Car Price Prediction - Sriharsha Velicheti

Reading The Data set

In [1]:
```
df = pd.read_csv("C:/Users/sriharsha/Desktop/data science/data sets/Car dekho car
df
```

```
<IPython.core.display.Javascript object>
```

Out[1]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
...	...	...	...	...	...	...	...	...
4335	Hyundai i20 Magna 1.4 CRDi (Diesel)	2014	409999	80000	Diesel	Individual	Manual	Second Owner
4336	Hyundai i20 Magna 1.4 CRDi	2014	409999	80000	Diesel	Individual	Manual	Second Owner
4337	Maruti 800 AC BSIII	2009	110000	83000	Petrol	Individual	Manual	Second Owner
4338	Hyundai Creta 1.6 CRDi SX Option	2016	865000	90000	Diesel	Individual	Manual	First Owner
4339	Renault KWID RXT	2016	225000	40000	Petrol	Individual	Manual	First Owner

4340 rows × 8 columns

In [2]:
```
df.head()
```

Out[2]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

In [ ]:
```
```

In [3]:
```
df.info()#No null values
```

```
<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
```

In [4]:
```
df.year.unique()
```

Out[4]:
```
array([2007, 2012, 2017, 2014, 2016, 2015, 2018, 2019, 2013, 2011, 2010,
       2009, 2006, 1996, 2005, 2008, 2004, 1998, 2003, 2002, 2020, 2000,
       1999, 2001, 1995, 1997, 1992], dtype=int64)
```

There are no null values in the dataset

In [5]:
```
pd.crosstab(index=df['fuel'],columns=df['seller_type'],margins = True)
```

```
<IPython.core.display.Javascript object>
```

Out[5]:

seller_type fuel	Dealer	Individual	Trustmark Dealer	All
CNG	9	31	0	40
Diesel	529	1576	48	2153
Electric	1	0	0	1
LPG	1	22	0	23
Petrol	454	1615	54	2123
All	994	3244	102	4340

inferences from the crosstab

1. There is only 1 electric vehicle in the entire data set
2. There 63 vehciles that are running on gas
3. Most of the vehicles are running with Petrol and diesel
4. Most of the sellers are individual in the car dekho platform and there are very few Trustmark Dealers

In [6]:
```
pd.crosstab(index=df['fuel'],columns=df['seller_type']).plot(kind='bar',figsize=(
```

```
<IPython.core.display.Javascript object>
```

Out[6]:

```
<AxesSubplot:xlabel='fuel'>
```

In [7]:
```
pd.crosstab(index=df['fuel'],columns=df['owner'],margins = True,normalize=True)
```

```
<IPython.core.display.Javascript object>
```

Out[7]:

owner	First Owner	Fourth & Above Owner	Second Owner	Test Drive Car	Third Owner	All
fuel						
CNG	0.004839	0.000691	0.003226	0.000000	0.000461	0.009217
Diesel	0.323502	0.007834	0.127189	0.001613	0.035945	0.496083
Electric	0.000000	0.000000	0.000230	0.000000	0.000000	0.000230
LPG	0.002304	0.000230	0.002304	0.000000	0.000461	0.005300
Petrol	0.321889	0.009908	0.121889	0.002304	0.033180	0.489171
All	0.652535	0.018664	0.254839	0.003917	0.070046	1.000000

inferences

1. The Platform Car Dekho is Famous place for first hand and second hand sellers and buyers
2. There are very few who are trading a new car or a car that is traded more than 4 times

In [8]:
```
pd.crosstab(index=df['year'],columns=df['fuel'],margins = True)
```

```
<IPython.core.display.Javascript object>
```

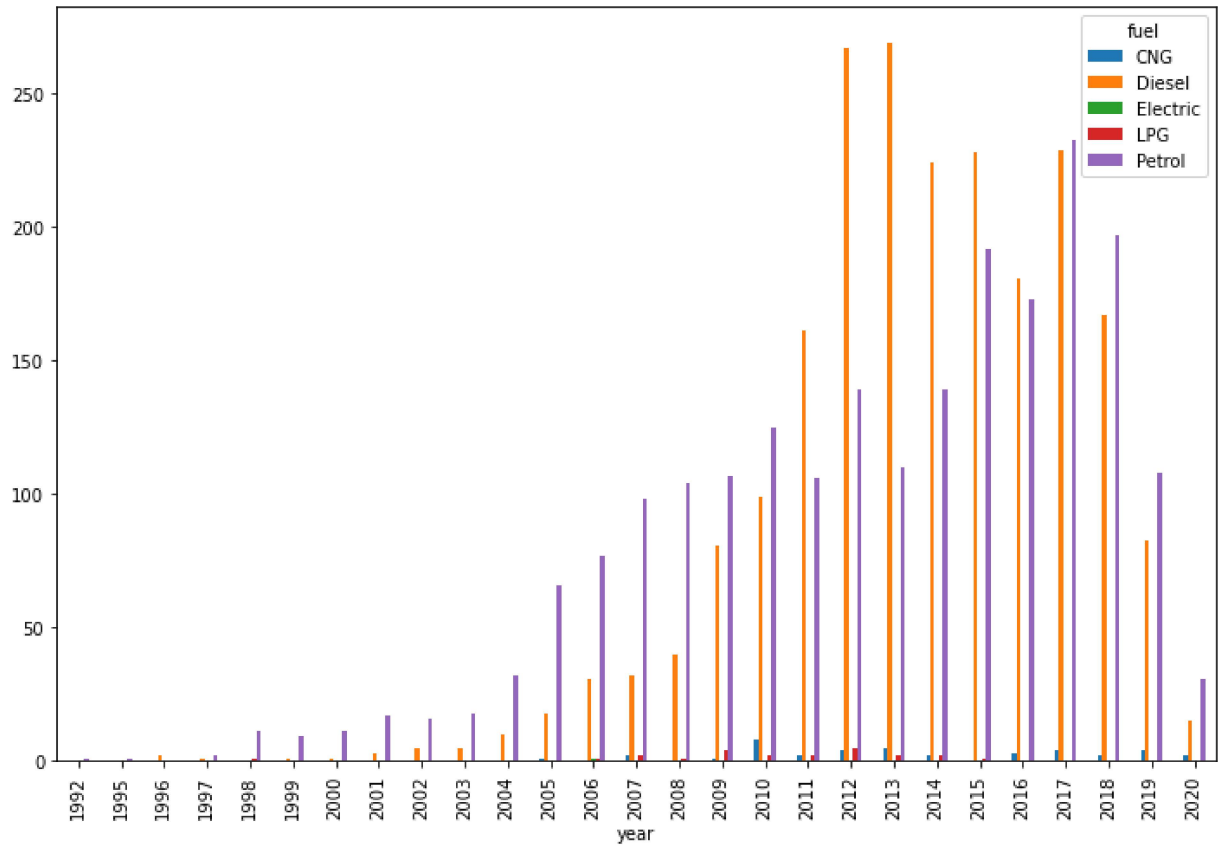
Out[8]:

fuel / year	CNG	Diesel	Electric	LPG	Petrol	All
1992	0	0	0	0	1	1
1995	0	0	0	0	1	1
1996	0	2	0	0	0	2
1997	0	1	0	0	2	3
1998	0	0	0	1	11	12
1999	0	1	0	0	9	10
2000	0	1	0	0	11	12
2001	0	3	0	0	17	20
2002	0	5	0	0	16	21
2003	0	5	0	0	18	23
2004	0	10	0	0	32	42
2005	1	18	0	0	66	85
2006	0	31	1	1	77	110
2007	2	32	0	2	98	134
2008	0	40	0	1	104	145
2009	1	81	0	4	107	193
2010	8	99	0	2	125	234
2011	2	161	0	2	106	271
2012	4	267	0	5	139	415
2013	5	269	0	2	110	386
2014	2	224	0	2	139	367
2015	0	228	0	1	192	421
2016	3	181	0	0	173	357
2017	4	229	0	0	233	466
2018	2	167	0	0	197	366
2019	4	83	0	0	108	195
2020	2	15	0	0	31	48
All	40	2153	1	23	2123	4340

In [9]:
```
pd.crosstab(index=df['year'],columns=df['fuel']).plot(kind='bar',figsize = (12,8)
```

```
<IPython.core.display.Javascript object>
```

Out[9]: <AxesSubplot:xlabel='year'>



inferences:

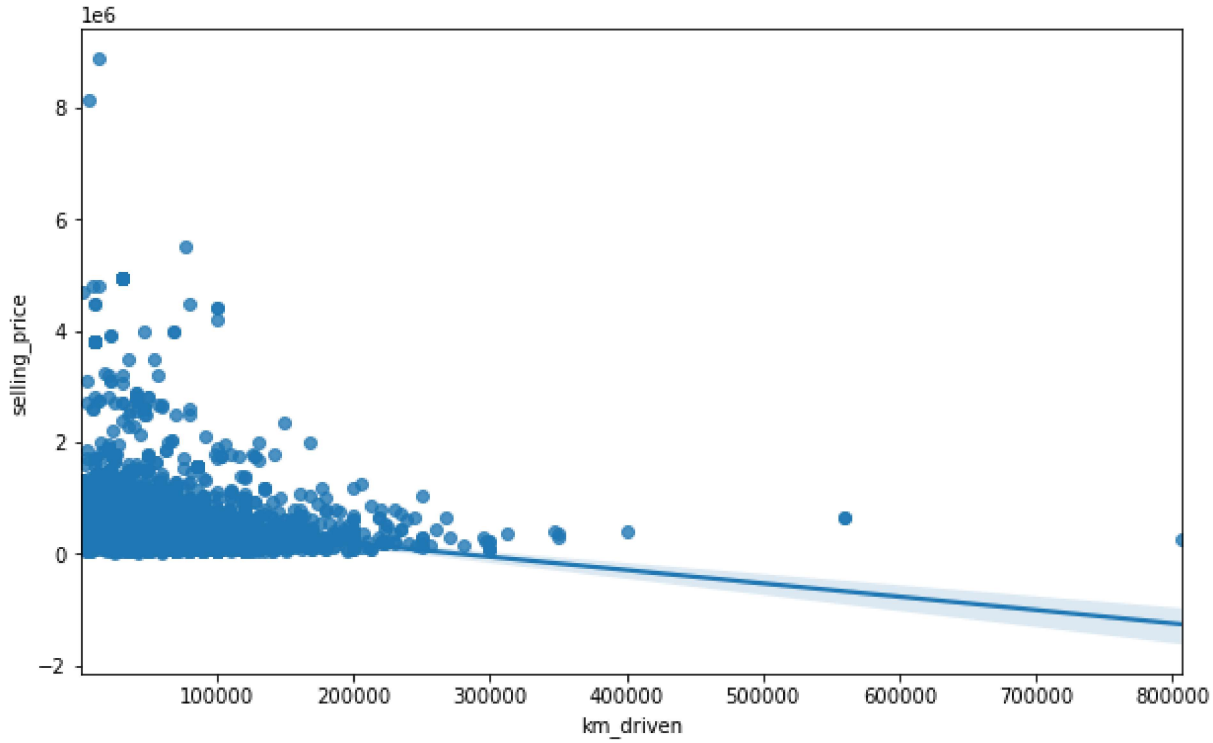
1. Till 1997 the number of cars traded were in single digit
2. from 1995 to 2010 the no. of petrol cars were more in number when compared with diesel cars or anyother
3. Gas (CNG/LPG) were started to raise from the year 2005.
4. Diesel cars were quit dominating between the years 2011-2016

In [10]:
```
plt.figure(figsize=(10,6))
sns.regplot(x=df['km_driven'],y = df['selling_price'],marker='o')
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>
```

Out[10]: <AxesSubplot:xlabel='km_driven', ylabel='selling_price'>



**** As The kilometers increases the selling price is decreasing**

In [11]:
```
pd.crosstab(index=df['transmission'],columns=df['fuel'],margins = True)
```

```
<IPython.core.display.Javascript object>
```

