Capstone_Project

August 26, 2024

1 Data Pre-processing

[97]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
#
```

[98]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[99]:
```
df = pd.read_csv('/content/drive/MyDrive/Colab data files/CAR DETAILS.csv')
df.head()
```

[99]:

	name	year	selling_price	km_driven	fuel	\
0	Maruti 800 AC	2007	60000	70000	Petrol	
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	
3	Datsun RediGO T Option	2017	250000	46000	Petrol	
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	

	seller_type	transmission	owner
0	Individual	Manual	First Owner
1	Individual	Manual	First Owner
2	Individual	Manual	First Owner
3	Individual	Manual	First Owner
4	Individual	Manual	Second Owner

[100]:
```
df['Brand'] = df['name'].str.split(expand=True)[0]
```

[101]:
```
df.head()
```

[101]:
```
                name            year  selling_price  km_driven   fuel   \
0             Maruti 800 AC  2007      60000        70000    Petrol
1  Maruti Wagon R LXI Minor  2007     135000        50000    Petrol
2      Hyundai Verna 1.6 SX  2012     600000       100000    Diesel
3    Datsun RediGO T Option  2017     250000        46000    Petrol
4     Honda Amaze VX i-DTEC  2014     450000       141000    Diesel

  seller_type transmission     owner      Brand
0  Individual    Manual      First Owner   Maruti
1  Individual    Manual      First Owner   Maruti
2  Individual    Manual      First Owner  Hyundai
3  Individual    Manual      First Owner   Datsun
4  Individual    Manual     Second Owner    Honda
```

[102]:
```
df.drop('name', axis=1, inplace=True)
```

[103]:
```
df.shape
```

[103]:
```
(4340, 8)
```

[104]:
```
df.describe()
```

[104]:
```
          year      selling_price    km_driven
count  4340.000000  4.340000e+03     4340.000000
mean   2013.090783  5.041273e+05    66215.777419
std       4.215344  5.785487e+05    46644.102194
min    1992.000000  2.000000e+04        1.000000
25%    2011.000000  2.087498e+05    35000.000000
50%    2014.000000  3.500000e+05    60000.000000
75%    2016.000000  6.000000e+05    90000.000000
max    2020.000000  8.900000e+06   806599.000000
```

[105]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   year           4340 non-null   int64
 1   selling_price  4340 non-null   int64
 2   km_driven      4340 non-null   int64
 3   fuel           4340 non-null   object
 4   seller_type    4340 non-null   object
 5   transmission   4340 non-null   object
 6   owner          4340 non-null   object
 7   Brand          4340 non-null   object
```

```
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

[106]:
```
df.duplicated().sum()
```

[106]:
```
790
```

[107]:
```
df.drop_duplicates(inplace=True)
```

[108]:
```
df.duplicated().sum()
```

[108]:
```
0
```

[109]:
```
df.shape
```

[109]:
```
(3550, 8)
```

[110]:
```
df.isna().sum()
```

[110]:
```
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
Brand            0
dtype: int64
```

[111]:
```
df.dtypes
```

[111]:
```
year              int64
selling_price     int64
km_driven         int64
fuel             object
seller_type      object
transmission     object
owner            object
Brand            object
dtype: object
```

[112]:
```
cat_cols = df.select_dtypes(include='object').columns.to_list()
num_cols = df.select_dtypes(exclude='object').columns.to_list()

print('Categorical Columns:', cat_cols)
print('Numerical Columns:', num_cols)
```

```
Categorical Columns: ['fuel', 'seller_type', 'transmission', 'owner', 'Brand']
Numerical Columns: ['year', 'selling_price', 'km_driven']
```

[113]:
```
df['Brand'].value_counts()
```

[113]:
```
Brand
Maruti           1057
Hyundai           631
Mahindra          324
Tata              308
Ford              220
Honda             216
Toyota            170
Chevrolet         151
Renault           108
Volkswagen         93
Nissan             52
Skoda              49
Fiat               32
Audi               31
Datsun             29
BMW                25
Mercedes-Benz      21
Jaguar              5
Mitsubishi          5
Land                5
Volvo               4
Jeep                3
Ambassador          3
MG                  2
OpelCorsa           2
Daewoo              1
Force               1
Isuzu               1
Kia                 1
Name: count, dtype: int64
```

[114]:
```
Brand_list = df["Brand"].value_counts().index.tolist()
```

[114]:
```
['Maruti',
 'Hyundai',
 'Mahindra',
 'Tata',
 'Ford',
 'Honda',
 'Toyota',
 'Chevrolet',
```

```
 'Renault',
 'Volkswagen',
 'Nissan',
 'Skoda',
 'Fiat',
 'Audi',
 'Datsun',
 'BMW',
 'Mercedes-Benz',
 'Jaguar',
 'Mitsubishi',
 'Land',
 'Volvo',
 'Jeep',
 'Ambassador',
 'MG',
 'OpelCorsa',
 'Daewoo',
 'Force',
 'Isuzu',
 'Kia']
```

[115]:
```
df_B = df.copy()
```

[116]:
```
# df = df_B.copy()
```

[117]:
```
# Get the last 12 elements (from -12 to -1)
last_12_brands = Brand_list[-19:]

last_12_brands  # This will give you the desired list of brands
```

[117]:
```
['Nissan',
 'Skoda',
 'Fiat',
 'Audi',
 'Datsun',
 'BMW',
 'Mercedes-Benz',
 'Jaguar',
 'Mitsubishi',
 'Land',
 'Volvo',
 'Jeep',
 'Ambassador',
 'MG',
 'OpelCorsa',
 'Daewoo',
```

```
 'Force',
 'Isuzu',
 'Kia']
```

[118]:
```
df["Brand"] = df["Brand"].apply(lambda x: x if x not in last_12_brands else␣
↪"Other")
```

[119]:
```
df['Brand'].value_counts()
```

[119]:
```
Brand
Maruti        1057
Hyundai        631
Mahindra       324
Tata           308
Other          272
Ford           220
Honda          216
Toyota         170
Chevrolet      151
Renault        108
Volkswagen      93
Name: count, dtype: int64
```

[120]:
```
df.Brand.unique().tolist()
```

[120]:
```
['Maruti',
 'Hyundai',
 'Other',
 'Honda',
 'Tata',
 'Chevrolet',
 'Toyota',
 'Mahindra',
 'Ford',
 'Renault',
 'Volkswagen']
```

[121]:
```
Brand_list = df["Brand"].value_counts().index.tolist()
Brand_list
```

[121]:
```
['Maruti',
 'Hyundai',
 'Mahindra',
 'Tata',
 'Other',
 'Ford',
 'Honda',
```

```
 'Toyota',
 'Chevrolet',
 'Renault',
 'Volkswagen']
```

[122]:
```
# Brand_List = ['Maruti',␣
 ↪'Hyundai','Mahindra','Tata','Other','Ford','Honda','Toyota','Chevrolet','Renault','Volkswag
```

[123]:
```
for i in cat_cols:
  print(f'for {i} column')
  print('\n')
  print(df[i].value_counts())
  print('\n')
```

```
for fuel column


fuel
Diesel      1789
Petrol      1701
CNG           37
LPG           22
Electric       1
Name: count, dtype: int64


for seller_type column


seller_type
Individual          2805
Dealer               712
Trustmark Dealer      33
Name: count, dtype: int64


for transmission column


transmission
Manual       3238
Automatic     312
Name: count, dtype: int64


for owner column
```

```
owner
First Owner             2199
Second Owner             970
Third Owner              289
Fourth & Above Owner      75
Test Drive Car            17
Name: count, dtype: int64


for Brand column


Brand
Maruti        1057
Hyundai        631
Mahindra       324
Tata           308
Other          272
Ford           220
Honda          216
Toyota         170
Chevrolet      151
Renault        108
Volkswagen      93
Name: count, dtype: int64
```

[124]:
```
df.shape
```

[124]:
```
(3550, 8)
```

[125]:
```
df1 = df.copy()
```

[126]:
```
df.head()
```

[126]:
```
   year  selling_price  km_driven    fuel  seller_type transmission  \
0  2007          60000      70000  Petrol   Individual       Manual
1  2007         135000      50000  Petrol   Individual       Manual
2  2012         600000     100000  Diesel   Individual       Manual
3  2017         250000      46000  Petrol   Individual       Manual
4  2014         450000     141000  Diesel   Individual       Manual

        owner      Brand
0  First Owner   Maruti
1  First Owner   Maruti
```

```
2   First Owner  Hyundai
3   First Owner    Other
4  Second Owner    Honda
```

[127]:
```
df.shape
```

[127]:
```
(3550, 8)
```

1.1 EDA

[310]:
```
!pip install pandas-profiling autoviz --quiet

import plotly
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
from autoviz.AutoViz_Class import AutoViz_Class
from IPython.display import Image
%matplotlib inline

template_style = 'plotly_white'
```

[129]:
```
df.nlargest(1, 'selling_price')
```

[129]:
```
      year  selling_price  km_driven   fuel  seller_type transmission  \
3872  2016      8900000       13000    Petrol    Dealer     Automatic

        owner     Brand
3872  First Owner  Other
```

[130]:
```
def grouped_data(column_name):
  '''
  This function will group the Data
  '''
  df_tmp = df.groupby(column_name).agg({'selling_price':'mean'}).reset_index()
  return df_tmp
```

[131]:
```
grouped_data('Brand')
```

[131]:
```
      Brand     selling_price
0    Chevrolet  2.327132e+05
1         Ford  5.636272e+05
2        Honda  5.399444e+05
3      Hyundai  4.134120e+05
4     Mahindra  5.847469e+05
5       Maruti  3.323959e+05
6        Other  1.073787e+06
```

