Importing requried labries

In [1]:

import pandas as pd

import numpy as np

from sklearn import linear\_model

from sklearn.preprocessing import LabelEncoder

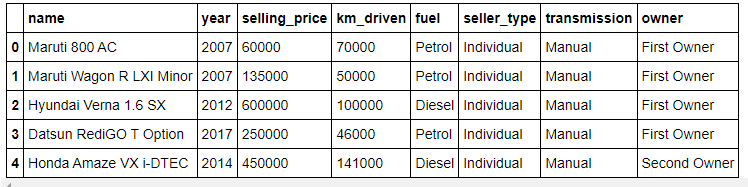
from sklearn.model\_selection import train\_test\_split

In [2]:

cardata=pd.read\_csv("/content/CAR DETAILS FROM CAR DEKHO.csv")

cardata.head()

Out[2]:



information about dataframe

In [3]:

cardata.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 4340 entries, 0 to 4339

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 name 4340 non-null object

1 year 4340 non-null int64

2 selling\_price 4340 non-null int64

3 km\_driven 4340 non-null int64

4 fuel 4340 non-null object

5 seller\_type 4340 non-null object

6 transmission 4340 non-null object

7 owner 4340 non-null object

dtypes: int64(3), object(5)

memory usage: 271.4+ KB

seperating dependent and independent variables

In [4]:

input\_data=cardata.drop(["name"],axis=1)

target\_data=cardata.selling\_price

Checking data

In [5]:

print(input\_data.head())

print(target\_data.head())

year selling\_price km\_driven ... seller\_type transmission owner

0 2007 60000 70000 ... Individual Manual First Owner

1 2007 135000 50000 ... Individual Manual First Owner

2 2012 600000 100000 ... Individual Manual First Owner

3 2017 250000 46000 ... Individual Manual First Owner

4 2014 450000 141000 ... Individual Manual Second Owner

[5 rows x 7 columns]

0 60000

1 135000

2 600000

3 250000

4 450000

Name: selling\_price, dtype: int64

Encoding the categorical columns

In [6]:

enc=LabelEncoder()

input\_data['seller\_type']=enc.fit\_transform(input\_data['seller\_type'])

input\_data["transmission"]=enc.fit\_transform(input\_data['transmission'])

input\_data["owner"]=enc.fit\_transform(input\_data['owner'])

input\_data["fuel"]=enc.fit\_transform(input\_data['fuel'])

In [8]:

cardata.fuel.value\_counts()

Out[8]:

Diesel 2153

Petrol 2123

CNG 40

LPG 23

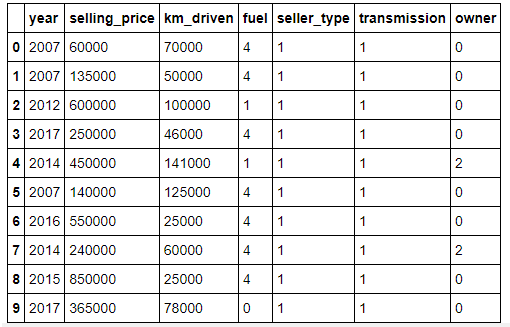
Electric 1

Name: fuel, dtype: int64

In [9]:

input\_data.head(10)

Out[9]:



In [10]:

import seaborn as sns

import matplotlib.pyplot as plt

plt.style.use('seaborn-white')

plotting relation of each independent variable with dependent variable

In [11]:

plt.figure(figsize=(25,5))

plt.subplot(1,5,1)

sns.scatterplot(x=input\_data["km\_driven"],y=input\_data['selling\_price'])

plt.subplot(1,5,2)

sns.scatterplot(x=input\_data['transmission'],y=input\_data['selling\_price'])

plt.subplot(1,5,3)

sns.scatterplot(x=input\_data['seller\_type'],y=input\_data['selling\_price'])

plt.subplot(1,5,4)

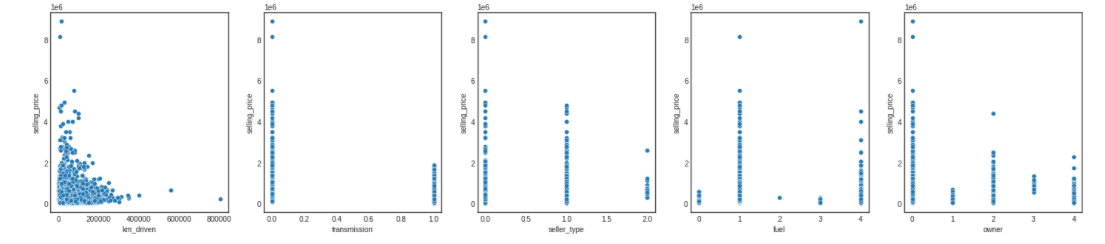
sns.scatterplot(x=input\_data['fuel'],y=input\_data['selling\_price'])

plt.subplot(1,5,5)

sns.scatterplot(x=input\_data['owner'],y=input\_data['selling\_price'])

Out[11]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7ff71916f450>



Splitting data into training and testing data

In [12]:

x\_train,x\_test,y\_train,y\_test=train\_test\_split(input\_data,target\_data,test\_size=0.3,random\_state=32)

print(x\_train.shape)

print(x\_test.shape)

(3038, 7)

(1302, 7)

Instantiating Linear Regression model

In [13]:

model=linear\_model.LinearRegression()

In [14]:

model.fit(x\_train,y\_train)

Out[14]:

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

In [15]:

y\_pred=model.predict(x\_test)

Evaluation metrics

In [16]:

from sklearn.metrics import r2\_score,mean\_squared\_error,mean\_absolute\_error

r2\_score(y\_test,y\_pred)

Out[16]:

1.0

In [17]:

mean\_squared\_error(y\_test,y\_pred)

Out[17]:

1.660773336078694e-20

In [18]:

mean\_absolute\_error(y\_test,y\_pred)

Out[18]:

5.902355170584128e-11

Testing the model with real independent values from dataset

In [19]:

pred=model.predict([[2007,135000,50000,4,1,1,0]])

print(pred)

[135000.]

In [20]:

target\_data[1]

Out[20]:

135000

In [ ]:

Both the predicted and actual values are same so our model is performing well