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 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

```
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

data = pd.read_excel("Cardekho.xlsx")
data.head(5)
```

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

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

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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15411 entries, 0 to 15410
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   car_name           15411 non-null  object 
 1   brand              15411 non-null  object 
 2   model              15411 non-null  object 
 3   vehicle_age        15411 non-null  int64  
 4   km_driven          15411 non-null  int64  
 5   seller_type        15411 non-null  object 
 6   fuel_type          15411 non-null  object 
 7   transmission_type  15411 non-null  object 
 8   mileage            15411 non-null  float64
 9   engine             15411 non-null  int64  
 10  max_power          15411 non-null  float64
 11  seats              15411 non-null  int64  
 12  selling_price      15411 non-null  int64  
dtypes: float64(2), int64(5), object(6)
memory usage: 1.5+ MB
```

```
data.shape
```

```
(15411, 13)
```

```
# summary statistics
data.describe()
```

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

        max_power  \
count  15411.000000   
mean      100.588254   
std        42.972979   
min        38.400000   
25%        74.000000   
50%        88.500000   
75%       117.300000   
max       626.000000   

              seats  selling_price  
count  15411.000000   1.541100e+04  
mean       5.325482   7.749711e+05  
```

```
std         0.807628   8.941284e+05
min         0.000000   4.000000e+04
25%         5.000000   3.850000e+05
50%         5.000000   5.560000e+05
75%         5.000000   8.250000e+05
max         9.000000   3.950000e+07
```

```
data['car_name'].value_counts()
```

```
car_name
Hyundai i20               906
Maruti Swift Dzire        890
Maruti Swift              781
Maruti Alto               778
Honda City                757
                         ...
Mercedes-AMG C              1
Rolls-Royce Ghost           1
Maserati Quattroporte       1
Isuzu MUX                   1
Force Gurkha                1
Name: count, Length: 121, dtype: int64
```

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

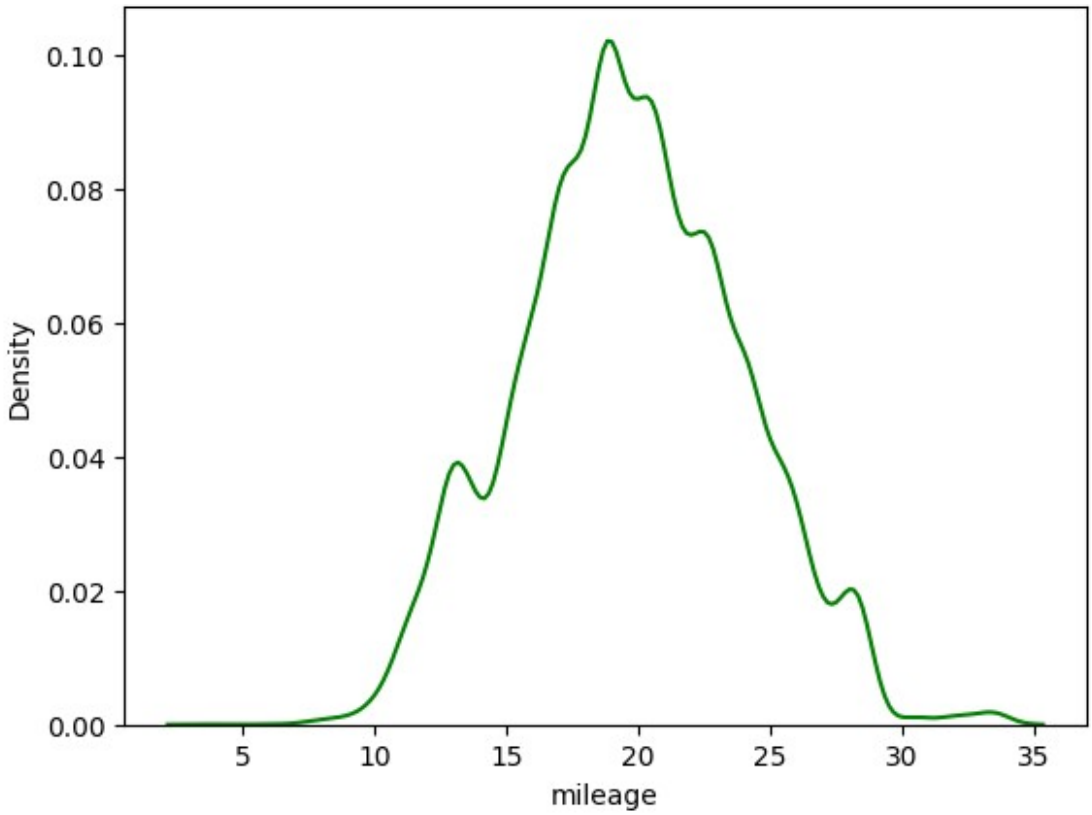
```
fuel_type
Petrol      7643
Diesel      7419
CNG          301
LPG           44
Electric       4
Name: count, dtype: int64
```

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

```
np.float64(19.70115112581922)
```

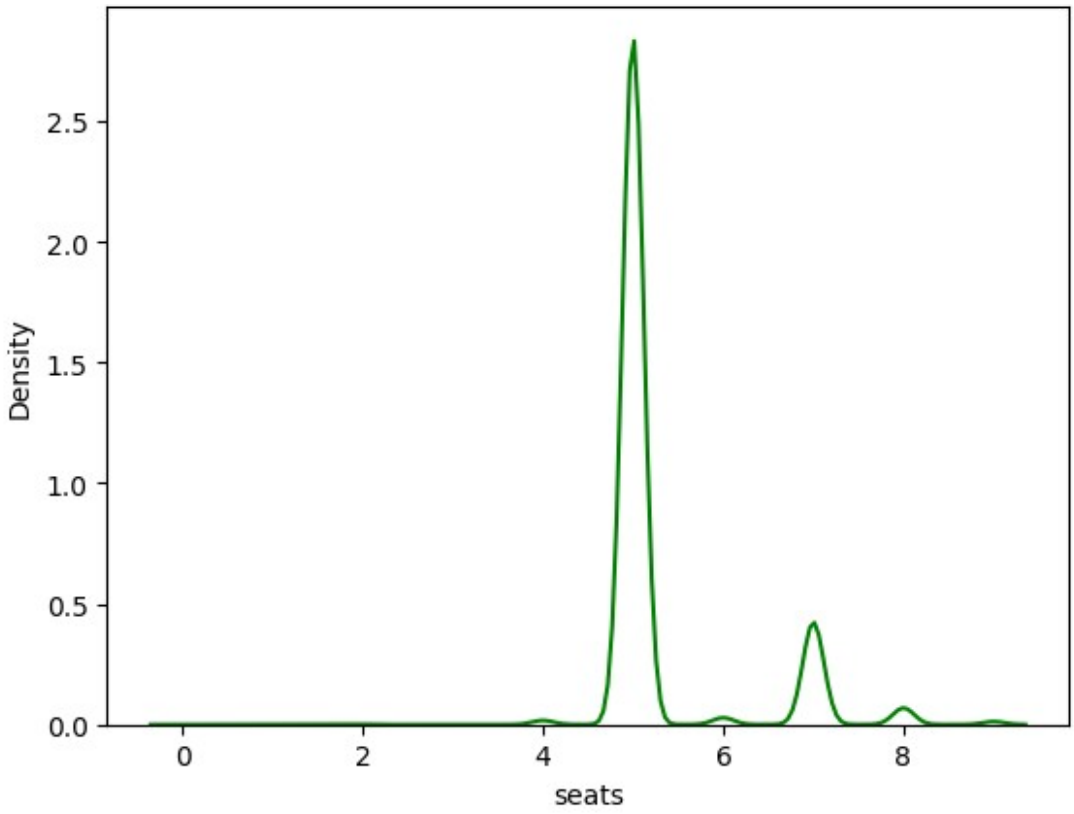
```
sns.kdeplot(x = data['mileage'],color = 'g')
```

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



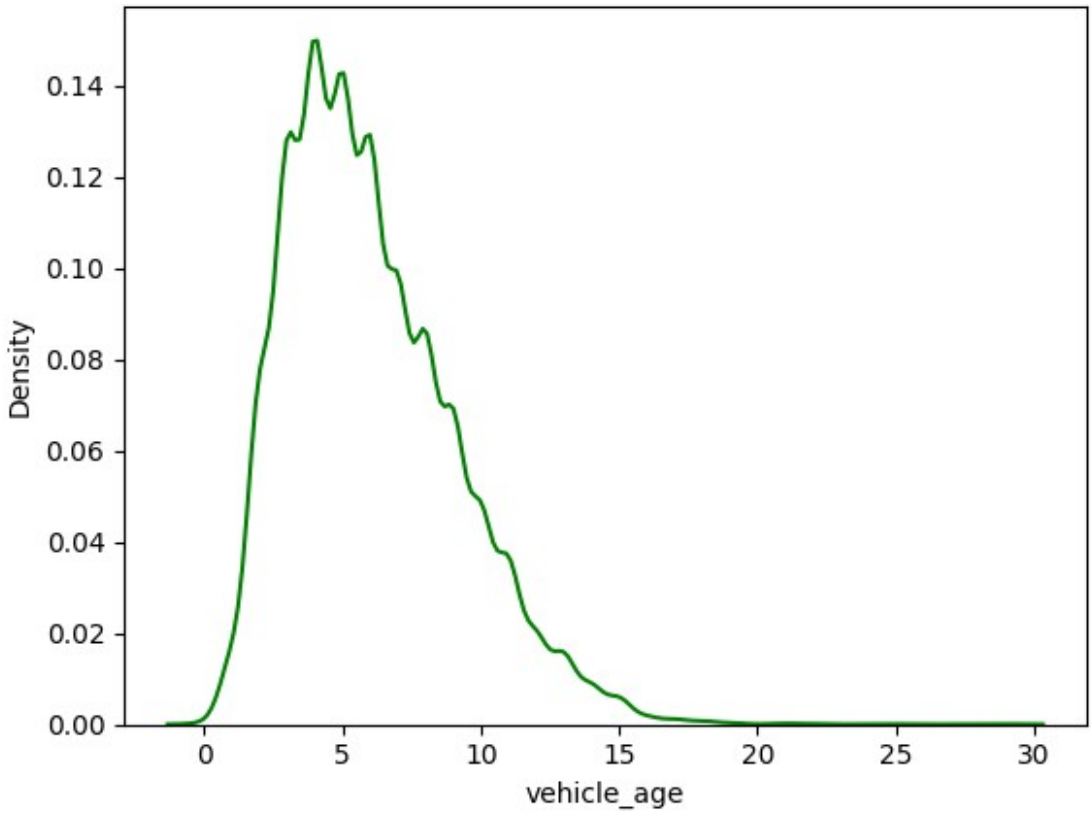

```
sns.kdeplot(x = data['seats'],color = 'g')
```

```
<Axes: xlabel='seats', ylabel='Density'>
```



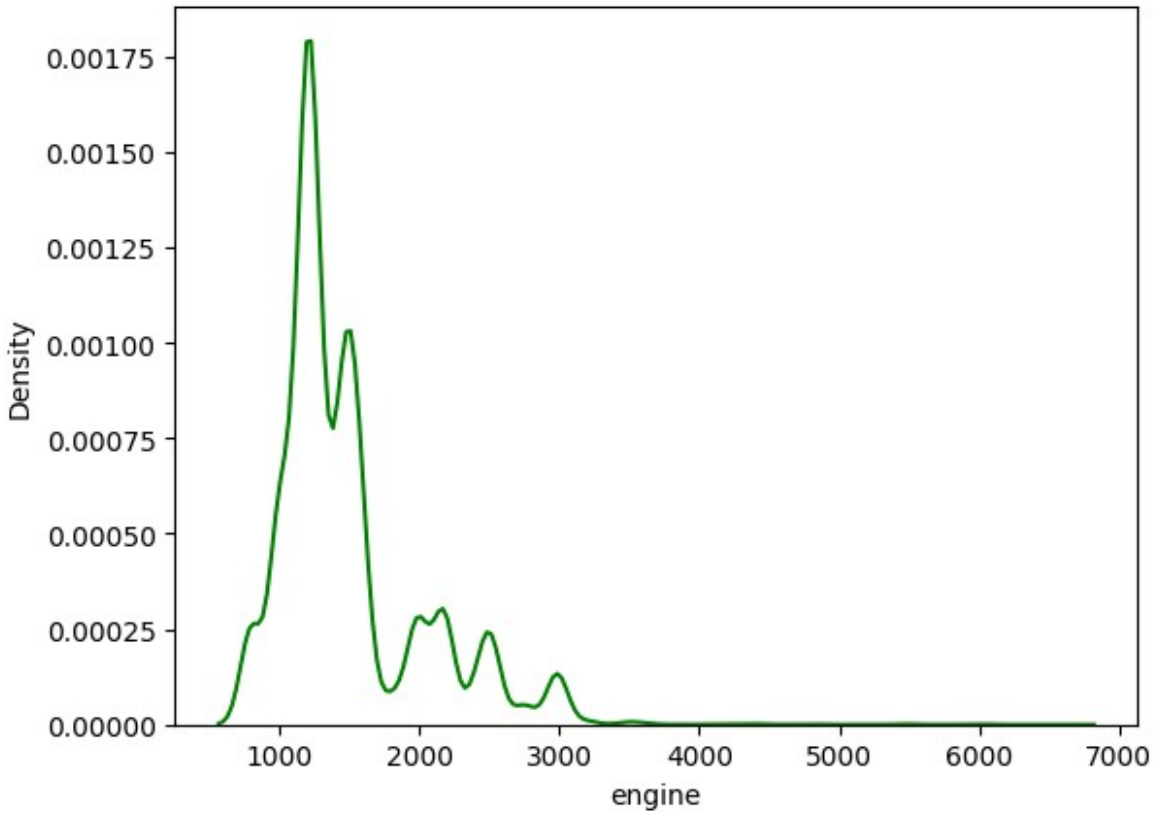

```
sns.kdeplot(x = data['vehicle_age'],color = 'g')
```

```
<Axes: xlabel='vehicle_age', ylabel='Density'>
```



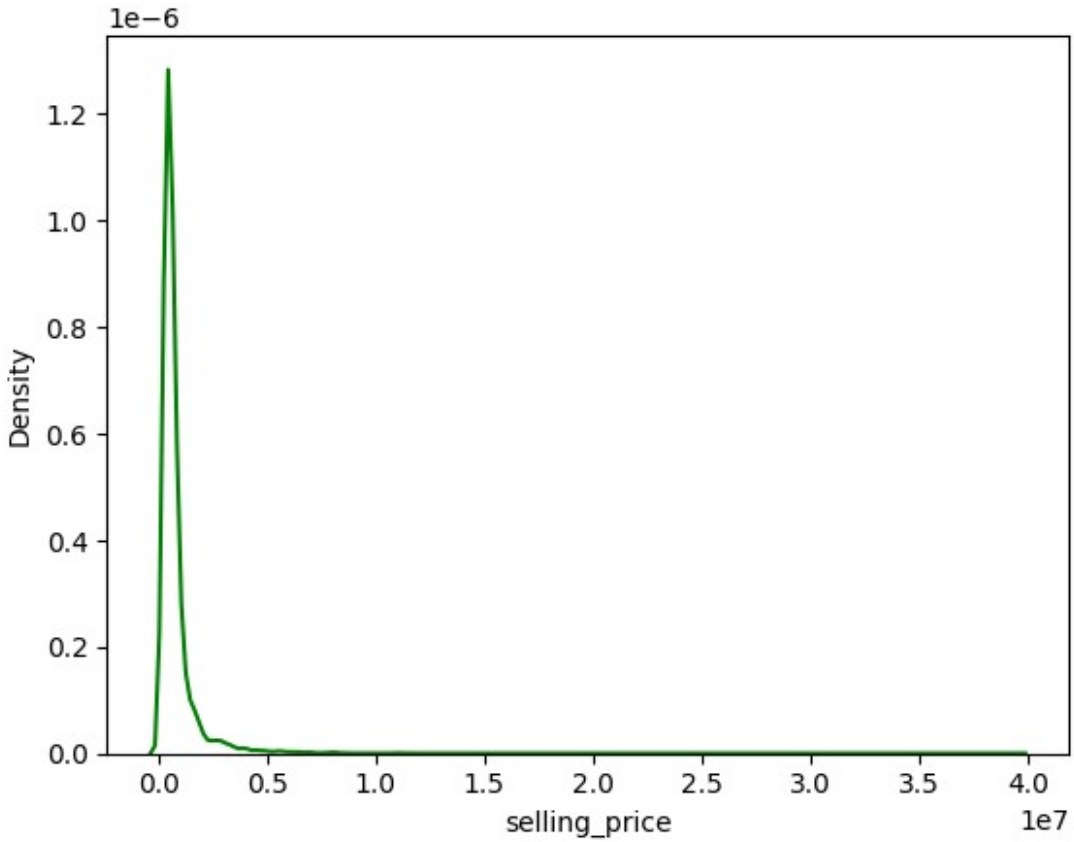

```
sns.kdeplot(x = data['engine'],color = 'g')
```

```
<Axes: xlabel='engine', ylabel='Density'>
```



