Market Segment Analysis of EV Vehicles

by

Prajapati Taksh Rajeshkumar

Dataset used:

https://drive.google.com/file/d/1yeTKNvAxCALz4QIKluGZqDFc6GbHt9dV/view?usp=sharing

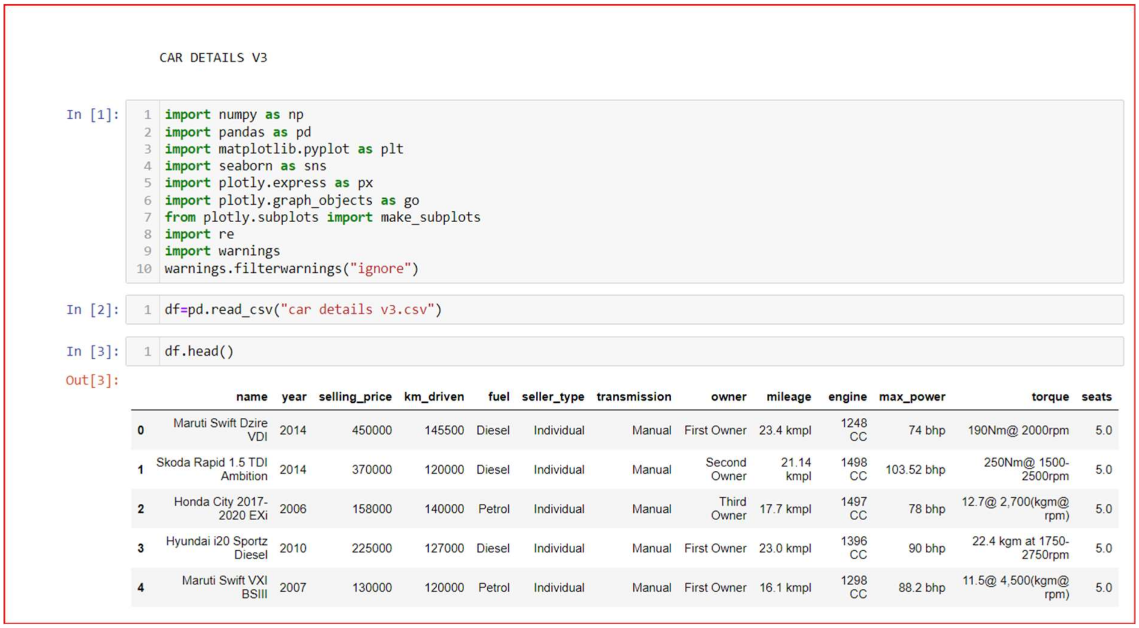
Project link:

https://github.com/TakshPrajapati/Intern_Feynnlabs

1. Data Pre-Processing:

Data preprocessing is a crucial step in preparing raw data to make it suitable for machine learning models. The process involves cleaning the data, removing any errors or inconsistencies, and transforming it into a format that can be easily analyzed. It is essential to preprocess the data before performing any segmentation analysis.

To preprocess data, the first step is to import the raw data in a suitable format and create a data frame for further analysis. The next step is to identify any null values in the dataset and remove them to avoid any data inconsistencies.



CAR DETAILS V3

In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import re
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

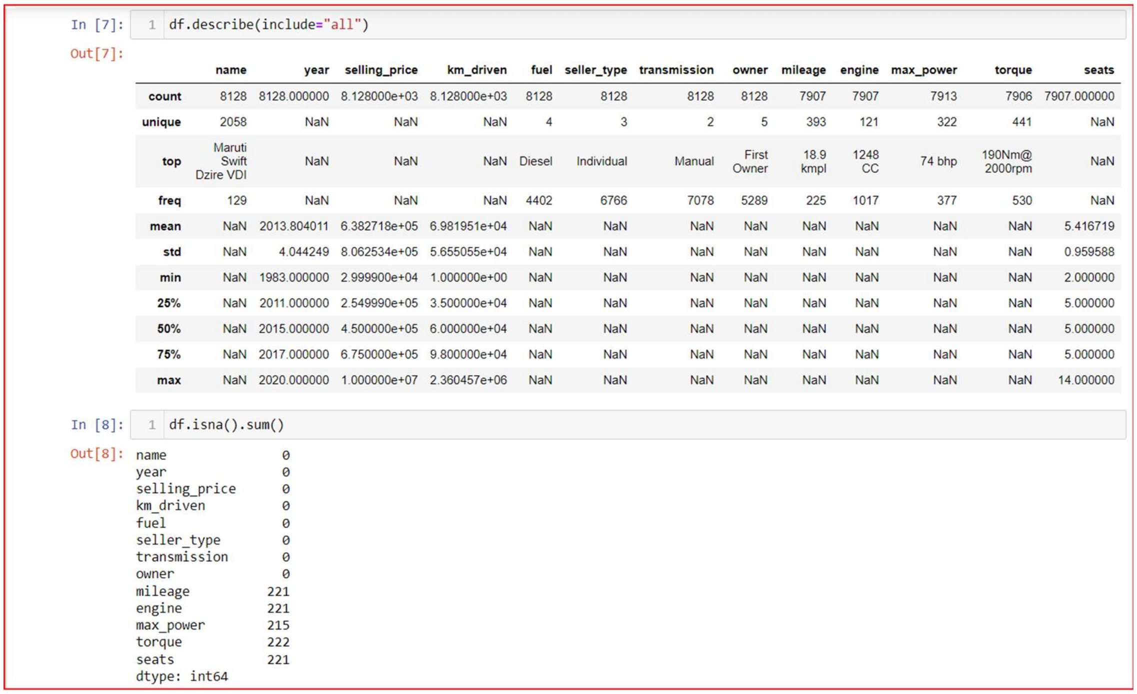
```python
df=pd.read_csv("car details v3.csv")
```

In [3]:

```python
df.head()
```

Out[3]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0



In [7]:

```python
df.describe(include="all")
```

Out[7]:

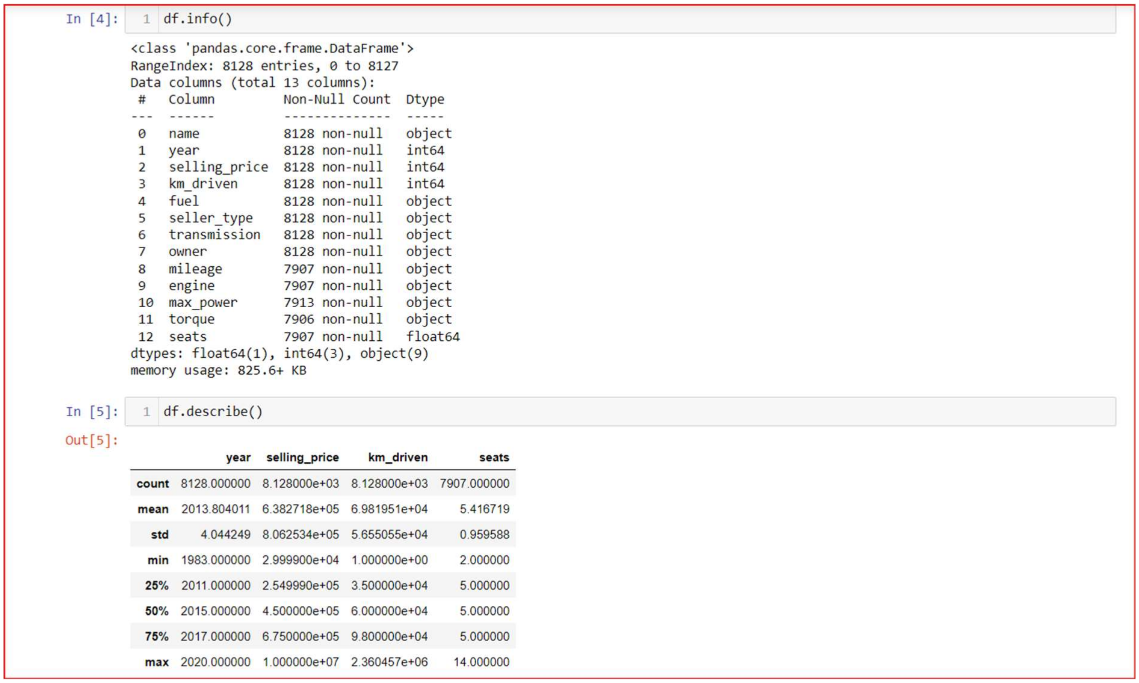
	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
count	8128	8128.000000	8.128000e+03	8.128000e+03	8128	8128	8128	8128	7907	7907	7913	7906	7907.000000
unique	2058	NaN	NaN	NaN	4	3	2	5	393	121	322	441	NaN
top	Maruti Swift Dzire VDI	NaN	NaN	NaN	Diesel	Individual	Manual	First Owner	18.9 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	NaN
freq	129	NaN	NaN	NaN	4402	6766	7078	5289	225	1017	377	530	NaN
mean	NaN	2013.804011	6.382718e+05	6.981951e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.416719
std	NaN	4.044249	8.062534e+05	5.655055e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.959588
min	NaN	1983.000000	2.999900e+04	1.000000e+00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.000000
25%	NaN	2011.000000	2.549990e+05	3.500000e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.000000
50%	NaN	2015.000000	4.500000e+05	6.000000e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.000000
75%	NaN	2017.000000	6.750000e+05	9.800000e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.000000
max	NaN	2020.000000	1.000000e+07	2.360457e+06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	14.000000

In [8]:

```python
df.isna().sum()
```

Out[8]:

```
name               0
year               0
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage          221
engine           221
max_power        215
torque           222
seats            221
dtype: int64
```



In [4]:
```
df.info()
```

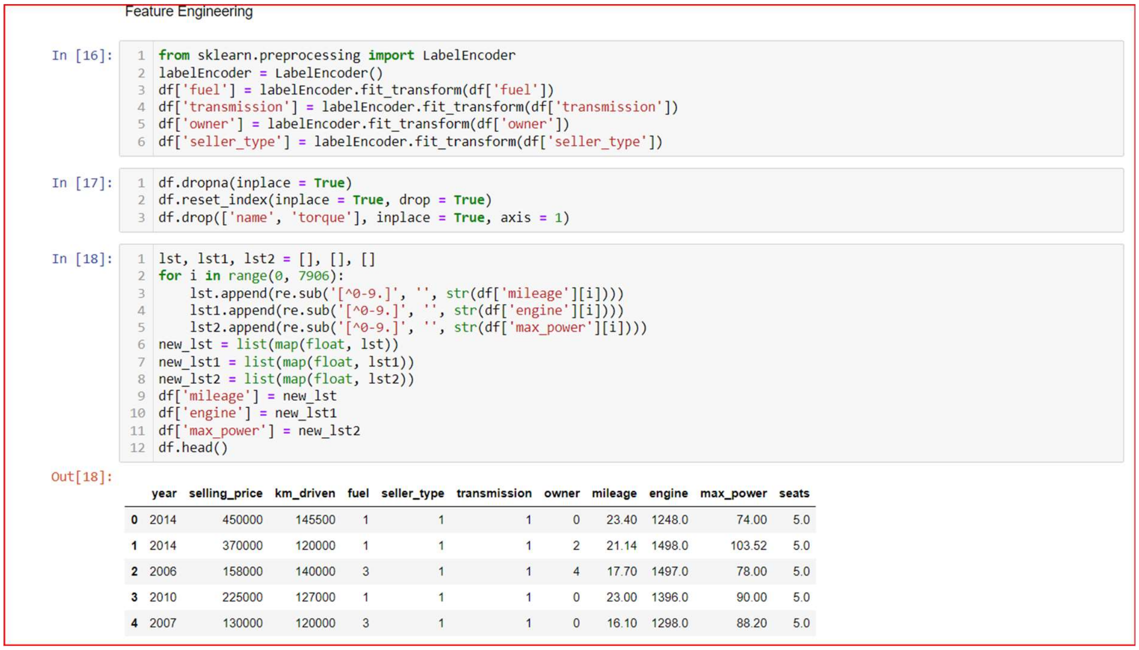
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           8128 non-null   object
 1   year           8128 non-null   int64
 2   selling_price  8128 non-null   int64
 3   km_driven      8128 non-null   int64
 4   fuel           8128 non-null   object
 5   seller_type    8128 non-null   object
 6   transmission   8128 non-null   object
 7   owner          8128 non-null   object
 8   mileage        7907 non-null   object
 9   engine         7907 non-null   object
 10  max_power      7913 non-null   object
 11  torque         7906 non-null   object
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	year	selling_price	km_driven	seats
count	8128.000000	8.128000e+03	8.128000e+03	7907.000000
mean	2013.804011	6.382718e+05	6.981951e+04	5.416719
std	4.044249	8.062534e+05	5.655055e+04	0.959588
min	1983.000000	2.999900e+04	1.000000e+00	2.000000
25%	2011.000000	2.549990e+05	3.500000e+04	5.000000
50%	2015.000000	4.500000e+05	6.000000e+04	5.000000
75%	2017.000000	6.750000e+05	9.800000e+04	5.000000
max	2020.000000	1.000000e+07	2.360457e+06	14.000000

To make the attributes of data easier to understand we make changes to it known as Label encoding which is a technique used to represent categorical variables as numerical variables so that machine learning models can use them as inputs.



Feature Engineering

In [16]:
```
from sklearn.preprocessing import LabelEncoder
labelEncoder = LabelEncoder()
df['fuel'] = labelEncoder.fit_transform(df['fuel'])
df['transmission'] = labelEncoder.fit_transform(df['transmission'])
df['owner'] = labelEncoder.fit_transform(df['owner'])
df['seller_type'] = labelEncoder.fit_transform(df['seller_type'])
```

In [17]:
```
df.dropna(inplace = True)
df.reset_index(inplace = True, drop = True)
df.drop(['name', 'torque'], inplace = True, axis = 1)
```

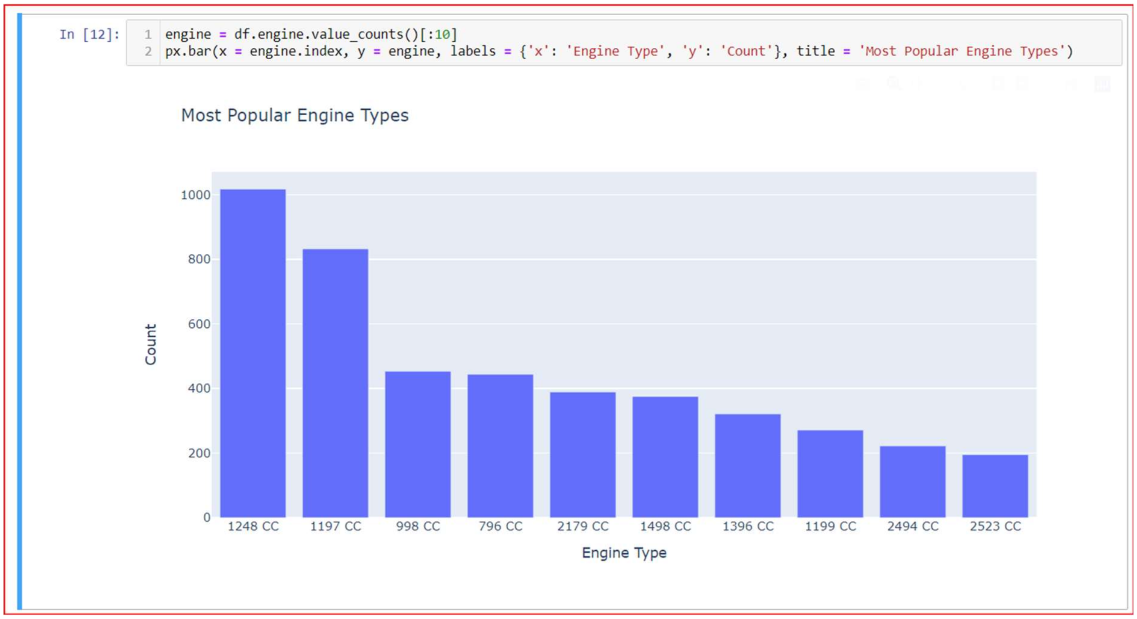
In [18]:
```
lst, lst1, lst2 = [], [], []
for i in range(0, 7906):
    lst.append(re.sub('[^0-9.]', '', str(df['mileage'][i])))
    lst1.append(re.sub('[^0-9.]', '', str(df['engine'][i])))
    lst2.append(re.sub('[^0-9.]', '', str(df['max_power'][i])))
new_lst = list(map(float, lst))
new_lst1 = list(map(float, lst1))
new_lst2 = list(map(float, lst2))
df['mileage'] = new_lst
df['engine'] = new_lst1
df['max_power'] = new_lst2
df.head()
```

Out[18]:

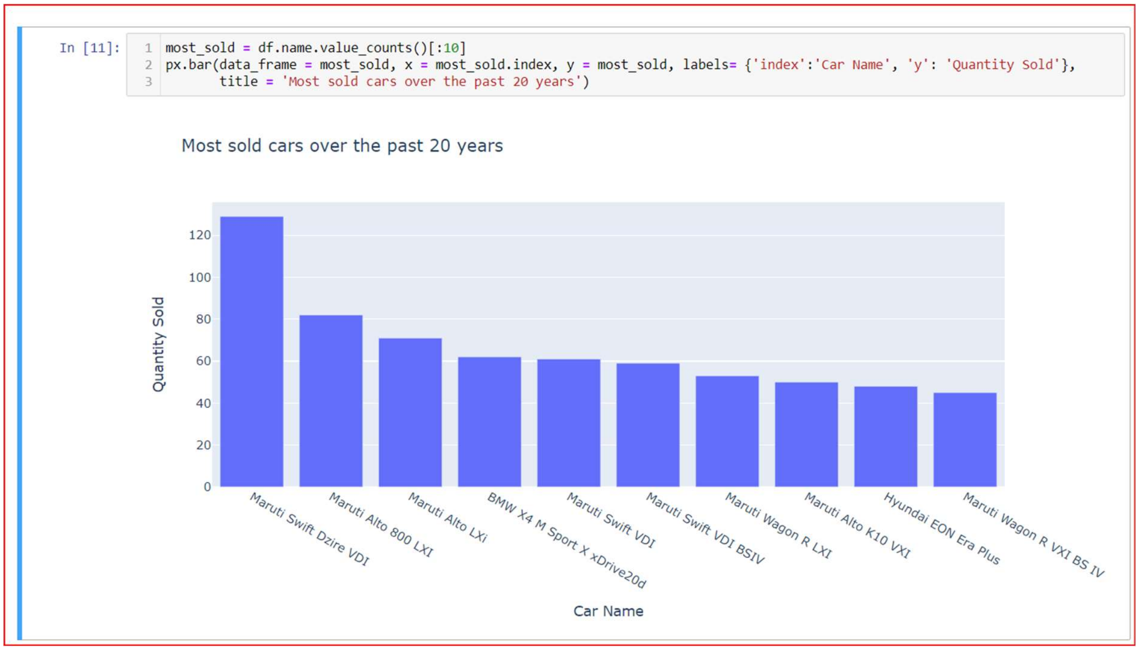
	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats
0	2014	450000	145500	1	1	1	0	23.40	1248.0	74.00	5.0
1	2014	370000	120000	1	1	1	2	21.14	1498.0	103.52	5.0
2	2006	158000	140000	3	1	1	4	17.70	1497.0	78.00	5.0
3	2010	225000	127000	1	1	1	0	23.00	1396.0	90.00	5.0
4	2007	130000	120000	3	1	1	0	16.10	1298.0	88.20	5.0

2. Visualization

Data visualization is used to make complex data easier to understand, identify relationships and correlations, and communicate insights and findings to others. It also makes data more engaging, which can encourage people to explore it further. Finally, data visualization supports decision-making by providing a clear, visual representation of the data that can help identify trends and patterns that might be missed in other forms of analysis.



```
engine = df.engine.value_counts()[:10]
px.bar(x = engine.index, y = engine, labels = {'x': 'Engine Type', 'y': 'Count'}, title = 'Most Popular Engine Types')
```

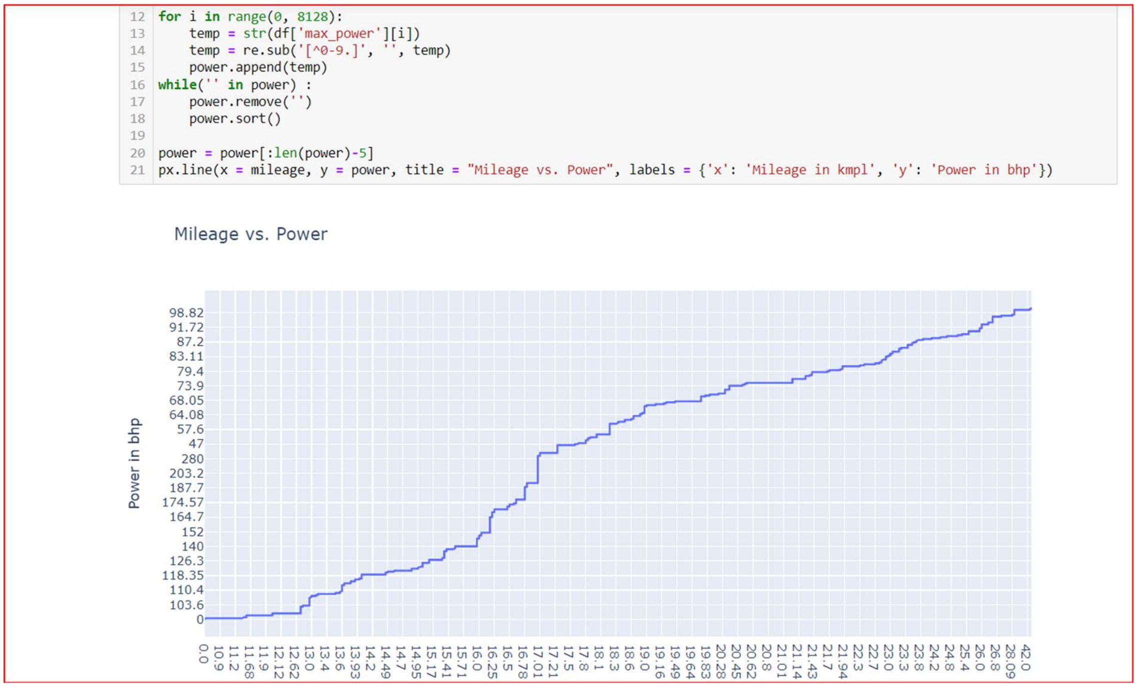


In [11]:

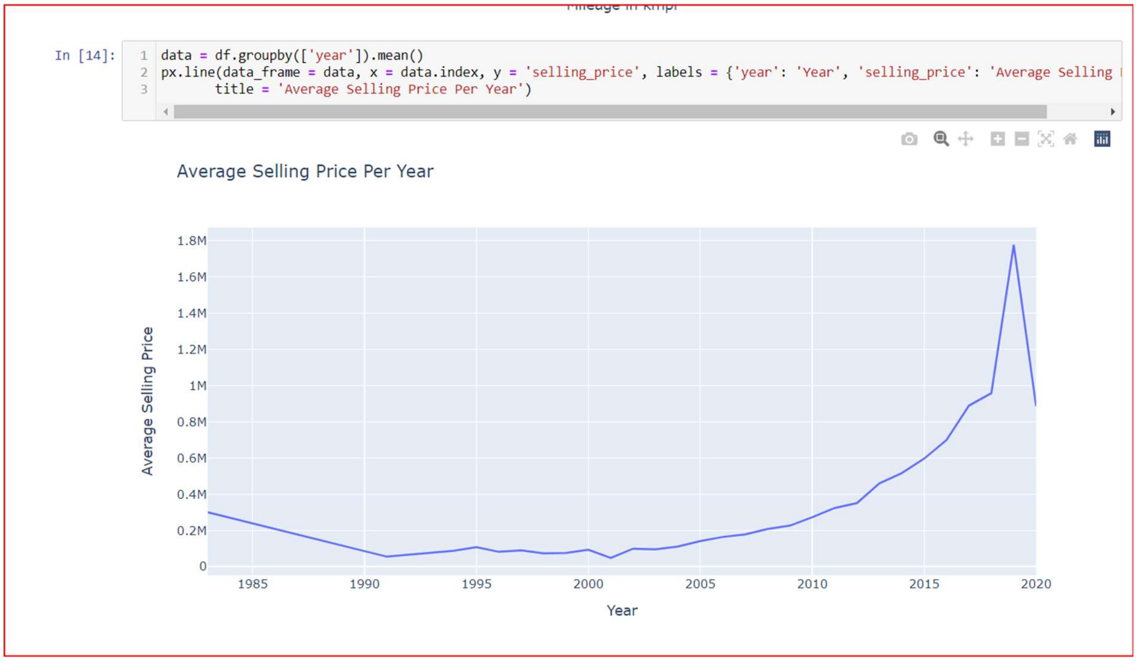
```
most_sold = df.name.value_counts()[:10]
px.bar(data_frame = most_sold, x = most_sold.index, y = most_sold, labels= {'index':'Car Name', 'y': 'Quantity Sold'},
       title = 'Most sold cars over the past 20 years')
```

3. Geometric Analysis

Geometric analysis is used to study geometric objects and their properties such as shape, size, and position. It is used to provide a rigorous mathematical foundation for various areas such as physics, engineering, and computer science. Geometric analysis enables the development of powerful tools to solve complex problems in these fields.

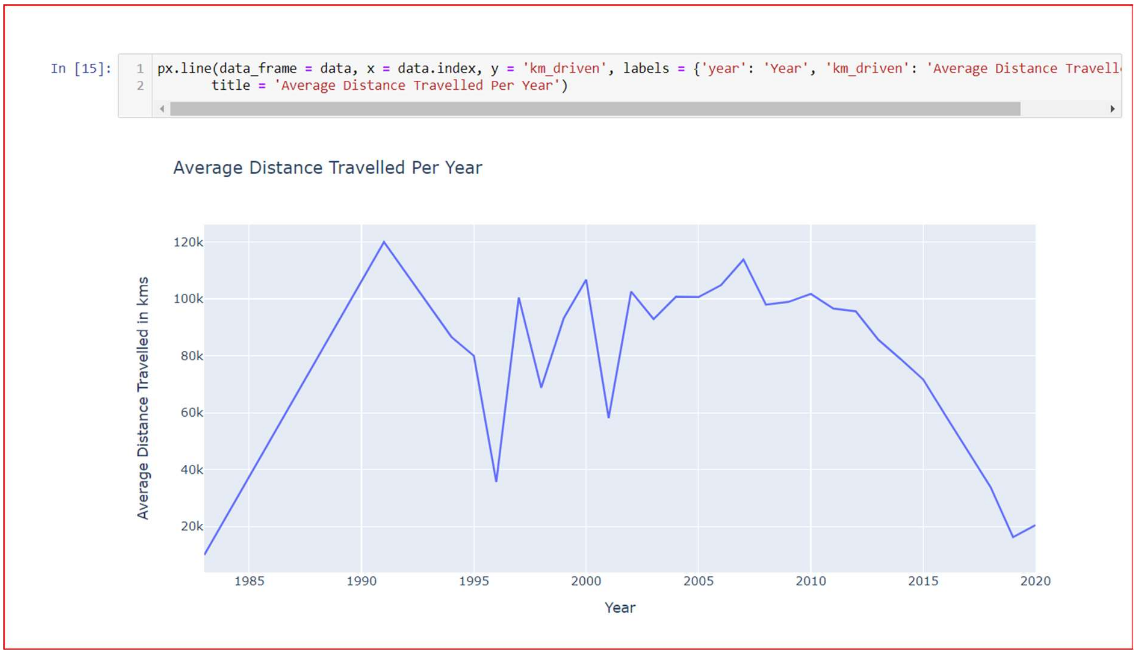


```
for i in range(0, 8128):
    temp = str(df['max_power'][i])
    temp = re.sub('[^0-9.]', '', temp)
    power.append(temp)
while('' in power) :
    power.remove('')
    power.sort()

power = power[:len(power)-5]
px.line(x = mileage, y = power, title = "Mileage vs. Power", labels = {'x': 'Mileage in kmpl', 'y': 'Power in bhp'})
```



In [14]:

```
data = df.groupby(['year']).mean()
px.line(data_frame = data, x = data.index, y = 'selling_price', labels = {'year': 'Year', 'selling_price': 'Average Selling 
        title = 'Average Selling Price Per Year')
```



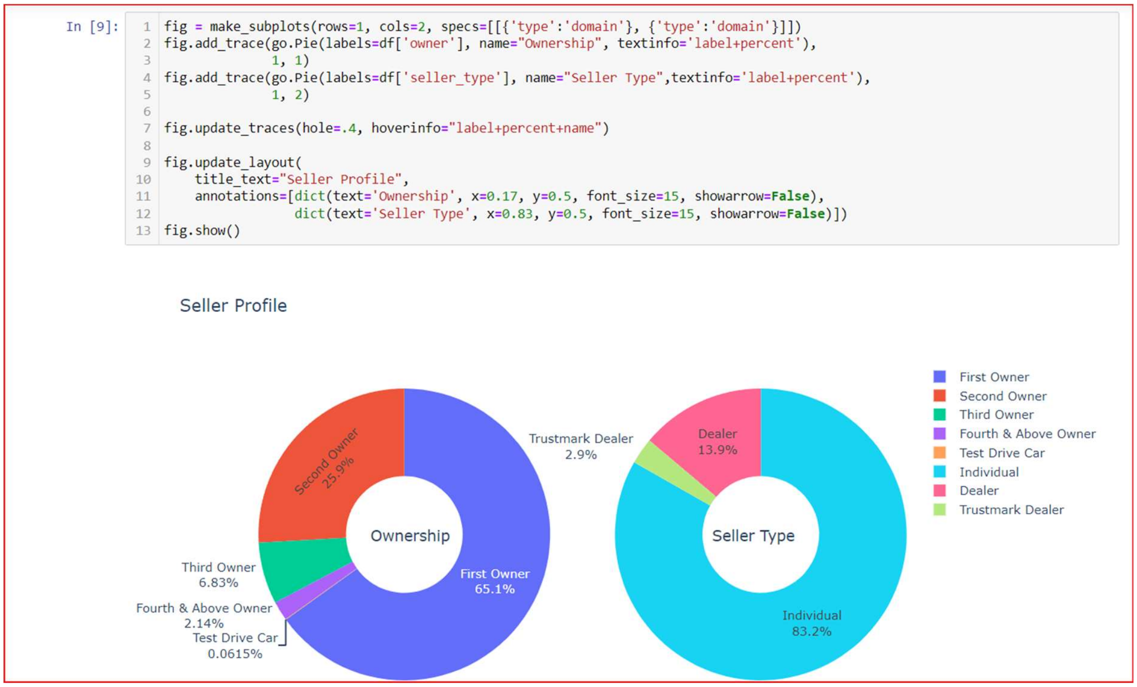
In [15]:

```
px.line(data_frame = data, x = data.index, y = 'km_driven', labels = {'year': 'Year', 'km_driven': 'Average Distance Travell
        title = 'Average Distance Travelled Per Year')
```

Average Distance Travelled Per Year
Average Distance Travelled in kms
120k
100k
80k
60k
40k
20k
1985
1990
1995
2000
2005
2010
2015
2020
Year

4. Psychographic Analysis

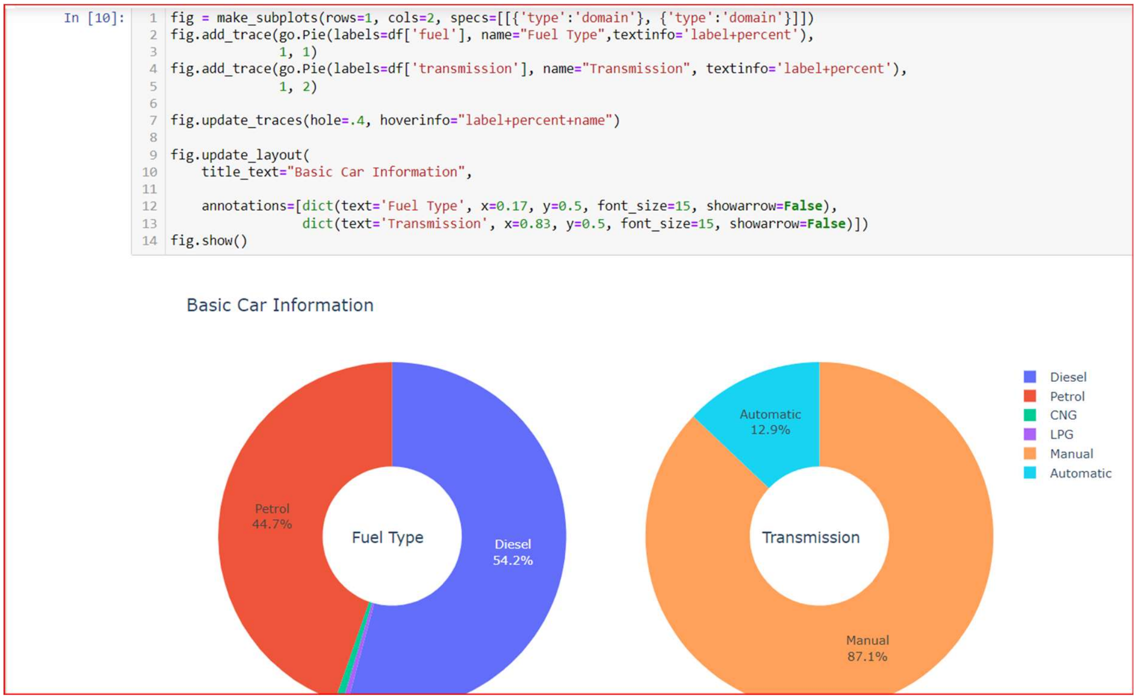
Psychographics helps in understanding consumer behaviour by analyzing their personality, values, interests, and lifestyle. It provides insights into the motivations and attitudes of the target audience, which can help marketers create more effective marketing strategies. By understanding the psychographics of their target audience, businesses can tailor their products and services to better meet customer needs and preferences.



In [9]:

```
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=df['owner'], name="Ownership", textinfo='label+percent'),
              1, 1)
fig.add_trace(go.Pie(labels=df['seller_type'], name="Seller Type",textinfo='label+percent'),
              1, 2)

fig.update_traces(hole=.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="Seller Profile",
    annotations=[dict(text='Ownership', x=0.17, y=0.5, font_size=15, showarrow=False),
                 dict(text='Seller Type', x=0.83, y=0.5, font_size=15, showarrow=False)])
fig.show()
```

Seller Profile



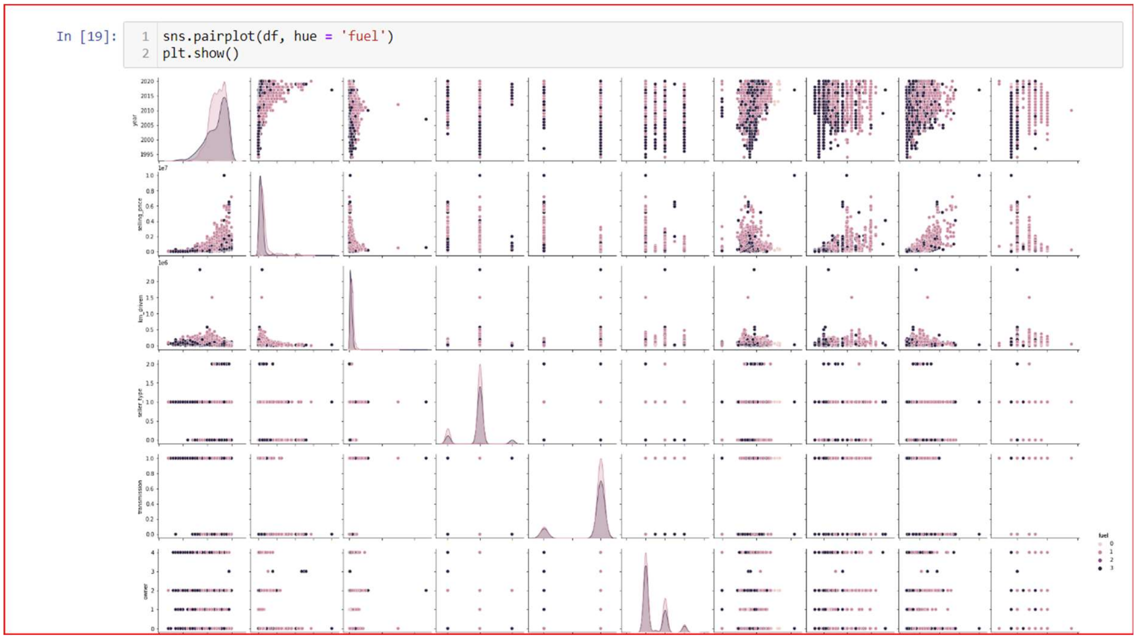
In [10]:

```
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=df['fuel'], name="Fuel Type",textinfo='label+percent'),
              1, 1)
fig.add_trace(go.Pie(labels=df['transmission'], name="Transmission", textinfo='label+percent'),
              1, 2)

fig.update_traces(hole=.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="Basic Car Information",

    annotations=[dict(text='Fuel Type', x=0.17, y=0.5, font_size=15, showarrow=False),
                 dict(text='Transmission', x=0.83, y=0.5, font_size=15, showarrow=False)])
fig.show()
```

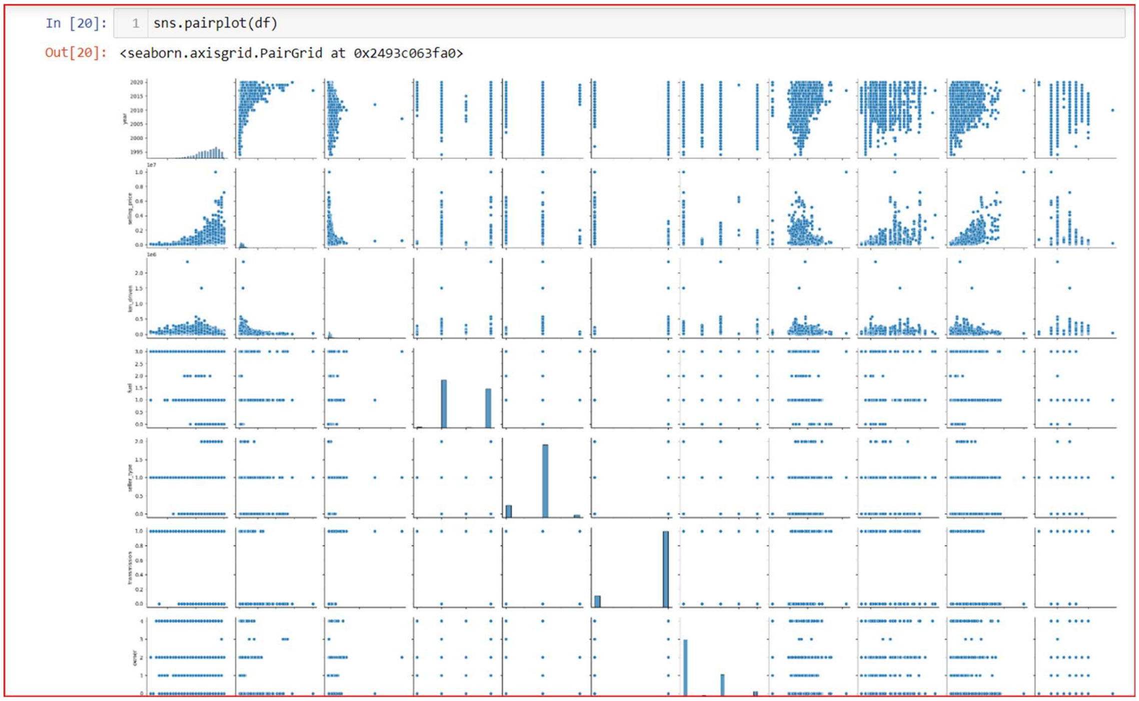
Basic Car Information

5. Demographic Analysis:

Demographic analysis helps in understanding the characteristics of a population, such as age, gender, income, and education. It provides insights into the preferences and behaviors of a particular group, which can help in developing effective marketing strategies. By understanding the demographic makeup of their target audience, businesses can tailor their products and services to better meet customer needs and preferences.



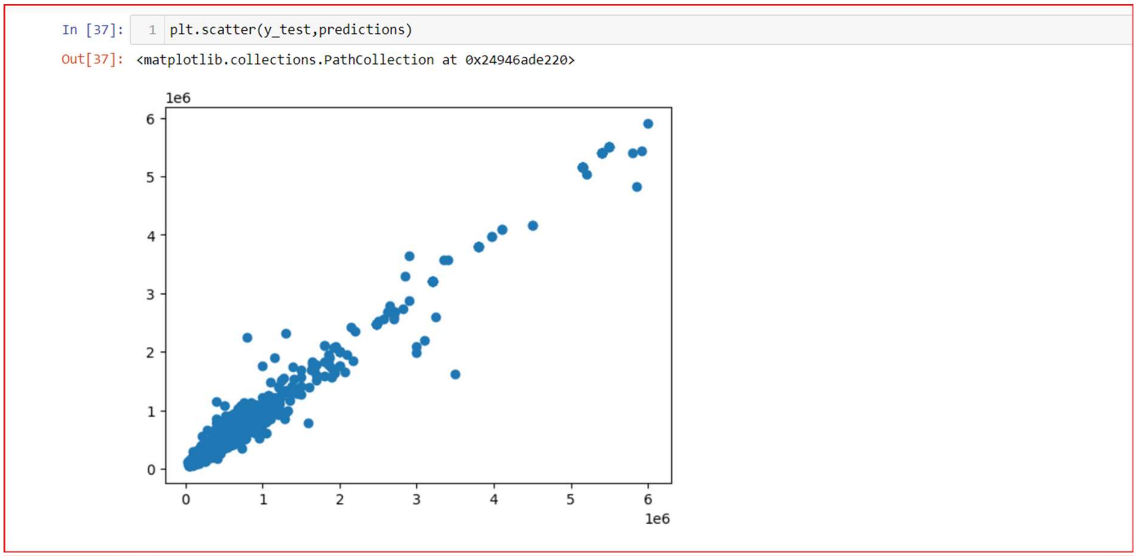
```
In [19]:  sns.pairplot(df, hue = 'fuel')
          plt.show()
```



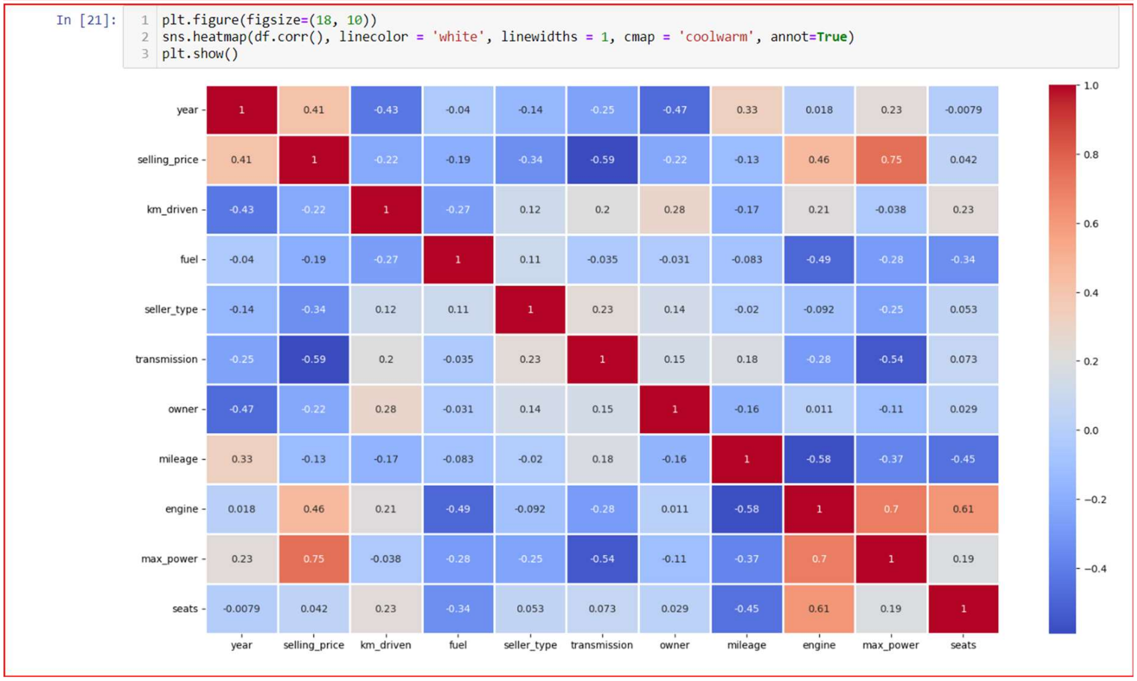
```
In [20]:  sns.pairplot(df)
Out[20]:  <seaborn.axisgrid.PairGrid at 0x2493c063fa0>
```

6. Behaviour Analysis:

Behaviour analysis helps in understanding the actions and choices made by individuals, providing insights into their preferences and motivations. It helps businesses identify the factors that influence consumer behaviour and develop effective marketing strategies. By understanding consumer behaviour, businesses can improve their products and services, enhance customer satisfaction, and increase profitability.

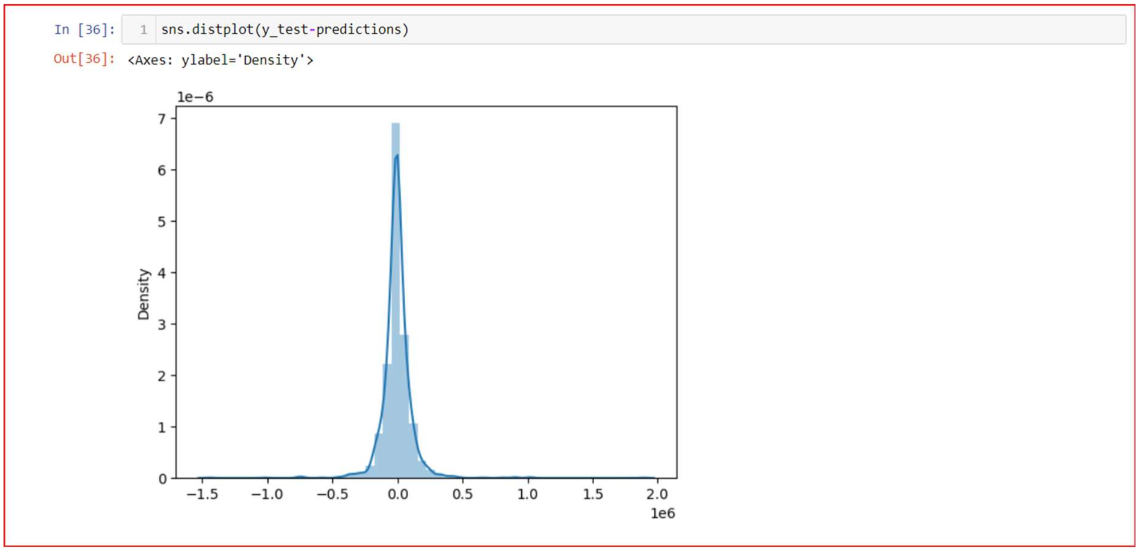


```
In [37]: plt.scatter(y_test,predictions)
Out[37]: <matplotlib.collections.PathCollection at 0x24946ade220>
```



```
In [21]: plt.figure(figsize=(18, 10))
         sns.heatmap(df.corr(), linecolor = 'white', linewidths = 1, cmap = 'coolwarm', annot=True)
         plt.show()
```

	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats
year	1	0.41	-0.43	-0.04	-0.14	-0.25	-0.47	0.33	0.018	0.23	-0.0079
selling_price	0.41	1	-0.22	-0.19	-0.34	-0.59	-0.22	-0.13	0.46	0.75	0.042
km_driven	-0.43	-0.22	1	-0.27	0.12	0.2	0.28	-0.17	0.21	-0.038	0.23
fuel	-0.04	-0.19	-0.27	1	0.11	-0.035	-0.031	-0.083	-0.49	-0.28	-0.34
seller_type	-0.14	-0.34	0.12	0.11	1	0.23	0.14	-0.02	-0.092	-0.25	0.053
transmission	-0.25	-0.59	0.2	-0.035	0.23	1	0.15	0.18	-0.28	-0.54	0.073
owner	-0.47	-0.22	0.28	-0.031	0.14	0.15	1	-0.16	0.011	-0.11	0.029
mileage	0.33	-0.13	-0.17	-0.083	-0.02	0.18	-0.16	1	-0.58	-0.37	-0.45
engine	0.018	0.46	0.21	-0.49	-0.092	-0.28	0.011	-0.58	1	0.7	0.61
max_power	0.23	0.75	-0.038	-0.28	-0.25	-0.54	-0.11	-0.37	0.7	1	0.19
seats	-0.0079	0.042	0.23	-0.34	0.053	0.073	0.029	-0.45	0.61	0.19	1



In [36]:

```
sns.distplot(y_test-predictions)
```

Out[36]:

```
<Axes: ylabel='Density'>
```