```
7      Renault  4.072592e+05
8         Tata  2.786724e+05
9       Toyota  8.389176e+05
10  Volkswagen  4.616666e+05
```

[132]:
```
grouped_data('fuel')
```

[132]:
```
      fuel    selling_price
0      CNG  273162.081081
1   Diesel  614484.033538
2  Electric  310000.000000
3      LPG  171818.136364
4   Petrol  335894.316285
```

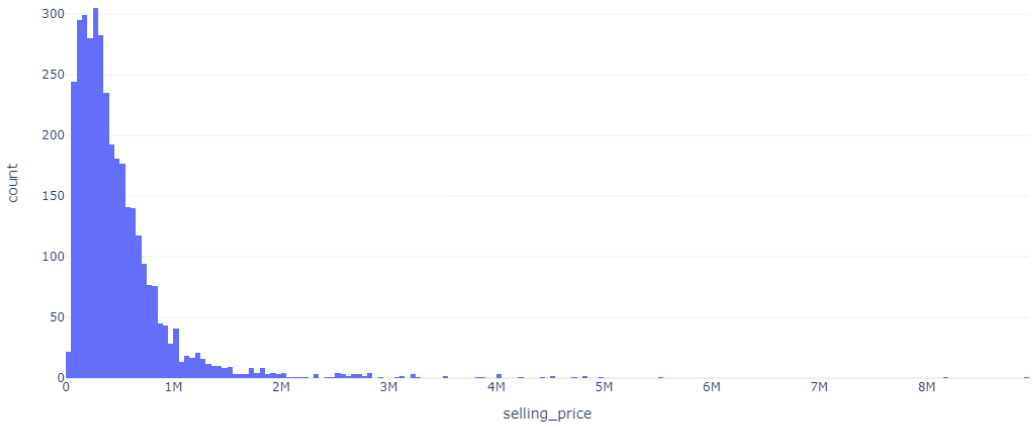
[303]:
```
!pip install -q kaleido --quiet

import kaleido
```

[133]:
```
fig = px.histogram(df, 'selling_price', template = template_style)
fig.show()
```

Show the distribution and skewness of the scale [Boxplot]

[311]:
```
Image("img1.png")
```

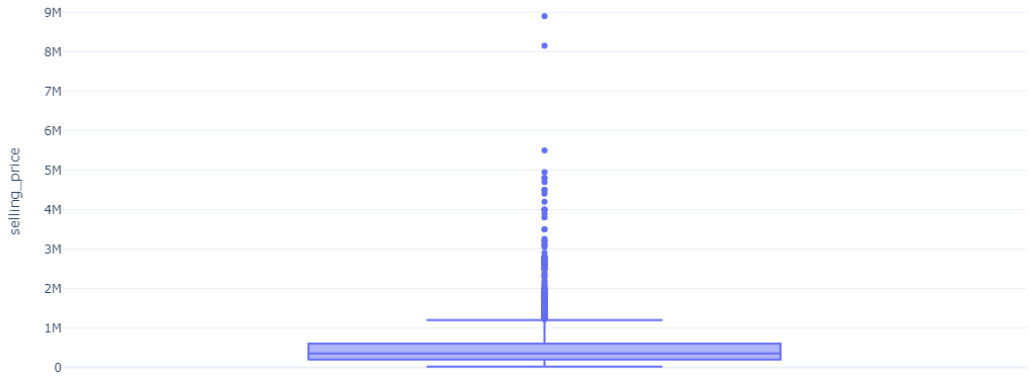
[311]:



[134]:
```
# Create Chart

fig = px.box(df, y ='selling_price', template = template_style)
fig.show()
```

[312]:
```
Image("img2.png")
```

[312]:



[135]:
```
df.head()
```

[135]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Brand
0	2007	60000	70000	Petrol	Individual	Manual	First Owner	Maruti
1	2007	135000	50000	Petrol	Individual	Manual	First Owner	Maruti
2	2012	600000	100000	Diesel	Individual	Manual	First Owner	Hyundai
3	2017	250000	46000	Petrol	Individual	Manual	First Owner	Other
4	2014	450000	141000	Diesel	Individual	Manual	Second Owner	Honda

[136]:
```
owner_type = grouped_data('owner')
owner_type
```

[136]:

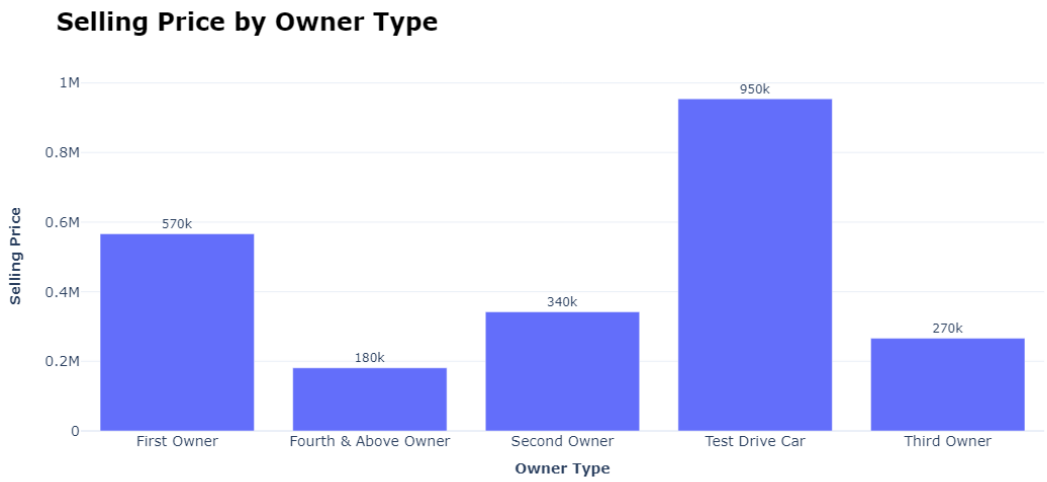
	owner	selling_price
0	First Owner	566632.524784
1	Fourth & Above Owner	181213.293333
2	Second Owner	342382.628866
3	Test Drive Car	954293.941176
4	Third Owner	266142.207612

[137]:
```
import plotly.express as px
import plotly.graph_objects as go

# Your existing code
fig = px.bar(
    owner_type,
    x='owner',
    y='selling_price',
    title='<b>Selling Price by Owner Type</b>',
    template=template_style  # Example of a dark theme
)

# Add labels on the bars
fig.update_traces(
    texttemplate='%{y:.2s}',
    textposition='outside',
    # marker_color='rgba(100, 150, 255, 0.7)'  # Customize bar color
)

# Customize layout (e.g., axis titles, font size)
fig.update_layout(
    xaxis_title='<b>Owner Type</b>',
    yaxis_title='<b>Selling Price</b>',
    title_font=dict(size=24, color = 'black'),
    xaxis=dict(tickfont=dict(size=14)),
    yaxis=dict(tickfont=dict(size=14)),
    # plot_bgcolor='rgba(0, 0, 0, 0)',  # Transparent background
    # paper_bgcolor='rgba(0, 0, 0, 0)'  # Transparent paper background
)

# Add hover effects (show more details on hover)
fig.update_traces(
    hovertemplate='<b>Owner: %{x}</b><br>Selling Price: %{y}<extra></extra>'
)

fig.show()

# Export the chart as an HTML file
plotly.offline.plot(fig, filename='sales_by_owner.html', auto_open=False)
```

[137]:
```
'sales_by_owner.html'
```

[313]:
```
Image("img3.png")
```

[313]:



[138]:
```
seller_type = grouped_data('seller_type')
seller_type
```

[138]:

	seller_type	selling_price
0	Dealer	652699.390449
1	Individual	425314.865597
2	Trustmark Dealer	822272.727273

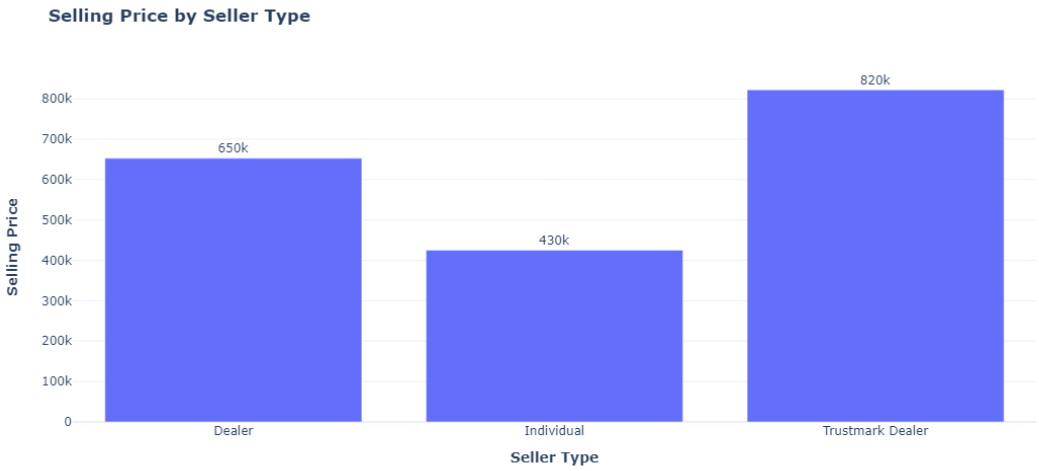
[139]:
```
fig = px.bar(seller_type,
             x='seller_type',
             y='selling_price',
             title = '<b>Selling Price by Seller Type<B>',
             template = template_style)

fig.update_traces(
    texttemplate='%{y:.2s}',
    textposition='outside',
    hovertemplate='<b>Seller: %{x}</b><br>Selling Price: %{y}<extra></extra>'
    # marker_color='rgba(100, 150, 255, 0.7)'  # Customize bar color
)

fig.update_layout(
    xaxis_title='<b>Seller Type</b>',
    yaxis_title='<b>Selling Price</b>',
)

fig.show()

# Export Chart
```

```
plotly.offline.plot(fig, filename = 'sales_by_seller.html', auto_open = False)
```

[139]:
```
'sales_by_seller.html'
```

[314]:
```
Image("img4.png")
```

[314]:



[140]:
```
fuel_type = grouped_data('fuel')
fuel_type
```

[140]:

	fuel	selling_price
0	CNG	273162.081081
1	Diesel	614484.033538
2	Electric	310000.000000
3	LPG	171818.136364
4	Petrol	335894.316285

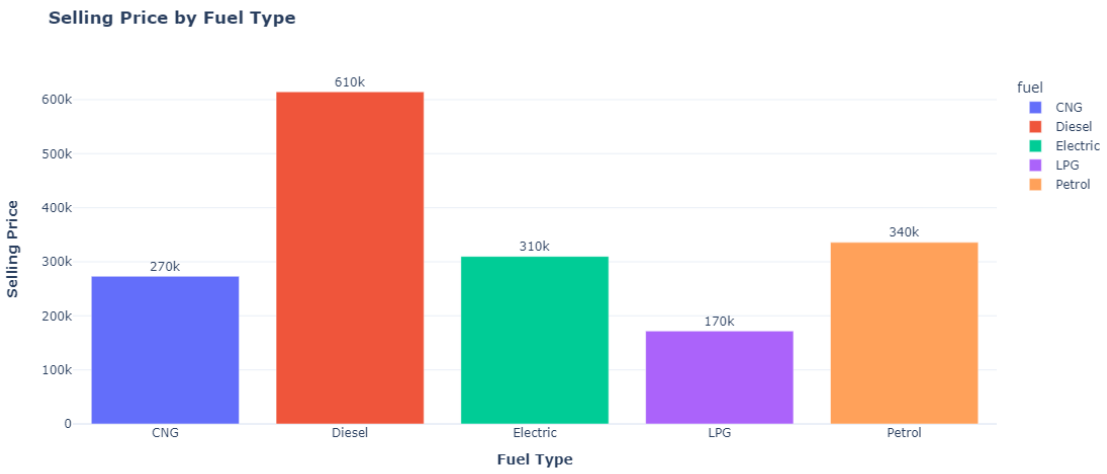
[141]:
```
fig = px.bar(fuel_type,
             x='fuel',
             y='selling_price',
             title = '<b>Selling Price by Fuel Type<B>',
             color = 'fuel',
             template = template_style)

fig.update_traces(
    texttemplate='%{y:.2s}',
    textposition='outside',
    hovertemplate='<b>Fuel: %{x}</b><br>Selling Price: %{y}<extra></extra>'
)
```