Out[11]:

transmission	CNG	Diesel	Electric	LPG	Petrol	All
Automatic	0	254	1	0	193	448
Manual	40	1899	0	23	1930	3892
All	40	2153	1	23	2123	4340

In [ ]:
```

```

Making every categorical object into Numerical(int) data

In [12]:
```
df.owner.value_counts()
```

Out[12]:
```
First Owner             2832
Second Owner            1106
Third Owner              304
Fourth & Above Owner      81
Test Drive Car            17
Name: owner, dtype: int64
```

In [13]:
```
df.owner.replace(['First Owner', 'Second Owner', 'Fourth & Above Owner','Third Ow
```

In [14]:
```
df.transmission.value_counts()
```

Out[14]:
```
Manual       3892
Automatic     448
Name: transmission, dtype: int64
```

In [15]:
```
df.transmission.replace(['Manual','Automatic'],[0,1],inplace=True)
```

In [16]:
```
df.seller_type.unique()
```

Out[16]:
```
array(['Individual', 'Dealer', 'Trustmark Dealer'], dtype=object)
```

In [17]:
```
df.seller_type.replace(['Individual', 'Dealer', 'Trustmark Dealer'],[1,2,3],inpla
```

In [18]:
```
df.fuel.unique()
```

Out[18]:
```
array(['Petrol', 'Diesel', 'CNG', 'LPG', 'Electric'], dtype=object)
```

In [19]:
```
df.fuel.replace(['Petrol', 'Diesel', 'CNG', 'LPG', 'Electric'],[1,2,3,4,5],inplac
```

In [20]:
```
df.fuel.replace([1,2,3,4,5],[1,2,3,3,5],inplace = True)
```

In [21]:
```
df.drop(df[df['fuel']==5].index,inplace=True)
```

In [22]:
```
df.corr()
```

Out[22]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner
year	1.000000	0.413932	-0.419861	0.103796	0.183127	0.145136	-0.468563
selling_price	0.413932	1.000000	-0.192298	0.242688	0.240841	0.530972	-0.228050
km_driven	-0.419861	-0.192298	1.000000	0.281301	-0.187147	-0.120285	0.321887
fuel	0.103796	0.242688	0.281301	1.000000	0.014325	0.027616	0.017522
seller_type	0.183127	0.240841	-0.187147	0.014325	1.000000	0.199450	-0.259875
transmission	0.145136	0.530972	-0.120285	0.027616	0.199450	1.000000	-0.083843
owner	-0.468563	-0.228050	0.321887	0.017522	-0.259875	-0.083843	1.000000

In [ ]:
```

```

In [23]:
```
plt.figure(figsize=(12,8))
plt.pie(df['transmission'].value_counts(),labels=['Manual','Automatic'],explode=[
plt.legend()
plt.title("Mode of Transmission of Vehicle")
```

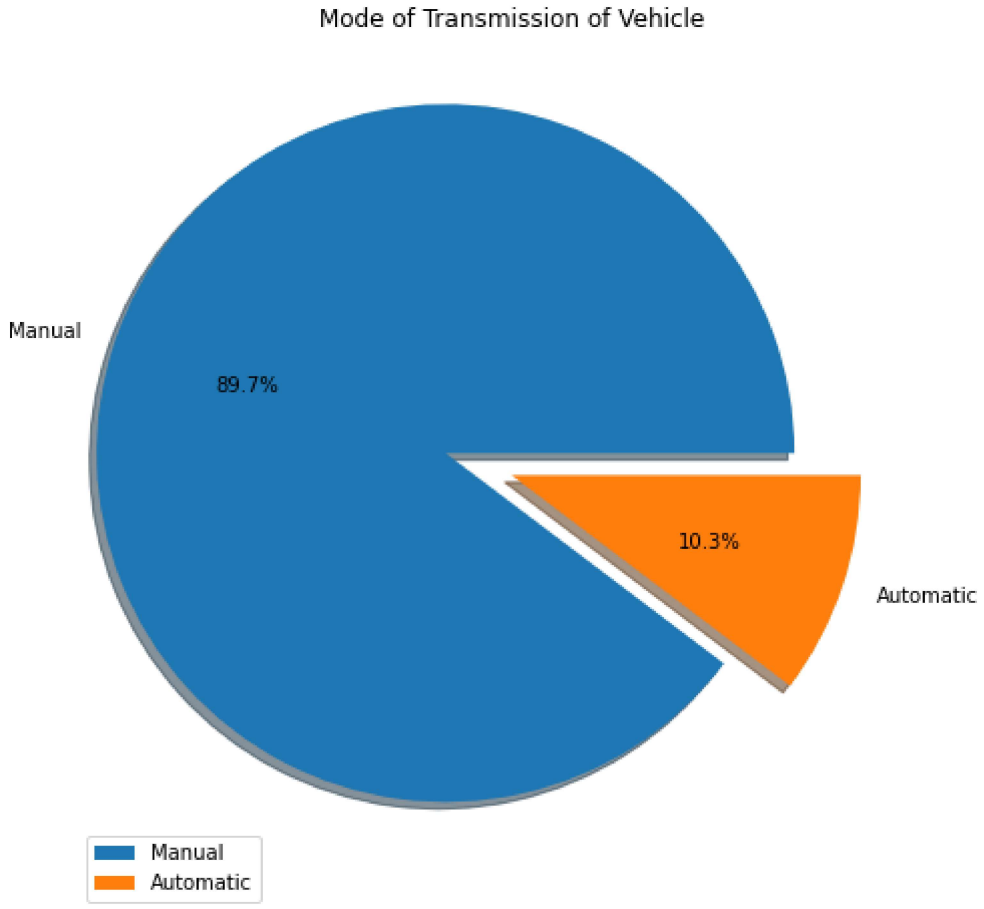
```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>
```

