My Capstone Project - Assignment

- Used Cars Price Prediction
- Uses a Kaggle Dataset

```
%cd /content/drive/MyDrive/My Capstone Project
```

```
/content/drive/MyDrive/My Capstone Project
```

```
!ls /content/drive/MyDrive/ST1_DATASET
```

```
test-data.csv  train-data.csv
```

This project is based on Used Cars Price Prediction data available from Kaggle repository.

(https://www.kaggle.com/datasets/avikasliwal/used-cars-price-prediction)

- It contains 1233 test data and 6018 train data of used cars.
- My project task is to create a machine learning model which can predict the price of used cars.
- For solving this problem, I will approach the task with a step by step approach to create a data analysis and prediction model based on (machine learning, regression algorithm for example) available from different python packages, modules and classes.

Step 1: Reading the data with Python

```
import warnings
warnings.filterwarnings('ignore')
```

```
import pandas as pd
import numpy as np
Used_Car_Data=pd.read_csv('/content/drive/MyDrive/ST1_DATASET/train-data.csv', encoding='latin')
print('Shape before deleting duplicate values:', Used_Car_Data.shape)

Used_Car_Data=Used_Car_Data.drop_duplicates()
print('Shape after deleting duplictae values:', Used_Car_Data.shape)

Used_Car_Data.head(10)
```

```
Shape before deleting duplicate values: (6019, 14)
Shape after deleting duplictae values: (6019, 14)
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	
4	4	Audi A4 New 2.0 TDI	Coimbatore	2013	40670	Diesel	Automatic	

Step 2: Problem Statement Definition

- Creating a prediction model to predict the price of an used car
- Target variable: Price Predictors/features: Location, Year, Kilometers_Driven, Fuel_Type, Transmission, Owner_Type, Mileage, Engine, Power, Seats etc

Step 3: Choosing the appropriate ML/AI Algorithm for Data Analysis

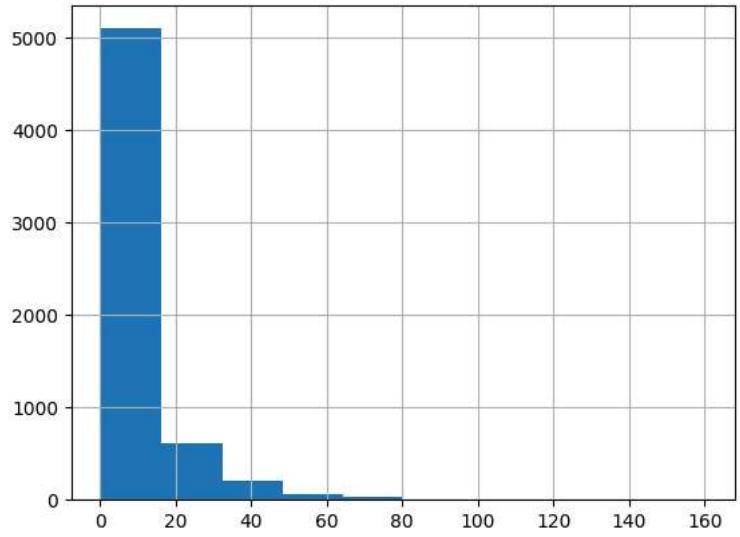
- Based on the problem statement we need to create a supervised a supervised ML Regression model, as the target variable is continuous.

Step 4: Looking at the class distribution (Target variable distribution to check if the data is balanced or skewed)

- If target variable's distribution is too skewed then the predictve modelling will lead to poor results.
- Ideally Bell curve is desirable but sightly positive skew or negative skew is also fine
- When performing Regression algorithm modelling and analysis, we need to make sure the histogram looks like a bell curve or slight skewed version of it.
- Otherwise It impacts the Machine Learning algorithm ability to leaen all the scenarios from the data

```
%matplotlib inline
#creating histogram as the target variable is continuous
# This will help us to understand the distribution of the Price values
Used_Car_Data['Price'].hist()
```

```
<Axes: >
```



Observations from step 4

- The data distribution of the target variable is skewed and is satisfactory to process further.
- There are sufficient number of rows for each type of values to learn from

Step 5: Basic Exploratory Data Analysis

- This step is performed to guage the overall data

```
# Looking at sample rows in the data
Used_Car_Data.head()
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Own
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	

```
# Looking at the sample rows in the data
Used_Car_Data.tail()
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
6014	6014	Maruti Swift VDI	Delhi	2014	27365	Diesel	Manual	
6015	6015	Hyundai Xcent 1.1 CRDi S	Jaipur	2015	100000	Diesel	Manual	
		Mahindra						

```
#Observing the summarized information of data
#Data types, missing values based on number of non-null values Vs total rows etc.
#Remove those variables from data which have too many missing values (Mssing values > 30%)
#Remove qualitative variables which cannot be used in machine learning
Used_Car_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         6019 non-null   int64  
 1   Name               6019 non-null   object 
 2   Location           6019 non-null   object 
 3   Year               6019 non-null   int64  
 4   Kilometers_Driven  6019 non-null   int64  
 5   Fuel_Type          6019 non-null   object 
 6   Transmission       6019 non-null   object 
 7   Owner_Type         6019 non-null   object 
 8   Mileage            6017 non-null   object 
 9   Engine             5983 non-null   object 
 10  Power              5983 non-null   object 
 11  Seats              5977 non-null   float64
 12  New_Price          824 non-null    object 
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
memory usage: 658.5+ KB
```

```
# Looking at the descriptive statistics of the data
Used_Car_Data.describe(include='all')
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transi
count	6019.000000	6019	6019	6019.000000	6.019000e+03	6019	
unique	NaN	1876	11	NaN	NaN	5	
top	NaN	Mahindra XUV500 W8 2WD	Mumbai	NaN	NaN	Diesel	
freq	NaN	49	790	NaN	NaN	3205	
mean	3009.000000	NaN	NaN	2013.358199	5.873838e+04	NaN	
std	1737.679967	NaN	NaN	3.269742	9.126884e+04	NaN	
min	0.000000	NaN	NaN	1998.000000	1.710000e+02	NaN	
25%	1504.500000	NaN	NaN	2011.000000	3.400000e+04	NaN	
50%	3009.000000	NaN	NaN	2014.000000	5.300000e+04	NaN	
75%	4513.500000	NaN	NaN	2016.000000	7.300000e+04	NaN	

```
#Finging unique values for each column
#To understand which column is categorical and which one is continuous
#Typically if the numer of unique values are < 20 then the variable is likely to be category otherwise continuous
Used_Car_Data.nunique()
```

```
Unnamed: 0          6019
Name                1876
Location              11
Year                  22
Kilometers_Driven   3093
Fuel_Type              5
Transmission           2
Owner_Type             4
Mileage              442
Engine               146
```

```
Power                372
Seats                  9
New_Price            540
Price               1373
dtype: int64
```

Step 7: Removing Unwanted Columns

```
# used for cleaning up the data frame by removing unneccessary colums and duplicate rows
Used_Car_Data.isna().sum()
```

```
Unnamed: 0              0
Name                    0
Location                0
Year                    0
Kilometers_Driven       0
Fuel_Type               0
Transmission            0
Owner_Type              0
Mileage                 2
Engine                 36
Power                  36
Seats                  42
New_Price            5195
Price                   0
dtype: int64
```

```
Used_Car_Data.drop(columns = ['Unnamed: 0','New_Price'],inplace = True)
Used_Car_Data.duplicated().sum()
```

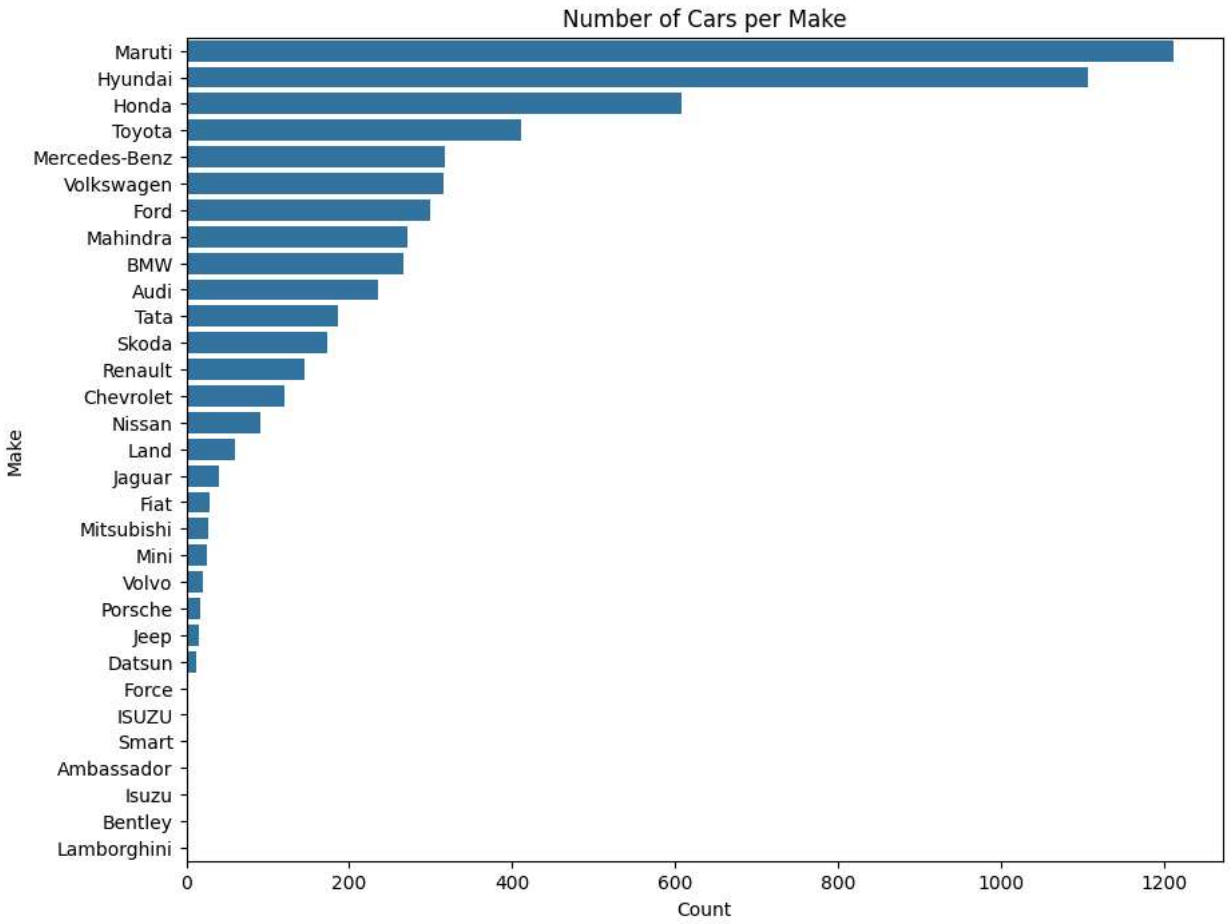
```
0
```

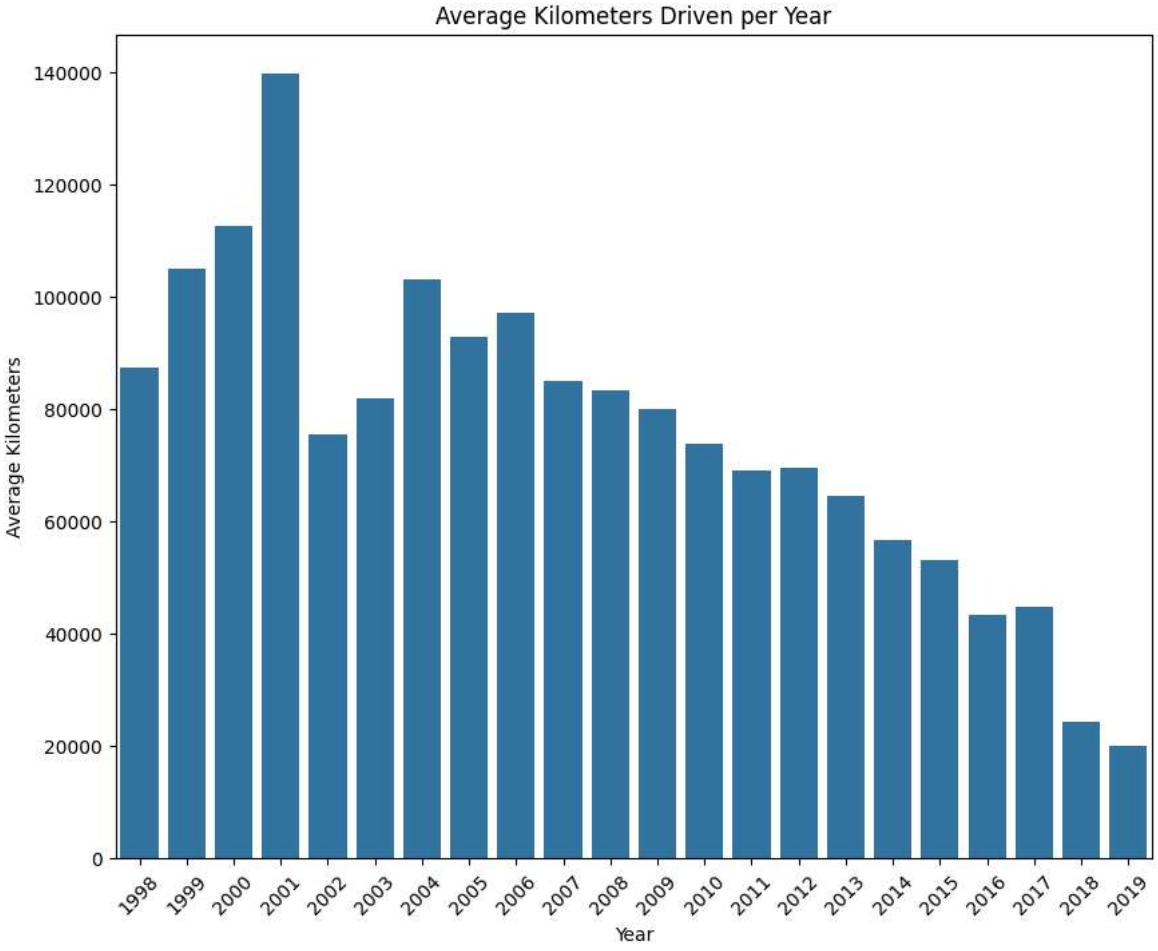
Step 8: Visual Exploratory Data Analysis

```
pip install pandas matplotlib seaborn
```

