Assignment 17

kartik Thakur

In [36]:
```
# Predict the selling price of car from the given dataset using Linear Regression Model
# Column Name:
# name: Name of Car
# year: In which year car is purchased
# selling_price: Selling price of Car
# km_driven: Car driven in Kms
# fuel: Type of car Petrol,Diesel or CNG
# seller_type: who is selling the car
# transmission: car is automatic or manual
# mileage: Mileage of Car
# engine: Car Engine in cc
# max_power: Maximum power of car
# seats: Number of seats in car
```

step 1 : firstly import the necessarry libraries

In [59]:
```
# firstly import the necessarry libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
%matplotlib inline
```

step 2 : Import the csv file and clean the car data

In [60]:
```
car=pd.read_csv('car_details.csv')
```

In [61]:
```
car.head()
```

Out[61]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	mileage	engine	max_power
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	23.4 kmpl	1248 CC	74 bhp
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	21.14 kmpl	1498 CC	103.52 bhp
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	17.7 kmpl	1497 CC	78 bhp
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	23.0 kmpl	1396 CC	90 bhp
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	16.1 kmpl	1298 CC	88.2 bhp

step 3: cleaning the data from null values and more

In [62]:
```
car.isnull().sum()     # checking null values
```

Out[62]:
```
name              0
year              0
selling_price     0
km_driven         0
fuel              0
seller_type       0
transmission      0
mileage         221
engine          221
max_power       215
seats           221
dtype: int64
```

In [63]:
```
car.dropna(inplace=True)
```

In [64]:
```
car.isnull().sum()
```

Out[64]:
```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
mileage          0
engine           0
max_power        0
seats            0
dtype: int64
```

In [65]:
```
car.info() # getting info of car data
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7907 entries, 0 to 8127
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   name           7907 non-null   object 
 1   year           7907 non-null   int64  
 2   selling_price  7907 non-null   int64  
 3   km_driven      7907 non-null   int64  
 4   fuel           7907 non-null   object 
 5   seller_type    7907 non-null   object 
 6   transmission   7907 non-null   object 
 7   mileage        7907 non-null   object 
 8   engine         7907 non-null   object 
 9   max_power      7907 non-null   object 
 10  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(7)
memory usage: 741.3+ KB
```

In [66]:
```
car.head()
```

Out[66]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	mileage	engine	max_power
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	23.4 kmpl	1248 CC	74 bhp
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	21.14 kmpl	1498 CC	103.52 bhp
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	17.7 kmpl	1497 CC	78 bhp
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	23.0 kmpl	1396 CC	90 bhp
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	16.1 kmpl	1298 CC	88.2 bhp

In [67]:
```
car['seats']=car['seats'].astype('int32')  # changing float to int32
car['mileage']=car['mileage'].str.replace('kmpl|km/kg','')
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_10736\1728237567.py:2: FutureWarning: The
default value of regex will change from True to False in a future version.
  car['mileage']=car['mileage'].str.replace('kmpl|km/kg','')
```

In [68]:
```
car['mileage']=car['mileage'].astype('float')
car['engine']=car['engine'].str.replace('CC','')
car['engine']=car['engine'].astype('int64')
```

In [69]:
```
car['max_power']=car['max_power'].str.replace('bhp','')
car['max_power']=pd.to_numeric(car['max_power'],errors='coerce')
car['max_power']=car['max_power'].astype('float')
```

In [70]:
```
### AND EXCLUDING UNNECESSARY COLUMNS
car
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7907 entries, 0 to 8127
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   name           7907 non-null   object 
 1   year           7907 non-null   int64  
 2   selling_price  7907 non-null   int64  
 3   km_driven      7907 non-null   int64  
 4   fuel           7907 non-null   object 
 5   seller_type    7907 non-null   object 
 6   transmission   7907 non-null   object 
 7   mileage        7907 non-null   float64
 8   engine         7907 non-null   int64  
 9   max_power      7906 non-null   float64
 10  seats          7907 non-null   int32  
dtypes: float64(2), int32(1), int64(4), object(4)
memory usage: 710.4+ KB
```

In [71]:
```

car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7907 entries, 0 to 8127
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   name           7907 non-null   object 
 1   year           7907 non-null   int64  
 2   selling_price  7907 non-null   int64  
 3   km_driven      7907 non-null   int64  
 4   fuel           7907 non-null   object 
 5   seller_type    7907 non-null   object 
 6   transmission   7907 non-null   object 
 7   mileage        7907 non-null   float64
 8   engine         7907 non-null   int64  
 9   max_power      7906 non-null   float64
 10  seats          7907 non-null   int32  
dtypes: float64(2), int32(1), int64(4), object(4)
memory usage: 710.4+ KB
```

In [235]:
```
car.dropna(inplace=True)
car.isnull().sum()
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7906 entries, 0 to 8127
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   name           7906 non-null   object 
 1   year           7906 non-null   int64  
 2   selling_price  7906 non-null   int64  
 3   km_driven      7906 non-null   int64  
 4   fuel           7906 non-null   object 
 5   seller_type    7906 non-null   object 
 6   transmission   7906 non-null   object 
 7   mileage        7906 non-null   float64
 8   engine         7906 non-null   int64  
 9   max_power      7906 non-null   float64
 10  seats          7906 non-null   int32  
dtypes: float64(2), int32(1), int64(4), object(4)
memory usage: 968.3+ KB
```

In [236]:
```
car.describe(include='all')    # describing the car after cleaning the data
```

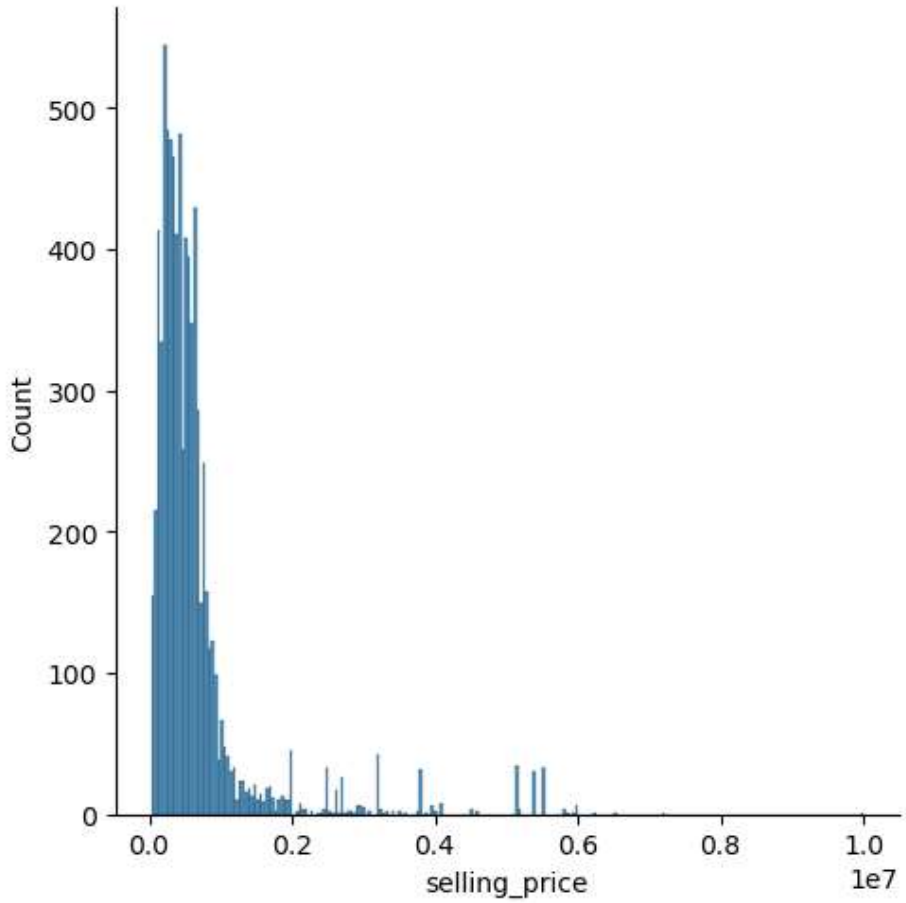
Out[236]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	mileage	
count	7906	7906.000000	7.906000e+03	7.906000e+03	7906	7906	7906	7906.000000	79
unique	1982	NaN	NaN	NaN	4	3	2	NaN	
top	Maruti Swift Dzire VDI	NaN	NaN	NaN	Diesel	Individual	Manual	NaN	
freq	129	NaN	NaN	NaN	4299	6563	6865	NaN	
mean	NaN	2013.983936	6.498137e+05	6.918866e+04	NaN	NaN	NaN	19.419861	14
std	NaN	3.863695	8.135827e+05	5.679230e+04	NaN	NaN	NaN	4.036263	5
min	NaN	1994.000000	2.999900e+04	1.000000e+00	NaN	NaN	NaN	0.000000	6
25%	NaN	2012.000000	2.700000e+05	3.500000e+04	NaN	NaN	NaN	16.780000	11
50%	NaN	2015.000000	4.500000e+05	6.000000e+04	NaN	NaN	NaN	19.300000	12
75%	NaN	2017.000000	6.900000e+05	9.542500e+04	NaN	NaN	NaN	22.320000	15
max	NaN	2020.000000	1.000000e+07	2.360457e+06	NaN	NaN	NaN	42.000000	36