Out[23]:
```
Text(0.5, 1.0, 'Mode of Transmission of Vehicle')
```



In [24]:
```
df.columns
```

Out[24]:
```
Index(['name', 'year', 'selling_price', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner'],
      dtype='object')
```

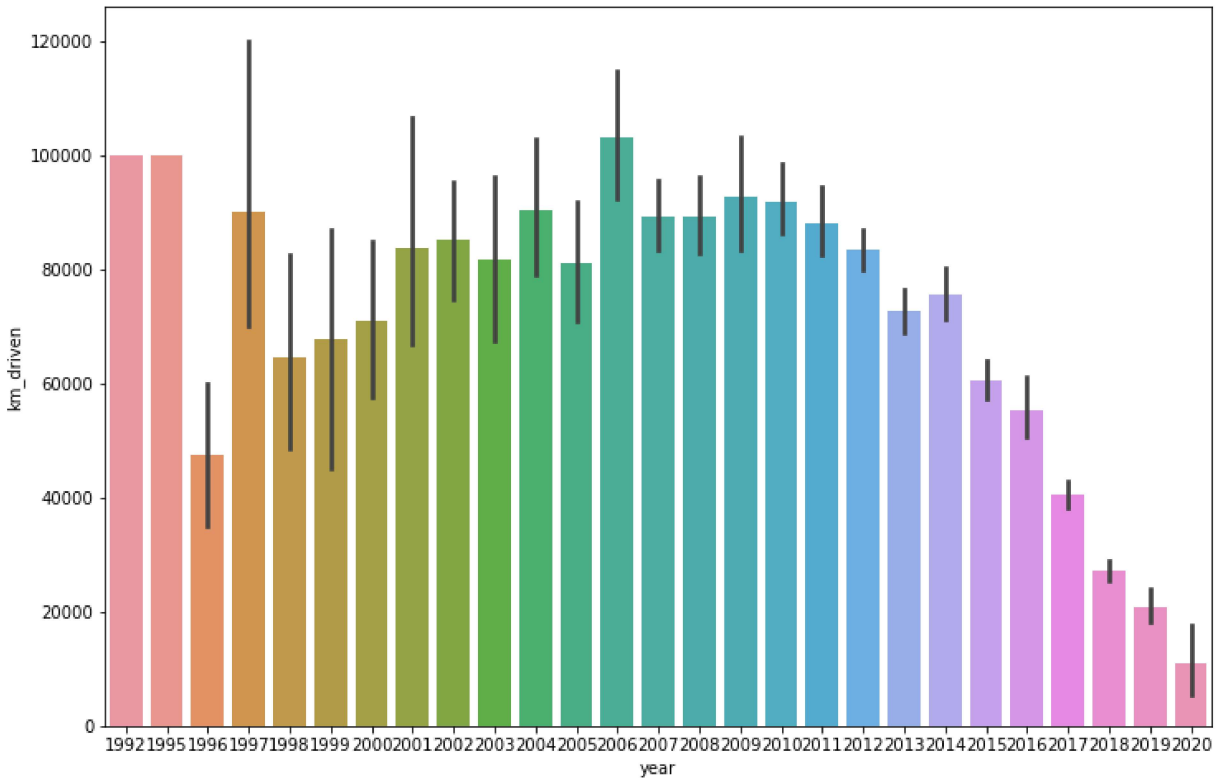
Outlier Detection And Treatment

In [25]:
```
plt.figure(figsize=(12,8))
sns.barplot(x = df['year'],y = df['km_driven'])
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>
```

Out[25]:
```
<AxesSubplot:xlabel='year', ylabel='km_driven'>
```



In [26]:
```
df.drop(df[df['year']<=1995].index,inplace=True)
```

In [27]:
```
df.corr()
```

Out[27]:

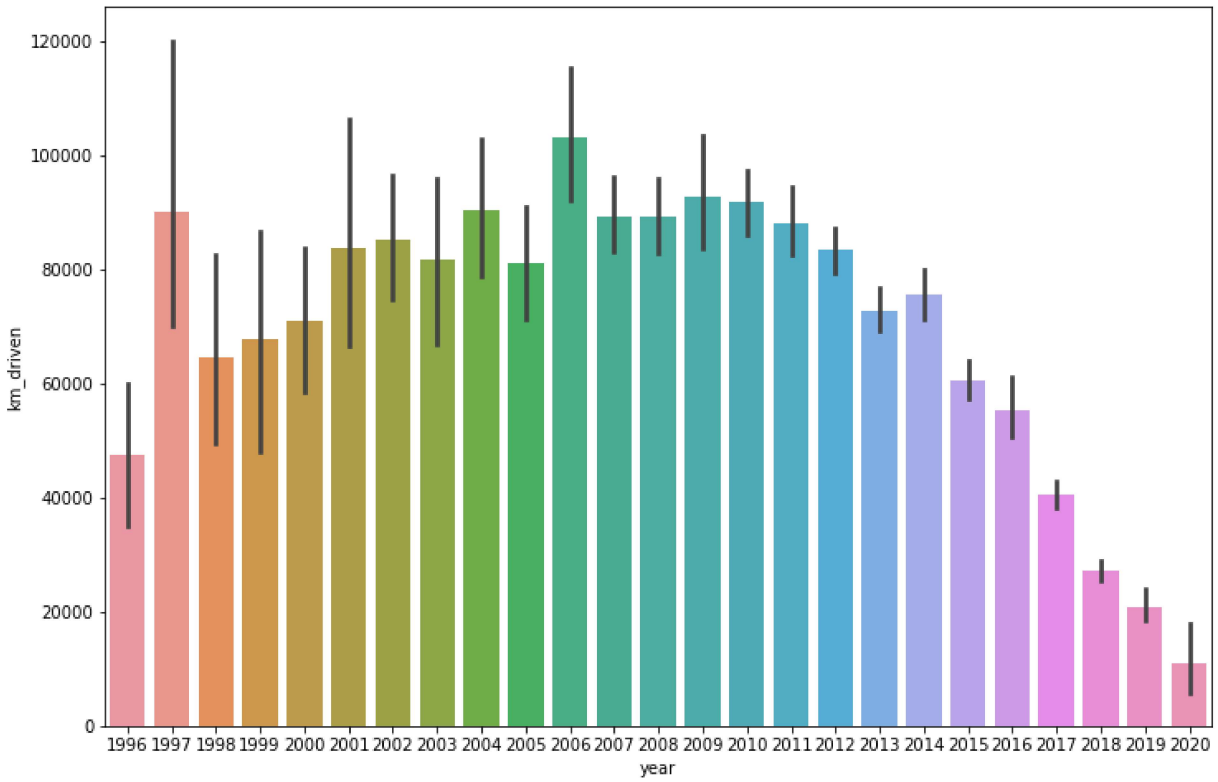
	year	selling_price	km_driven	fuel	seller_type	transmission	owner
year	1.000000	0.414462	-0.420472	0.102199	0.182869	0.145143	-0.466723
selling_price	0.414462	1.000000	-0.192097	0.242432	0.240698	0.530938	-0.227657
km_driven	-0.420472	-0.192097	1.000000	0.281731	-0.186997	-0.120190	0.321697
fuel	0.102199	0.242432	0.281731	1.000000	0.014074	0.027468	0.018556
seller_type	0.182869	0.240698	-0.186997	0.014074	1.000000	0.199382	-0.259738
transmission	0.145143	0.530938	-0.120190	0.027468	0.199382	1.000000	-0.083634
owner	-0.466723	-0.227657	0.321697	0.018556	-0.259738	-0.083634	1.000000

In [28]:
```
plt.figure(figsize=(12,8))
sns.barplot(x = df['year'],y = df['km_driven'])
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>
```

Out[28]:
```
<AxesSubplot:xlabel='year', ylabel='km_driven'>
```

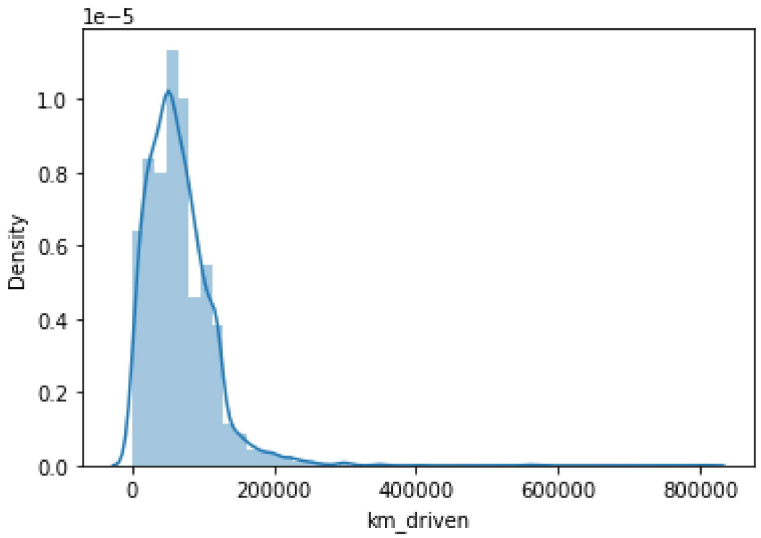


In [29]:
```
sns.distplot(df['km_driven'])#Not Normalized data
```

```
<IPython.core.display.Javascript object>
```

```
C:\Users\sriharsha\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[29]: <AxesSubplot:xlabel='km_driven', ylabel='Density'>

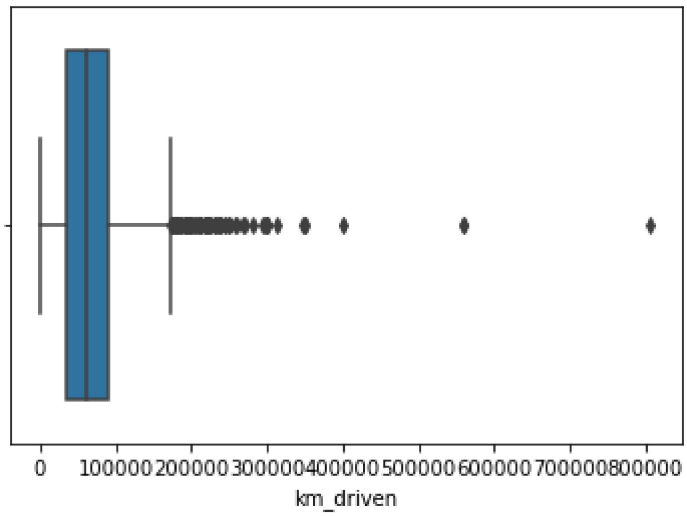


In [30]:
```
#Since there is normalized distribution check for outliers
sns.boxplot(df['km_driven'])
#There are some outliers on upper side from box plot which is more than max limit
```

```
<IPython.core.display.Javascript object>

C:\Users\sriharsha\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[30]: <AxesSubplot:xlabel='km_driven'>



In [31]:
```
df.describe()
```

Out[31]:

	year	selling_price	km_driven	fuel	seller_type	transmission	ov
count	4337.000000	4.337000e+03	4337.000000	4337.000000	4337.000000	4337.000000	4337.000
mean	2013.101453	5.043711e+05	66201.169933	1.525478	1.275997	0.103067	1.446
std	4.194239	5.786668e+05	46654.547491	0.527700	0.496907	0.304081	0.71
min	1996.000000	2.000000e+04	1.000000	1.000000	1.000000	0.000000	0.000
25%	2011.000000	2.100000e+05	35000.000000	1.000000	1.000000	0.000000	1.000
50%	2014.000000	3.500000e+05	60000.000000	2.000000	1.000000	0.000000	1.000
75%	2016.000000	6.000000e+05	90000.000000	2.000000	2.000000	0.000000	2.000
max	2020.000000	8.900000e+06	806599.000000	3.000000	3.000000	1.000000	4.000