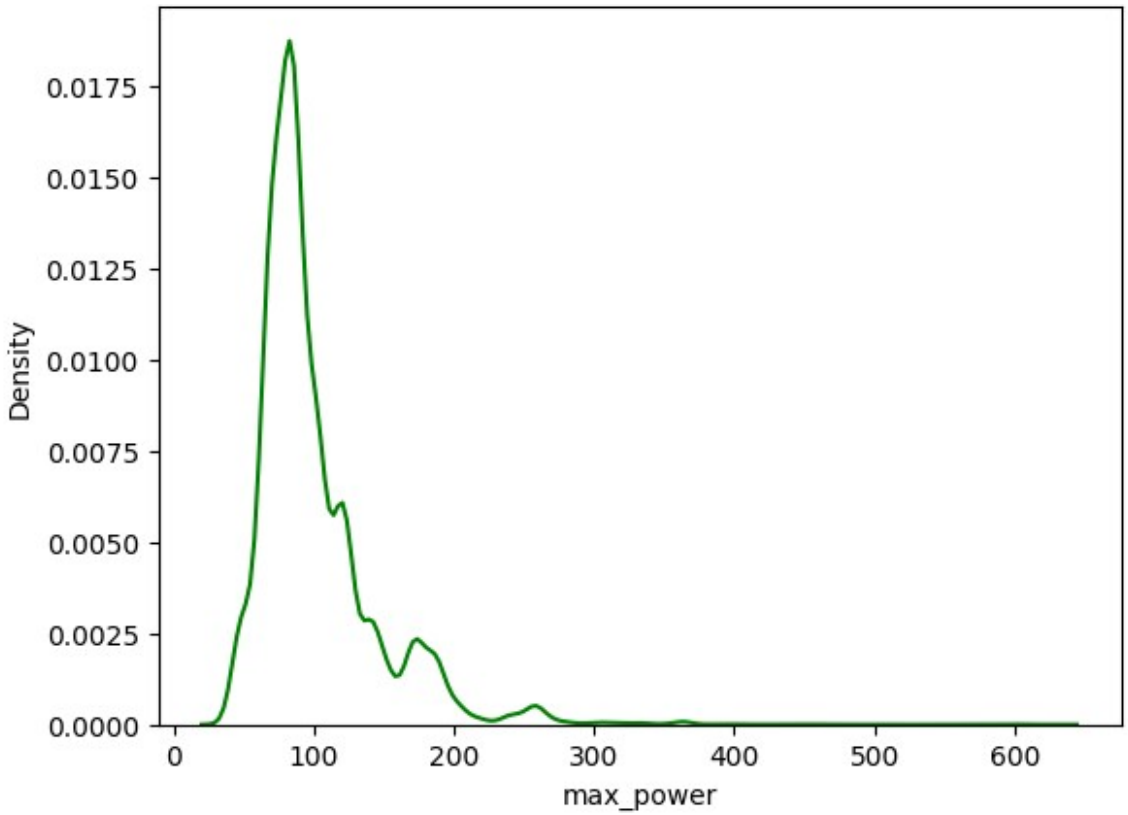

```
sns.kdeplot(x = data['selling_price'],color = 'g')
```

```
<Axes: xlabel='selling_price', ylabel='Density'>
```



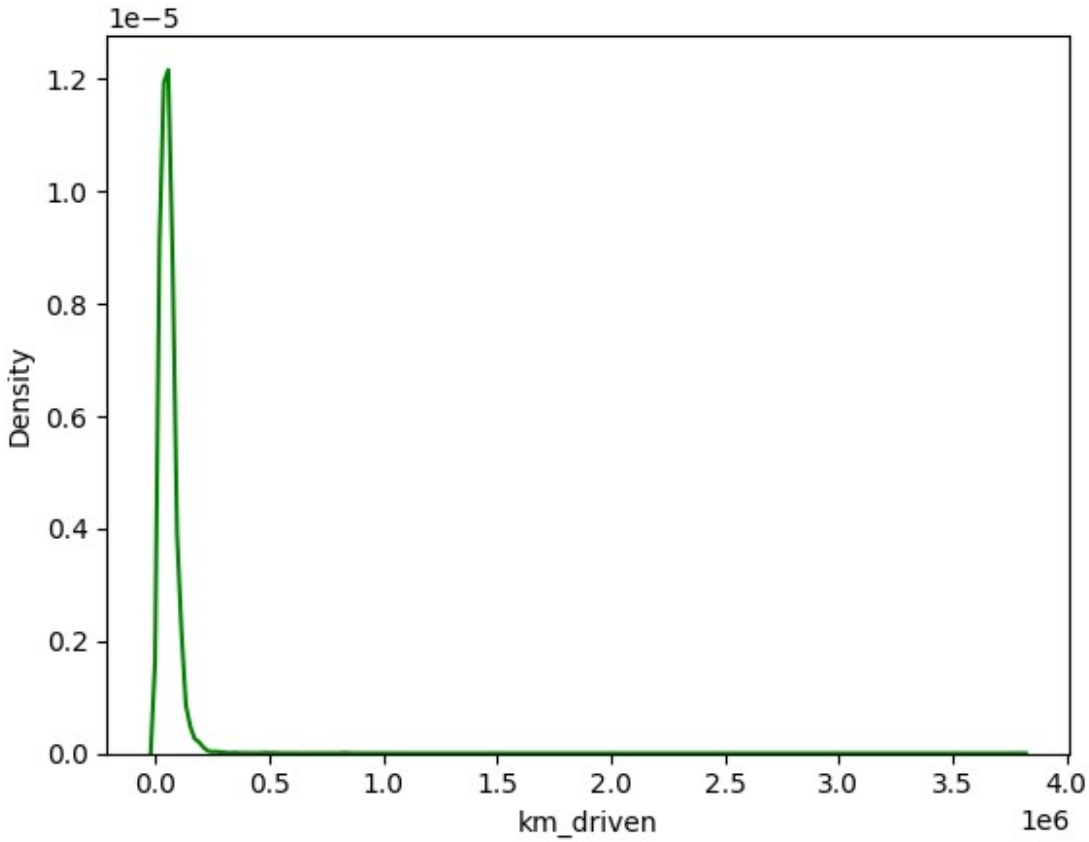

```
sns.kdeplot(x = data['max_power'],color = 'g')
```

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




```
sns.kdeplot(x = data['km_driven'],color = 'g')
```

```
<Axes: xlabel='km_driven', ylabel='Density'>
```



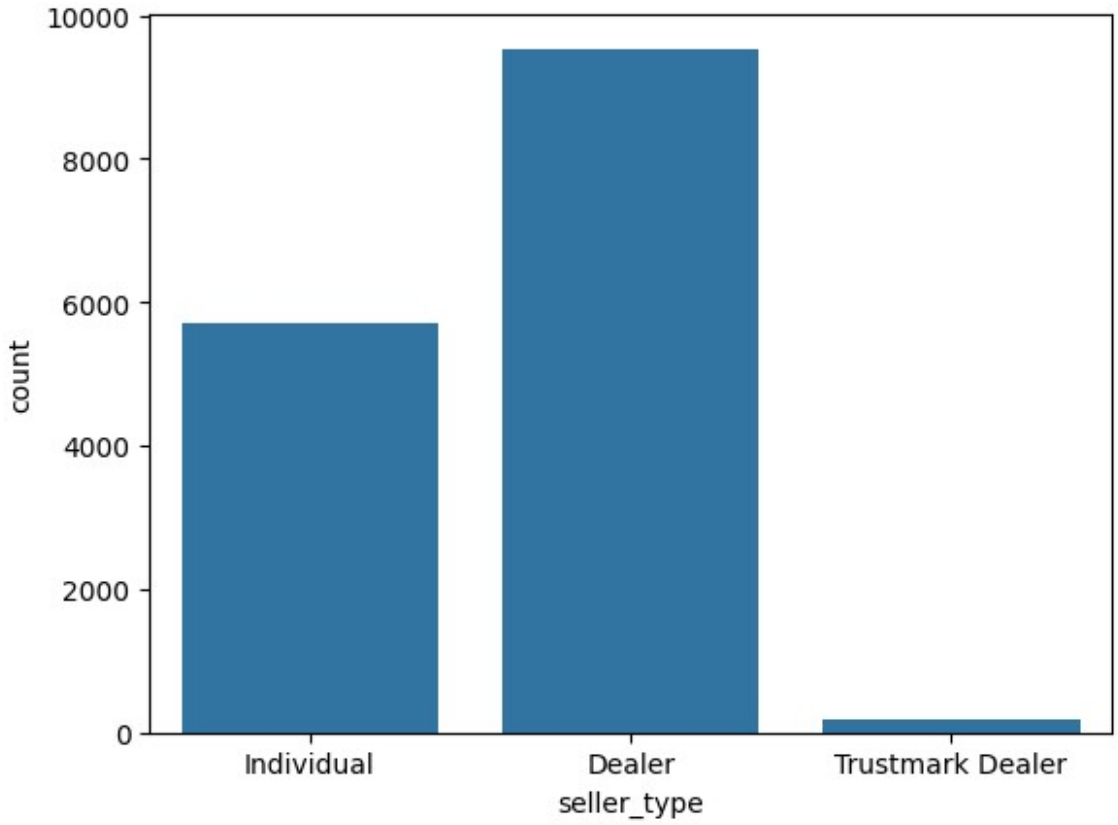

```
data[data['max_power'] >= 400]
```

	car_name	brand	model	vehicle_age
1172	Bentley Continental	Bentley	Continental	9
1209	Porsche Cayenne	Porsche	Cayenne	4
3799	Ferrari GTC4Lusso	Ferrari	GTC4Lusso	2
9190	Porsche Cayenne	Porsche	Cayenne	12
9364	Porsche Cayenne	Porsche	Cayenne	4
9450	BMW 6	BMW	6	12
9722	Mercedes-Benz S-Class	Mercedes-Benz	S-Class	3
10040	Bentley Continental	Bentley	Continental	9
10969	Rolls-Royce Ghost	Rolls-Royce	Ghost	4
12067	BMW 7	BMW	7	11

```
12997    Bentley Continental        Bentley  Continental           10


       km_driven seller_type fuel_type transmission_type  mileage
engine  \
1172        9000      Dealer    Petrol         Automatic     9.50
5998
1209       36000      Dealer    Petrol         Automatic    12.50
3604
3799        3800      Dealer    Petrol         Automatic     4.00
3855
9190      126000  Individual    Petrol         Automatic     8.50
4806
9364       24000      Dealer    Petrol         Automatic    12.50
3604
9450       65000      Dealer    Petrol         Automatic     7.94
4395
9722        4000      Dealer    Petrol         Automatic     7.81
4663
10040      37500      Dealer    Petrol         Automatic     6.00
5998
10969       5000  Individual    Petrol         Automatic    10.20
6592
12067      64000      Dealer    Petrol         Automatic     8.77
4395
12997      30000      Dealer    Petrol         Automatic     8.60
5998

       max_power  seats  selling_price
1172       626.0      4       14500000
1209       420.0      5        7800000
3799       601.0      4       39500000
9190       500.0      5        2000000
9364       440.0      5       11100000
9450       450.0      4        1500000
9722       459.0      4       13000000
10040      600.0      5        5200000
10969      563.0      4       24200000
12067      402.0      5        1499000
12997      552.0      4        8100000
```

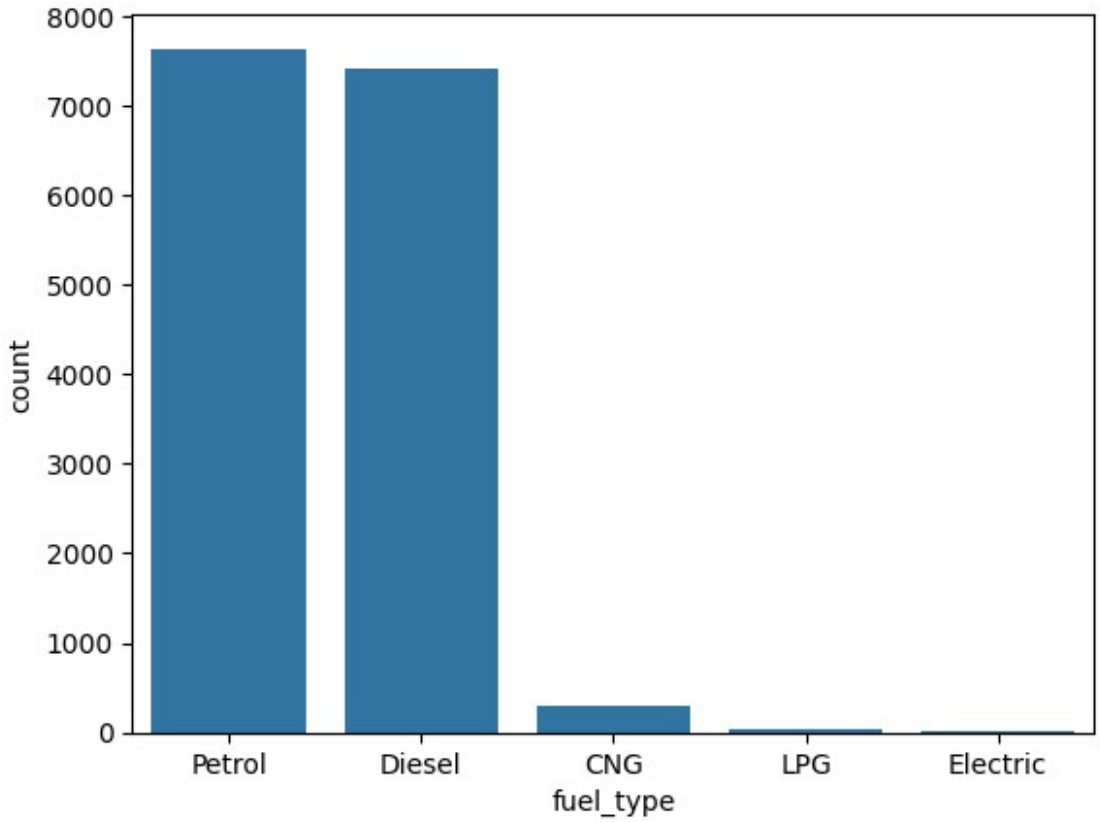
```
sns.countplot(x = data['seller_type'])
```

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



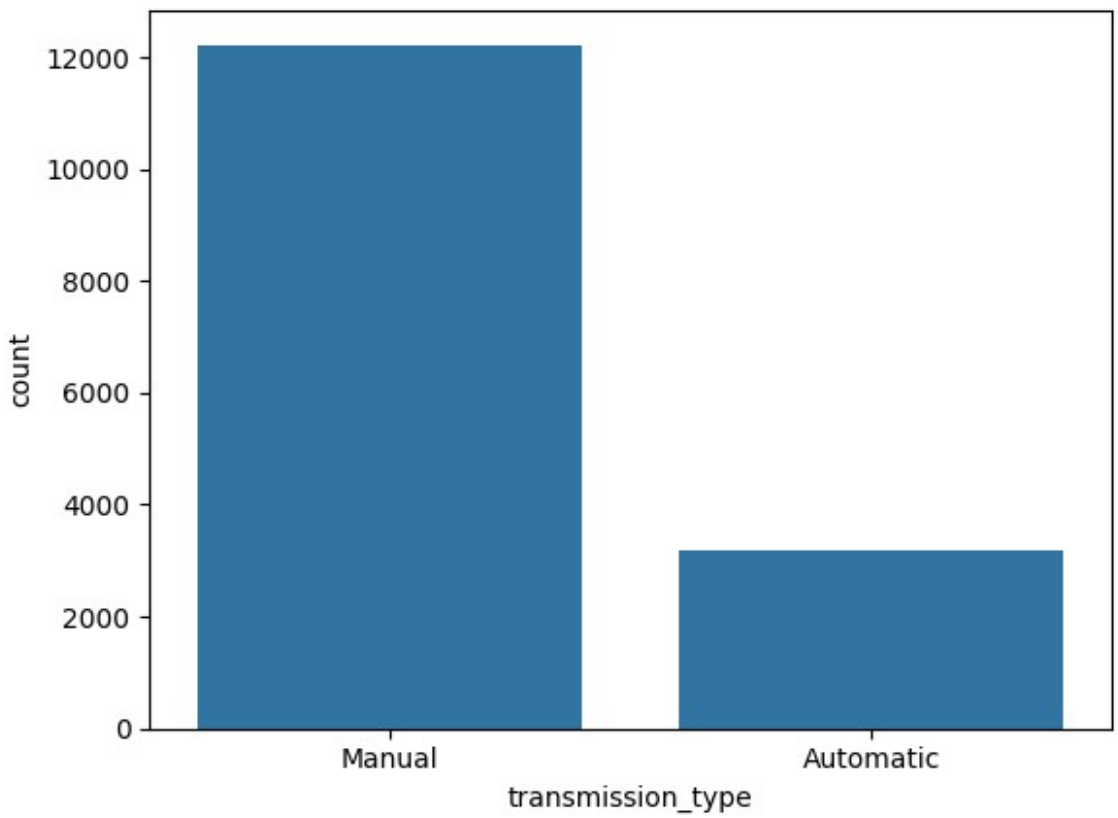

```
sns.countplot(x = data['fuel_type'])
```

```
<Axes: xlabel='fuel_type', ylabel='count'>
```

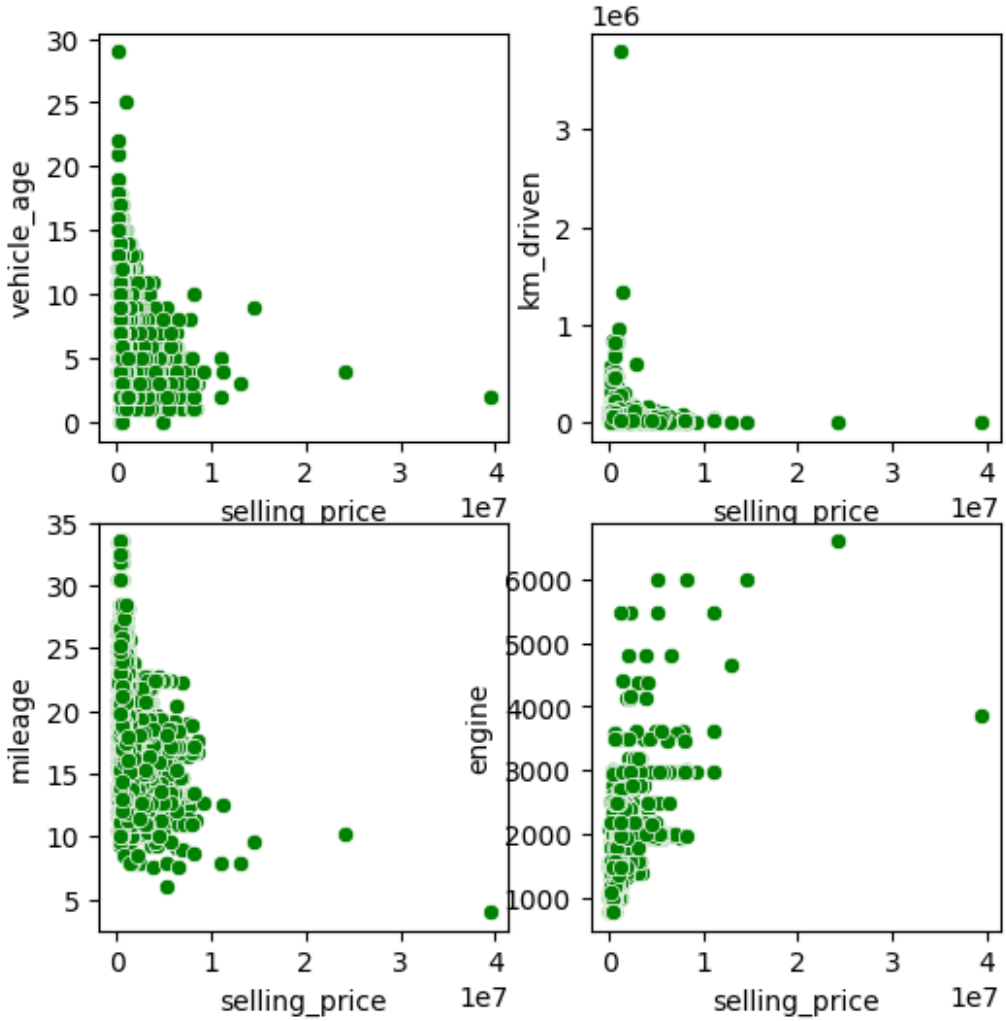



```
sns.countplot(x = data['transmission_type'])
```

```
<Axes: xlabel='transmission_type', ylabel='count'>
```

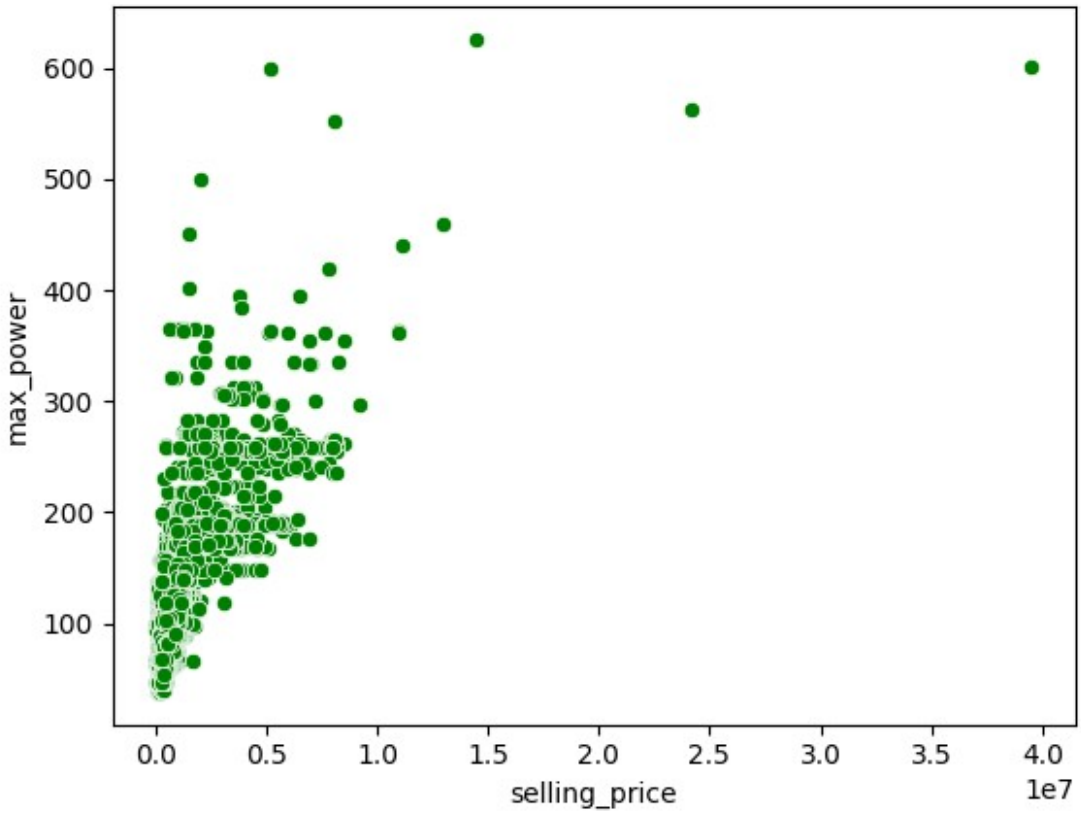



```
#Lets look at the relationship of each variable with the selling price
(Target variable)

fig = plt.figure(figsize = (6,6))

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')
```



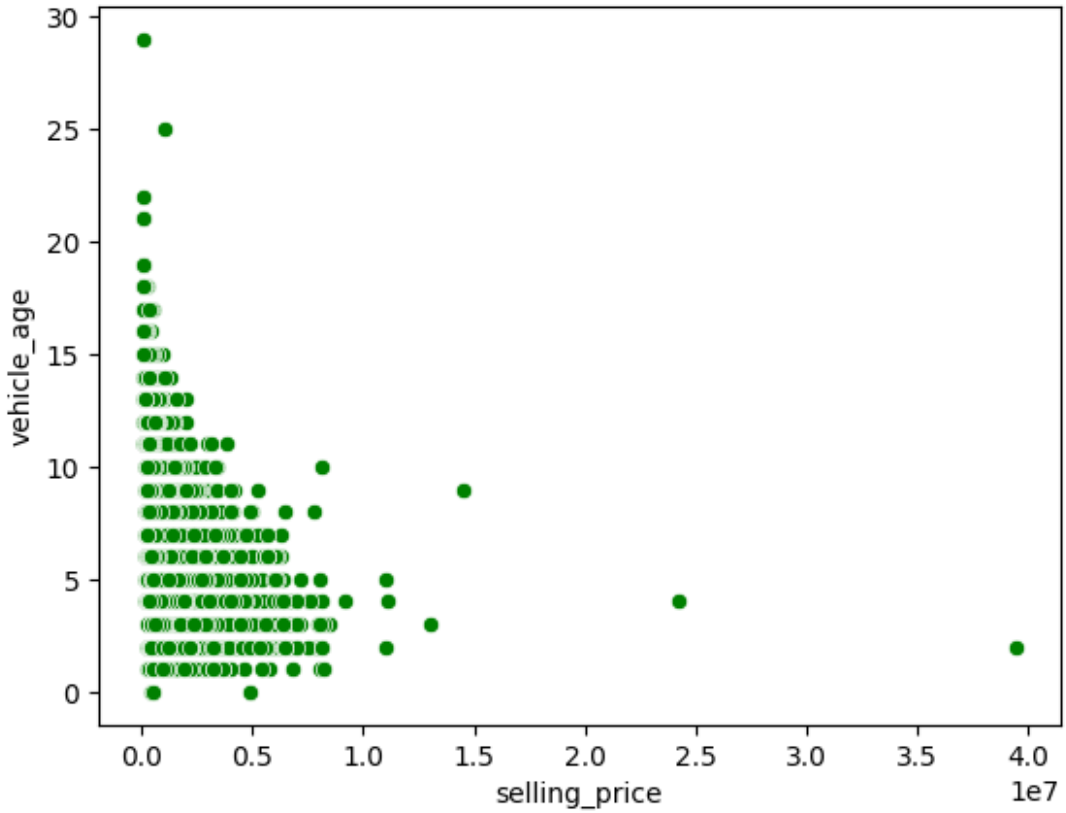

```
sns.scatterplot(data = data, x = 'selling_price',y = 'max_power',color
= 'g')
```

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



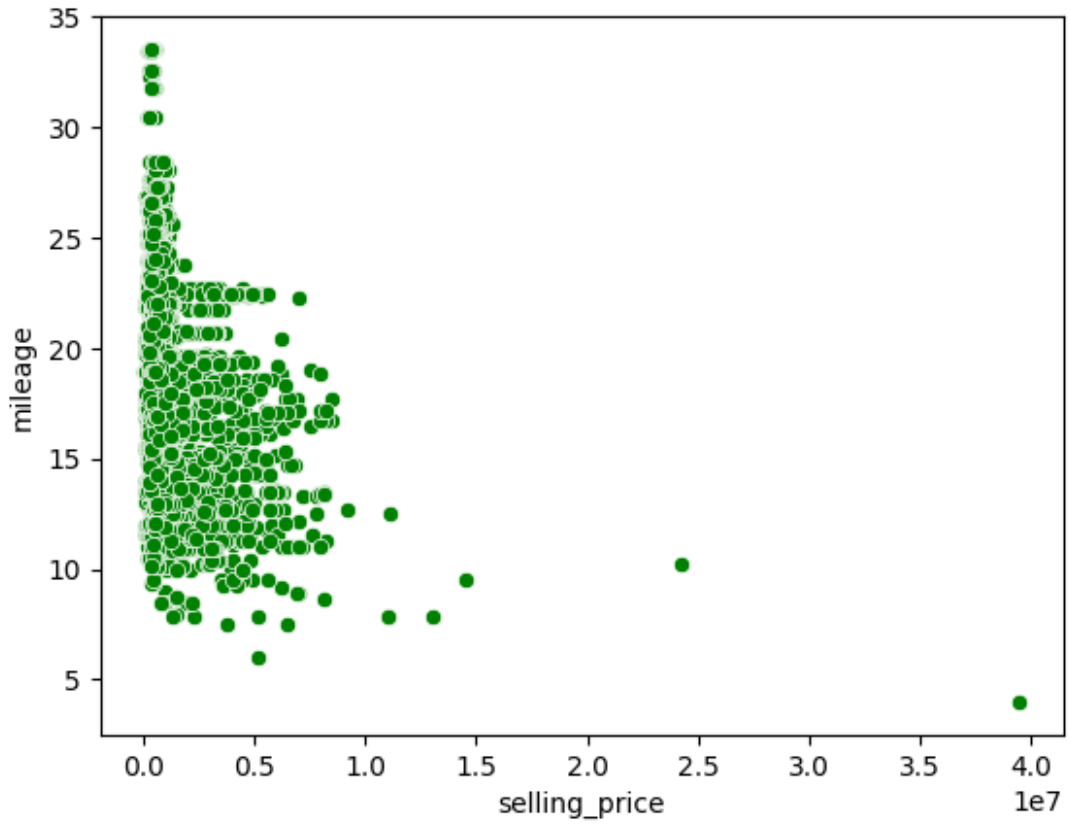

```
sns.scatterplot(data = data, x = 'selling_price',y =
'vehicle_age',color = 'g')
```

```
<Axes: xlabel='selling_price', ylabel='vehicle_age'>
```



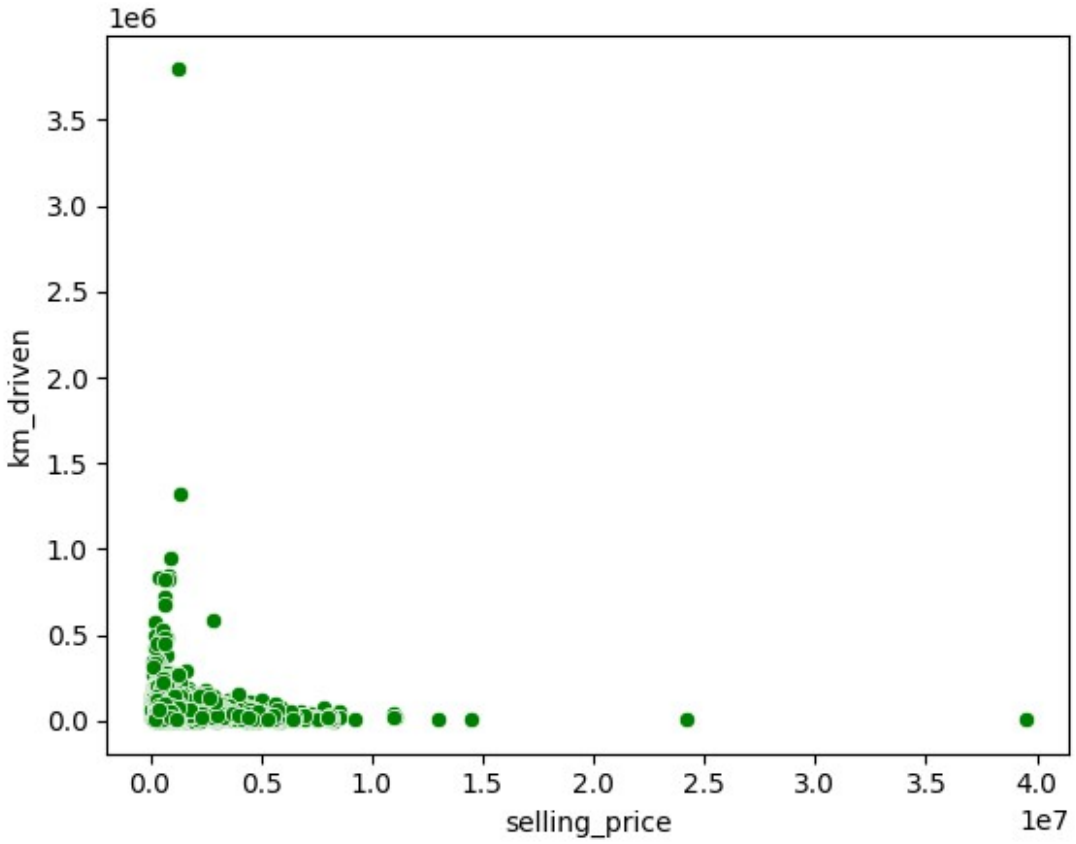

```
sns.scatterplot(data = data, x = 'selling_price',y = 'mileage',color = 'g')
```

```
<Axes: xlabel='selling_price', ylabel='mileage'>
```



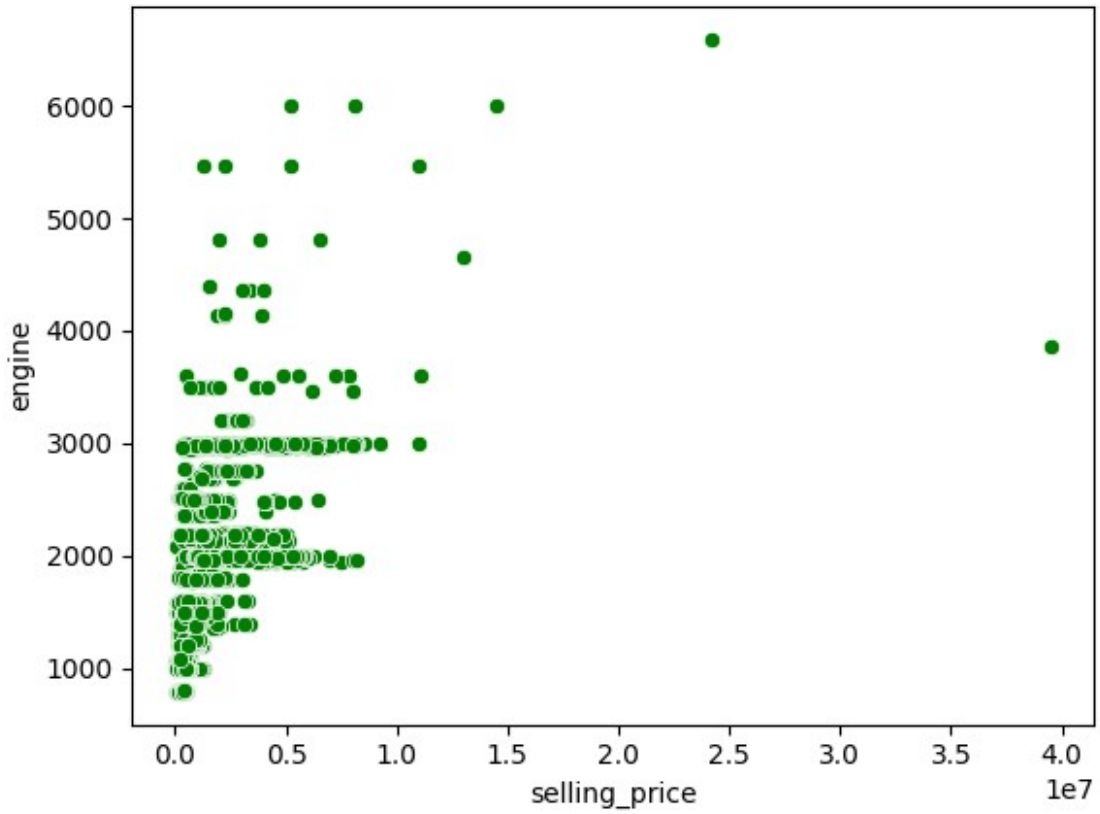

```
sns.scatterplot(data = data, x = 'selling_price',y = 'km_driven',color
= 'g')
```

```
<Axes: xlabel='selling_price', ylabel='km_driven'>
```



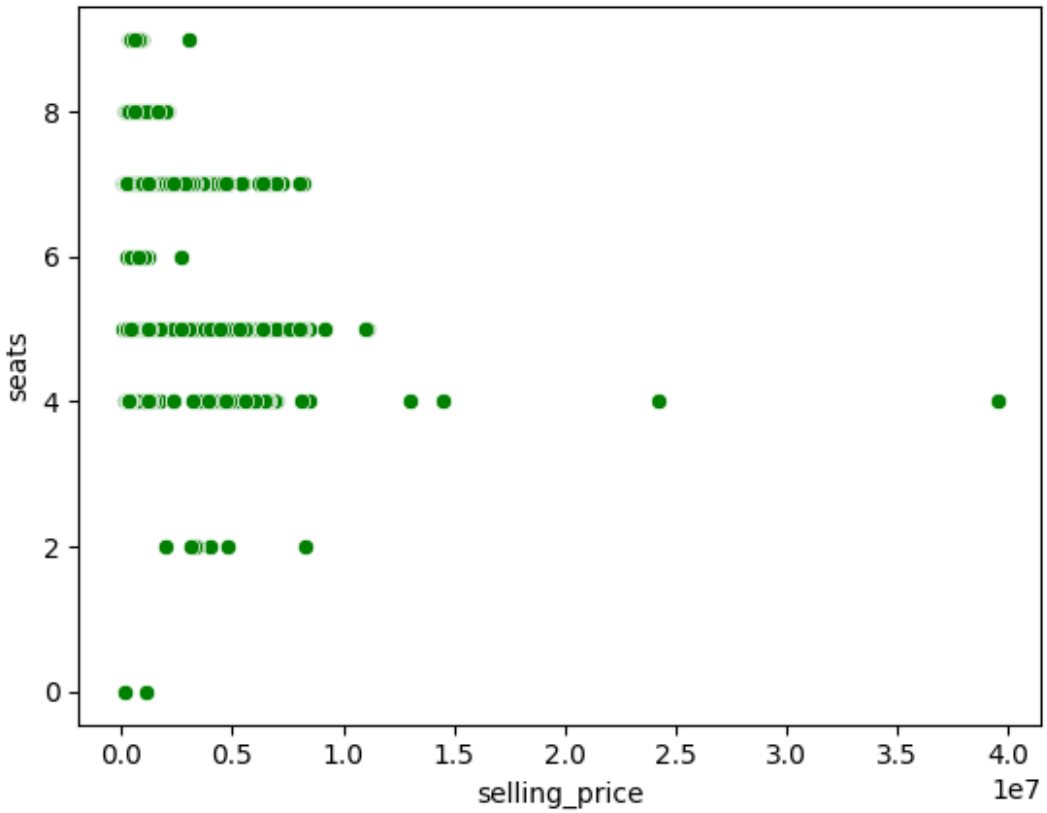

```
sns.scatterplot(data = data, x = 'selling_price',y = 'engine',color = 'g')
```

```
<Axes: xlabel='selling_price', ylabel='engine'>
```




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

```
<Axes: xlabel='selling_price', ylabel='seats'>
```



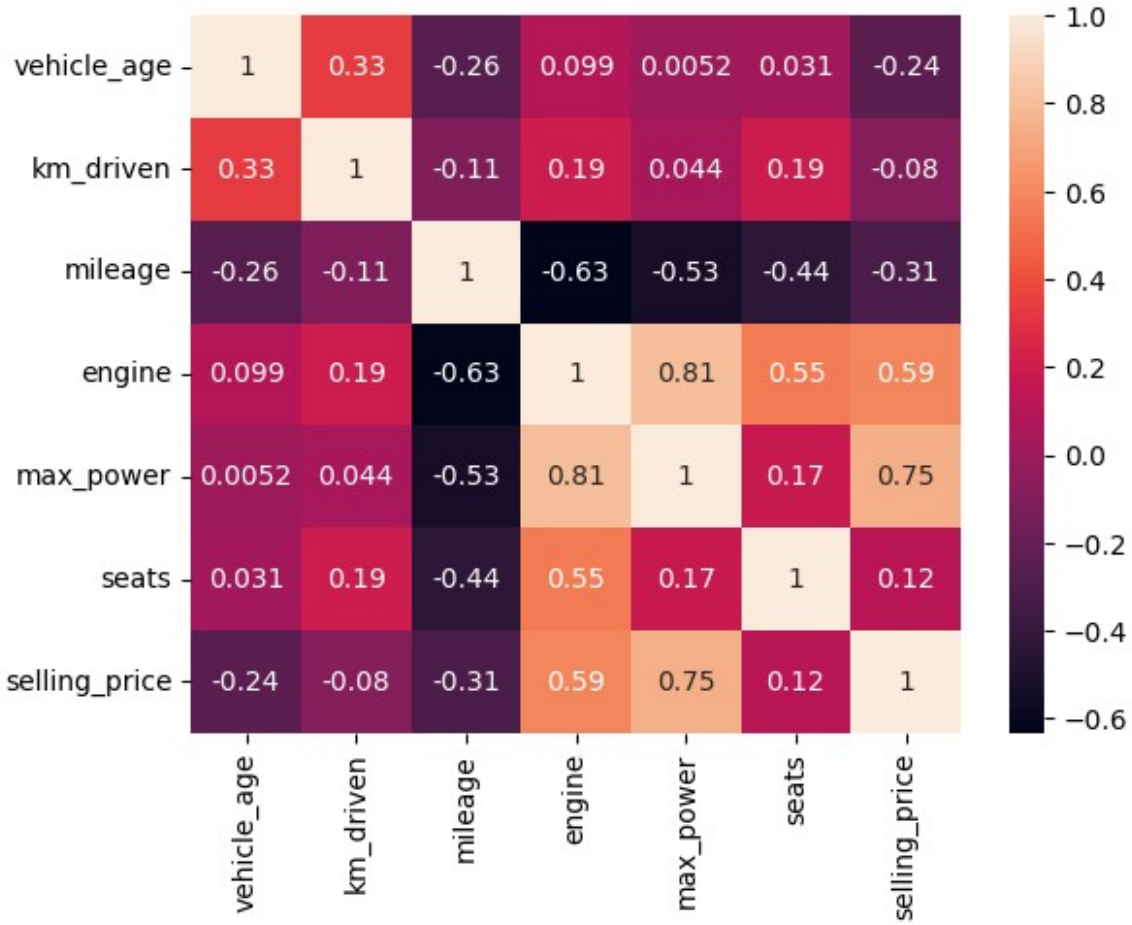

```
#Multi-variate analysis - to check correlation between all the
combination of numerical features

features =
['vehicle_age','km_driven','mileage','engine','max_power','seats','sel
ling_price']

data[features].corr()
```

	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
```

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

```
<Axes: >
```




```
data.head()
```

```
        car_name    brand     model  vehicle_age  km_driven
seller_type  \
0    Maruti Alto   Maruti      Alto            9     120000
Individual
1  Hyundai Grand  Hyundai     Grand            5      20000
```

```
Individual
2    Hyundai i20  Hyundai       i20            11      60000
Individual
3    Maruti Alto   Maruti      Alto             9      37000
Individual
4  Ford Ecosport     Ford  Ecosport             6      30000
Dealer

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

   selling_price
0         120000
1         550000
2         215000
3         226000
4         570000

model_data = data.copy()
model_data.head()

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

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

   selling_price
0         120000
1         550000
2         215000
```

```
3         226000
4         570000

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

```
       vehicle_age  km_driven fuel_type transmission_type  mileage
engine  \
0                9     120000    Petrol            Manual    19.70
796
1                5      20000    Petrol            Manual    18.90
1197
2               11      60000    Petrol            Manual    17.00
1197
3                9      37000    Petrol            Manual    20.92
998
4                6      30000    Diesel            Manual    22.77
1498
...            ...        ...       ...               ...      ...
...
15406            9      10723    Petrol            Manual    19.81
1086
15407            2      18000    Petrol            Manual    17.50
1373
15408            6      67000    Diesel            Manual    21.14
1498
15409            5    3800000    Diesel            Manual    16.00
2179
15410            2      13000    Petrol         Automatic    18.00
1497

       max_power  seats  selling_price
0          46.30      5         120000
1          82.00      5         550000
2          80.00      5         215000
3          67.10      5         226000
4          98.59      5         570000
...          ...    ...            ...
15406      68.05      5         250000
15407      91.10      7         925000
15408     103.52      5         425000
15409     140.00      7        1225000
15410     117.60      5        1200000

[15411 rows x 9 columns]
```

```
model_data = pd.get_dummies(model_data,dtype = float)
model_data
```

```
       vehicle_age  km_driven  mileage  engine  max_power  seats  \
0                9     120000    19.70     796      46.30      5
1                5      20000    18.90    1197      82.00      5
2               11      60000    17.00    1197      80.00      5
3                9      37000    20.92     998      67.10      5
4                6      30000    22.77    1498      98.59      5
...            ...        ...      ...     ...        ...    ...
15406            9      10723    19.81    1086      68.05      5
15407            2      18000    17.50    1373      91.10      7
15408            6      67000    21.14    1498     103.52      5
15409            5    3800000    16.00    2179     140.00      7
15410            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 14 columns]
```

```
"""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
```

```
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
```

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

```
       vehicle_age  km_driven  mileage  engine  max_power  seats  \
4865             4      41275    13.00    1591     121.30      5
6946             2      16000    19.50    1199      88.76      5
```

```
9904              5      30000    13.00    1591     121.30      5
2084              5      50000    19.87    1461      83.80      5
9323              4      56000    26.60     998      58.16      5
...             ...        ...      ...     ...        ...    ...
15392             5     128000    12.63    2179     147.50      5
4360              6      89635    13.60    2523      63.00      9
1738              3      36000    24.04     999      67.00      5
4613              3      25000    25.50    1498      98.60      5
7246              8      23000    16.47    1198      73.90      5

       fuel_type_CNG  fuel_type_Diesel  fuel_type_Electric
fuel_type_LPG  \
4865               0.0               0.0                 0.0
0.0
6946               0.0               0.0                 0.0
0.0
9904               0.0               0.0                 0.0
0.0
2084               0.0               1.0                 0.0
0.0
9323               1.0               0.0                 0.0
0.0
...                ...               ...                 ...
...
15392              0.0               1.0                 0.0
0.0
4360               0.0               1.0                 0.0
0.0
1738               0.0               0.0                 0.0
0.0
4613               0.0               1.0                 0.0
0.0
7246               0.0               0.0                 0.0
0.0

       fuel_type_Petrol  transmission_type_Automatic
transmission_type_Manual
4865                1.0                          1.0
0.0
6946                1.0                          0.0
1.0
9904                1.0                          0.0
1.0
2084                0.0                          0.0
1.0
9323                0.0                          0.0
1.0
...                 ...                          ...
...
```

```
15392                0.0                            1.0
0.0
4360                 0.0                            0.0
1.0
1738                 1.0                            1.0
0.0
4613                 0.0                            0.0
1.0
7246                 1.0                            0.0
1.0

[12328 rows x 13 columns]
```

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

```
LinearRegression()
```

```
# Getting the predictions
prediction = Regressor.predict(test_X)

print(prediction)

print(test_Y)
```

```
[ 360355.67045608  406417.15819262 1614168.2401381  ...
1133774.82204532
  925209.76577257 -199352.56474736]
3446      675000
13961     435000
3519     1750000
725       335000
4978      700000
          ...
4326      775000
14089     900000
11815     850000
8139      800000
6676      130000
Name: selling_price, Length: 3083, dtype: int64
```

```
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
```

```
[array([464368.35038147])]
```