```
fig.update_layout(
    xaxis_title='<b>Fuel Type</b>',
    yaxis_title='<b>Selling Price</b>',
    plot_bgcolor='rgba(0, 0, 0, 0)',  # Transparent background
)

fig.show()

# Export Chart
plotly.offline.plot(fig, filename = 'sales_by_fuel.html', auto_open = False)
```

[141]:
```
'sales_by_fuel.html'
```

[315]:
```
Image("img5.png")
```

[315]:



[143]:
```
df.head()
```

[143]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Brand
0	2007	60000	70000	Petrol	Individual	Manual	First Owner	Maruti
1	2007	135000	50000	Petrol	Individual	Manual	First Owner	Maruti
2	2012	600000	100000	Diesel	Individual	Manual	First Owner	Hyundai
3	2017	250000	46000	Petrol	Individual	Manual	First Owner	Other

```
4  Second Owner    Honda
```

[144]:
```
fuel_type_owner = df.groupby(['fuel', 'owner']).agg({'selling_price':'mean'}).
↪reset_index()
fuel_type_owner
```

[144]:

	fuel	owner	selling_price
0	CNG	First Owner	3.736314e+05
1	CNG	Fourth & Above Owner	1.083333e+05
2	CNG	Second Owner	1.871538e+05
3	CNG	Third Owner	1.250000e+05
4	Diesel	First Owner	7.303064e+05
5	Diesel	Fourth & Above Owner	2.514062e+05
6	Diesel	Second Owner	4.493627e+05
7	Diesel	Test Drive Car	1.056286e+06
8	Diesel	Third Owner	3.656283e+05
9	Electric	Second Owner	3.100000e+05
10	LPG	First Owner	2.110000e+05
11	LPG	Fourth & Above Owner	6.000000e+04
12	LPG	Second Owner	1.488888e+05
13	LPG	Third Owner	1.350000e+05
14	Petrol	First Owner	4.042061e+05
15	Petrol	Fourth & Above Owner	1.323333e+05
16	Petrol	Second Owner	2.316879e+05
17	Petrol	Test Drive Car	8.828998e+05
18	Petrol	Third Owner	1.626431e+05

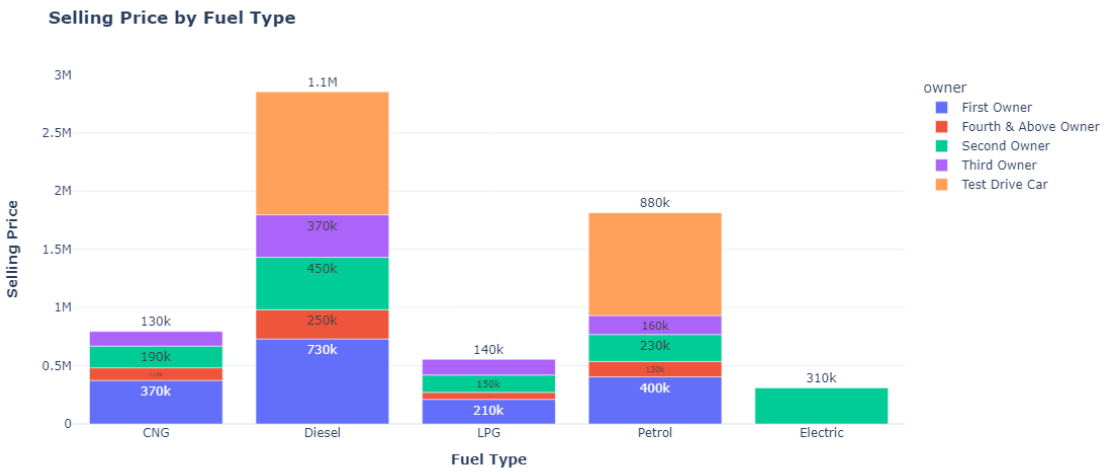
[145]:
```
import plotly.express as px
import plotly.graph_objects as go
import plotly

# Create the bar chart
fig = px.bar(fuel_type_owner,
             x='fuel',
             y='selling_price',
             title = '<b>Selling Price by Fuel Type</b>',
             color = 'owner',
             template = template_style,
             hover_data=['owner'])  # Ensure 'owner' is included in hover data

# Update traces with correct hovertemplate
fig.update_traces(
    texttemplate='%{y:.2s}',
    textposition='outside',
    hovertemplate='<b>Fuel: %{x}</b><br>Selling Price: %{y}<br>Owner:␣
↪%{customdata[0]}<extra></extra>'
)
```

```
fig.update_layout(
    xaxis_title='<b>Fuel Type</b>',
    yaxis_title='<b>Selling Price</b>',
)

fig.show()

# Export Chart
plotly.offline.plot(fig, filename='sales_by_fuel.html', auto_open=False)
```

[145]: 'sales_by_fuel.html'

[316]:
```
Image("img6.png")
```

[316]:



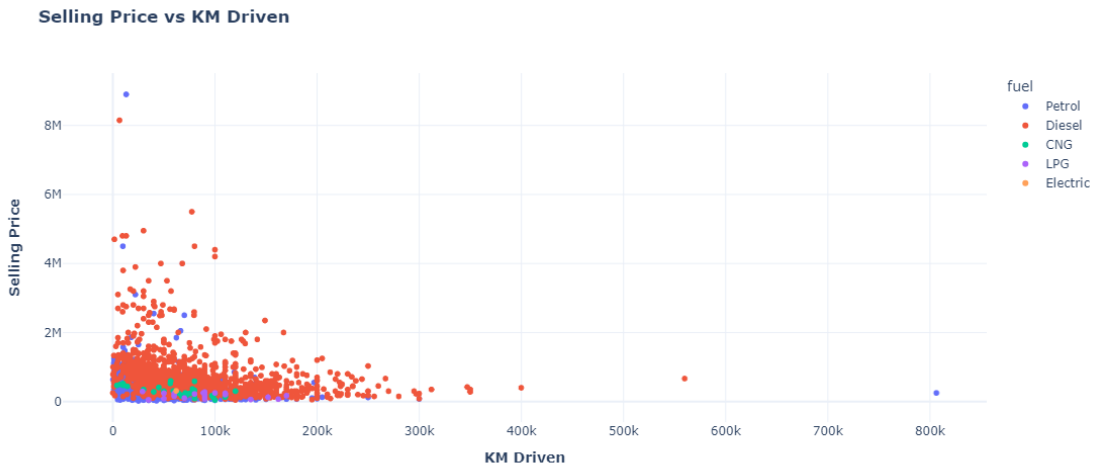
Scatter Chart between Selling Price and km_driven

[146]:
```
fig = px.scatter(df,
                 x = 'km_driven',
                 y = 'selling_price',
                 color = 'fuel',
                 title = '<b>Selling Price vs KM Driven</b>',
                 template = template_style,
                 hover_data = {'fuel':True})

fig.update_traces(
    hovertemplate='<b>Fuel: %{customdata[0]}</b><br>Selling Price: %{y}<br>KM␣
 ↪Driven: %{x}<extra></extra>'
)
```

```
fig.update_layout(
    xaxis_title='<b>KM Driven</b>',
    yaxis_title='<b>Selling Price</b>',
)

fig.show()
```

[317]:
```
Image("img7.png")
```

[317]:



[147]:
```
df.head()
```

[147]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Brand
0	2007	60000	70000	Petrol	Individual	Manual	First Owner	Maruti
1	2007	135000	50000	Petrol	Individual	Manual	First Owner	Maruti
2	2012	600000	100000	Diesel	Individual	Manual	First Owner	Hyundai
3	2017	250000	46000	Petrol	Individual	Manual	First Owner	Other
4	2014	450000	141000	Diesel	Individual	Manual	Second Owner	Honda

[148]:
```
fuel_type_km_driven = df.groupby(['fuel']).agg({'km_driven':'mean',␣
 ↪'selling_price':'sum'})
fuel_type_km_driven
```

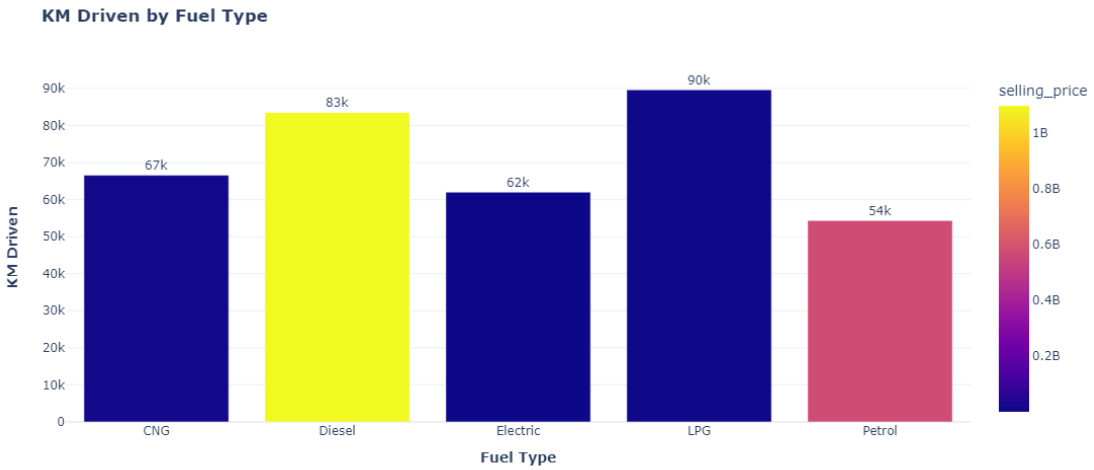
[148]:

fuel	km_driven	selling_price
CNG	66551.081081	10106997
Diesel	83495.335383	1099311936
Electric	62000.000000	310000
LPG	89618.181818	3779999
Petrol	54338.657848	571356232

[149]:
```
fig = px.bar(fuel_type_km_driven,
             x=fuel_type_km_driven.index,
             y='km_driven',
             title = '<b>KM Driven by Fuel Type<B>',
             color = 'selling_price',
             hover_data = {'selling_price':True},
             template = template_style)

fig.update_traces(
    texttemplate='%{y:.2s}',
    textposition='outside',
    hovertemplate='<b>Fuel: %{x}</b><br>KM_Driven: %{y}<br>Selling_price:␣
 ↪%{customdata[0]:2s}<extra></extra>'
)

fig.update_layout(
    xaxis_title='<b>Fuel Type</b>',
    yaxis_title='<b>KM Driven</b>',
)

fig.show()
```

[318]:
```
Image("img8.png")
```

[318]:



[150]:
```
df.head()
```

[150]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Brand
0	2007	60000	70000	Petrol	Individual	Manual	First Owner	Maruti
1	2007	135000	50000	Petrol	Individual	Manual	First Owner	Maruti
2	2012	600000	100000	Diesel	Individual	Manual	First Owner	Hyundai
3	2017	250000	46000	Petrol	Individual	Manual	First Owner	Other
4	2014	450000	141000	Diesel	Individual	Manual	Second Owner	Honda

[151]:
```
df_o = df.sort_values(by = 'year')
df_o.head()
```

[151]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Brand
3334	1992	50000	100000	Petrol	Individual	Manual	Fourth & Above Owner	Maruti
631	1995	95000	100000	Petrol	Individual	Manual	Second Owner	Maruti
61	1996	250000	35000	Diesel	Individual	Manual	Second Owner	Mahindra
2972	1996	200000	60000	Diesel	Individual	Manual	First Owner	Mahindra
1669	1997	150000	120000	Diesel	Individual	Manual	Third Owner	Mahindra

[152]:
```
sp_year = df_o.groupby('year').agg({'selling_price':'mean', 'km_driven':␣
↪'mean'}).reset_index()
sp_year.head() # Average profit
```

[152]:

	year	selling_price	km_driven
0	1992	50000.000000	100000.0
1	1995	95000.000000	100000.0
2	1996	225000.000000	47500.0
3	1997	93000.000000	90000.0
4	1998	165111.111111	70000.0

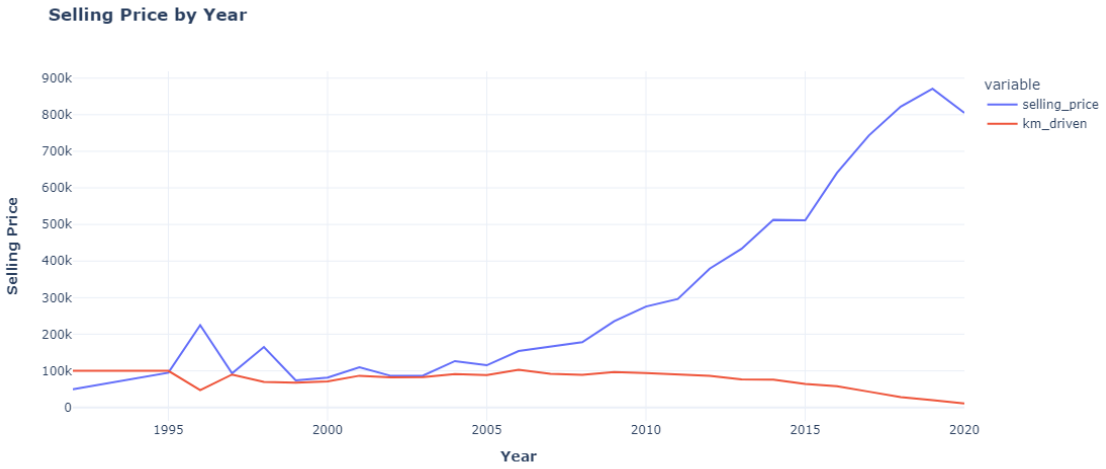
[153]:
```
sp_year.shape
```

[153]: (27, 3)

[154]:
```
fig = px.line(sp_year,
              x='year',
              y=['selling_price', 'km_driven'],
              title = '<b>Selling Price by Year<B>',
              template = template_style)

fig.update_traces(
    hovertemplate='<b>Year: %{x}</b><br>Selling Price: %{y}<extra></extra>'
)

fig.update_layout(
    xaxis_title='<b>Year</b>',
    yaxis_title='<b>Selling Price</b>',
)

fig.show()
```

[319]:
```
Image("img9.png")
```

[319]:



[155]:
```
cat_cols = df.select_dtypes(include='object').columns.to_list()
num_cols = df.select_dtypes(exclude='object').columns.to_list()

print('Categorical Columns:', cat_cols)
print('Numerical Columns:', num_cols)
```

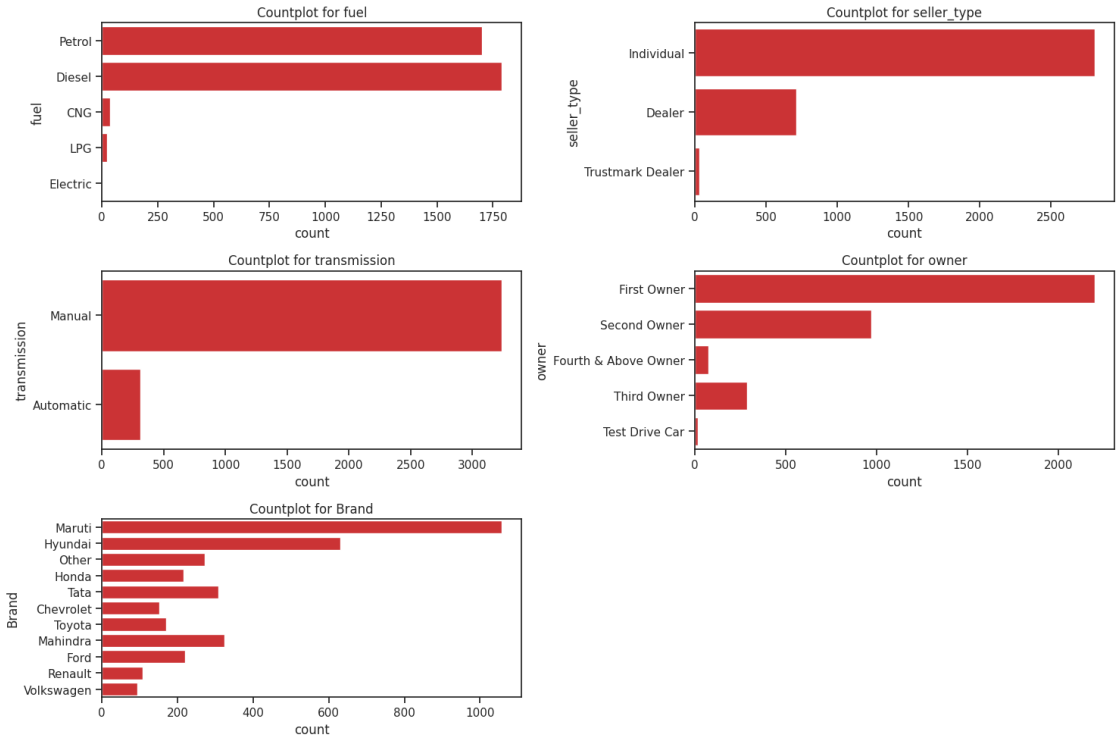
```
Categorical Columns: ['fuel', 'seller_type', 'transmission', 'owner', 'Brand']
Numerical Columns: ['year', 'selling_price', 'km_driven']
```

[156]:
```
sns.heatmap(df[num_cols].corr(), annot=True,cmap = plt.cm.CMRmap_r)
plt.show()
```

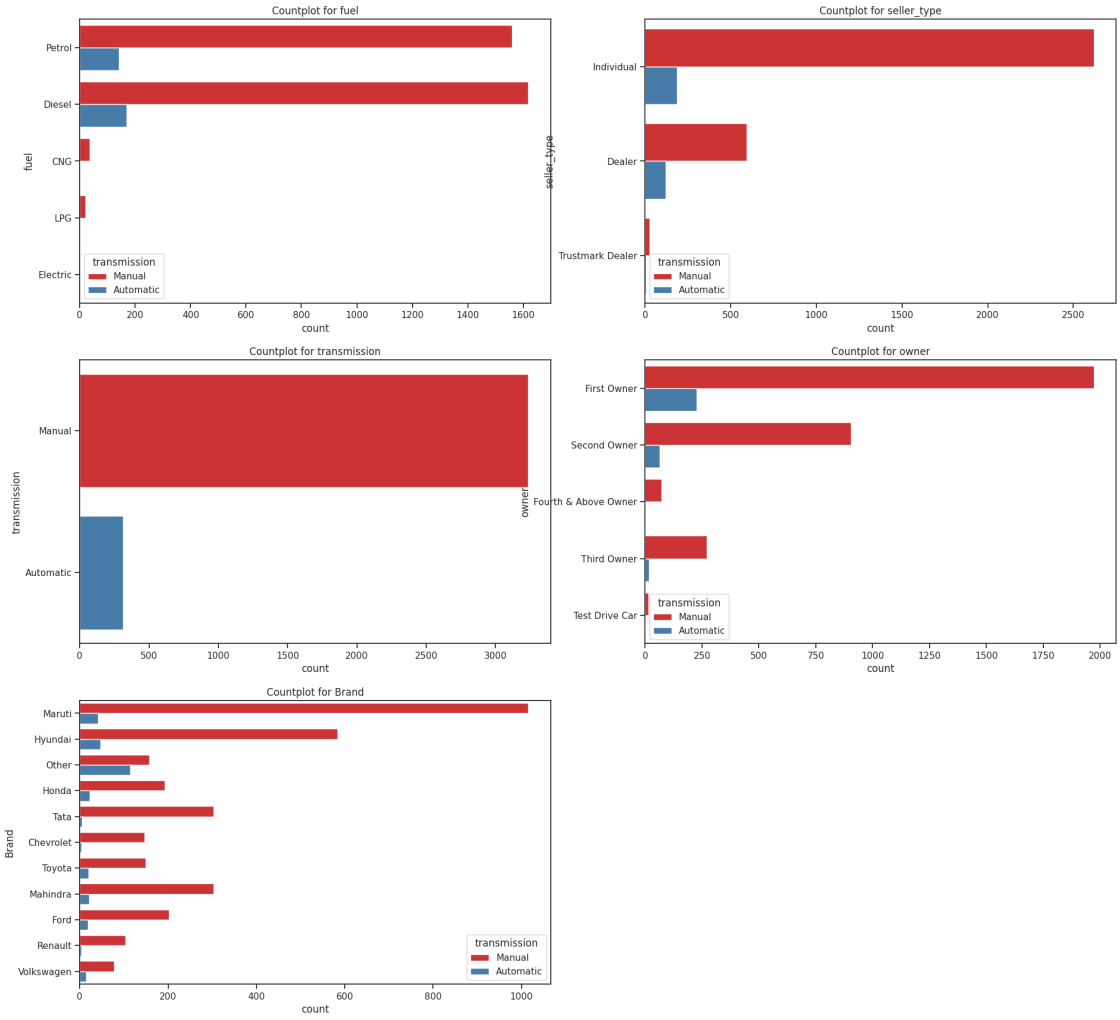
[157]:
```
cat_cols
```

[157]:
```
['fuel', 'seller_type', 'transmission', 'owner', 'Brand']
```

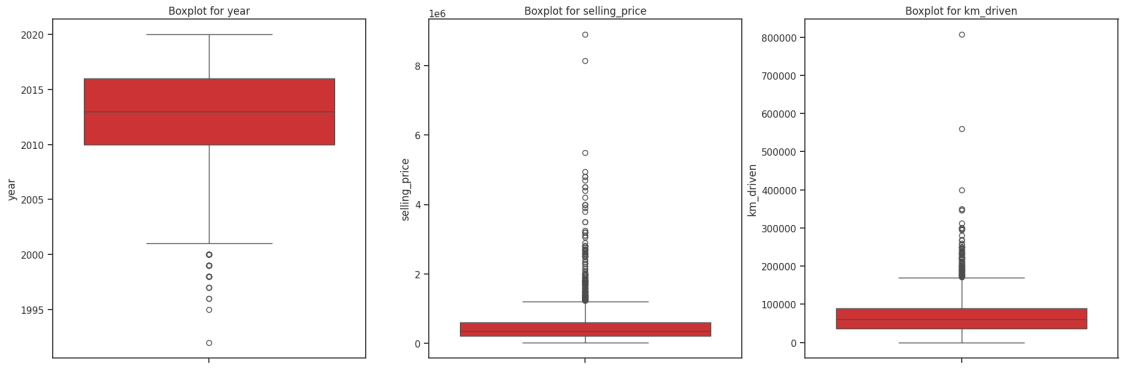
[158]:
```
plt.figure(figsize=(15, 10))
for i in range(len(cat_cols)):
  plt.subplot(3,2,i+1)
  sns.countplot(y = df[cat_cols[i]])
  plt.title(f'Countplot for {cat_cols[i]}')

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



[159]:
```
plt.figure(figsize=(20,20))
for i in range(len(cat_cols)):
    plt.subplot(3,2,i+1)
    sns.countplot(y=df[cat_cols[i]], hue = df['transmission'])
    plt.title(f'Countplot for {cat_cols[i]}')

plt.show()
```



[160]:
```
plt.figure(figsize=(20, 15)),
for i in range(len(num_cols)):
  plt.subplot(2,3,i+1)
  sns.boxplot(y = df[num_cols[i]])
  plt.title(f'Boxplot for {num_cols[i]}')

plt.show()
```



[161]:
```
det = df[num_cols].describe(percentiles = [0.01, 0.05, 0.10, 0.25, 0.50, 0.75,␣
↪0.85, 0.90, 0.95, 0.99]).T
det = det.iloc[:,3:]
det
```

[161]:

	min	1%	5%	10%	25%	50%	\
year	1992.0	2000.00	2005.0	2007.0	2010.0	2013.0	
selling_price	20000.0	51546.61	80000.0	110000.0	200000.0	350000.0	
km_driven	1.0	1728.42	10000.0	19104.0	36000.0	60658.0	

	75%	85%	90%	95%	99%	max
year	2016.0	2017.0	2018.0	2019.0	2020.0	2020.0
selling_price	600000.0	750000.0	875500.0	1200000.0	2675000.0	8900000.0
km_driven	90000.0	110000.0	120000.0	149853.3	223336.6	806599.0

[162]:
```
a =  df[num_cols].describe(percentiles=[0.01,0.02,0.03,0.05,0.75,0.80,0.85,0.
↪90,0.95,0.97,0.98,0.99]).T
a = a.iloc[:,3:]
a
```

[162]:

	min	1%	2%	3%	5%	50%	\
year	1992.0	2000.00	2002.98	2004.0	2005.0	2013.0	
selling_price	20000.0	51546.61	60000.00	70000.0	80000.0	350000.0	
km_driven	1.0	1728.42	5000.00	7000.0	10000.0	60658.0	

	75%	80%	85%	90%	95%	97%	\
year	2016.0	2017.0	2017.0	2018.0	2019.0	2019.0	
selling_price	600000.0	650000.0	750000.0	875500.0	1200000.0	1500000.0	
km_driven	90000.0	100000.0	110000.0	120000.0	149853.3	170000.0	

	98%	99%	max
year	2019.0	2020.0	2020.0
selling_price	1800500.0	2675000.0	8900000.0
km_driven	195000.0	223336.6	806599.0

[163]:
```
num_cols
```

[163]:
```
['year', 'selling_price', 'km_driven']
```

[164]:
```
df.head()
```

[164]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Brand
0	2007	60000	70000	Petrol	Individual	Manual	First Owner	Maruti
1	2007	135000	50000	Petrol	Individual	Manual	First Owner	Maruti
2	2012	600000	100000	Diesel	Individual	Manual	First Owner	Hyundai
3	2017	250000	46000	Petrol	Individual	Manual	First Owner	Other
4	2014	450000	141000	Diesel	Individual	Manual	Second Owner	Honda

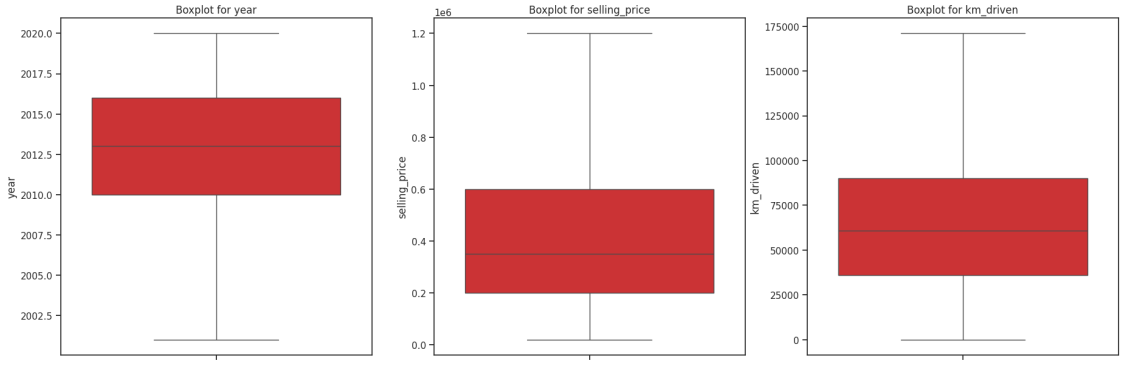
[165]:
```
df2 = df.copy()
```

[166]:
```
q1 = df['selling_price'].quantile(0.25)
q1
```

[166]:
```
200000.0
```

[168]:
```
# Removing Outliers

for i in num_cols:
  Q1 = df[i].quantile(0.25)
  Q3 = df[i].quantile(0.75)

  IQR = Q3 - Q1

  Upper = Q3 + 1.5 * IQR
  Lower = Q1 - 1.5 * IQR

  df[i] = df[i].clip(lower=Lower, upper=Upper)
```

[169]:
```
plt.figure(figsize=(20, 15)),
for i in range(len(num_cols)):
  plt.subplot(2,3,i+1)
  sns.boxplot(y = df[num_cols[i]])
  plt.title(f'Boxplot for {num_cols[i]}')

plt.show()
```



1.2 Standard Scaling

[170]:
```
df.shape
```

[170]:
```
(3550, 8)
```

[171]:
```
df['fuel'].value_counts()
```

[171]:
```
fuel
Diesel      1789
Petrol      1701
CNG           37
LPG           22
Electric       1
Name: count, dtype: int64
```

[172]:
```
df['fuel'] = df['fuel'].replace({'CNG': 'other','LPG': 'other','Electric':␣
↪'other'})
```

[173]:
```
df['fuel'].value_counts()
```

[173]:
```
fuel
Diesel    1789
Petrol    1701
other       60
Name: count, dtype: int64
```

[174]:
```
df['seller_type'].value_counts()
```

[174]:
```
seller_type
Individual          2805
Dealer               712
Trustmark Dealer      33
Name: count, dtype: int64
```

[175]:
```
df['seller_type'] = df['seller_type'].replace({'Trustmark Dealer': 'Other␣
↪Dealer', 'Dealer': 'Other Dealer'})
```

[176]:
```
df['seller_type'].value_counts()
```

[176]:
```
seller_type
Individual      2805
Other Dealer     745
Name: count, dtype: int64
```

[177]:
```
df['owner'].value_counts()
```

[177]:
```
owner
First Owner             2199
Second Owner             970
Third Owner              289
Fourth & Above Owner      75
Test Drive Car            17
Name: count, dtype: int64
```

[178]:
```
df['owner'] = df['owner'].replace({'Third Owner': 'Other Owner', 'Fourth &␣
↪Above Owner': 'Other Owner', 'Test Drive Car': 'Other Owner'})
```

[179]:
```
df['owner'].value_counts()
```

[179]:
```
owner
First Owner     2199
Second Owner     970
Other Owner      381
Name: count, dtype: int64
```

[180]:
```
df[df['km_driven'] == df['km_driven'].min()]
```

[180]:
```
      year  selling_price  km_driven   fuel  seller_type transmission  \
1312  2014         250000          1  Diesel   Individual       Manual

          owner       Brand
1312  Second Owner  Mahindra
```

[181]:
```
!pip install autoviz --quiet
```

[182]:
```
from autoviz.AutoViz_Class import AutoViz_Class
%matplotlib inline
```

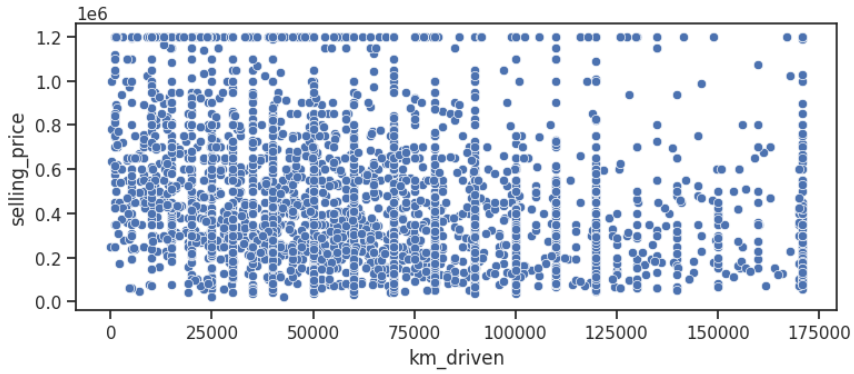
[183]:
```
from autoviz.AutoViz_Class import AutoViz_Class

# Initialize AutoViz class
```

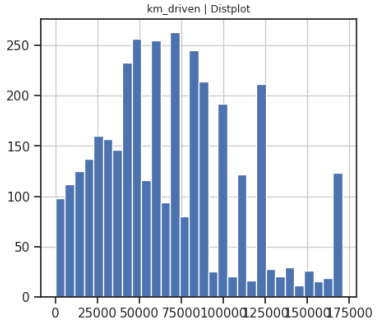
```
AV = AutoViz_Class()

df_AV = AV.AutoViz(df, depVar='selling_price')
```

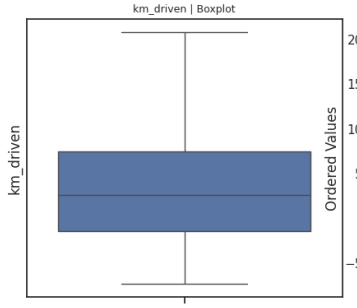
```
Shape of your Data Set loaded: (3550, 8)
################################################################################
#######
######################## C L A S S I F Y I N G  V A R I A B L E S
####################
################################################################################
#######
Classifying variables in data set…
    Number of Numeric Columns =  0
    Number of Integer-Categorical Columns =  1
    Number of String-Categorical Columns =  3
    Number of Factor-Categorical Columns =  0
    Number of String-Boolean Columns =  2
    Number of Numeric-Boolean Columns =  0
    Number of Discrete String Columns =  0
    Number of NLP String Columns =  0
    Number of Date Time Columns =  1
    Number of ID Columns =  0
    Number of Columns to Delete =  0
    7 Predictors classified…
        No variables removed since no ID or low-information variables found in
data set

################ Regression problem ####################
To fix these data quality issues in the dataset, import FixDQ from autoviz…
There are 7 duplicate rows in your dataset
    Alert: Dropping duplicate rows can sometimes cause your column data types to
change to object!
    All variables classified into correct types.
```

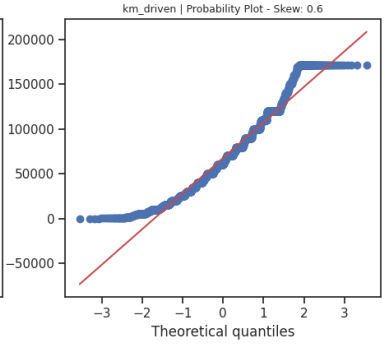
```
<pandas.io.formats.style.Styler at 0x7bd73a95f8e0>
```

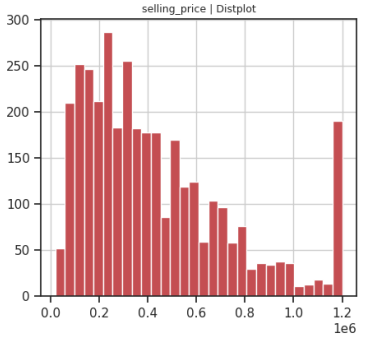
Scatter Plot of each Continuous Variable vs Target

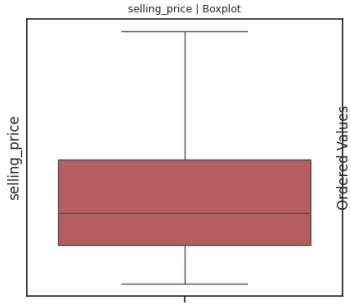


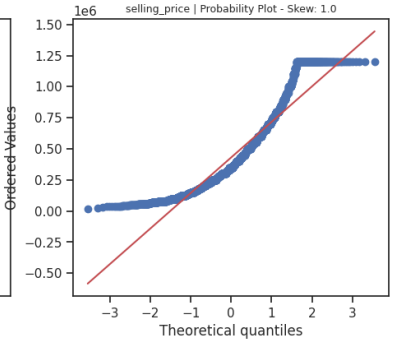

km_driven | Distplot
0
50
100
150
200
250
0
25000
50000
75000
100000
125000
150000
175000

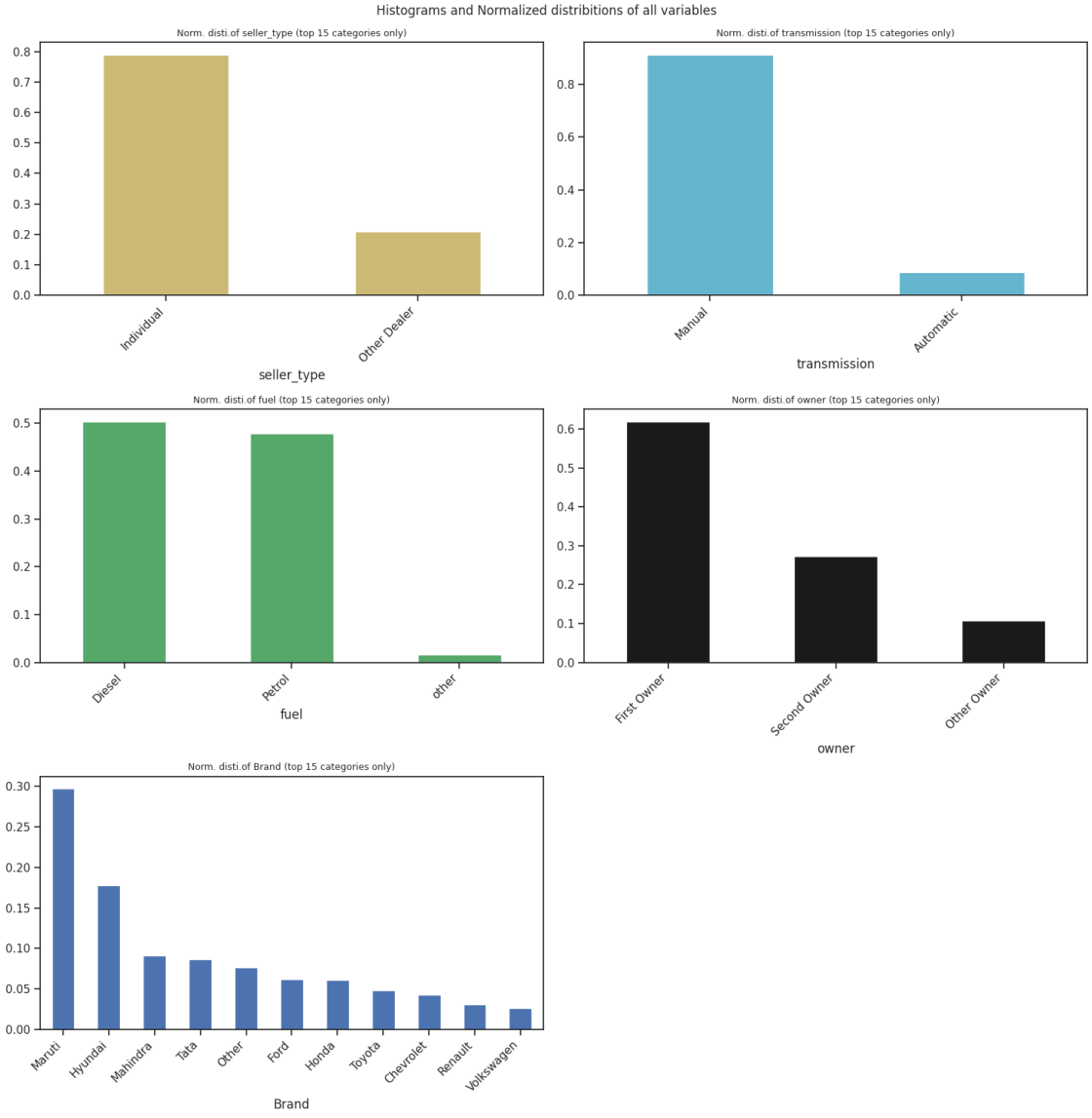

km_driven | Boxplot
km_driven

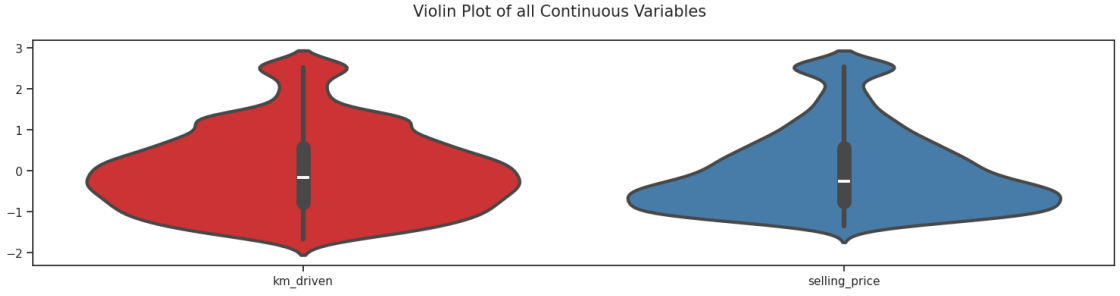

km_driven | Probability Plot - Skew: 0.6
Ordered Values
200000
150000
100000
50000
0
−50000
−3
−2
−1
0
1
2
3
Theoretical quantiles


selling_price | Distplot
0
50
100
150
200
250
300
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6


selling_price | Boxplot
selling_price

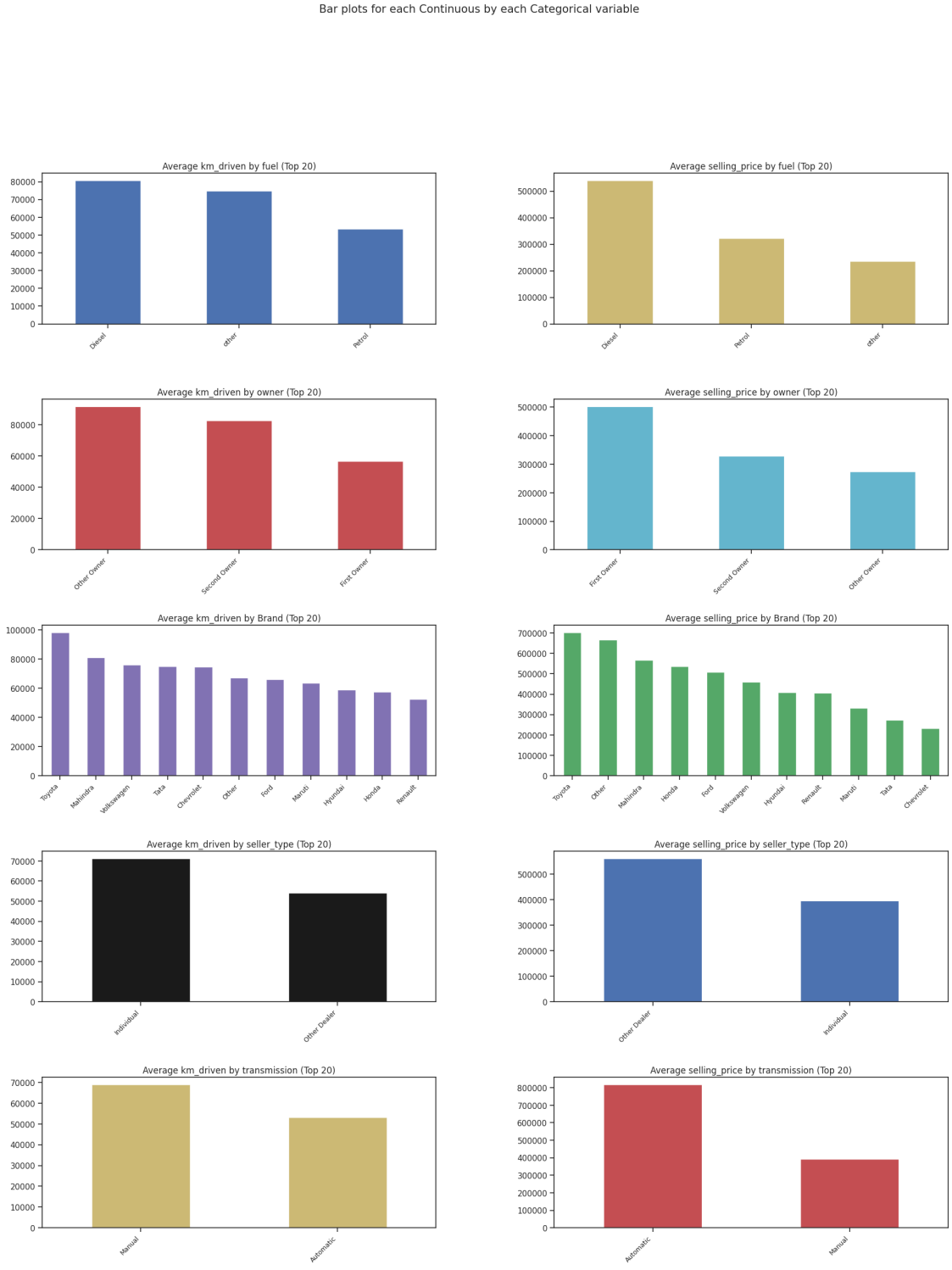

selling_price | Probability Plot - Skew: 1.0
1e6
Ordered Values
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−3
−2
−1
0
1
2
3
Theoretical quantiles





Could not draw some Time Series plots. agg function failed

[how->mean,dtype->object]



All Plots done

```
Time to run AutoViz = 15 seconds

 ###################### AUTO VISUALIZATION Completed ########################
```

[184]:
```
df.head()
```

[184]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	Brand
0	2007	60000	70000	Petrol	Individual	Manual	First Owner	Maruti
1	2007	135000	50000	Petrol	Individual	Manual	First Owner	Maruti
2	2012	600000	100000	Diesel	Individual	Manual	First Owner	Hyundai
3	2017	250000	46000	Petrol	Individual	Manual	First Owner	Other
4	2014	450000	141000	Diesel	Individual	Manual	Second Owner	Honda

[185]:
```
num_cols = ['year', 'km_driven']

x_sd = df.drop('selling_price', axis=1)
y = df['selling_price']

from sklearn.preprocessing import StandardScaler

sd = StandardScaler()
x_sd[num_cols] = sd.fit_transform(x_sd[num_cols])
```

2 Model Building

[216]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

def eval_metrics(model, M_name):
  model.fit(x_train, y_train)
  y_pred = model.predict(x_test)
  train_r2 = model.score(x_train, y_train)
  test_r2 = model.score(x_test, y_test)
  mae = mean_absolute_error(y_test,y_pred)
  mse = mean_squared_error(y_test,y_pred)
  rmse = np.sqrt(mse)
  R2_Score = r2_score(y_test,y_pred)
  res = pd.DataFrame({'Train_R2':train_r2,'Test_R2':test_r2,'MAE':mae,
                      'MSE':mse,'RMSE':rmse, 'R2_score':␣
 ↪R2_Score},index=[M_name])
```

```
  return res, y_pred
```

[217]:
```
# x_new = A.copy()
```

[218]:
```
x_new = x_sd.copy()
```

[219]:
```
x_new.head()
```

[219]:

	year	km_driven	fuel	seller_type	transmission	owner	Brand
0	-1.436687	0.057505	Petrol	Individual	Manual	First Owner	Maruti
1	-1.436687	-0.436835	Petrol	Individual	Manual	First Owner	Maruti
2	-0.236122	0.799016	Diesel	Individual	Manual	First Owner	Hyundai
3	0.964442	-0.535703	Petrol	Individual	Manual	First Owner	Other
4	0.244103	1.812414	Diesel	Individual	Manual	Second Owner	Honda

[220]:
```
x_new.shape
```

[220]:
```
(3550, 7)
```

[221]:
```
x_new.columns
```

[221]:
```
Index(['year', 'km_driven', 'fuel', 'seller_type', 'transmission', 'owner',
       'Brand'],
      dtype='object')
```

[222]:
```
x_new['fuel'].value_counts()
```

[222]:
```
fuel
Diesel    1789
Petrol    1701
other       60
Name: count, dtype: int64
```

[223]:
```
x_dummy = pd.get_dummies(x_new, drop_first=True)
x_dummy.shape
```

[223]:
```
(3550, 18)
```

[224]:
```
x_dummy.columns
```

[224]:
```
Index(['year', 'km_driven', 'fuel_Petrol', 'fuel_other',
       'seller_type_Other Dealer', 'transmission_Manual', 'owner_Other Owner',
       'owner_Second Owner', 'Brand_Ford', 'Brand_Honda', 'Brand_Hyundai',
       'Brand_Mahindra', 'Brand_Maruti', 'Brand_Other', 'Brand_Renault',
       'Brand_Tata', 'Brand_Toyota', 'Brand_Volkswagen'],
      dtype='object')
```

[225]:
```
x_dummy.head()
```

[225]:

	year	km_driven	fuel_Petrol	fuel_other	seller_type_Other Dealer	transmission_Manual	owner_Other Owner	owner_Second Owner	Brand_Ford	Brand_Honda	Brand_Hyundai	Brand_Mahindra	Brand_Maruti	Brand_Other	Brand_Renault	Brand_Tata	Brand_Toyota	Brand_Volkswagen
0	-1.436687	0.057505	True	False	False	True	False	False	False	False	False	False	True	False	False	False	False	False
1	-1.436687	-0.436835	True	False	False	True	False	False	False	False	False	False	True	False	False	False	False	False
2	-0.236122	0.799016	False	False	False	True	False	False	False	False	True	False	False	False	False	False	False	False
3	0.964442	-0.535703	True	False	False	True	False	False	False	False	False	False	False	True	False	False	False	False
4	0.244103	1.812414	False	False	False	True	False	True	False	True	False	False	False	False	False	False	False	False

[226]:
```
from sklearn.model_selection import train_test_split

x = x_dummy
y = y

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=42)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2485, 18)
(1065, 18)
(2485,)
(1065,)
```

[227]:
```
x.head()
```

[227]:

	year	km_driven	fuel_Petrol	fuel_other	seller_type_Other Dealer	transmission_Manual	owner_Other Owner	owner_Second Owner	Brand_Ford	Brand_Honda	Brand_Hyundai	Brand_Mahindra	Brand_Maruti	Brand_Other	Brand_Renault	Brand_Tata	Brand_Toyota	Brand_Volkswagen
0	-1.436687	0.057505	True	False	False	True	False	False	False	False	False	False	True	False	False	False	False	False
1	-1.436687	-0.436835	True	False	False	True	False	False	False	False	False	False	True	False	False	False	False	False
2	-0.236122	0.799016	False	False	False	True	False	False	False	False	True	False	False	False	False	False	False	False
3	0.964442	-0.535703	True	False	False	True	False	False	False	False	False	False	False	True	False	False	False	False
4	0.244103	1.812414	False	False	False	True	False	True	False	True	False	False	False	False	False	False	False	False

[228]:
```
x.columns
```

[228]:
```
Index(['year', 'km_driven', 'fuel_Petrol', 'fuel_other',
       'seller_type_Other Dealer', 'transmission_Manual', 'owner_Other Owner',
       'owner_Second Owner', 'Brand_Ford', 'Brand_Honda', 'Brand_Hyundai',
       'Brand_Mahindra', 'Brand_Maruti', 'Brand_Other', 'Brand_Renault',
       'Brand_Tata', 'Brand_Toyota', 'Brand_Volkswagen'],
      dtype='object')
```

[229]:
```
import pickle

dummy_columns = x.columns.tolist()

with open('dummy_columns.sav', 'wb') as f:
    pickle.dump(dummy_columns, f)
```

[230]:
```
dummy_columns
```

[230]:
```
['year',
 'km_driven',
 'fuel_Petrol',
 'fuel_other',
 'seller_type_Other Dealer',
 'transmission_Manual',
 'owner_Other Owner',
 'owner_Second Owner',
 'Brand_Ford',
 'Brand_Honda',
 'Brand_Hyundai',
 'Brand_Mahindra',
 'Brand_Maruti',
 'Brand_Other',
 'Brand_Renault',
 'Brand_Tata',
 'Brand_Toyota',
 'Brand_Volkswagen']
```

2.1 Linear Regression

[231]:
```
from sklearn.linear_model import LinearRegression

lr_1 = LinearRegression()
lr1_1_eval, y_pred_lr1 = eval_metrics(lr_1, 'Linear Regression')
lr1_1_eval
```

[231]:

	Train_R2	Test_R2	MAE	MSE	RMSE	R2_score
Linear Regression	0.650711	0.649719	136844.72932	3.295923e+10	181546.762446	0.649719

[232]:
```
from sklearn.linear_model import LinearRegression

lr_1 = LinearRegression()
lr1_1_eval, y_pred_lr1 = eval_metrics(lr_1, 'Linear Regression')
lr1_1_eval
```

[232]:

	Train_R2	Test_R2	MAE	MSE	RMSE	R2_score
Linear Regression	0.650711	0.649719	136844.72932	3.295923e+10	181546.762446	0.649719

2.2 KNN

[233]:
```
from sklearn.neighbors import KNeighborsRegressor

knn_1 = KNeighborsRegressor(n_neighbors=7)
knn1_1_eval, y_pred_knn1 = eval_metrics(knn_1, 'KNN') # Overfitting
knn1_1_eval
```

[233]:

	Train_R2	Test_R2	MAE	MSE	RMSE	R2_score
KNN	0.762238	0.702047	118970.336821	2.803546e+10	167437.936695	0.702047

[234]:
```
from sklearn.neighbors import KNeighborsRegressor

knn_2 = KNeighborsRegressor(n_neighbors=7, metric = 'manhattan')
knn1_2_eval, y_pred_knn1 = eval_metrics(knn_2, 'KNN') # Overfitting
knn1_2_eval
```

[234]:

	Train_R2	Test_R2	MAE	MSE	RMSE	R2_score
KNN	0.767171	0.704728	119320.839571	2.778326e+10	166683.106951	0.704728

2.3 Decision Tree

[235]:
```
from sklearn.tree import DecisionTreeRegressor

dt_1 = DecisionTreeRegressor()
dt1_1_eval, y_pred_dt1 = eval_metrics(dt_1, 'Decision Tree')
dt1_1_eval
```

[235]:

	Train_R2	Test_R2	MAE	MSE	RMSE	\
Decision Tree	0.971164	0.559764	137817.195696	4.142345e+10	203527.514178	

	R2_score
Decision Tree	0.559764

2.4 Random Forest

[236]:
```
from sklearn.ensemble import RandomForestRegressor

rf_1 = RandomForestRegressor()
rf_1_eval, y_pred_rf1 = eval_metrics(rf_1, 'Random Forest')
rf_1_eval
```

[236]:

	Train_R2	Test_R2	MAE	MSE	RMSE	\
Random Forest	0.935623	0.706575	117471.369151	2.760948e+10	166161.014001	

	R2_score
Random Forest	0.706575

[237]:
```
rf_2 = RandomForestRegressor(n_estimators=200,
                             min_samples_split=7,
                             min_samples_leaf=3,
                             max_features='sqrt',
                             max_depth=None,
                             bootstrap=False)

rf_2_eval, y_pred_rf = eval_metrics(rf_2, 'Best Random Forest')
rf_2_eval
```

[237]:
```
                     Train_R2   Test_R2       MAE            MSE      \
Best Random Forest   0.836567   0.720041   117276.410373   2.634243e+10

                         RMSE       R2_score
Best Random Forest   162303.506098   0.720041
```

2.5 Bagging Linear Regression

[238]:
```
from sklearn.ensemble import BaggingRegressor

bag_lr_1 = BaggingRegressor(base_estimator=LinearRegression())
bag_lr_1_eval, y_pred_bag_lr1 = eval_metrics(bag_lr_1, 'Bagging Linear␣
 ↪Regression')
bag_lr_1_eval
```

[238]:
```
                           Train_R2   Test_R2       MAE            MSE      \
Bagging Linear Regression  0.650426   0.649331   136993.268518   3.299571e+10

                               RMSE       R2_score
Bagging Linear Regression   181647.199499   0.649331
```

[239]:
```
bag_rf_1 = BaggingRegressor(base_estimator=RandomForestRegressor())
bag_rf_1_eval, y_pred_bag_rf1 = eval_metrics(bag_rf_1, 'Baggi RF Regression')
bag_rf_1_eval
```

[239]:
```
                      Train_R2   Test_R2       MAE            MSE      \
Baggi RF Regression   0.881955   0.726761   114954.002959   2.571007e+10

                          RMSE       R2_score
Baggi RF Regression   160343.586928   0.726761
```

2.6 XGBoost Model

[240]:
```
from xgboost import XGBRegressor

xgb_1 = XGBRegressor()
xgb1_1_eval, y_pred_xgb1 = eval_metrics(xgb_1, 'XGBoost')
```

```
xgb1_1_eval
```

[240]:
```
          Train_R2   Test_R2       MAE             MSE          RMSE       \
XGBoost   0.919266  0.705096  117160.497681  2.774863e+10  166579.197682

          R2_score
XGBoost   0.705096
```

[241]:
```
# {'subsample': 0.9,
#  'reg_lambda': 0,
#  'reg_alpha': 0.1,
#  'random_state': 42,
#  'objective': 'reg:squarederror',
#  'n_estimators': 500,
#  'min_child_weight': 7,
#  'max_depth': 3,
#  'learning_rate': 0.05,
#  'gamma': 0,
#  'colsample_bytree': 0.8}
```

[242]:
```
xgb_2 = XGBRegressor(n_estimators=1000,
                     learning_rate=0.05,
                     max_depth=3,
                     min_child_weight=7,
                     gamma=0,
                     subsample=0.9,
                     colsample_bytree=0.8,
                     reg_alpha=0.1,
                     reg_lambda=0)

xgb_2_eval, y_pred_xgb = eval_metrics(xgb_2, 'Best XGBoost')
xgb_2_eval
```

[242]:
```
              Train_R2   Test_R2       MAE             MSE          RMSE       \
Best XGBoost  0.817077  0.717132  115444.969507  2.661607e+10  163144.314079

              R2_score
Best XGBoost  0.717132
```

[243]:
```
result_df = pd.concat([lr1_1_eval, knn1_1_eval, dt1_1_eval, rf_1_eval,
                       rf_2_eval, bag_lr_1_eval, bag_rf_1_eval,
                       xgb1_1_eval, xgb_2_eval], axis=0).
 ↪sort_values(by='R2_score', ascending=False)
result_df
```

[243]:
```
                           Train_R2   Test_R2       MAE             MSE      \
Baggi RF Regression        0.881955  0.726761  114954.002959  2.571007e+10
```

```
Best Random Forest          0.836567  0.720041  117276.410373  2.634243e+10
Best XGBoost                0.817077  0.717132  115444.969507  2.661607e+10
Random Forest               0.935623  0.706575  117471.369151  2.760948e+10
XGBoost                     0.919266  0.705096  117160.497681  2.774863e+10
KNN                         0.762238  0.702047  118970.336821  2.803546e+10
Linear Regression           0.650711  0.649719  136844.729320  3.295923e+10
Bagging Linear Regression   0.650426  0.649331  136993.268518  3.299571e+10
Decision Tree               0.971164  0.559764  137817.195696  4.142345e+10
```

```
                                RMSE       R2_score
Baggi RF Regression         160343.586928  0.726761
Best Random Forest          162303.506098  0.720041
Best XGBoost                163144.314079  0.717132
Random Forest               166161.014001  0.706575
XGBoost                     166579.197682  0.705096
KNN                         167437.936695  0.702047
Linear Regression           181546.762446  0.649719
Bagging Linear Regression   181647.199499  0.649331
Decision Tree               203527.514178  0.559764
```

[244]:
```
import pickle

pickle.dump(bag_rf_1, open('bag_rf_model.sav', 'wb'))
pickle.dump(knn_2, open('knn_model.sav', 'wb'))
pickle.dump(xgb_2, open('xgb_model.sav', 'wb'))
```

[245]:
```
pickle.dump(sd, open('sd.sav', 'wb'))
```

3 Sampel Testing

[247]:
```
original_df = pd.read_csv('/content/drive/MyDrive/Colab data files/CAR DETAILS.
 ↪csv')
original_df.head()
```

[247]:

	name	year	selling_price	km_driven	fuel	\
0	Maruti 800 AC	2007	60000	70000	Petrol	
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	
3	Datsun RediGO T Option	2017	250000	46000	Petrol	
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	

	seller_type	transmission	owner
0	Individual	Manual	First Owner
1	Individual	Manual	First Owner
2	Individual	Manual	First Owner
3	Individual	Manual	First Owner

```
4  Individual    Manual     Second Owner
```

[ ]:

[249]:
```
sampel_df = original_df.sample(20)
sampel_df
```

[249]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
685	Maruti Wagon R LX Minor	2013	250000	38000	Petrol	Individual	Manual	First Owner
1488	Maruti 800 AC BSIII	2005	120000	20000	Petrol	Individual	Manual	Second Owner
1118	Hyundai i20 Asta 1.2	2016	500000	40000	Petrol	Individual	Manual	First Owner
3134	Chevrolet Cruze LTZ	2011	345000	65500	Diesel	Individual	Manual	First Owner
621	Maruti S-Cross Zeta DDiS 200 SH	2015	750000	45974	Diesel	Trustmark Dealer	Manual	First Owner
438	Honda Accord 2.4 MT	2009	350000	57035	Petrol	Dealer	Manual	First Owner
616	Maruti Swift ZXI BSIV	2016	670000	7104	Petrol	Trustmark Dealer	Manual	First Owner
1646	BMW X1 sDrive 20d xLine	2019	2600000	9500	Diesel	Individual	Automatic	First Owner
567	Hyundai Santro GS	2005	80000	56580	Petrol	Dealer	Manual	First Owner
676	Mahindra Bolero SLX	2011	311000	140000	Diesel	Individual	Manual	First Owner
2039	Hyundai Grand i10 1.2 CRDi Sportz Option	2017	509999	44000	Diesel	Dealer	Manual	First Owner
2327	Maruti Alto 800 VXI	2016	245000	60000	Petrol	Individual	Manual	First Owner
1082	Hyundai Santro Xing XL eRLX Euro III	2005	114999	90000	Petrol	Dealer	Manual	Second Owner
3435	Ford Figo 1.5D Ambiente ABS MT	2016	350000	60000	Diesel	Individual	Manual	Second Owner
1162	Maruti Alto LX	2004	110000	60000	Petrol	Individual	Manual	First Owner
2226	Hyundai EON D Lite	2016	210000	30000	Petrol	Individual	Manual	First Owner
4282	Maruti Wagon R LX Minor	2013	290000	52000	Petrol	Individual	Manual	First Owner
4208	Toyota Qualis FS B3	2001	150000					
2702	Hyundai Verna 1.6 CRDI	2012	500000					
1345	Maruti A-Star Vxi	2009	120000					

```
4208   256000    Diesel            Dealer      Manual    First Owner
2702    26000    Diesel        Individual      Manual    First Owner
1345    76000    Petrol        Individual      Manual   Second Owner
```

[261]:
```
df = sampel_df.copy()
```

[253]:
```
df.shape
```

[253]:
```
(20, 8)
```

[263]:
```
df.head()
```

[263]:
```
                    name                year  selling_price  km_driven   fuel   \
685           Maruti Wagon R LX Minor  2013     250000        38000    Petrol
1488              Maruti 800 AC BSIII  2005     120000        20000    Petrol
1118             Hyundai i20 Asta 1.2  2016     500000        40000    Petrol
3134              Chevrolet Cruze LTZ  2011     345000        65500    Diesel
621   Maruti S-Cross Zeta DDiS 200 SH  2015     750000        45974    Diesel

        seller_type    transmission     owner
685         Individual    Manual      First Owner
1488        Individual    Manual     Second Owner
1118        Individual    Manual      First Owner
3134        Individual    Manual      First Owner
621   Trustmark Dealer    Manual      First Owner
```

[250]:
```
# All the Preprocessing steps for Trained Dataset must also be applied here
```

[264]:
```
df['Brand'] = df['name'].str.split(expand=True)[0]
df["Brand"] = df["Brand"].apply(lambda x: x if x not in last_12_brands else␣
 ↪"Other")
```

[265]:
```
# applying all the settings

df.drop('name', axis=1, inplace=True)
df.drop_duplicates(inplace=True)
df['fuel'] = df['fuel'].replace({'CNG': 'other','LPG': 'other','Electric':␣
 ↪'other'})
df['seller_type'] = df['seller_type'].replace({'Trustmark Dealer': 'Other␣
 ↪Dealer', 'Dealer': 'Other Dealer'})
df['owner'] = df['owner'].replace({'Third Owner': 'Other Owner', 'Fourth &␣
 ↪Above Owner': 'Other Owner', 'Test Drive Car': 'Other Owner'})
```

[280]:
```
num_cols = ['year', 'km_driven']

x_smpl = df.drop('selling_price', axis=1)
# y = df['selling_price']
```

```
x_sam = x_smpl.copy()

from sklearn.preprocessing import StandardScaler

sd = StandardScaler()
x_sam[num_cols] = sd.fit_transform(x_sam[num_cols])
```

[281]:
```
x_new = x_sam.copy()
```

[282]:
```
x_dummy = pd.get_dummies(x_new, drop_first=True)
x_dummy.shape
```

[282]:
```
(20, 13)
```

[283]:
```
x_test = x_dummy
```

[284]:
```
with open('dummy_columns.sav', 'rb') as f:
    dummy_columns = pickle.load(f)
```

[285]:
```
x_test = x_test.reindex(columns=dummy_columns, fill_value=0)
```

[286]:
```
model = pickle.load(open('bag_rf_model.sav', 'rb'))
```

[287]:
```
y_pred = model.predict(x_test)
```

[288]:
```
y_pred
```

[288]:
```
array([ 265783.98545238,  119148.13687143,  480072.58158333,
        283534.54128205,  650180.43809957,  352435.891     ,
        443479.79798968, 1184796.999     ,  158966.30333333,
        370657.97619048,  954623.43458333,  453965.14713309,
        143847.26371429,  738057.07945761,  101595.02468254,
        537279.11107857,  239997.08571905,  306455.02970818,
        482716.03727706,  191308.38168153])
```

[293]:
```
x_smpl["predicted_price"] = y_pred.round(-4)
```

[295]:
```
x_smpl.head()
```

[295]:
```
      year  km_driven   fuel   seller_type  transmission     owner      \
685   2013    38000    Petrol    Individual    Manual      First Owner
1488  2005    20000    Petrol    Individual    Manual     Second Owner
1118  2016    40000    Petrol    Individual    Manual      First Owner
3134  2011    65500    Diesel    Individual    Manual      First Owner
621   2015    45974    Diesel  Other Dealer    Manual      First Owner

       Brand     predicted_price
```

```
685       Maruti     270000.0
1488      Maruti     120000.0
1118     Hyundai     480000.0
3134   Chevrolet     280000.0
621       Maruti     650000.0
```

[296]:
```
x_smpl.to_csv('sample_prediction.csv', index=False)
```