```
# -*- coding: utf-8 -*-
"""Untitled39.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1PHKKbwR8IWqrMBjR-x5ZAruC
LyO5YcV2
"""

# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

# %matplotlib inline
plt.style.use('ggplot')
import missingno as msno

pd.set_option('display.float_format', lambda x: '%.3f' % x)
pd.set_option('display.max_rows', 300)
pd.set_option('display.max_colwidth',400)

#This cell imports necessary libraries like numpy, pandas, matplotli
b, seaborn,
#warnings, and missingno, and sets display options for pandas DataFra
mes to
# enhance readability.

data =pd.read_csv('/content/used_cars_data.csv')
#This cell loads the used_cars_data.csv file into a pandas DataFrame
named data.

data.head()

#This displays the first 5 rows of the DataFrame, providing an initia
l look at
```

```
#the data structure and content.

data.shape

#This shows the dimensions (number of rows and columns) of the DataFr
ame,
#which is (7253, 14).

data.info()

##This provides a concise summary of the DataFrame, including column
names,
#non-null counts, and data types, indicating the presence of missing
values
#and various data types.

data.isnull().sum()
#This calculates and displays the count of missing values for each co
lumn,
#highlighting columns like New_Price and Price with
#a significant number of missing entries.

msno.bar(data)
#This generates a bar plot using missingno to visually represent the
#completeness of each column, with longer bars indicating fewer missi
ng values.

data.nunique()
#This counts the number of unique values in each column,
#which helps identify categorical features and
#potential primary keys or highly varied columns.

data = data.drop(['S.No.'],axis=1)
#This drops the 'S.No.' column as it's likely an identifier and not u
seful for analysis.

from datetime import date
date.today().year
data['Car_Age']= date.today().year - data['Year']
data.head()
```

```
#This calculates the 'Car_Age' by subtracting the 'Year' of manufactu
re
#from the current year and adds it as a new column,
#then displays the updated DataFrame head.

data['Name'].unique()

#This shows all unique values in the 'Name' column,
#indicating a wide variety of car models and brands.

data['Brand']=data.Name.str.split().str.get(0)

#This extracts the first word from the 'Name' column to create a new
'Brand' column.

data['Brand'].sample(20)

data['Model']=data.Name.str.split().str.get(1).fillna(' ') +' '+data.
Name.str.split().str.get(2).fillna('').str.strip()

#This extracts the second and third words from the 'Name' column to c
reate a 'Model' column, handling potential missing values.

data['Model'].sample(20)

data['Brand'].unique()

data['Brand'].replace({"ISUZU":"Isuzu","Mini":"Mini Cooper","Land":"L
and Rover"},inplace=True)

#This standardizes brand names by replacing 'ISUZU' with 'Isuzu', 'Mi
ni' with 'Mini Cooper', and 'Land' with 'Land Rover'.

data['Brand'].unique()

data['Mileage'].sample(10)

data['Mileage_value']=data['Mileage'].astype(str).str.extract('(\d+
\.?\d*)').astype('float')
if data['Mileage_value'].isnull().any():
  data['Mileage_value'].fillna(data['Mileage_value'].median(),inplace
=True)

  #This extracts the unit part of the 'Mileage' column into a new 'Mi
leage_Unit' column and fills any missing values with the median of th
e column.

data['Mileage_Unit']=data['Mileage'].astype(str).str.extract('([a-zA-
Z/]+)')
if data['Mileage_Unit'].isnull().any():
  data['Mileage_Unit'].fillna(data['Mileage_Unit'].median(),inplace=T
rue)

#This extracts the unit part of the 'Mileage' column into a new 'Mile
age_Unit' column and fills any missing values with the median of the
column.

is_cng_Lpg = (data['Fuel_Type']=='CNG')|(data['Fuel_Type']=='LPG')
data.loc[is_cng_Lpg,'Mileage_value'] *= 1.4

data['Mileage_Unit'].fillna(data['Mileage_value'].median(),inplace=Tr
ue)

#  #This adjusts 'Mileage_value' for CNG/LPG cars by multiplying by
1.4 and fills any remaining missing 'Mileage_Unit' values with the me
dian of 'Mileage_value'.

data['Engine'] = data['Engine'].astype(str).str.replace('CC', '', reg
ex=False)
data['Engine'] =pd.to_numeric(data['Engine'],errors='coerce')

if data['Engine'].isnull().any():
  data['Engine'].fillna(data['Engine'].median(),inplace=True)

#This cleans the 'Engine' column by removing 'CC', converts it to num
eric, and fills missing values with the median.

data['Power']= data['Power'].astype(str).str.extract('(\\d+\\.?\\d
*)').astype('float')

if data['Power'].isnull().any():
    power_median = data['Power'].median()
    if pd.isna(power_median):
        # If median is NaN, it implies all values in the column are N
aN.
        # Fill with 0 as a fallback to ensure no missing values remai
n.
        data['Power'].fillna(0, inplace=True)
    else:
        # If median is a valid number, fill NaNs with it.
        data['Power'].fillna(power_median, inplace=True)

if data['Seats'].isnull().any():
  data['Seats'].fillna(data['Seats'].median(),inplace=True)

  #This fills any missing values in the 'Seats' column with its media
n

if data['Price'].isnull().any():
  data['Price'].fillna(data['Price'].median(),inplace=True)

  #This fills any missing values in the 'Seats' column with its media
n

data.isnull().sum()
#This re-checks for missing values in the DataFrame, confirming that
'Mileage', 'New_Price', and 'Mileage_Unit' still have missing values,
while others have been addressed.

data.head()

data.drop('Mileage', axis=1, inplace=True)
data.drop('New_Price', axis=1, inplace=True)
data.drop('Mileage_Unit', axis=1, inplace=True)
#This drops the original 'Mileage' and 'New_Price' columns, as well a
s 'Mileage_Unit' since new processed mileage values have been create
d.

cat_cols = data.select_dtypes(include='object').columns
cat_cols
# This identifies and displays the list of categorical columns remain
ing in the DataFrame.
```

```
num_cols =data.select_dtypes(include='number').columns
num_cols
#This identifies and displays the list of numerical columns remaining
in the DataFrame.

for col in num_cols:
  plt.figure(figsize=(10,5))
  plt.subplot(1,2,1)
  data[col].hist(grid = False, bins = 20, color = 'blue')
  plt.subplot(1,2,2)
  sns.boxplot(data[col])
  plt.show()

  #Histograms: These plots show the distribution of each numerical fe
ature. For example, Year is heavily skewed towards newer cars, Kilome
ters_Driven and Price are right-skewed with many cars having lower va
lues, and Car_Age reflects the Year distribution. Engine and Power al
so show particular distributions, indicating common engine sizes and
power outputs. Mileage_value might show a bimodal distribution if the
re are significant differences between fuel types or car categories.
#Box Plots: These plots visualize the spread and presence of outliers
for each numerical column. They effectively highlight the median, int
erquartile range, and potential extreme values for Kilometers_Driven,
Engine, Power, Price, and Mileage_value, indicating a considerable nu
mber of outliers in these features.

for col in cat_cols:
  plt.figure(figsize=(10,5))
  top_10 = data[col].value_counts().head(10).index
  sns.countplot(data=data, x=col, order=top_10)
  plt.xticks(rotation=90)
  plt.show()

  #These plots illustrate the frequency of the top 10 most common cat
egories within each categorical column (Name, Location, Fuel_Type, Tr
ansmission, Owner_Type, Brand, Model). For instance, you can see the
most frequent car brands, locations where cars are sold, dominant fue
l types (Petrol/Diesel), and the prevalence of Manual vs. Automatic t
ransmission. This helps identify popular choices and market trends.

plt.figure(figsize=(15,7))
```

```
sns.heatmap(data[num_cols].corr(), annot=True, fmt='.2f', vmin =-1, v
max =1, cmap ="Spectral")
#This heatmap displays the pairwise correlation coefficients between
all numerical features. It helps in understanding linear relationship
s between variables. For example, Year and Car_Age will have a strong
negative correlation (as one increases, the other decreases). Engine
and Power often show a positive correlation. The correlation with Pri
ce is particularly interesting, indicating which numerical factors ar
e most strongly associated with a car's price.

plt.figure(figsize=(15,7))
sns.lineplot(data= data, x='Year', y='Price', hue ='Transmission')

#This plot shows the trend of car prices over different years, differ
entiated by transmission type. It helps to observe if automatic cars
generally fetch higher prices than manual cars across different manuf
acturing years.

plt.figure(figsize=(15,7))
sns.lineplot(data= data, x='Year', y='Price', hue ='Fuel_Type')

#This plot illustrates how car prices vary with year of manufacture f
or different fuel types. It can reveal if certain fuel types maintain
value better or worse over time.

plt.figure(figsize=(15,7))
sns.lineplot(data= data, x='Year', y='Price', hue ='Owner_Type')

#This plot visualizes the relationship between car age and price, bro
ken down by owner type (First, Second, etc.). It typically shows that
cars with more owners tend to have lower prices.

plt.figure(figsize=(15,7))
sns.barplot(data= data, x='Seats', y='Price')
plt.grid()

#This bar plot shows the average price of cars based on the number of
seats. It can indicate if cars with more seats (e.g., SUVs or MUVs) g
enerally command higher prices than those with fewer seats.
```

```
"""**Used Car Price Prediction & Market Analysis**



**Business Analysis Highlights **

**Data Quality & Preprocessing:**
 You effectively handled missing values in critical columns like Mile
age, Engine, Power, Seats, and Price, ensuring data integrity. You al
so engineered new features such as Car_Age, Brand, Model, and Mileage
_value from raw data, which are crucial for richer analysis.
Market Trends by Age & Fuel Type:
 Your analysis of Year vs. Price (with Transmission, Fuel_Type, and O
wner_Type hues) revealed significant trends. For instance:
Newer cars generally command higher prices, as expected.
There are discernible price differences and depreciation patterns bet
ween different fuel types (e.g., Diesel vs. Petrol) and transmission
types (Automatic vs. Manual).
The number of previous owners (Owner_Type) directly impacts a car's r
esale value, with cars having fewer owners typically fetching higher
prices.
Key Influencers of Price:
 The correlation heatmap and individual feature distributions showed
strong relationships. Engine size and Power are positively correlated
with Price, indicating that more powerful cars with larger engines te
nd to be more expensive. Kilometers_Driven often shows an inverse rel
ationship, where higher mileage correlates with lower prices.
Feature Importance for Pricing:

 By extracting Brand and Model, you've laid the groundwork for identi
fying which specific brands and models hold their value better or are
in higher demand, providing granular insights into market dynamics.
Impact of Seating Capacity:
The bar plot of Seats vs. Price provides insight into how seating cap
acity influences average car prices, potentially highlighting segment
s like SUVs/MUVs with higher average prices due to more seats.


"""
```

```
"""**Business Recommendations (What can be done with Bthese insight
s):**
**Pricing Strategy Optimization: **Car dealerships or individual sell
ers can use these insights to set more competitive and accurate price
s for used cars, considering factors like age, mileage, fuel type, tr
ansmission, brand, model, and number of owners.
**Inventory Management:** Dealerships can optimize their inventory by
focusing on car types (fuel, transmission, brand, model) that show be
tter resale value retention or higher demand based on the observed pr
ice trends.
Marketing & Sales Targeting: Tailor marketing campaigns based on popu
lar car segments, fuel types, or brands that are currently performing
well in the market.
**Customer Advisory:** Advise potential buyers on factors that signif
icantly affect car value and depreciation, empowering them to make in
formed purchasing decisions.
Future Feature Development: The engineered features (e.g., Car_Age, M
ileage_value, Brand, Model) are robust inputs for building a machine
learning model to predict used car prices more accurately.
"""
```