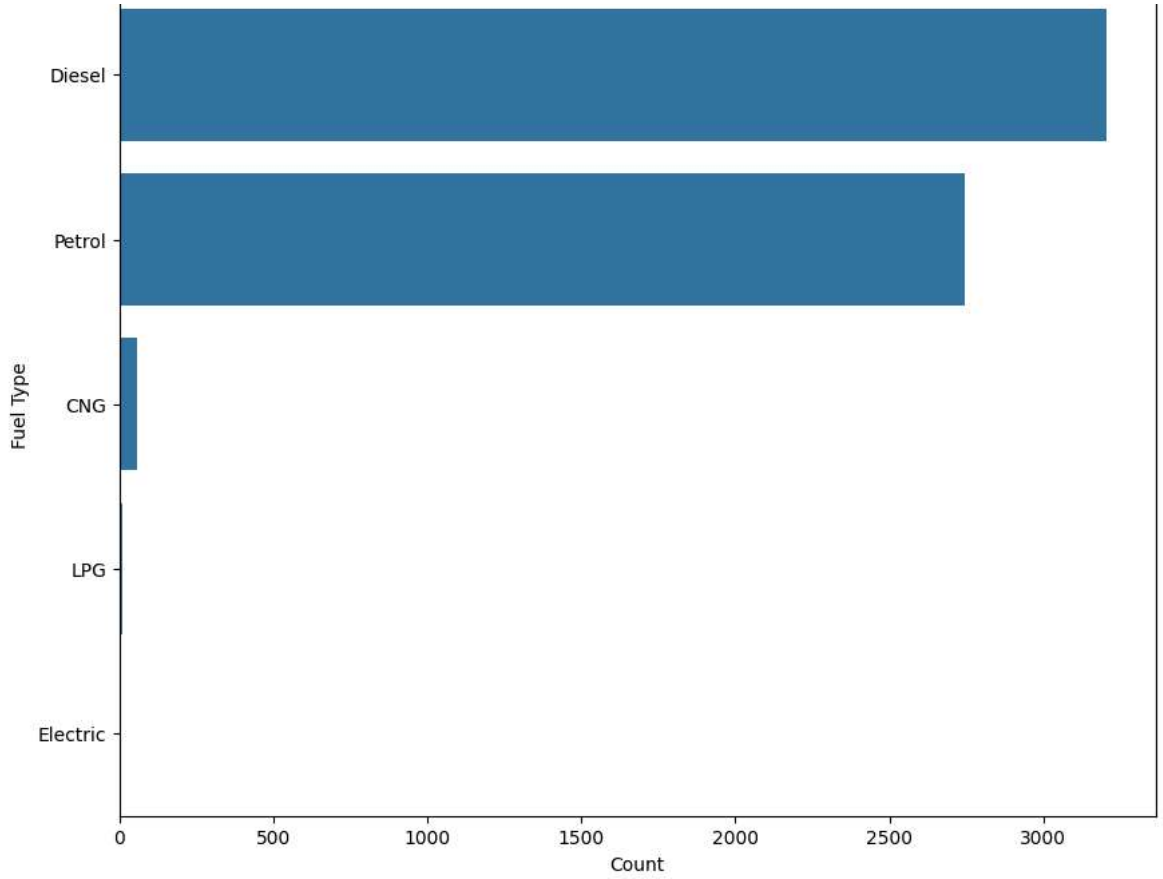
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
used_car_data = pd.read_csv(file_path)

# Drop unnecessary columns if they exist to avoid KeyError
columns_to_drop = ['Unnamed: 0', 'New_Price']
used_car_data.drop(columns=[col for col in columns_to_drop if col in used_car_data.columns], inplace=True)

# Remove duplicate rows if any
used_car_data.drop_duplicates(inplace=True)

# Extract 'Make' from the 'Name' column for car makes
used_car_data['Make'] = used_car_data['Name'].apply(lambda x: x.split()[0])
car_make_count = used_car_data['Make'].value_counts().reset_index()
car_make_count.columns = ['Make', 'Count']

# Plot: Number of cars per Make
plt.figure(figsize=(10, 8))
sns.barplot(x="Count", y="Make", data=car_make_count)
plt.title("Number of Cars per Make")
plt.xlabel("Count")
plt.ylabel("Make")
plt.show()

# Assuming Kilometers_Driven as a proxy for insights, as actual price data wasn't specified
avg_km_year = used_car_data.groupby('Year')['Kilometers_Driven'].mean().reset_index()
avg_km_year.columns = ['Year', 'Average Kilometers']

# Plot: Average Kilometers Driven per Year
plt.figure(figsize=(10, 8))
sns.barplot(x="Year", y="Average Kilometers", data=avg_km_year)
plt.title("Average Kilometers Driven per Year")
plt.xlabel("Year")
plt.ylabel("Average Kilometers")
```

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

    # Count cars by Fuel Type
    fuel_type_count = used_car_data['Fuel_Type'].value_counts().reset_index()
    fuel_type_count.columns = ['Fuel_Type', 'Count']

    # Plot: Number of cars per Fuel Type
    plt.figure(figsize=(10, 8))
    sns.barplot(x="Count", y="Fuel_Type", data=fuel_type_count)
    plt.title("Number of Cars per Fuel Type")
    plt.xlabel("Count")
    plt.ylabel("Fuel Type")
    plt.show()
```