In [32]:
```
df.km_driven.quantile(0.95)
```

Out[32]:
```
140000.0
```

In [33]:
```
(df.km_driven>140000).sum()
```

Out[33]:
```
208
```

In [34]:
```
df[df.km_driven>140000].index
```

Out[34]:
```
Int64Index([   4,   17,   32,   69,   70,   75,  159,  166,  196,  197,
            ...
            4231, 4251, 4255, 4271, 4275, 4286, 4314, 4326, 4331, 4334],
           dtype='int64', length=208)
```

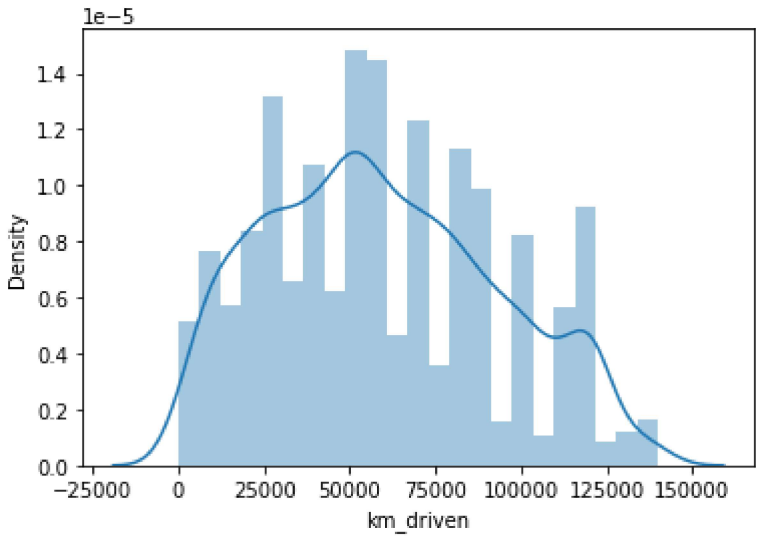
In [35]:
```
df.drop(df[df.km_driven>140000].index,inplace = True)
```

In [36]:
```
sns.distplot(df['km_driven'])#Now distribution is Normalized in a better way
```

```
<IPython.core.display.Javascript object>

C:\Users\sriharsha\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for histog
rams).
  warnings.warn(msg, FutureWarning)
```

Out[36]:
```
<AxesSubplot:xlabel='km_driven', ylabel='Density'>
```



In [37]:
```
df.corr()
```

Out[37]:

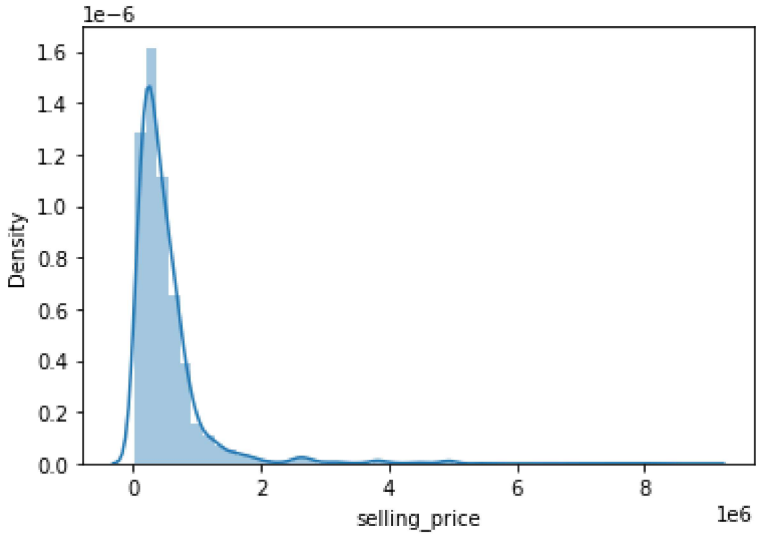
	year	selling_price	km_driven	fuel	seller_type	transmission	owner
year	1.000000	0.413258	-0.491067	0.122520	0.175248	0.145925	-0.467883
selling_price	0.413258	1.000000	-0.234899	0.254106	0.237313	0.534351	-0.228926
km_driven	-0.491067	-0.234899	1.000000	0.269267	-0.218493	-0.130611	0.373943
fuel	0.122520	0.254106	0.269267	1.000000	0.025767	0.034684	-0.001563
seller_type	0.175248	0.237313	-0.218493	0.025767	1.000000	0.199553	-0.256243
transmission	0.145925	0.534351	-0.130611	0.034684	0.199553	1.000000	-0.084641
owner	-0.467883	-0.228926	0.373943	-0.001563	-0.256243	-0.084641	1.000000

In [38]:
```
sns.distplot(df['selling_price'])#right skewed data not normalized
```

```
<IPython.core.display.Javascript object>

C:\Users\sriharsha\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for histog
rams).
  warnings.warn(msg, FutureWarning)
```