Plotting graphs to understand the car details better

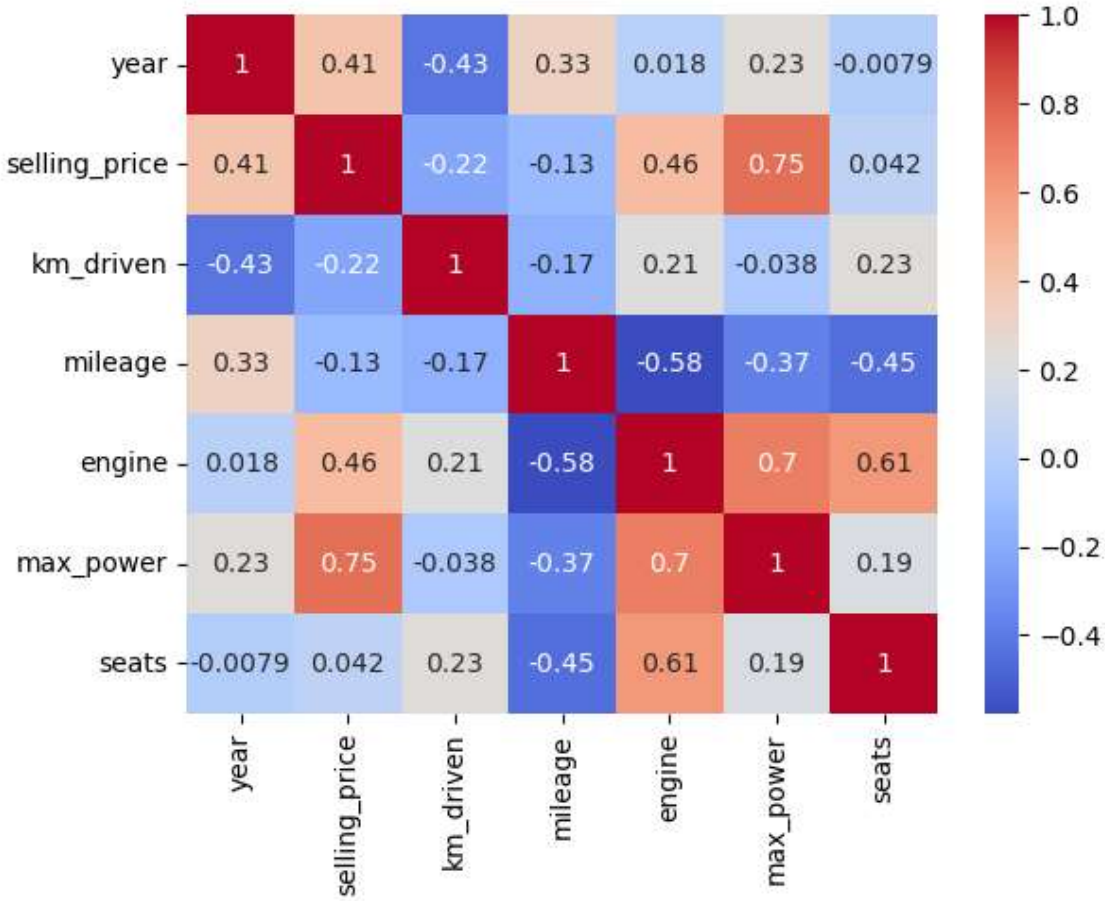
In [234]:
```
sns.displot(car['selling_price'])
```

Out[234]:
```
<seaborn.axisgrid.FacetGrid at 0x1ef1ce78c70>
```



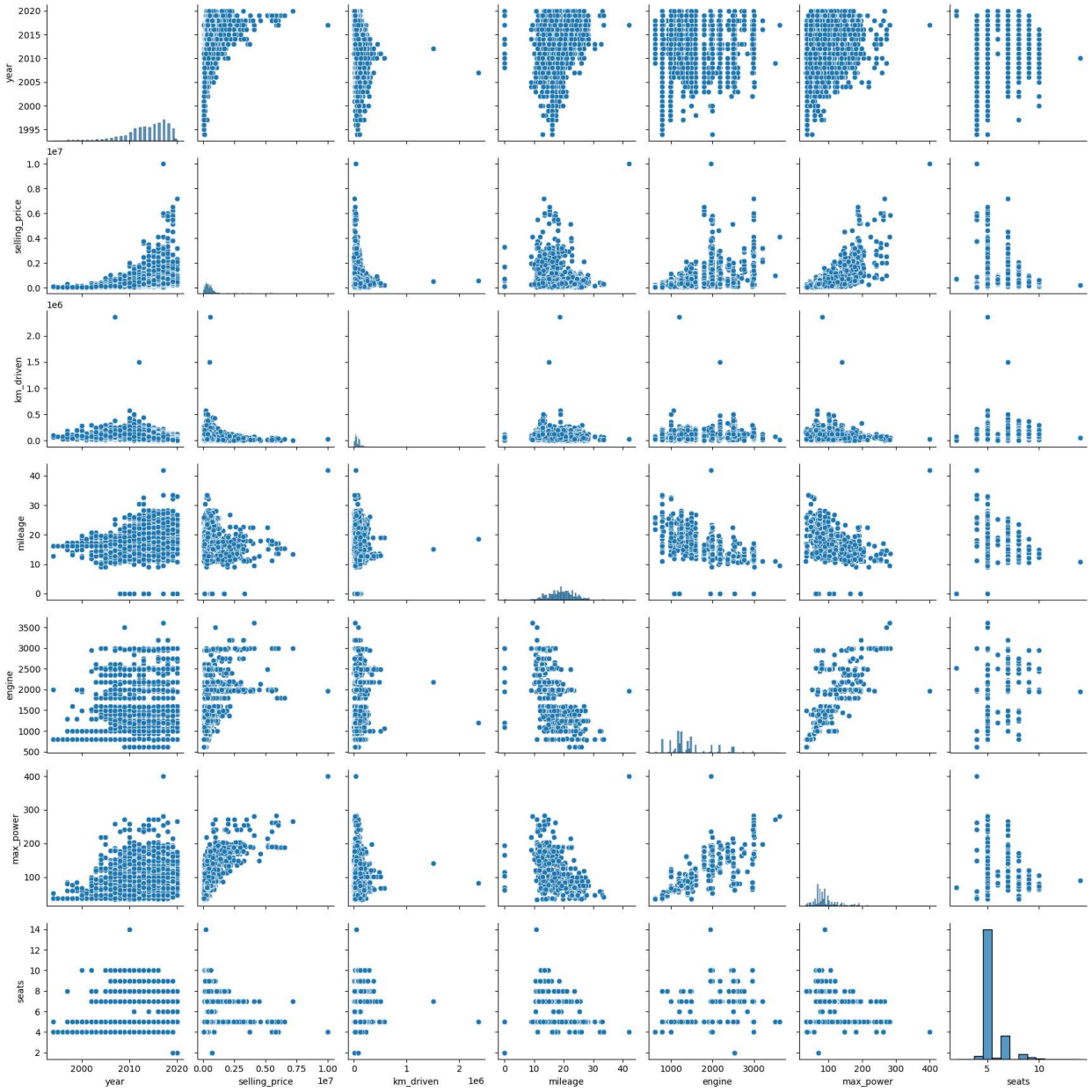
In [230]:
```
sns.heatmap(car.corr(),cmap='coolwarm',annot=True)
```

Out[230]: <AxesSubplot:>



In [73]:
```
sns.pairplot(car)
```

Out[73]:
```
<seaborn.axisgrid.PairGrid at 0x1ef17de7130>
```



step 4 : seprating dependent and independent variables into x and y

car.columns

In [493]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import mean_squared_error, r2_score
```

In [494]:
```
car_encoded = pd.get_dummies(car, columns=['fuel','seller_type','transmission','name'])
```

In [495]:
```
# car_encoded.drop(['name'],axis=1,inplace=True)  #  we can drop car names but we don't
```

In [496]:
```
x=car_encoded.drop(columns=['selling_price'],axis=1)

y=car_encoded['selling_price']
x
```

Out[496]:

	year	km_driven	mileage	engine	max_power	seats	fuel_CNG	fuel_Diesel	fuel_LPG	fuel_Petrol
0	2014	145500	23.40	1248	74.00	5	0	1	0	0
1	2014	120000	21.14	1498	103.52	5	0	1	0	0
2	2006	140000	17.70	1497	78.00	5	0	0	0	1
3	2010	127000	23.00	1396	90.00	5	0	1	0	0
4	2007	120000	16.10	1298	88.20	5	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...
8123	2013	110000	18.50	1197	82.85	5	0	0	0	1
8124	2007	119000	16.80	1493	110.00	5	0	1	0	0
8125	2009	120000	19.30	1248	73.90	5	0	1	0	0
8126	2013	25000	23.57	1396	70.00	5	0	1	0	0
8127	2013	25000	23.57	1396	70.00	5	0	1	0	0

7906 rows × 1997 columns

In [497]:
```
# split the training and test data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_state=86)
```

step5: Linear regression / Model training

In [498]:
```
regression =LinearRegression()
x_test
```

Out[498]:

	year	km_driven	mileage	engine	max_power	seats	fuel_CNG	fuel_Diesel	fuel_LPG	fuel_Petrol
2789	2009	80000	17.00	1405	70.00	5	0	1	0	0
4486	2019	7000	21.63	998	67.00	5	0	0	0	1
1777	2014	90000	23.20	1248	73.94	5	0	1	0	0
4458	2013	50000	15.64	1193	64.10	5	0	0	0	1
3687	2018	13000	17.00	2200	139.01	7	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...
151	2017	9000	13.60	1999	177.00	5	0	1	0	0
5738	2012	70000	19.70	796	46.30	5	0	0	0	1
2738	2012	95000	19.40	1405	70.00	5	0	1	0	0
3524	2016	56494	18.20	1199	88.70	5	0	0	0	1
979	2013	80000	22.07	1199	73.90	5	0	1	0	0

3163 rows × 1997 columns

In [499]:
```
reg_model=LinearRegression()
```

In [500]:
```
reg_model.fit(x_train,y_train)
```

Out[500]:
```
LinearRegression()
```

In [ ]:
```

```

In [501]:
```
price_predictions= reg_model.predict(x_train)
```

In [502]:
```
# R squared Error
```

In [503]:
```
error_score=metrics.r2_score(y_train,price_predictions)
print('R squared Error :',error_score)
```

```
R squared Error : 0.990078388727653
```

In [504]:
```
r2=r2_score(y_train,price_predictions)
r2
```

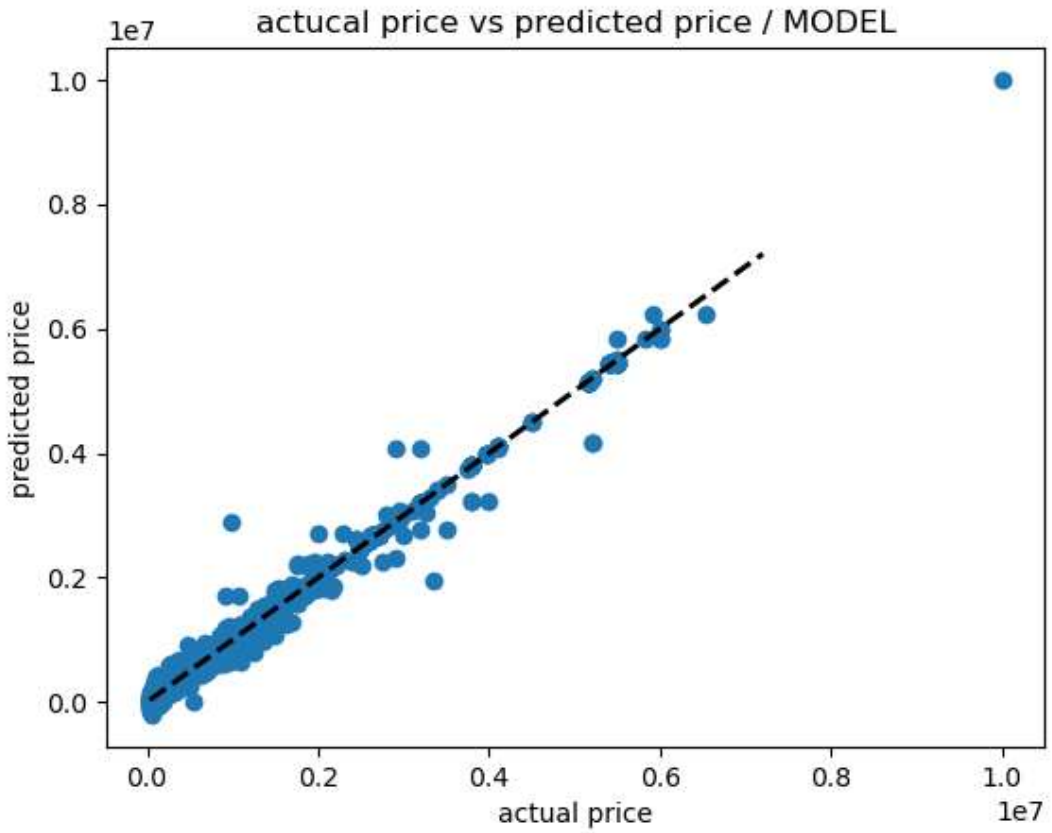
Out[504]:
```
0.990078388727653
```

In [ ]:
```

```

VISULIZATION OF ACTUCAL PRICE AND MODEL PREDICTED PRICE

In [509]:
```
plt.scatter(y_train,price_predictions)
plt.plot([min(y_test),max(y_test)],[min(y_test),max(y_test)],'k--',lw=2)
plt.xlabel('actual price')
plt.ylabel('predicted price')
plt.title('actucal price vs predicted price / MODEL')
plt.show()
```



STEP 6: Giving model car details and finding its car price

x train , y train and predicted price ..how it worked

In [463]:
```
x_train.head(2)
```

Out[463]:

	year	km_driven	mileage	engine	max_power	seats	fuel_CNG	fuel_Diesel	fuel_LPG	fuel_Petrol
7279	2016	60000	21.49	1498	108.5	5	0	1	0	0
2723	2010	100000	18.00	995	62.0	5	0	0	0	1

2 rows × 1997 columns

In [464]:
```
y_train.head()
```

Out[464]:
```
7279     591000
2723     140000
871     1050000
1871    5400000
618      135000
Name: selling_price, dtype: int64
```

In [427]:
```
pred= list(map(int,training_data_prediction))  # this was predicted price
```

In [428]:
```
pred
```

```
 788758,
 231218,
 400001,
 545274,
 535858,
 724568,
 600001,
 392238,
 444133,
 374994,
 476601,
 381025,
 350004,
 3200000,
 1177267,
 440996,
 2674615,
 638136,
 689780,
 432213.
```

In [429]:
```
y_train   # pred price should be close to y_train price ...
```

Out[429]:
```
7279     591000
2723     140000
871     1050000
1871    5400000
618      135000
         ...
8069     500000
1154     400000
3611    1650000
1944     200000
3322      90000
Name: selling_price, Length: 4743, dtype: int64
```

In [ ]:
```

```

In [459]:
```
new_car= x_test.sample()
i=new_car.index
new_car
```

Out[459]:

	year	km_driven	mileage	engine	max_power	seats	fuel_CNG	fuel_Diesel	fuel_LPG	fuel_Petrol
6959	2017	9000	13.6	1999	177.0	5	0	1	0	0

1 rows × 1997 columns

In [460]:
```
new_car_prediction= reg_model.predict(new_car)
```

In [461]:
```
price=list(map(int,new_car_prediction))
```

In [466]:
```
print('the model predicted selling price is', price ,'and actual is:',y_test.loc[i].va
actual_price=y_test.loc[i].values
```

```
the model predicted selling price is [2688066] and actual is: [2711000]
```