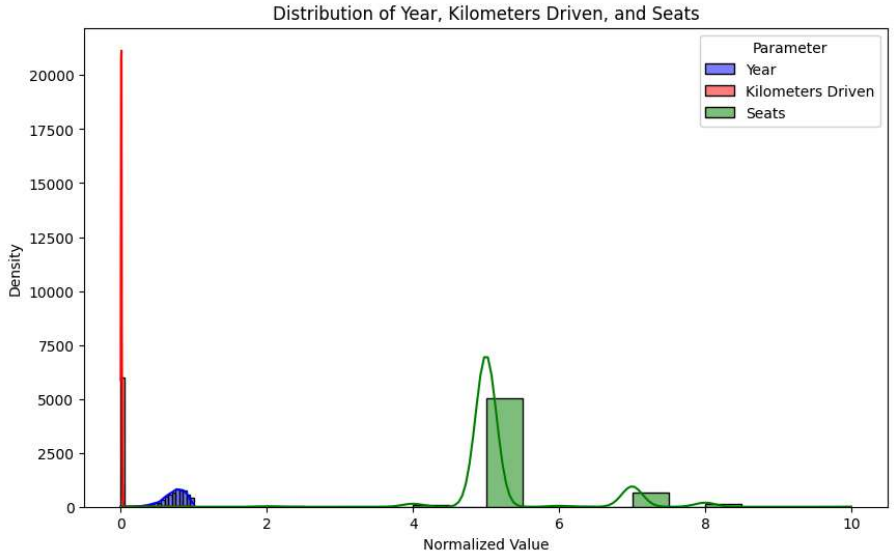


Number of Cars per Fuel Type



Step 9: Now Visualize distribution of all the continuous predictor variables in the data using histograms

```
# Normalize data to make it easier to compare on the same scale
used_car_data['Normalized Kilometers'] = (used_car_data['Kilometers_Driven'] - used_car_data['Kilometers_Driven'].min()) / (used_car_data['K
used_car_data['Normalized Year'] = (used_car_data['Year'] - used_car_data['Year'].min()) / (used_car_data['Year'].max() - used_car_data['Yea

# Plotting
plt.figure(figsize=(10, 6))
sns.histplot(used_car_data['Normalized Year'], bins=20, color='blue', kde=True, label='Year')
sns.histplot(used_car_data['Normalized Kilometers'], bins=20, color='red', kde=True, label='Kilometers Driven')
sns.histplot(used_car_data['Seats'], bins=20, color='green', kde=True, label='Seats')  # Note: this may not be as meaningful if 'Seats' are
plt.legend(title='Parameter')
plt.title('Distribution of Year, Kilometers Driven, and Seats')
plt.xlabel('Normalized Value')
plt.ylabel('Density')
plt.show()
```



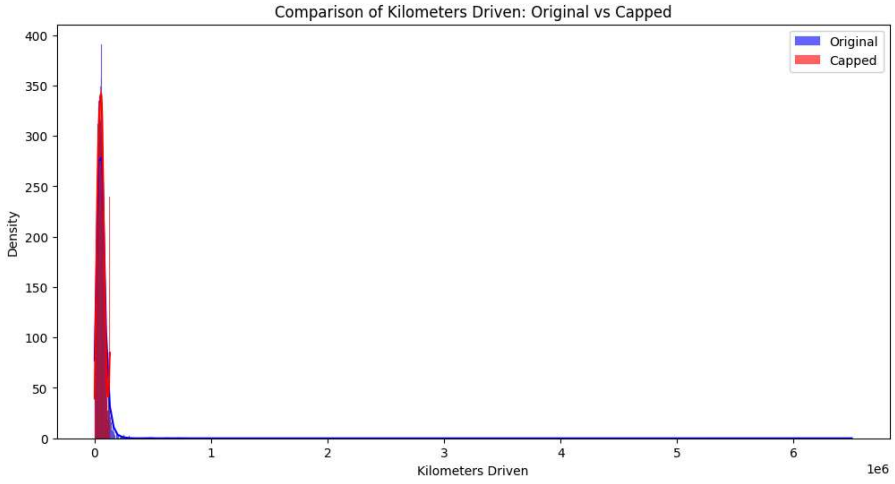
Step 10: Outlier Analysis

```
# Outlier analysis for 'Kilometers_Driven'
# Method 1: Using Z-Score
z_scores = np.abs((used_car_data['Kilometers_Driven'] - used_car_data['Kilometers_Driven'].mean()) / used_car_data['Kilometers_Driven'].std(
outliers_z = used_car_data[z_scores > 3]

# Method 2: Using IQR (Interquartile Range)
Q1 = used_car_data['Kilometers_Driven'].quantile(0.25)
Q3 = used_car_data['Kilometers_Driven'].quantile(0.75)
IQR = Q3 - Q1
outliers_iqr = used_car_data[(used_car_data['Kilometers_Driven'] < (Q1 - 1.5 * IQR)) | (used_car_data['Kilometers_Driven'] > (Q3 + 1.5 * IQR

# Print outliers
print("Outliers using Z-Score method:")
print(outliers_z[['Name', 'Kilometers_Driven']])
print("\nOutliers using IQR method:")
print(outliers_iqr[['Name', 'Kilometers_Driven']])
```

```
Outliers using Z-Score method:
                                                Name  Kilometers_Driven
340            Skoda Octavia Ambition Plus 2.0 TDI AT             775000
358                          Hyundai i10 Magna 1.2              620000
1860                Volkswagen Vento Diesel Highline             720000
2328                       BMW X5 xDrive 30d M Sport            6500000
2823  Volkswagen Jetta 2013-2015 2.0L TDI Highline AT             480000
3092                             Honda City i VTEC SV             480000
4491                  Hyundai i20 Magna Optional 1.2              445000

Outliers using IQR method:
                                     Name  Kilometers_Driven
29       Toyota Innova 2.5 V Diesel 7-seater             262000
64                       Tata Indica V2 eLS              178000
77                     Toyota Innova 2.0 G1              230000
154      Skoda Superb Elegance 2.0 TDI CR AT             136997
164        Ford Ecosport 1.5 DV5 MT Ambiente             147898
...                                      ...                ...
5852    Toyota Innova 2.5 G4 Diesel 8-seater             192000
5871  Ford Endeavour 4x2 XLT Limited Edition             180000
5914          Skoda Octavia Elegance 1.9 TDI             132000
5953                   Ford Figo Diesel EXI              140000
5957                     Honda City 1.5 EXI              186679

[202 rows x 2 columns]
```

Step 11: Visualising Data Distribution after outlier removal.

```
# Calculate IQR and determine boundaries for outliers
Q1 = used_car_data['Kilometers_Driven'].quantile(0.25)
Q3 = used_car_data['Kilometers_Driven'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Replace outliers with the nearest values within the IQR boundaries
used_car_data['Kilometers_Driven_Capped'] = used_car_data['Kilometers_Driven'].apply(
    lambda x: max(min(x, upper_bound), lower_bound)
)

# Check if the new column exists in the DataFrame
if 'Kilometers_Driven_Capped' in used_car_data.columns:
    # Visualization
    plt.figure(figsize=(12, 6))

    # Original Data Distribution
    sns.histplot(used_car_data['Kilometers_Driven'], color="blue", kde=True, label='Original', alpha=0.6)

    # Capped Data Distribution
    sns.histplot(used_car_data['Kilometers_Driven_Capped'], color="red", kde=True, label='Capped', alpha=0.6)

    plt.title('Comparison of Kilometers Driven: Original vs Capped')
    plt.xlabel('Kilometers Driven')
    plt.ylabel('Density')
    plt.legend()
    plt.show()
else:
    print("The column 'Kilometers_Driven_Capped' was not created successfully.")
```



Step 12: Missing Values Analysis

```
import pandas as pd

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
used_car_data = pd.read_csv(file_path)

# Calculate missing values in each column
missing_values = used_car_data.isna().sum()

# Print the number of missing values for each column
print("Number of missing values for each column:")
print(missing_values)
```

```
Number of missing values for each column:
Unnamed: 0              0
Name                    0
Location                0
Year                    0
Kilometers_Driven       0
Fuel_Type               0
Transmission            0
Owner_Type              0
Mileage                 2
Engine                 36
Power                  36
Seats                  42
New_Price            5195
Price                   0
dtype: int64
```

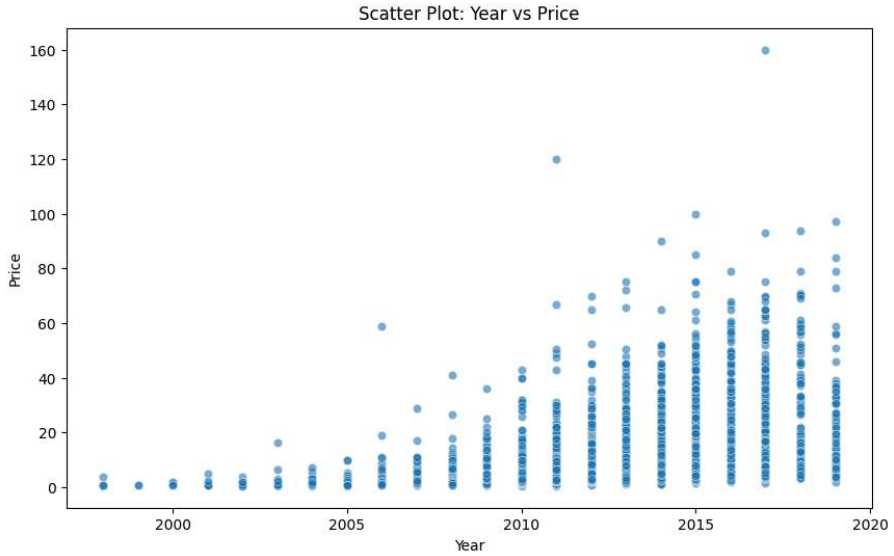
Handling the missing values

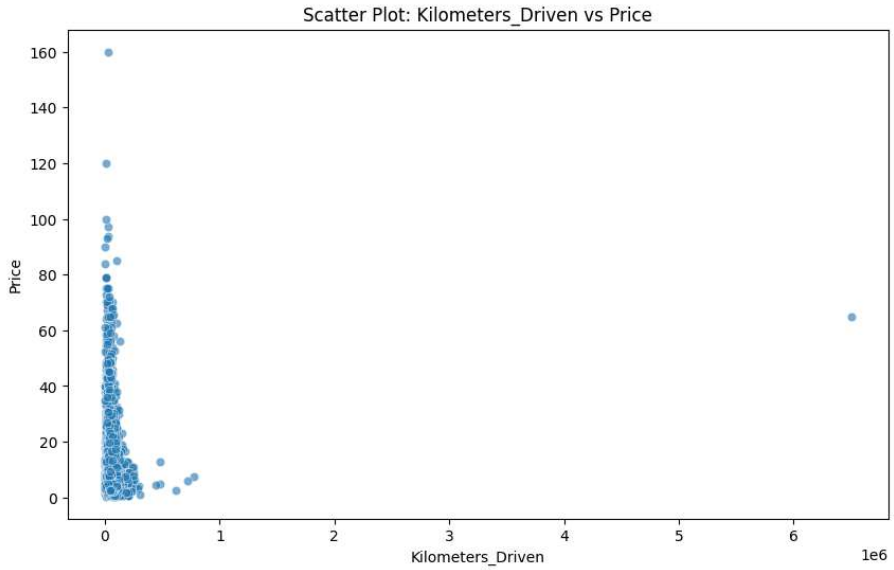
```
# Remove 'New_Price' if not required (due to high number of missing values)
used_car_data.drop(columns=['New_Price'], inplace=True)

# Convert 'Mileage', 'Engine', and 'Power' to numeric values, removing units
used_car_data['Mileage'] = used_car_data['Mileage'].str.extract('(\d+\.\d+)').astype(float)
used_car_data['Engine'] = used_car_data['Engine'].str.extract('(\d+)').astype(float)
used_car_data['Power'] = used_car_data['Power'].str.extract('(\d+\.\d+|\d+)').astype(float)

# Fill missing numeric values with the median for 'Mileage', 'Engine', 'Power'
for column in ['Mileage', 'Engine', 'Power']:
    median_value = used_car_data[column].median()
    used_car_data[column].fillna(median_value, inplace=True)

# Verify missing values after handling
print("\nMissing values after handling:")
print(used_car_data.isna().sum())
```

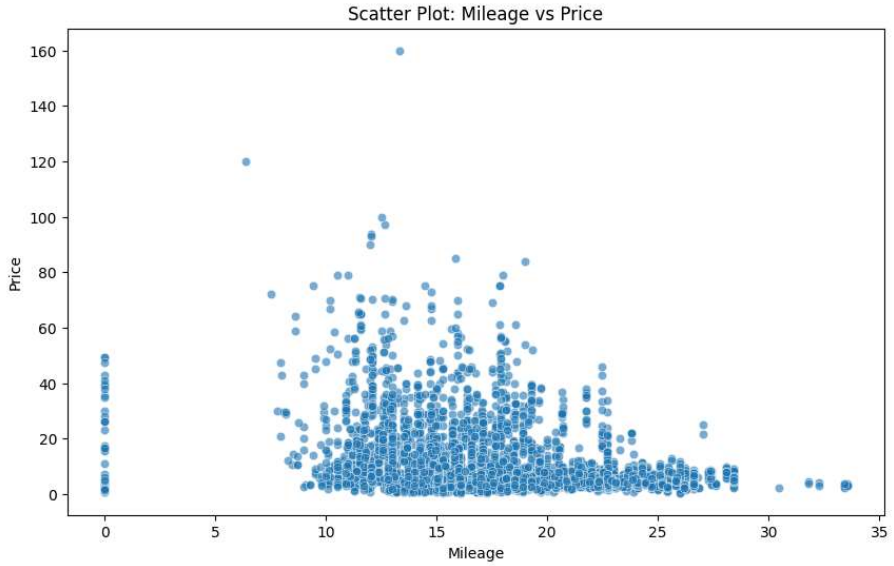
```
Missing values after handling:
Unnamed: 0          0
Name                0
Location            0
Year                0
Kilometers_Driven   0
Fuel_Type           0
Transmission        0
Owner_Type          0
Mileage             0
Engine              0
Power               0
Seats               0
Price               0
dtype: int64
```

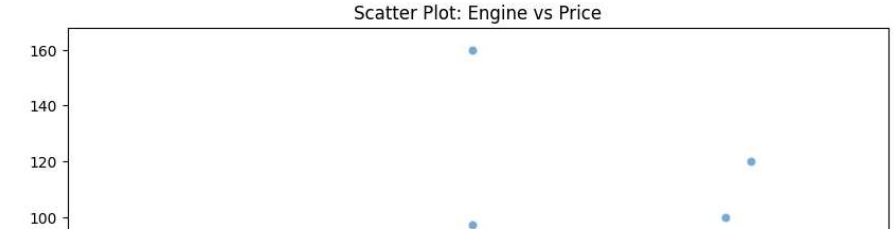
Step 13: Feature Selection (Attribute Selection)

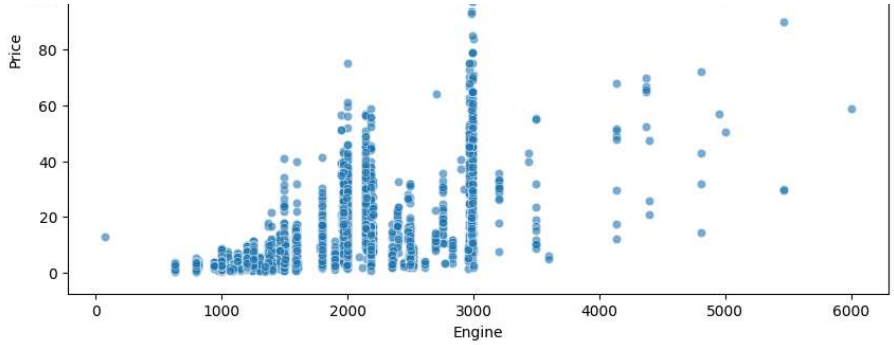
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
used_car_data = pd.read_csv(file_path)

# Check and drop any columns not used as predictors or if they contain high missing values
if 'New_Price' in used_car_data.columns:
    used_car_data.drop(columns=['New_Price'], inplace=True)

# Convert 'Mileage', 'Engine', and 'Power' to numeric values, removing units
used_car_data['Mileage'] = used_car_data['Mileage'].str.extract('(\d+\.\d+)').astype(float)
used_car_data['Engine'] = used_car_data['Engine'].str.extract('(\d+)').astype(float)
used_car_data['Power'] = used_car_data['Power'].str.extract('(\d+\.\d+|\d+)').astype(float)

# Fill missing values if any remain (using median for numerical columns)
numeric_columns = ['Mileage', 'Engine', 'Power', 'Seats']
for column in numeric_columns:
    if used_car_data[column].isna().any():
        median_value = used_car_data[column].median()
        used_car_data[column].fillna(median_value, inplace=True)

# List of predictor variables (excluding any identifiers or non-predictive information)
predictors = ['Year', 'Kilometers_Driven', 'Mileage', 'Engine', 'Power', 'Seats']

# Create scatter plots for each predictor against the 'Price'
for predictor in predictors:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=used_car_data, x=predictor, y='Price', alpha=0.6)
    plt.title(f'Scatter Plot: {predictor} vs Price')
    plt.xlabel(predictor)
    plt.ylabel('Price')
    plt.show()
```

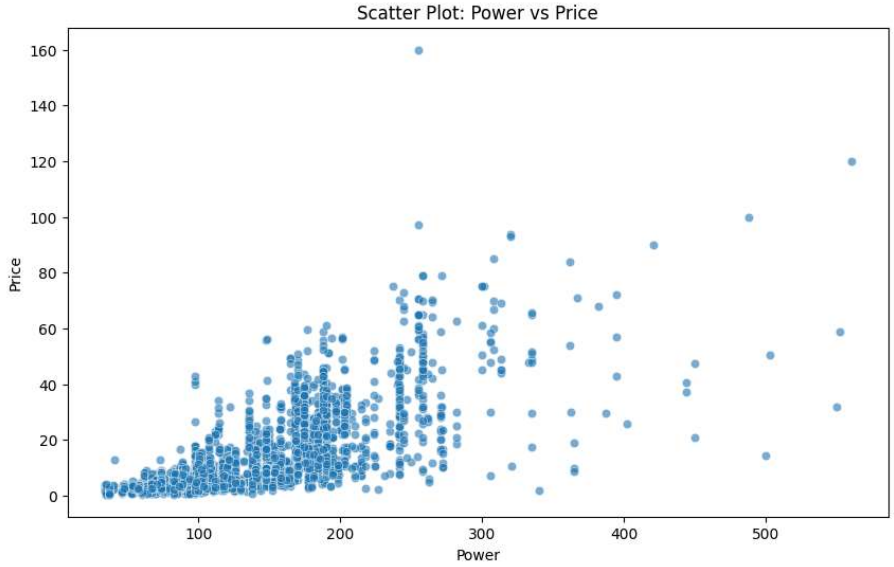

Scatter Plot: Year vs Price
Price
Year
0
20
40
60
80
100
120
140
160
2000
2005
2010
2015
2020

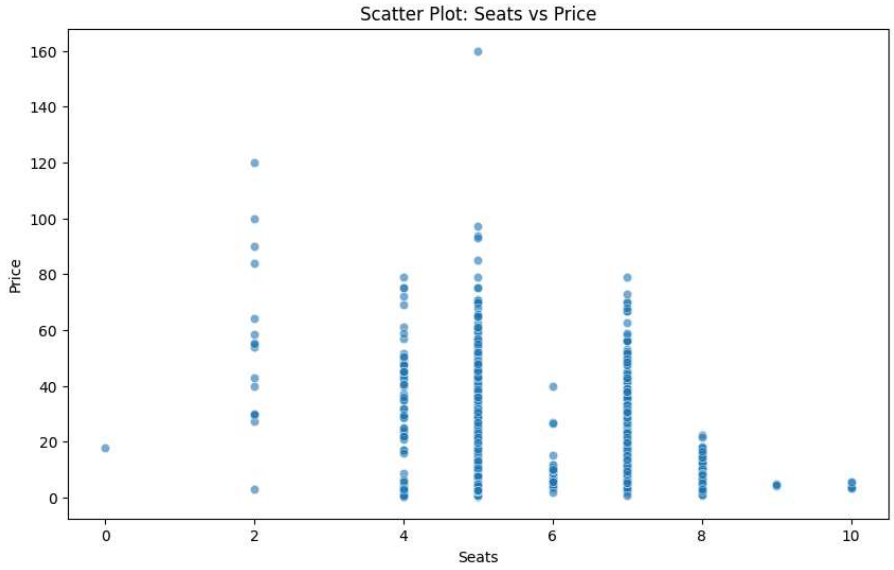

Scatter Plot: Kilometers_Driven vs Price
Price
Kilometers_Driven
1e6
0
20
40
60
80
100
120
140
160
0
1
2
3
4
5
6


Scatter Plot: Mileage vs Price
Price
Mileage
0
20
40
60
80
100
120
140
160
0
5
10
15
20
25
30
35


Scatter Plot: Engine vs Price
160
140
120
100

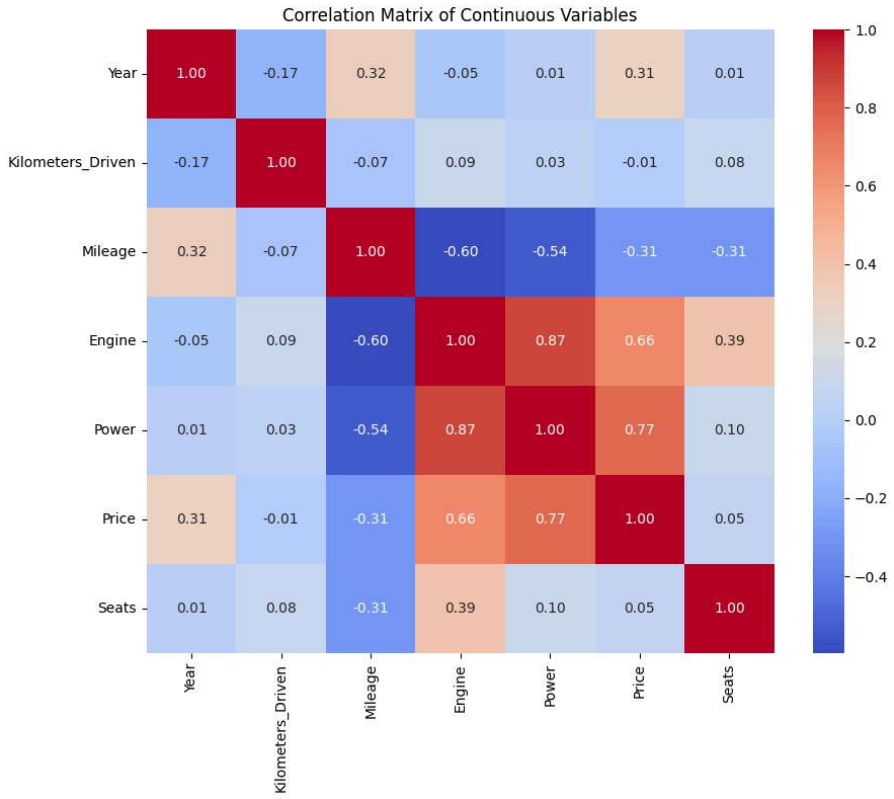

Price
Engine


Scatter Plot: Power vs Price
Price
Power


Scatter Plot: Seats vs Price
Price
Seats

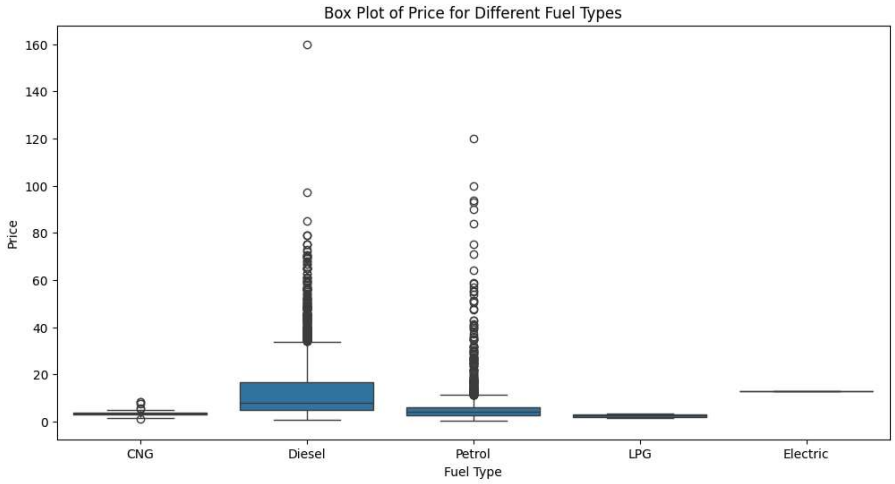
Step 14: Statistical Feature Selection (Continuous vs Continuous) using Correlation value.

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

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
used_car_data = pd.read_csv(file_path)

# Convert 'Mileage', 'Engine', and 'Power' to numeric values, removing units
used_car_data['Mileage'] = used_car_data['Mileage'].str.extract('(\d+\.\d+)').astype(float)
used_car_data['Engine'] = used_car_data['Engine'].str.extract('(\d+)').astype(float)
used_car_data['Power'] = used_car_data['Power'].str.extract('(\d+\.\d+|\d+)').astype(float)

# Selecting only the continuous variables for correlation analysis
continuous_columns = ['Year', 'Kilometers_Driven', 'Mileage', 'Engine', 'Power', 'Price', 'Seats']
continuous_data = used_car_data[continuous_columns]

# Calculate the correlation matrix
corr_matrix = continuous_data.corr()

# Visualizing the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', cbar=True)
plt.title('Correlation Matrix of Continuous Variables')
plt.show()
```



Step 15: Relationship Exploration: Categorical vs Continuous -- Box Plots.

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

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
used_car_data = pd.read_csv(file_path)

# Convert 'Price' to a numeric value, if it's not already
# This step is just a precaution, ensure your 'Price' column is appropriate for plotting
used_car_data['Price'] = pd.to_numeric(used_car_data['Price'], errors='coerce')

# Drop rows with missing 'Price' or 'Fuel_Type' as these are necessary for our analysis
used_car_data.dropna(subset=['Price', 'Fuel_Type'], inplace=True)

# Creating box plots for 'Price' across different 'Fuel_Type'
plt.figure(figsize=(12, 6))
sns.boxplot(x='Fuel_Type', y='Price', data=used_car_data)
plt.title('Box Plot of Price for Different Fuel Types')
plt.xlabel('Fuel Type')
plt.ylabel('Price')
plt.show()
```



Step 16: Statistical Feature Selection (Categorical vs Continuous) using ANOVA test.

```
import pandas as pd
from scipy import stats
import numpy as np

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
used_car_data = pd.read_csv(file_path)

# Ensure 'Price' is a numeric column
used_car_data['Price'] = pd.to_numeric(used_car_data['Price'], errors='coerce')

# Dropping rows with missing values in 'Price' or 'Fuel_Type' to ensure clean data for the test
used_car_data.dropna(subset=['Price', 'Fuel_Type'], inplace=True)

# Group data by 'Fuel_Type' and collect 'Price' for each group
grouped = used_car_data.groupby('Fuel_Type')['Price'].apply(list)

# Perform ANOVA test
f_value, p_value = stats.f_oneway(*grouped)

# Output the results
print(f"ANOVA test results for Price across different Fuel Types:")
print(f"F-Value: {f_value}, P-Value: {p_value}")
```

```
ANOVA test results for Price across different Fuel Types:
F-Value: 173.45164177692007, P-Value: 8.146026879615981e-141
```

Selecting final Predictors/Features for building Machine Learning/AI Model

```
import pandas as pd

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
data = pd.read_csv(file_path)

# Clean up the 'Mileage', 'Engine', and 'Power' columns by extracting numeric values
data['Mileage'] = data['Mileage'].str.extract('(\d+\.\d+)').astype(float)
data['Engine'] = data['Engine'].str.extract('(\d+)').astype(float)
data['Power'] = data['Power'].str.extract('(\d+\.\d+|\d+)').astype(float)

# Fill missing values with median for these columns if there are any
for col in ['Mileage', 'Engine', 'Power', 'Seats']:
    if data[col].isna().any():
        data[col].fillna(data[col].median(), inplace=True)

# Example list of selected columns after feature selection
selected_columns = ['Year', 'Mileage', 'Engine', 'Power', 'Seats', 'Price']  # Customize this list based on your feature selection outcome

# Create a new DataFrame using only the selected columns
data_for_ml = data[selected_columns]

# Display the first few rows of the new DataFrame
print(data_for_ml.head())
```

```
   Year  Mileage  Engine   Power  Seats  Price
0  2010    26.60   998.0   58.16    5.0   1.75
1  2015    19.67  1582.0  126.20    5.0  12.50
2  2011    18.20  1199.0   88.70    5.0   4.50
3  2012    20.77  1248.0   88.76    7.0   6.00
4  2013    15.20  1968.0  140.80    5.0  17.74
```

```
data_for_ml.to_pickle('data_for_ml.pkl')
```

Step 17: Data Pre-Processing for Machine Learning Model Building or Model Development.

```
import pandas as pd

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
data = pd.read_csv(file_path)

# Clean up the 'Mileage', 'Engine', and 'Power' columns by extracting numeric values
data['Mileage'] = data['Mileage'].str.extract('(\d+\.\d+|\d+)').astype(float)
data['Engine'] = data['Engine'].str.extract('(\d+)').astype(float)
data['Power'] = data['Power'].str.extract('(\d+\.\d+|\d+)').astype(float)

# Identify categorical columns you want to convert using get_dummies
categorical_cols = ['Fuel_Type', 'Transmission', 'Owner_Type']

# Applying get_dummies() to the categorical columns and dropping the first to avoid dummy variable trap
data_dummies = pd.get_dummies(data[categorical_cols], drop_first=True)

# Concatenate the original data frame with the new dummy variables (excluding original categorical columns)
data = pd.concat([data.drop(categorical_cols, axis=1), data_dummies], axis=1)

# Handle missing values: fill with the median of each column if there are any missing values
for col in data.columns:
    if data[col].dtype in ['float64', 'int64'] and data[col].isna().any():
        data[col].fillna(data[col].median(), inplace=True)

# Optionally, display the first few rows to verify
print(data.head())
```

```
   Unnamed: 0                              Name    Location  Year  \
0           0           Maruti Wagon R LXI CNG      Mumbai  2010
1           1  Hyundai Creta 1.6 CRDi SX Option        Pune  2015
2           2                     Honda Jazz V     Chennai  2011
3           3                Maruti Ertiga VDI     Chennai  2012
4           4   Audi A4 New 2.0 TDI Multitronic  Coimbatore  2013

   Kilometers_Driven  Mileage  Engine   Power  Seats  New_Price  Price  \
0              72000    26.60   998.0   58.16    5.0        NaN   1.75
1              41000    19.67  1582.0  126.20    5.0        NaN  12.50
```

```
2              46000    18.20  1199.0   88.70    5.0  8.61 Lakh   4.50
3              87000    20.77  1248.0   88.76    7.0        NaN   6.00
4              40670    15.20  1968.0  140.80    5.0        NaN  17.74

   Fuel_Type_Diesel  Fuel_Type_Electric  Fuel_Type_LPG  Fuel_Type_Petrol  \
0             False               False          False             False
1              True               False          False             False
2             False               False          False              True
3              True               False          False             False
4              True               False          False             False

   Transmission_Manual  Owner_Type_Fourth & Above  Owner_Type_Second  \
0                 True                      False              False
1                 True                      False              False
2                 True                      False              False
3                 True                      False              False
4                False                      False               True

   Owner_Type_Third
0             False
1             False
2             False
3             False
4             False
```

Step 18: Machine Learning Model Development.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Load the data
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
data = pd.read_csv(file_path)

# Remove columns with text data that are not relevant or cannot be converted
if 'Name' in data.columns:
    data.drop('Name', axis=1, inplace=True)

# Convert 'Price' from '8.61 Lakh' to 861000 and so on if it is expressed in lakhs
if 'Price' in data.columns:
    data['Price'] = data['Price'].replace(r'[^\d.]', '', regex=True).astype(float) * 100000
```

Step 19: Standardization of Data

```
categorical_cols = ['Fuel_Type', 'Transmission', 'Owner_Type', 'Location']  # Including 'Location'

import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load your dataset
file_path = '/content/drive/MyDrive/ST1_DATASET/train-data.csv'
data = pd.read_csv(file_path)

# Assuming 'Price' is not a feature but the target, and it's not included in the features set
features = data.drop(['Price'], axis=1)

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit and transform the features
# Note: StandardScaler expects numerical data, ensure non-numeric columns are either dropped or converted
# Convert all string data to float if necessary and fill NaNs
for column in features.columns:
    if features[column].dtype == 'object':
        features[column] = pd.to_numeric(features[column].str.replace('[^\d.]', '', regex=True), errors='coerce')
features.fillna(features.mean(), inplace=True)

# Scale the features
features_scaled = scaler.fit_transform(features)
```