Out[38]:
```
<AxesSubplot:xlabel='selling_price', ylabel='Density'>
```

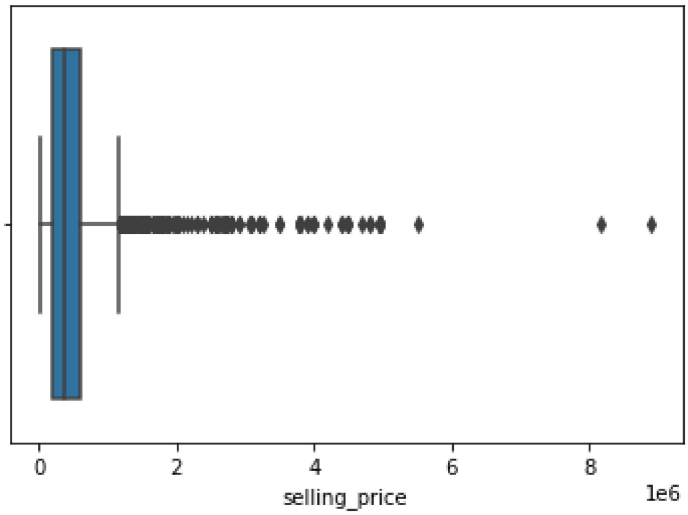


In [39]:
```
sns.boxplot(df['selling_price'])
```

```
<IPython.core.display.Javascript object>

C:\Users\sriharsha\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[39]: <AxesSubplot:xlabel='selling_price'>



In [40]:
```
df.selling_price.quantile(0.95)
```

Out[40]: 1300000.0

In [41]:
```
(df['selling_price']>1300000.0).sum()
```

Out[41]: 201

In [42]:
```
df[df['selling_price']>1300000.0].index
```

Out[42]:
```
Int64Index([  12,   25,   29,   30,   35,   36,   40,   43,   89,   94,
            ...
            4047, 4059, 4163, 4170, 4186, 4204, 4224, 4304, 4311, 4313],
           dtype='int64', length=201)
```

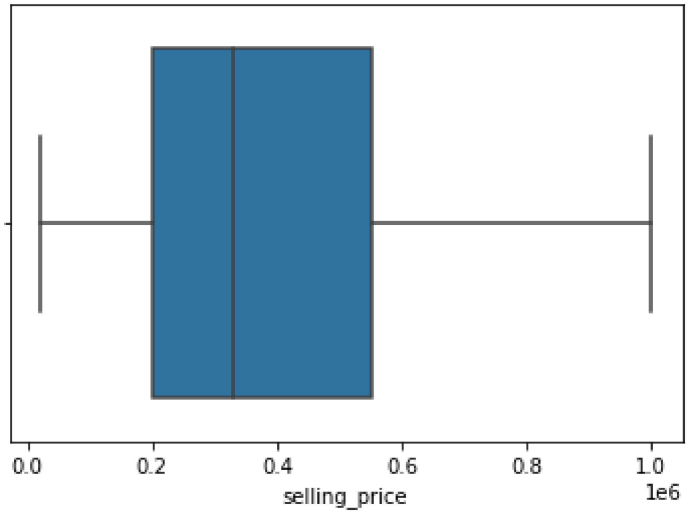
In [43]:
```
df.drop(df[df['selling_price']>1000000.0].index,inplace = True)
```

In [44]:
```
sns.boxplot(df['selling_price'])
```

```
<IPython.core.display.Javascript object>

C:\Users\sriharsha\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[44]:
```
<AxesSubplot:xlabel='selling_price'>
```



In [45]:
```
df.shape
```

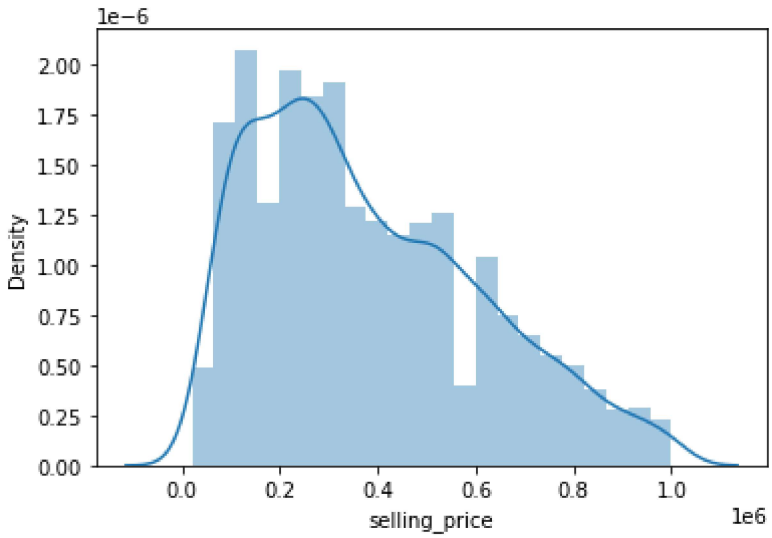
Out[45]:
```
(3796, 8)
```

In [46]:
```
sns.distplot(df['selling_price'])
```

```
<IPython.core.display.Javascript object>

C:\Users\sriharsha\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for histog
rams).
  warnings.warn(msg, FutureWarning)
```

Out[46]: <AxesSubplot:xlabel='selling_price', ylabel='Density'>



In [47]:
```
df.describe()
```

Out[47]:

	year	selling_price	km_driven	fuel	seller_type	transmission	
count	3796.000000	3796.000000	3796.000000	3796.000000	3796.000000	3796.000000	3796.0
mean	2012.943625	381876.594046	61043.067439	1.475764	1.260801	0.060590	1.4
std	4.212846	233018.217208	33460.839584	0.530188	0.487473	0.238608	0.7
min	1996.000000	20000.000000	1.000000	1.000000	1.000000	0.000000	0.0
25%	2010.000000	199750.000000	35000.000000	1.000000	1.000000	0.000000	1.0
50%	2014.000000	330000.000000	60000.000000	1.000000	1.000000	0.000000	1.0
75%	2016.000000	550000.000000	85000.000000	2.000000	1.000000	0.000000	2.0
max	2020.000000	1000000.000000	140000.000000	3.000000	3.000000	1.000000	4.0

In [ ]:

In [48]:
```
from sklearn.preprocessing import Normalizer
mms = Normalizer()
```

In [49]:
```
#kk = mms.fit(df[['selling_price']])
```

In [50]:
```
#df[['selling_price']]=kk.transform(df[['selling_price']])
```

In [ ]:

In [ ]:

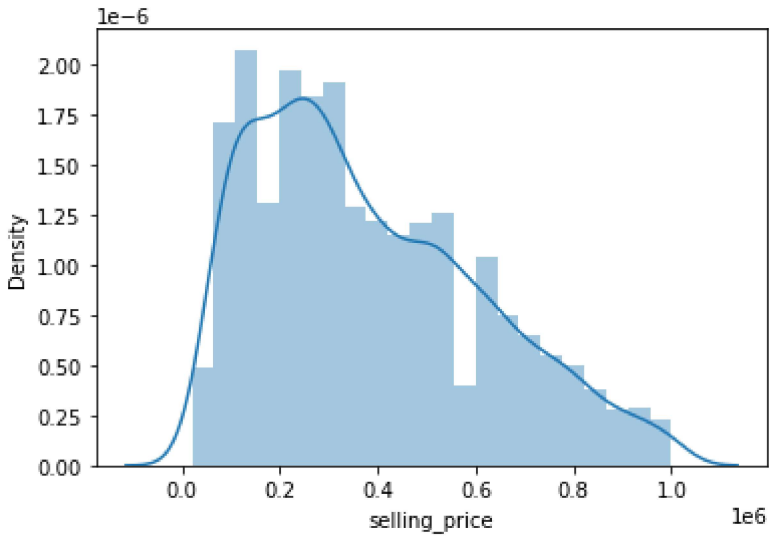
Target variable-----> Selling Price

In [51]:
```
sns.distplot(df['selling_price'])
```

```
<IPython.core.display.Javascript object>

C:\Users\sriharsha\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for histog
rams).
  warnings.warn(msg, FutureWarning)
```

Out[51]: <AxesSubplot:xlabel='selling_price', ylabel='Density'>



In [52]:
```
df.corr()
```

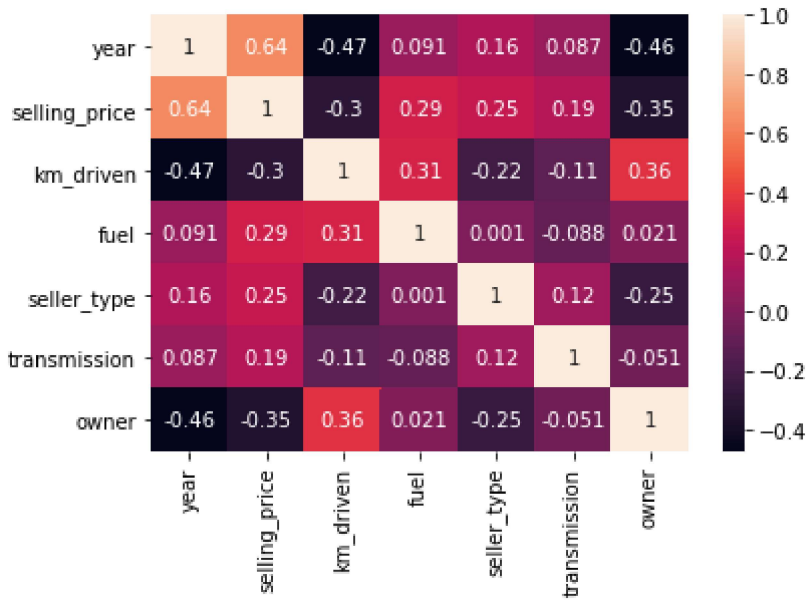
Out[52]:

	year	selling_price	km_driven	fuel	seller_type	transmission	owner
year	1.000000	0.644672	-0.472388	0.090581	0.164085	0.087020	-0.456525
selling_price	0.644672	1.000000	-0.304477	0.287646	0.250783	0.185398	-0.348208
km_driven	-0.472388	-0.304477	1.000000	0.310009	-0.222436	-0.114916	0.363629
fuel	0.090581	0.287646	0.310009	1.000000	0.001013	-0.088369	0.021207
seller_type	0.164085	0.250783	-0.222436	0.001013	1.000000	0.115573	-0.251082
transmission	0.087020	0.185398	-0.114916	-0.088369	0.115573	1.000000	-0.051477
owner	-0.456525	-0.348208	0.363629	0.021207	-0.251082	-0.051477	1.000000

In [53]:
```
sns.heatmap(df.corr(),annot=True)
```

```
<IPython.core.display.Javascript object>
```

Out[53]:
```
<AxesSubplot:>
```



In [54]:
```
df.columns
```

Out[54]:
```
Index(['name', 'year', 'selling_price', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner'],
      dtype='object')
```

In [55]:
```
x= df[['year', 'selling_price', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner']]
y = df['selling_price']
```

Train Test Split

In [56]:
```
from sklearn.model_selection import train_test_split
trainx,testx,trainy,testy = train_test_split(x,y,test_size=0.20)
```

Linear Regressor

In [57]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression(normalize=True)
```

In [58]:
```
lr.fit(trainx,trainy)
```

Out[58]:
```
LinearRegression(normalize=True)
```

In [59]:
```
yp = lr.predict(testx)
```

In [60]:
```
from sklearn.metrics import r2_score

r2_score(testy,yp)
```

Out[60]:
```
1.0
```

In [61]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(testy,yp)
mae
```

Out[61]:
```
1.3165653830296115e-10
```

In [62]:
```
100-mae*100
```

Out[62]:
```
99.99999998683434
```

Random Forest regrssor

In [63]:
```
from sklearn.ensemble import RandomForestRegressor
rgt = RandomForestRegressor(random_state=2,n_jobs=10)
```

In [64]:
```
rgt.fit(trainx,trainy)
```

Out[64]:
```
RandomForestRegressor(n_jobs=10, random_state=2)
```

In [65]:
```
yprgt = rgt.predict(testx)
```

In [66]:
```
ma = mean_absolute_error(testy,yprgt)
ma
```

Out[66]: 109.81268421052609

In [67]:
```
100-ma*100 #accuracy
```

Out[67]: -10881.26842105261

In [68]:
```
r2_score(testy,yprgt)
```

Out[68]: 0.9999960743419279

In [ ]:

Decision Tree Regressor

In [69]:
```
from sklearn.tree import  DecisionTreeRegressor
```

In [70]:
```
bb = DecisionTreeRegressor(random_state=2)
```

In [71]:
```
bb.fit(trainx,trainy)
```

Out[71]: DecisionTreeRegressor(random_state=2)

In [72]:
```
ypb = bb.predict(testx)
```

In [73]:
```
r2_score(testy,ypb)
```

Out[73]: 0.9999938084617163

In [74]:
```
m = mean_absolute_error(testy,ypb)
```

In [75]:
```
100-m*100
```

Out[75]: -7978.815789473684

In [ ]:

In [ ]: