Car_prediction_price

March 15, 2025

1 CarDekho Price Prediction

Steps for building a machine learning model: 1. Gaining the understanding of the project and what it is about 2. Import libraries (atleast initial ones) 3. Import the data/ Get the data 4. Data cleaning and understanding 5. EDA: Exploratory data analysis Univariate analysis - to look at the distribution in order to understand if there is an outlier present in the data Bi-variate analysis - When we look at the relationship between two variables (Typically between the target variable (Selling price in this case and all the other variables) Multi-variate analysis - to check correlation between all the combination of features

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

[4]:
```
data = pd.read_csv("E:\BHU\Data_Analyst\Ws_Cube_Tech\week_19 project\Cardekho.
↪csv")
data.head()
```

[4]:

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_type	transmission_type	mileage	engine	max_power	seats	selling_price
0	0	Maruti Alto	Maruti	Alto	9	120000	Individual	Petrol	Manual	19.70	796	46.30	5	120000
1	1	Hyundai Grand	Hyundai	Grand	5	20000	Individual	Petrol	Manual	18.90	1197	82.00	5	
2	2	Hyundai i20	Hyundai	i20	11	60000	Individual	Petrol	Manual	17.00	1197	80.00	5	
3	3	Maruti Alto	Maruti	Alto	9	37000	Individual	Petrol	Manual	20.92	998	67.10	5	
4	4	Ford Ecosport	Ford	Ecosport	6	30000	Dealer	Diesel	Manual	22.77	1498	98.59	5	

```
1          550000
2          215000
3          226000
4          570000
```

[5]:
```
# Basic data checks
data.info()
```

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

[6]:
```
data.shape
```

[6]:
```
(15411, 14)
```

[7]:
```
# Summary statistics
data.describe()
```

[7]:
```
         Unnamed: 0   vehicle_age     km_driven       mileage        engine  \
count  15411.000000  15411.000000  1.541100e+04  15411.000000  15411.000000
mean    9811.857699      6.036338  5.561648e+04     19.701151   1486.057751
std     5643.418542      3.013291  5.161855e+04      4.171265    521.106696
min        0.000000      0.000000  1.000000e+02      4.000000    793.000000
25%     4906.500000      4.000000  3.000000e+04     17.000000   1197.000000
50%     9872.000000      6.000000  5.000000e+04     19.670000   1248.000000
75%    14668.500000      8.000000  7.000000e+04     22.700000   1582.000000
max    19543.000000     29.000000  3.800000e+06     33.540000   6592.000000

          max_power         seats  selling_price
```

```
count  15411.000000  15411.000000   1.541100e+04
mean     100.588254      5.325482   7.749711e+05
std       42.972979      0.807628   8.941284e+05
min       38.400000      0.000000   4.000000e+04
25%       74.000000      5.000000   3.850000e+05
50%       88.500000      5.000000   5.560000e+05
75%      117.300000      5.000000   8.250000e+05
max      626.000000      9.000000   3.950000e+07
```

[8]:
```
data['fuel_type'].value_counts(normalize = True)*100
```

[8]:
```
Petrol      49.594446
Diesel      48.140938
CNG          1.953150
LPG          0.285510
Electric     0.025955
Name: fuel_type, dtype: float64
```

[9]:
```
data['fuel_type'].value_counts()
```

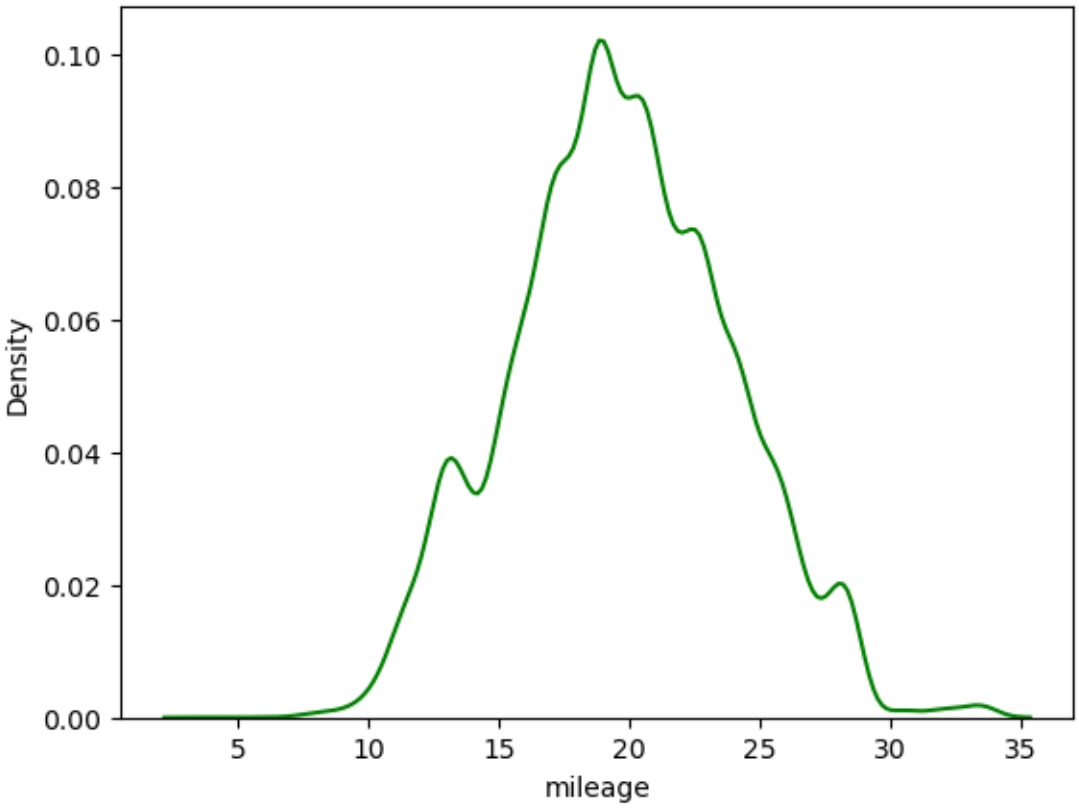
[9]:
```
Petrol      7643
Diesel      7419
CNG          301
LPG           44
Electric       4
Name: fuel_type, dtype: int64
```

[10]:
```
data['mileage'].mean()
```

[10]:
```
19.70115112581922
```

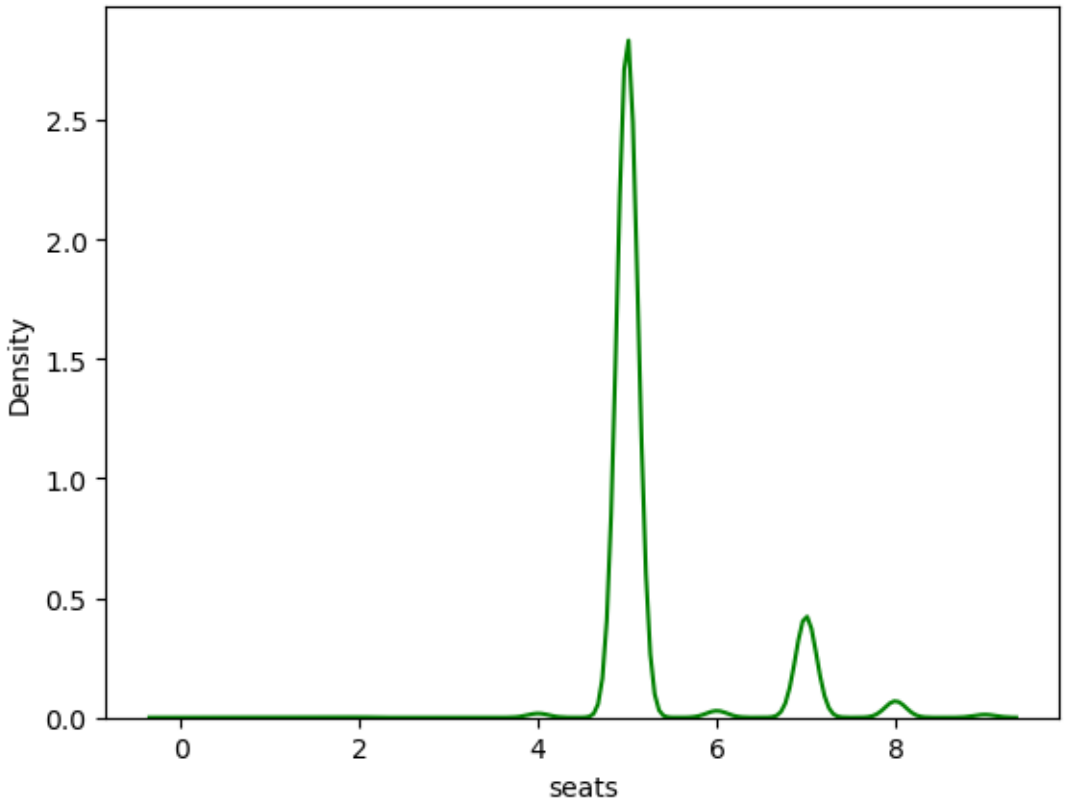
[11]:
```
sns.kdeplot(x = data['mileage'],color = 'g')
```

[11]:
```
<Axes: xlabel='mileage', ylabel='Density'>
```



[12]:
```
sns.kdeplot(x = data['seats'],color = 'g')
```

[12]: <Axes: xlabel='seats', ylabel='Density'>

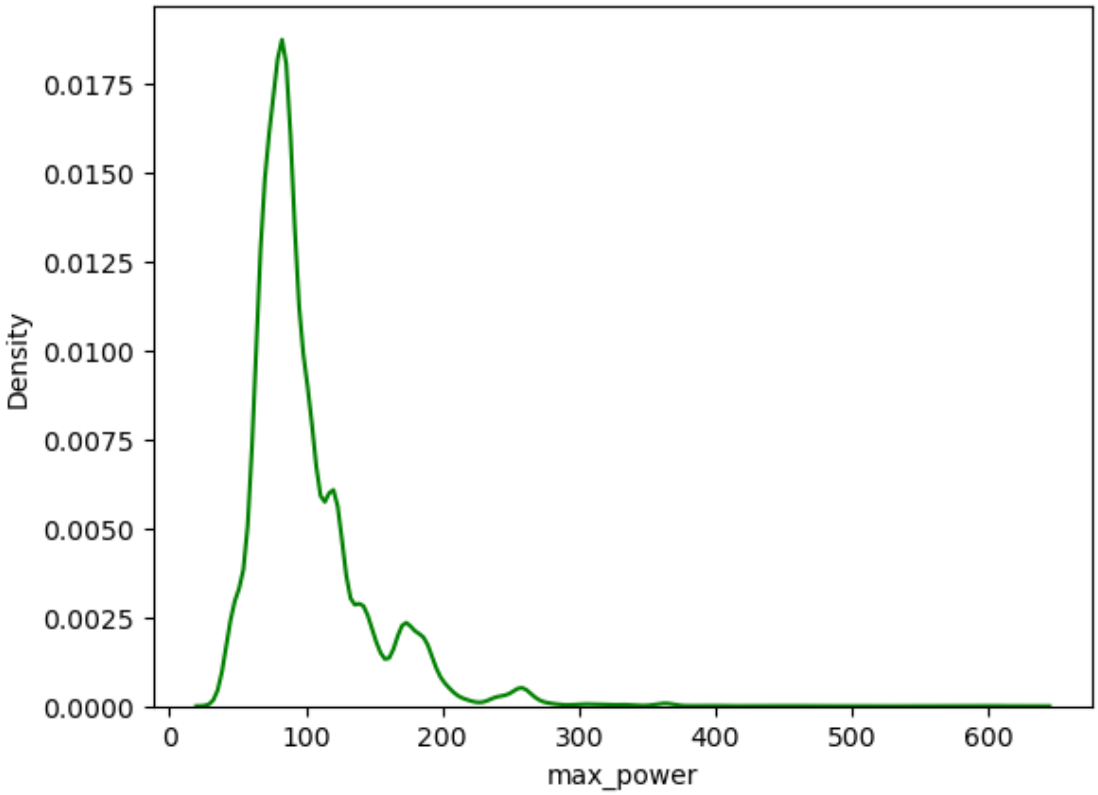


[13]:
```
data['seats'].value_counts()
```

[13]:
```
5    12910
7     1922
8      311
6      127
4       77
9       55
2        7
0        2
Name: seats, dtype: int64
```

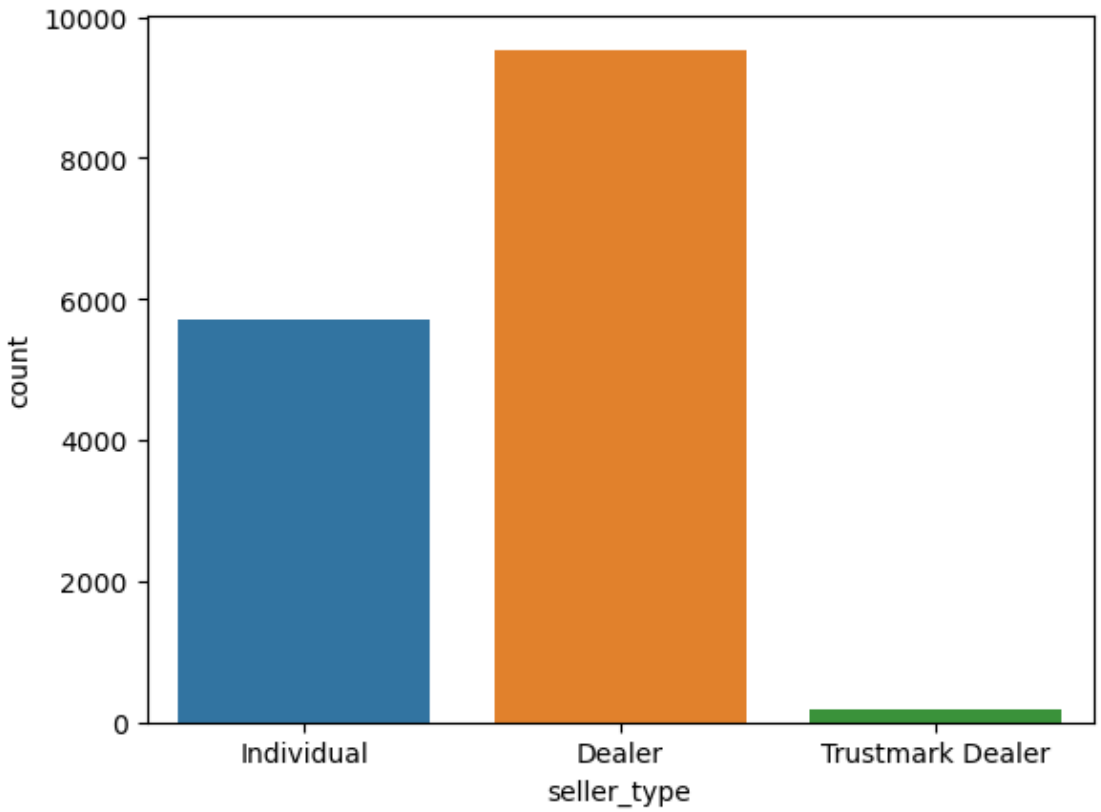
[14]:
```
sns.kdeplot(x = data['max_power'],color = 'g')
```

[14]:
```
<Axes: xlabel='max_power', ylabel='Density'>
```



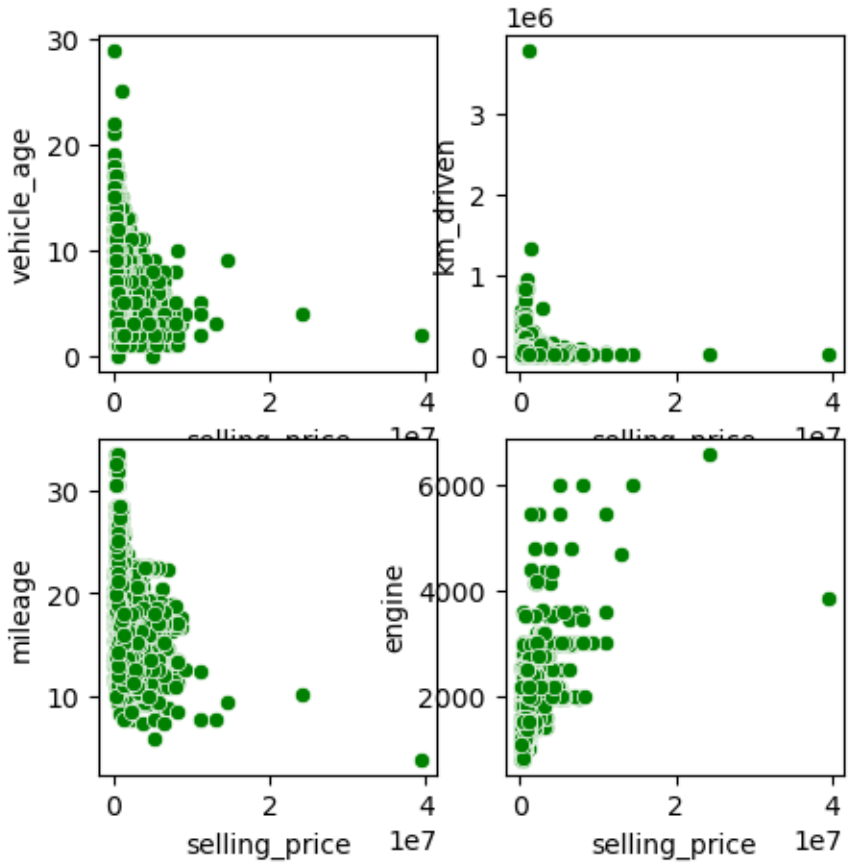
[15]:
```
sns.countplot(x = data['seller_type'])
```

[15]:
```
<Axes: xlabel='seller_type', ylabel='count'>
```



Please do this for all the categorical columns and then tell me the insights that you are getting by the graphs (Atleast 2 insights) Bi-variate analysis - When we look at the relationship between each column and the selling price

[17]:
```
#Lets look at the relationship of each variable with the selling price (Target variable)

fig = plt.figure(figsize = (5,5))

features = ['vehicle_age','km_driven','mileage','engine']

for i in range(len(features)):
    plt.subplot(2,2,i+1)
    sns.scatterplot(data = data, x = 'selling_price',y = features[i],color = 'g')
```

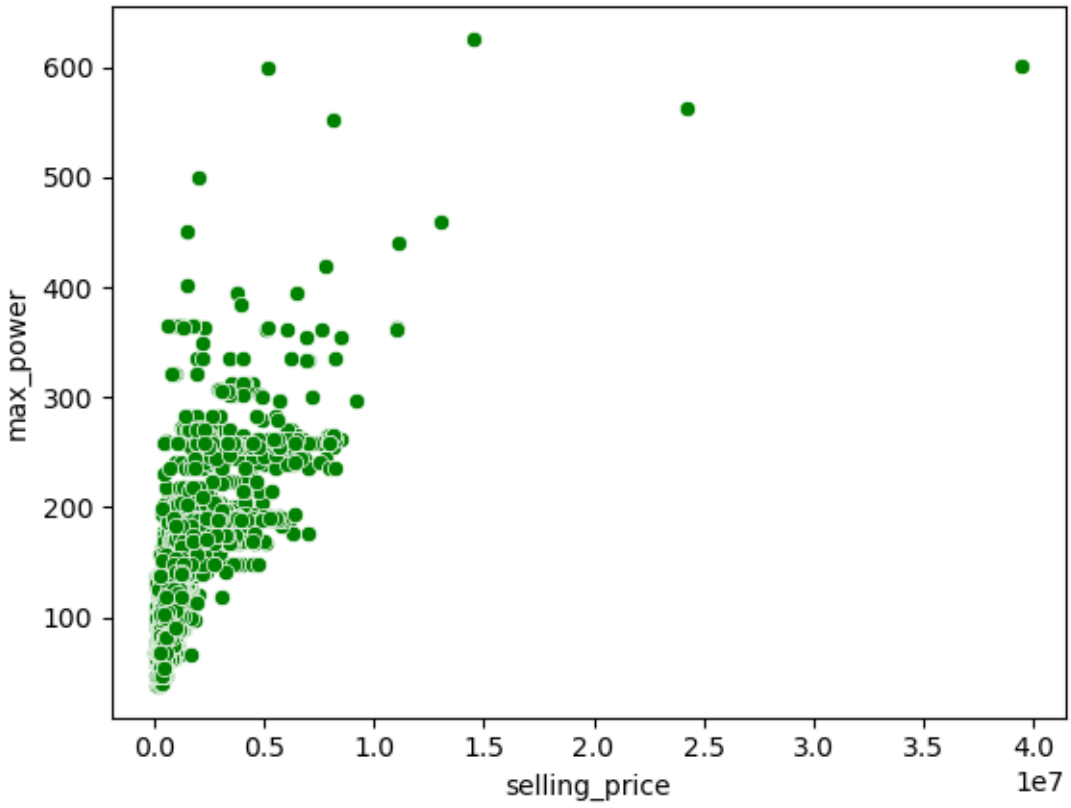


sns.scatterplot(data = data, x = 'selling_price',y = 'mileage',color = 'g') Insights: Vehicle age, Km_driven, mileage are impacting the selling_price negatively

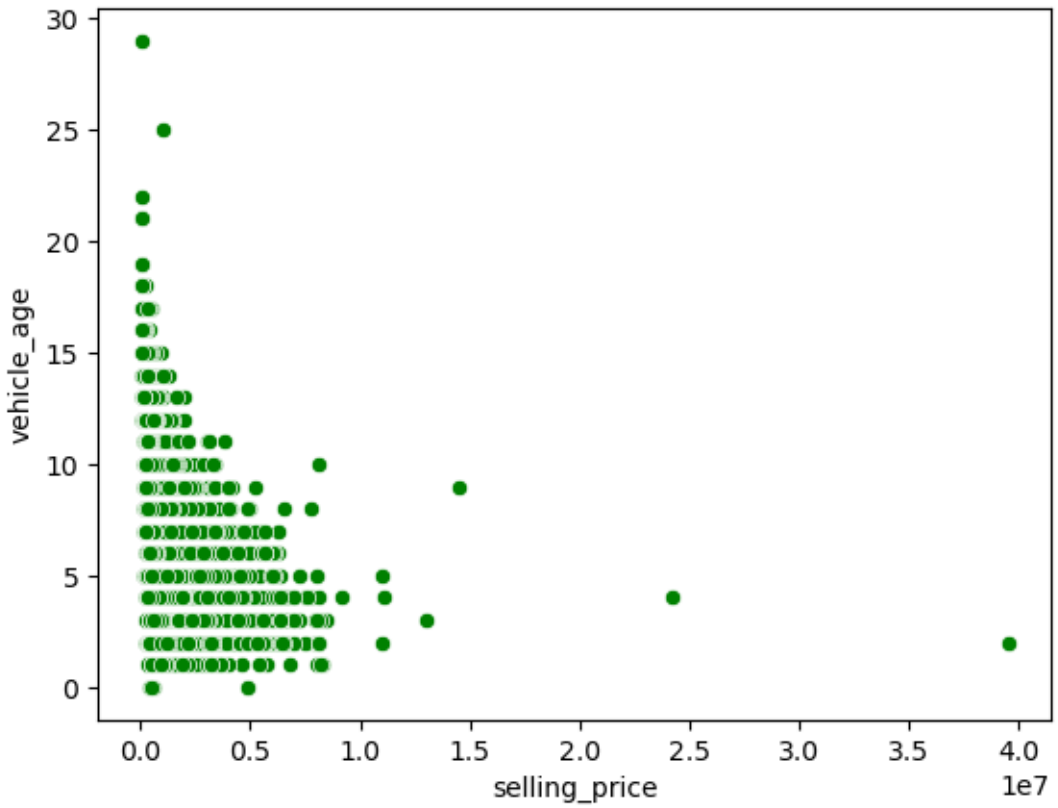
Engine, max_power will impact the selling_price positively

[18]:
```
sns.scatterplot(data = data, x = 'selling_price',y = 'max_power',color = 'g')
```

[18]:
```
<Axes: xlabel='selling_price', ylabel='max_power'>
```



[31]:
```
sns.scatterplot(data=data, x='selling_price', y='vehicle_age', color='g')
plt.show()
print(data.head())  # Check if the dataframe is loaded properly
print(data.columns)
```

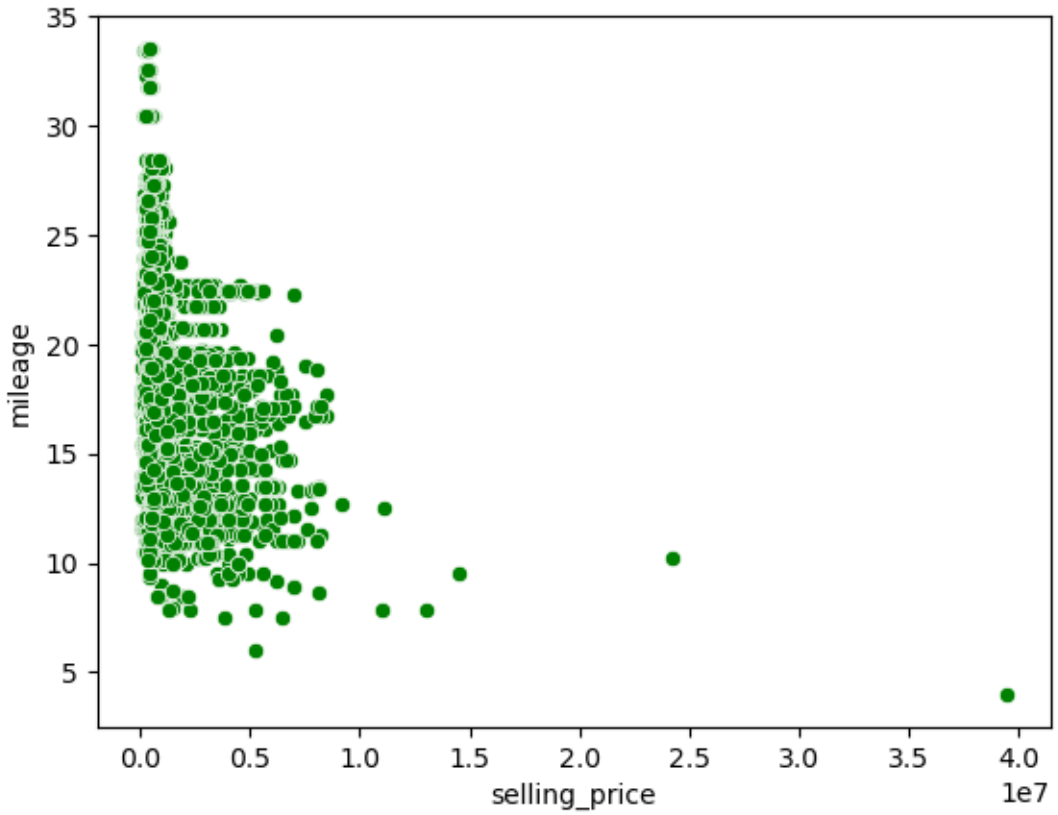



```
   Unnamed: 0       car_name    brand     model  vehicle_age  km_driven  \
0           0    Maruti Alto   Maruti      Alto            9     120000
1           1  Hyundai Grand  Hyundai     Grand            5      20000
2           2    Hyundai i20  Hyundai       i20           11      60000
3           3    Maruti Alto   Maruti      Alto            9      37000
4           4  Ford Ecosport     Ford  Ecosport            6      30000

  seller_type fuel_type transmission_type  mileage  engine  max_power  seats  \
0  Individual    Petrol            Manual    19.70     796      46.30      5
1  Individual    Petrol            Manual    18.90    1197      82.00      5
2  Individual    Petrol            Manual    17.00    1197      80.00      5
3  Individual    Petrol            Manual    20.92     998      67.10      5
4      Dealer    Diesel            Manual    22.77    1498      98.59      5

   selling_price
0         120000
1         550000
2         215000
3         226000
4         570000
Index(['Unnamed: 0', 'car_name', 'brand', 'model', 'vehicle_age', 'km_driven',
```

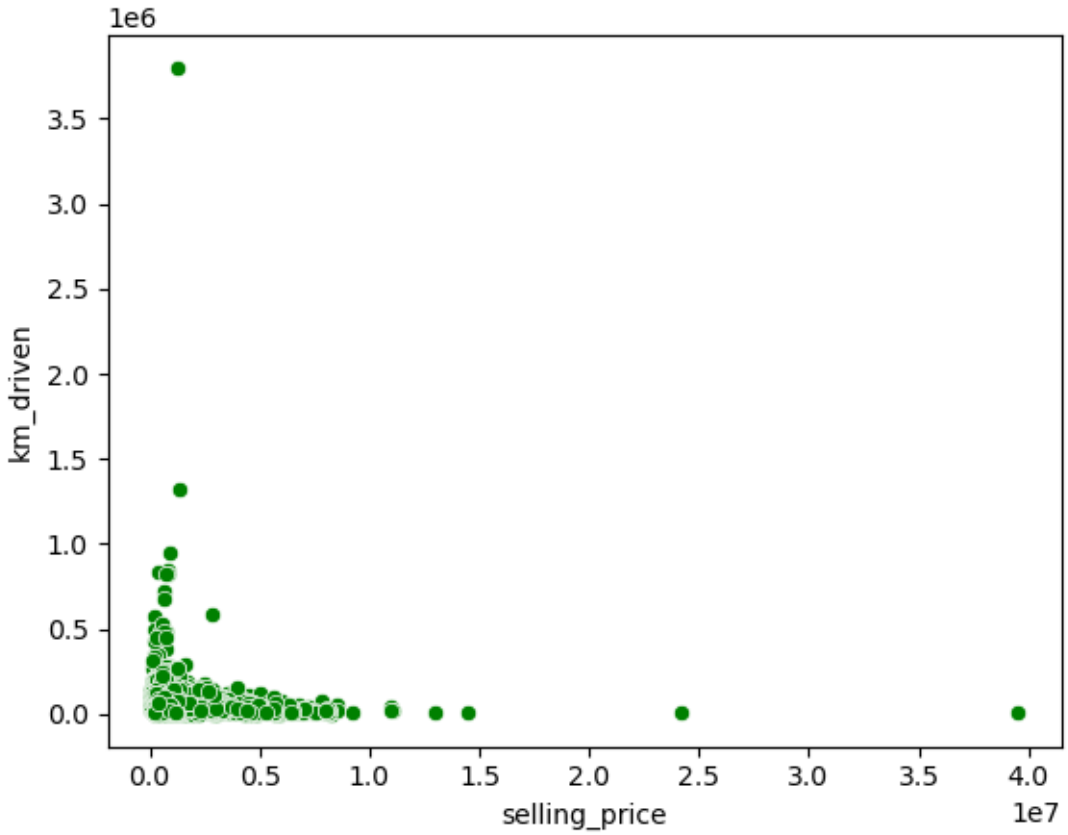
```
       'seller_type', 'fuel_type', 'transmission_type', 'mileage', 'engine',
       'max_power', 'seats', 'selling_price'],
      dtype='object')
```

[33]:
```
# Ensure 'data' is a DataFrame and check columns
print(type(data))  # Should be <class 'pandas.core.frame.DataFrame'>
print(data.columns)  # Verify column names

# Check for missing values and handle them
data = data.dropna(subset=['selling_price', 'mileage'])

# Ensure correct data types
data['selling_price'] = pd.to_numeric(data['selling_price'], errors='coerce')
data['mileage'] = pd.to_numeric(data['mileage'], errors='coerce')

# Create the scatter plot
sns.scatterplot(data=data, x='selling_price', y='mileage', color='g')

# Show the plot
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
Index(['Unnamed: 0', 'car_name', 'brand', 'model', 'vehicle_age', 'km_driven',
       'seller_type', 'fuel_type', 'transmission_type', 'mileage', 'engine',
       'max_power', 'seats', 'selling_price'],
      dtype='object')
```

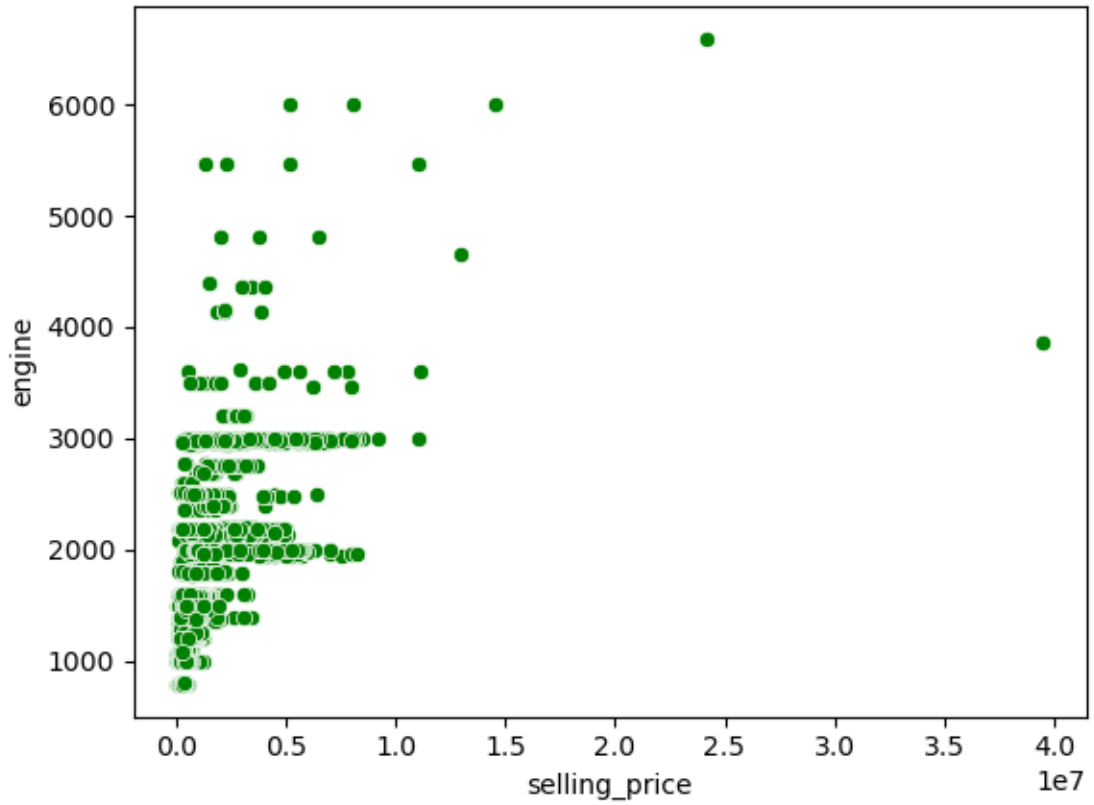


[35]:
```python
# Ensure 'data' is a DataFrame and check columns
if not isinstance(data, pd.DataFrame):
    raise TypeError("The variable 'data' must be a Pandas DataFrame.")

# Verify required columns exist
required_columns = {'selling_price', 'km_driven'}
if not required_columns.issubset(data.columns):
    raise KeyError(f"Missing required columns: {required_columns - set(data.
 ↪columns)}")

# Drop missing values and convert to numeric
data = data.dropna(subset=['selling_price', 'km_driven'])
data['selling_price'] = pd.to_numeric(data['selling_price'], errors='coerce')
data['km_driven'] = pd.to_numeric(data['km_driven'], errors='coerce')

# Plot the scatter plot
sns.scatterplot(data=data, x='selling_price', y='km_driven', color='g')

# Show the plot
plt.show()
```



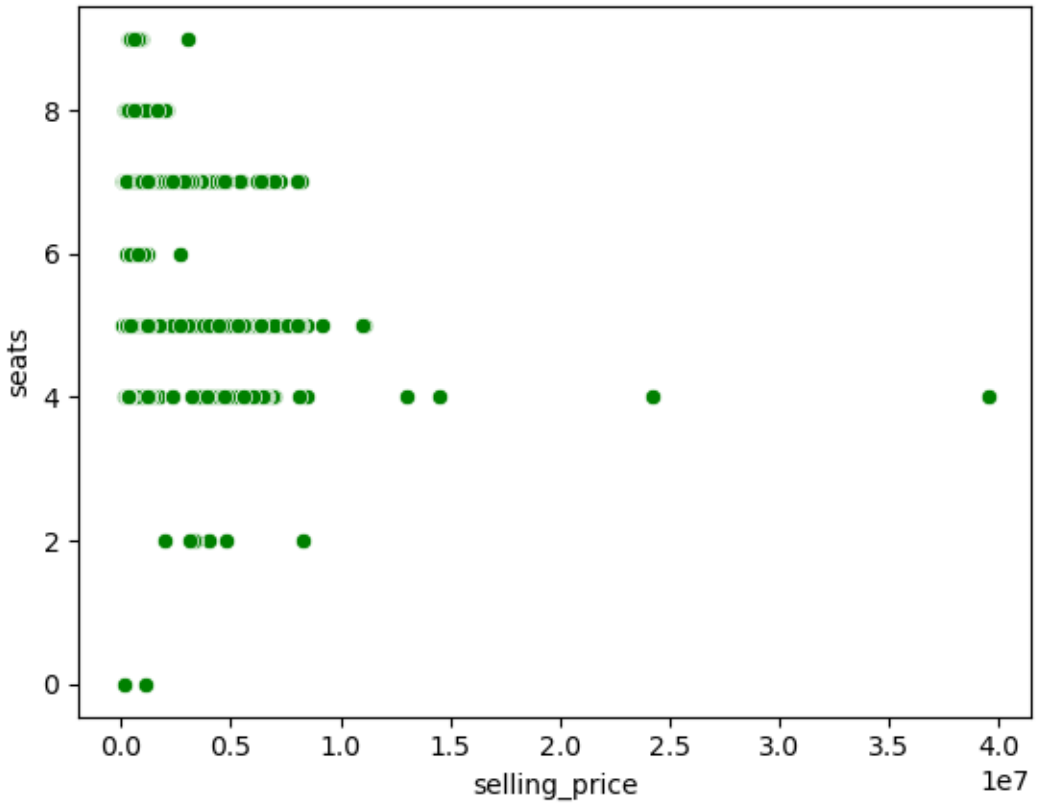
[36]:
```python
# Ensure 'data' is a DataFrame and check columns
if not isinstance(data, pd.DataFrame):
    raise TypeError("The variable 'data' must be a Pandas DataFrame.")

# Verify required columns exist
required_columns = {'selling_price', 'engine'}
if not required_columns.issubset(data.columns):
    raise KeyError(f"Missing required columns: {required_columns - set(data.
 ↪columns)}")

# Drop missing values and convert to numeric
data = data.dropna(subset=['selling_price', 'engine'])

# If 'engine' is stored as a string (e.g., '1197 CC'), extract numeric values
data['engine'] = data['engine'].astype(str).str.extract('(\d+)').astype(float)

# Convert 'selling_price' to numeric if necessary
data['selling_price'] = pd.to_numeric(data['selling_price'], errors='coerce')

# Plot the scatter plot
```

```
sns.scatterplot(data=data, x='selling_price', y='engine', color='g')

# Show the plot
plt.show()
```



[37]:
```
# Ensure 'data' is a DataFrame and check columns
if not isinstance(data, pd.DataFrame):
    raise TypeError("The variable 'data' must be a Pandas DataFrame.")

# Verify required columns exist
required_columns = {'selling_price', 'seats'}
if not required_columns.issubset(data.columns):
    raise KeyError(f"Missing required columns: {required_columns - set(data.
 ↪columns)}")

# Drop missing values and convert to numeric
data = data.dropna(subset=['selling_price', 'seats'])
data['selling_price'] = pd.to_numeric(data['selling_price'], errors='coerce')
data['seats'] = pd.to_numeric(data['seats'], errors='coerce')

# Plot the scatter plot
```

```
sns.scatterplot(data=data, x='selling_price', y='seats', color='g')

# Show the plot
plt.show()
```



[19]:
```
#Multi-variate analysis - to check correlation between all the combination of␣
 ↪numerical features

features =␣
 ↪['vehicle_age','km_driven','mileage','engine','max_power','seats','selling_price']

data[features].corr()
```

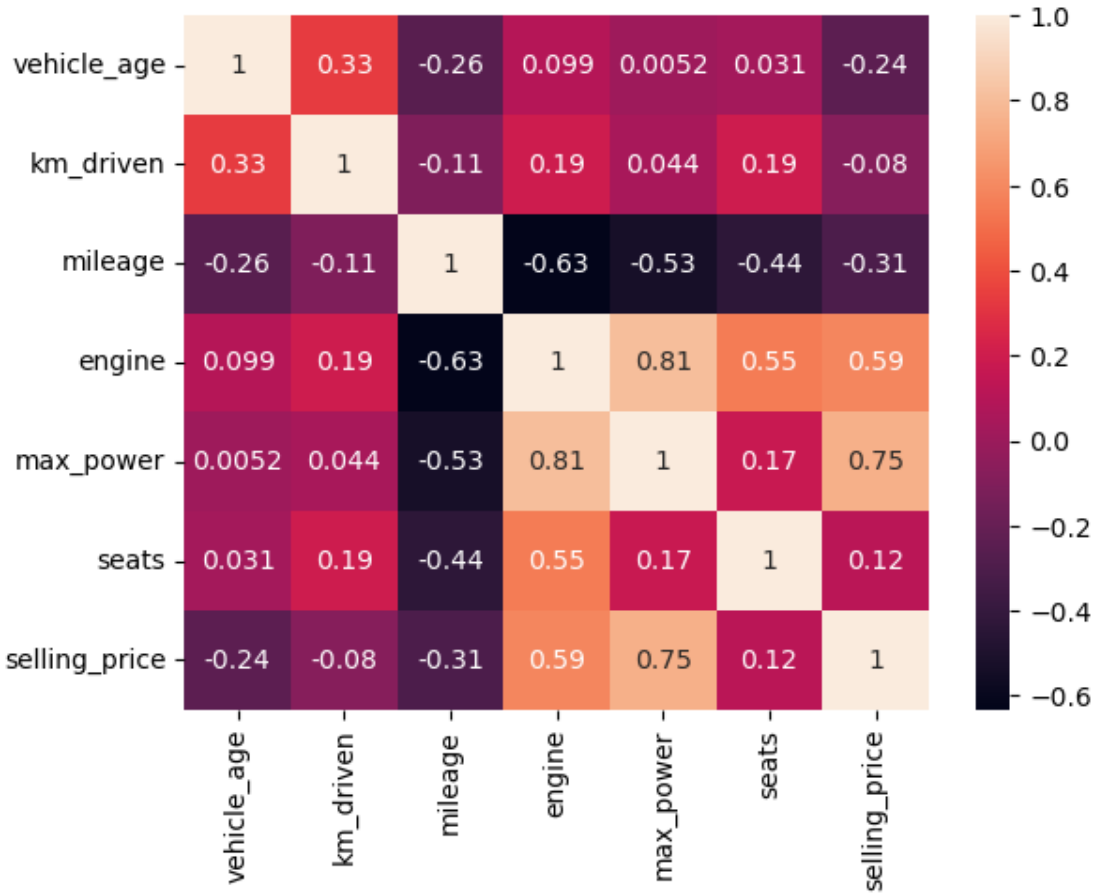
[19]:

	vehicle_age	km_driven	mileage	engine	max_power	\
vehicle_age	1.000000	0.333891	-0.257394	0.098965	0.005208	
km_driven	0.333891	1.000000	-0.105239	0.192885	0.044421	
mileage	-0.257394	-0.105239	1.000000	-0.632987	-0.533128	
engine	0.098965	0.192885	-0.632987	1.000000	0.807368	
max_power	0.005208	0.044421	-0.533128	0.807368	1.000000	
seats	0.030791	0.192830	-0.440280	0.551236	0.172257	
selling_price	-0.241851	-0.080030	-0.305549	0.585844	0.750236	

```
                   seats  selling_price
vehicle_age     0.030791      -0.241851
km_driven       0.192830      -0.080030
mileage        -0.440280      -0.305549
engine          0.551236       0.585844
max_power       0.172257       0.750236
seats           1.000000       0.115033
selling_price   0.115033       1.000000
```

[20]:
```
sns.heatmap(data= data[features].corr(),annot = True)
```

[20]:
```
<Axes: >
```



[21]:
```
data.head()
```

[21]:
```
   Unnamed: 0       car_name    brand     model  vehicle_age  km_driven  \
0           0    Maruti Alto   Maruti      Alto            9     120000
```

```
1           1  Hyundai Grand  Hyundai     Grand            5      20000
2           2    Hyundai i20  Hyundai       i20           11      60000
3           3    Maruti Alto   Maruti      Alto            9      37000
4           4  Ford Ecosport     Ford  Ecosport            6      30000

  seller_type fuel_type transmission_type  mileage  engine  max_power  seats  \
0  Individual    Petrol            Manual    19.70     796      46.30      5
1  Individual    Petrol            Manual    18.90    1197      82.00      5
2  Individual    Petrol            Manual    17.00    1197      80.00      5
3  Individual    Petrol            Manual    20.92     998      67.10      5
4      Dealer    Diesel            Manual    22.77    1498      98.59      5

   selling_price
0         120000
1         550000
2         215000
3         226000
4         570000
```

[22]:
```
model_data = data.copy()
model_data.head()
```

[22]:
```
   Unnamed: 0       car_name    brand     model  vehicle_age  km_driven  \
0           0    Maruti Alto   Maruti      Alto            9     120000
1           1  Hyundai Grand  Hyundai     Grand            5      20000
2           2    Hyundai i20  Hyundai       i20           11      60000
3           3    Maruti Alto   Maruti      Alto            9      37000
4           4  Ford Ecosport     Ford  Ecosport            6      30000

  seller_type fuel_type transmission_type  mileage  engine  max_power  seats  \
0  Individual    Petrol            Manual    19.70     796      46.30      5
1  Individual    Petrol            Manual    18.90    1197      82.00      5
2  Individual    Petrol            Manual    17.00    1197      80.00      5
3  Individual    Petrol            Manual    20.92     998      67.10      5
4      Dealer    Diesel            Manual    22.77    1498      98.59      5

   selling_price
0         120000
1         550000
2         215000
3         226000
4         570000
```

[23]:
```
model_data.drop(labels = ['car_name','brand','model','seller_type'],axis = 1,␣
↪inplace = True)
model_data
```

[23]:
```
       Unnamed: 0  vehicle_age  km_driven fuel_type transmission_type  \
0               0            9     120000    Petrol            Manual
1               1            5      20000    Petrol            Manual
2               2           11      60000    Petrol            Manual
3               3            9      37000    Petrol            Manual
4               4            6      30000    Diesel            Manual
…             …          …        …       …                 …
15406       19537            9      10723    Petrol            Manual
15407       19540            2      18000    Petrol            Manual
15408       19541            6      67000    Diesel            Manual
15409       19542            5    3800000    Diesel            Manual
15410       19543            2      13000    Petrol         Automatic

       mileage  engine  max_power  seats  selling_price
0        19.70     796      46.30      5         120000
1        18.90    1197      82.00      5         550000
2        17.00    1197      80.00      5         215000
3        20.92     998      67.10      5         226000
4        22.77    1498      98.59      5         570000
…        …     …        …       …              …
15406    19.81    1086      68.05      5         250000
15407    17.50    1373      91.10      7         925000
15408    21.14    1498     103.52      5         425000
15409    16.00    2179     140.00      7        1225000
15410    18.00    1497     117.60      5        1200000

[15411 rows x 10 columns]
```

[24]:
```
model_data = pd.get_dummies(model_data,dtype = float)
model_data
```

[24]:
```
       Unnamed: 0  vehicle_age  km_driven  mileage  engine  max_power  seats  \
0               0            9     120000    19.70     796      46.30      5
1               1            5      20000    18.90    1197      82.00      5
2               2           11      60000    17.00    1197      80.00      5
3               3            9      37000    20.92     998      67.10      5
4               4            6      30000    22.77    1498      98.59      5
…             …          …        …       …        …        …       …
15406       19537            9      10723    19.81    1086      68.05      5
15407       19540            2      18000    17.50    1373      91.10      7
15408       19541            6      67000    21.14    1498     103.52      5
15409       19542            5    3800000    16.00    2179     140.00      7
15410       19543            2      13000    18.00    1497     117.60      5

       selling_price  fuel_type_CNG  fuel_type_Diesel  fuel_type_Electric  \
0             120000            0.0               0.0                 0.0
1             550000            0.0               0.0                 0.0
```

```
2             215000            0.0               0.0                  0.0
3             226000            0.0               0.0                  0.0
4             570000            0.0               1.0                  0.0
…               …               …                 …                    …
15406         250000            0.0               0.0                  0.0
15407         925000            0.0               0.0                  0.0
15408         425000            0.0               1.0                  0.0
15409        1225000            0.0               1.0                  0.0
15410        1200000            0.0               0.0                  0.0

       fuel_type_LPG  fuel_type_Petrol  transmission_type_Automatic  \
0                0.0               1.0                          0.0
1                0.0               1.0                          0.0
2                0.0               1.0                          0.0
3                0.0               1.0                          0.0
4                0.0               0.0                          0.0
…                …                 …                            …
15406            0.0               1.0                          0.0
15407            0.0               1.0                          0.0
15408            0.0               0.0                          0.0
15409            0.0               0.0                          0.0
15410            0.0               1.0                          1.0

       transmission_type_Manual
0                           1.0
1                           1.0
2                           1.0
3                           1.0
4                           1.0
…                           …
15406                       1.0
15407                       1.0
15408                       1.0
15409                       1.0
15410                       0.0

[15411 rows x 15 columns]
```

[25]:
```
"""Linear regression - Modelling

Y (Target variable) = m1x1 + m2x2 + m3x3 ............

We will drop selling_price from independent variable"""

X = model_data.drop('selling_price', axis = 1)

# For getting the target variable we will just have selling_price
```

```
Y = model_data['selling_price']
Y
```

[25]:
```
0          120000
1          550000
2          215000
3          226000
4          570000
           …
15406      250000
15407      925000
15408      425000
15409     1225000
15410     1200000
Name: selling_price, Length: 15411, dtype: int64
```

[26]:
```
# To divide the data into Train and Test

train_X, test_X,train_Y,test_Y = train_test_split(X,Y,test_size = 0.2)

# 80% of the data goes to training and 20% of the data goes to testing
```

[27]:
```
# Applying regression for training the model
Regressor = LinearRegression().fit(train_X,train_Y)
```

[28]:
```
# Getting the predictions
prediction = Regressor.predict(test_X)

print(prediction)

print(test_Y)
```

```
[ 494091.22582228  182692.89262021 1329518.84047676 …  883910.65908778
  935937.9213368   440409.20396649]
482        475000
9020       315000
8076      1000000
11815      850000
3634       730000
           …
9834       485000
4335       600000
11408      690000
8204      1100000
11772      825000
Name: selling_price, Length: 3083, dtype: int64
```

[29]:
```
test_X['predicted_sales_price'] = prediction

test_X['Actual_price'] = test_Y

test_X['difference'] = test_X['predicted_sales_price'] - test_X['Actual_price']

test_X

mse = []
mse.append(mean_squared_error(y_true = test_Y,y_pred = prediction))

rmse = []
rmse.append(np.sqrt(mse))

rmse
```

[29]:
```
[array([470136.63866418])]
```

[ ]:

[ ]: