ANALYSIS

Strengths

1. Real-World Relevance:
    - The dataset focuses on the used car market, a thriving and highly relevant industry globally.
    - Provides valuable insights into pricing trends and customer preferences.
2. Comprehensive Features:
    - Dataset includes a variety of features such as fuel type, kilometers driven, transmission, and vehicle age, enabling detailed analysis.
3. Predictive Power:
    - The model can be used to accurately predict selling prices, aiding decision-making for buyers and sellers.

Weaknesses

1. Data Imbalance:
    - Uneven distribution of cars across fuel types or brands may lead to biased results.
2. Limited Dataset Scope:
    - Dataset may not include factors like region, insurance history, or market demand that also influence car prices.
3. Outlier Sensitivity:
    - Outliers in numerical fields like selling price or kilometers driven can skew the results.

Opportunities

1. Business Applications:
    - Can be used by online car dealerships to recommend pricing strategies.
    - Valuable for individual sellers to estimate fair market value.
2. Scalability:
    - With more data (e.g., customer demographics, location-based pricing), the model can become more robust and accurate.
3. Environmental Insights:
    - Analysis of fuel types and engine sizes can highlight trends in eco-friendly vehicle preferences.

Threats

1. Data Privacy Concerns:
    - Real-world applications may require sensitive customer or vehicle information, raising privacy issues.
2. Rapid Market Changes:
    - Sudden economic shifts or changes in fuel preferences could render the model less accurate.
3. Competition:
    - Similar projects or tools by competitors may overshadow the utility of this model.
4. Overfitting:
    - The model may perform well on training data but fail to generalize to unseen data.