In [39]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn import metrics

import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```python
data = pd.read_excel('CAR_DETAILS_FROM_CAR_DEKHO.csv.xlsx')
```

In [4]:
```python
data.head()
```

Out[4]:

	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 [5]:
```python
data.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
```

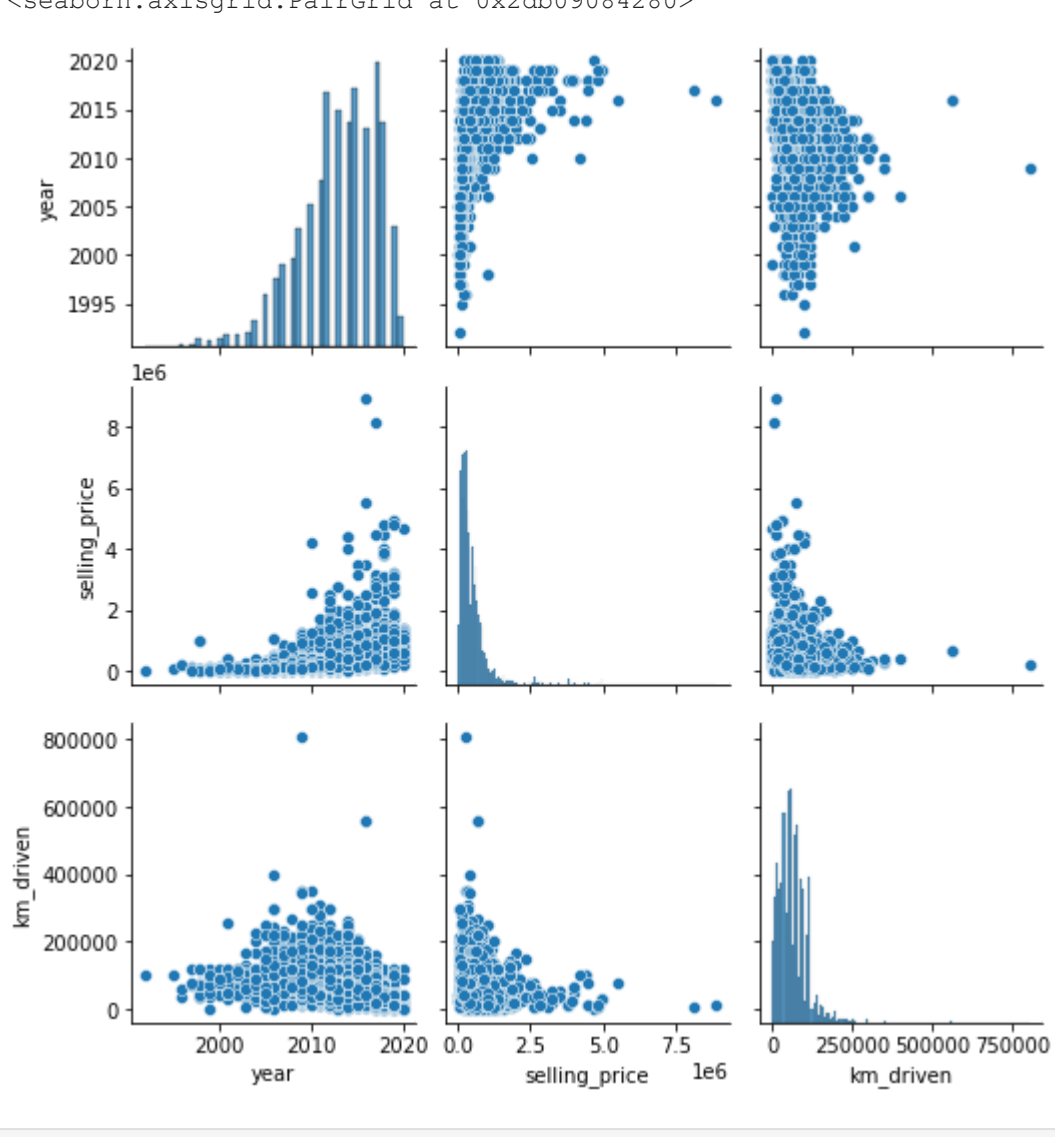
In [6]:
```python
data.describe()
```

Out[6]:

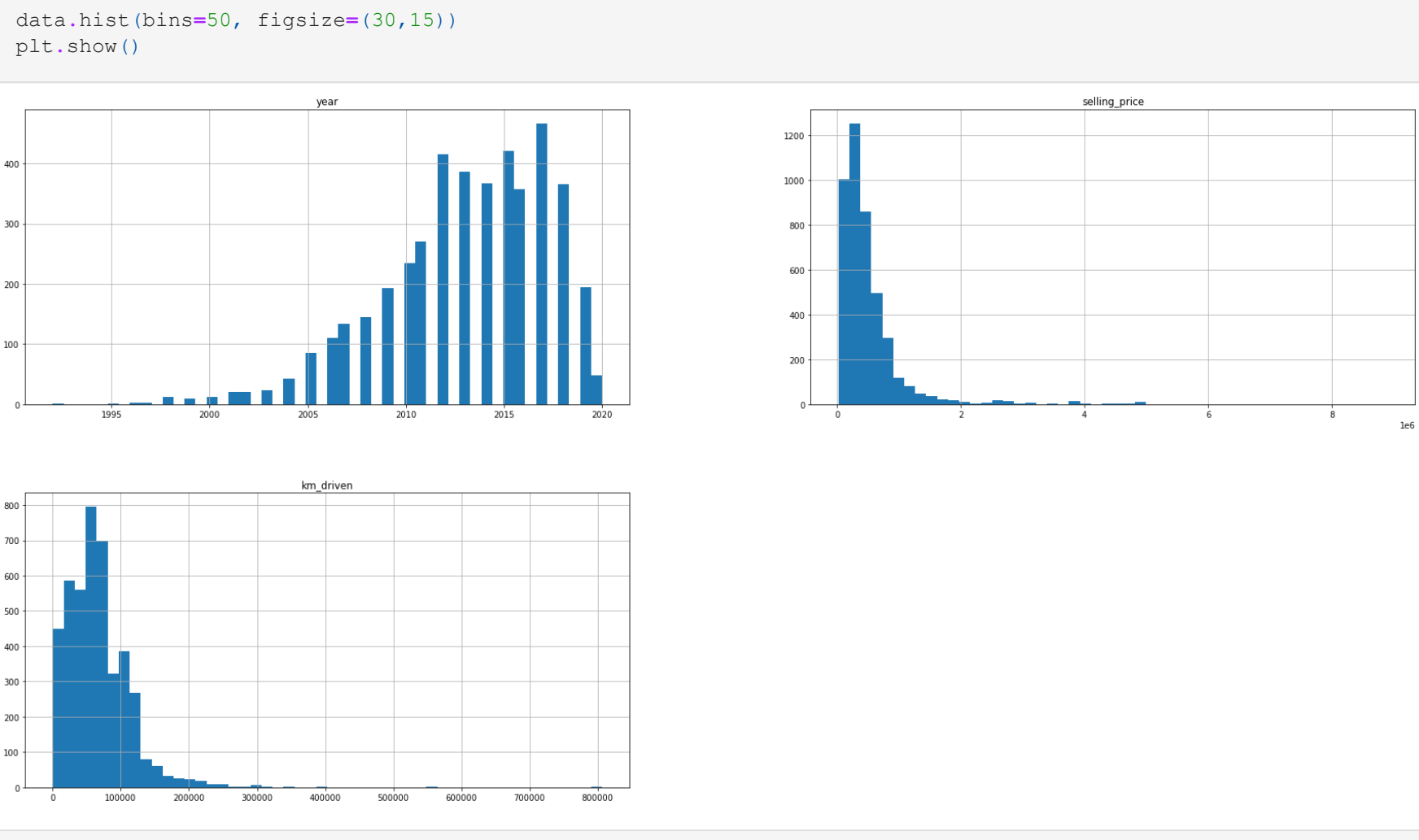
	year	selling_price	km_driven
count	4340.000000	4.340000e+03	4340.000000
mean	2013.090783	5.041273e+05	66215.777419
std	4.215344	5.785487e+05	46644.102194
min	1992.000000	2.000000e+04	1.000000
25%	2011.000000	2.087498e+05	35000.000000
50%	2014.000000	3.500000e+05	60000.000000
75%	2016.000000	6.000000e+05	90000.000000
max	2020.000000	8.900000e+06	806599.000000

In [7]:
```python
sns.pairplot(data)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x2db09084280>



In [8]:
```python
data.hist(bins=50, figsize=(30,15))
plt.show()
```



In [9]:
```python
data.isnull().sum()
```

Out[9]:
```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
dtype: int64
```

In [10]:
```python
data.nunique()
```

Out[10]:
```
name             1491
year               27
selling_price     445
km_driven         770
fuel                5
seller_type         3
transmission        2
owner               5
dtype: int64
```

In [11]:
```python
data.shape
```

Out[11]: (4340, 8)

In [12]:
```python
data.corr()
```

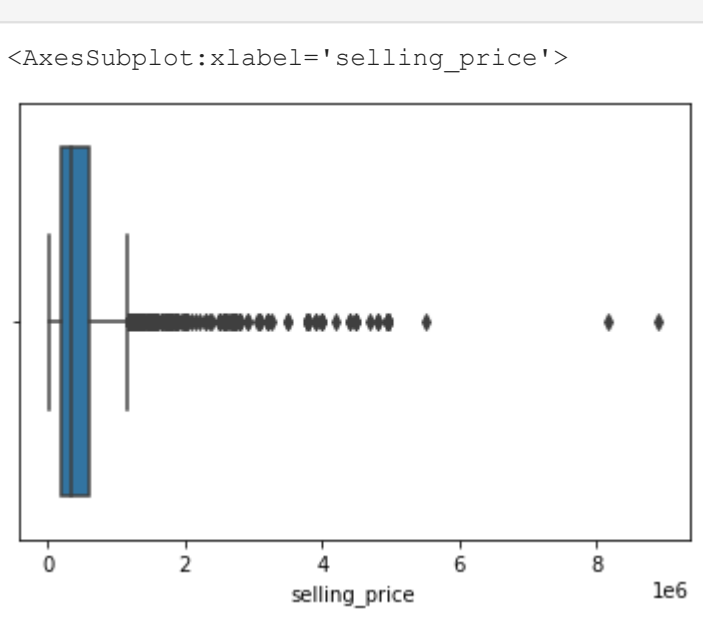
Out[12]:

	year	selling_price	km_driven
year	1.000000	0.413922	-0.419688
selling_price	0.413922	1.000000	-0.192289
km_driven	-0.419688	-0.192289	1.000000

In [13]:
```python
#how old the car is?
data['Car_Age'] = 2021 - data.year
```

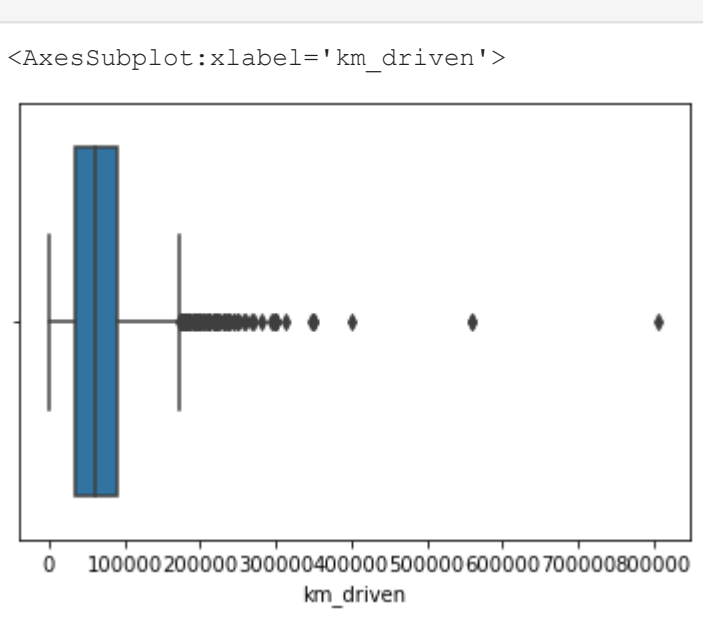
In [14]:
```python
sns.boxplot(x=data['selling_price'])
```

Out[14]: <AxesSubplot:xlabel='selling_price'>



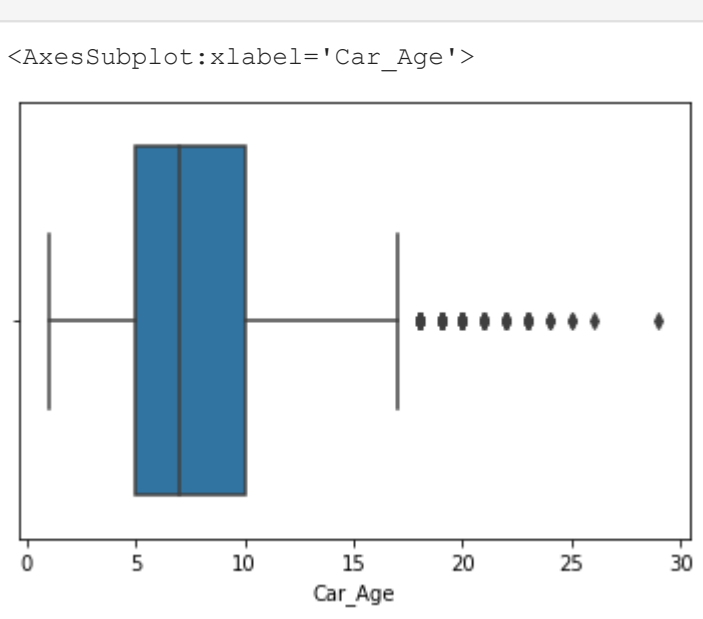
In [15]:
```python
sns.boxplot(x=data['km_driven'])
```

Out[15]: <AxesSubplot:xlabel='km_driven'>



In [16]:
```python
sns.boxplot(x=data['Car_Age'])
```

Out[16]: <AxesSubplot:xlabel='Car_Age'>

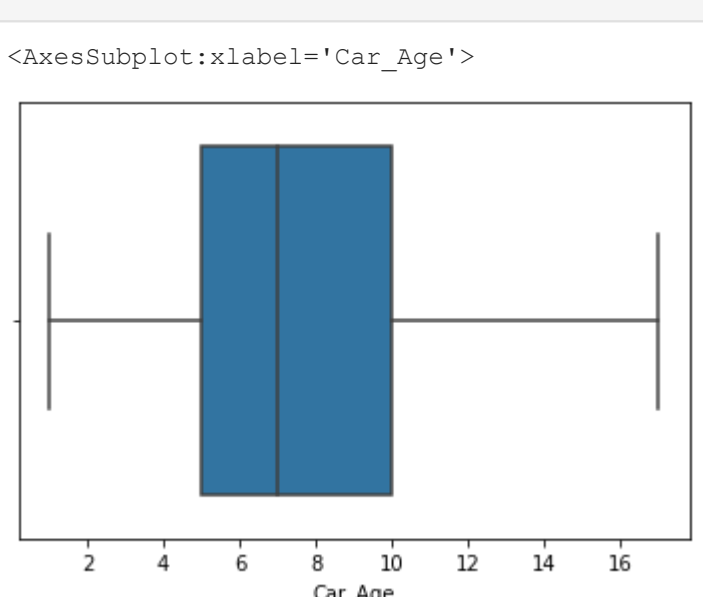


In [17]:
```python
def cap_data(data2):
    for col in data2.columns:
        print("capping the ",col)
        if (((data2[col].dtype)=='float64') | ((data2[col].dtype)=='int64')):
            Q1 = data2.quantile(0.25)
            Q3 = data2.quantile(0.75)
            IQR = Q3 - Q1
            data2 = data2[~((data2 < (Q1 - 1.5 * IQR)) |(data2 > (Q3 + 1.5 * IQR))).any(axis=1)]
        else:
            data2[col]=data2[col]
    return data2
data=cap_data(data)
```

```
capping the  name
capping the  year
capping the  selling_price
capping the  km_driven
capping the  fuel
capping the  seller_type
capping the  transmission
capping the  owner
capping the  Car_Age
```

In [18]:
```python
sns.boxplot(x=data['Car_Age'])
```

Out[18]: <AxesSubplot:xlabel='Car_Age'>



In [19]:
```python
#one hot encoding for categorical data
data = pd.get_dummies(data, drop_first=True)
data.head()
```

Out[19]:

	year	selling_price	km_driven	Car_Age	name_Ambassador Grand 1800 ISZ MPFI PW CL	name_Audi A6 2.7 TDI	name_Audi A6 2.8 FSI	name_BMW 5 Series 530i	name_BMW 7 Series 730Ld	name_BMW X1 sDrive20d	...	fuel_Electric
0	2007	60000	70000	14	0	0	0	0	0	0	...	0
1	2007	135000	50000	14	0	0	0	0	0	0	...	0
2	2012	600000	100000	9	0	0	0	0	0	0	...	0
3	2017	250000	46000	4	0	0	0	0	0	0	...	0
4	2014	450000	141000	7	0	0	0	0	0	0	...	0

5 rows × 1308 columns

In [29]:
```python
data.corr()
```

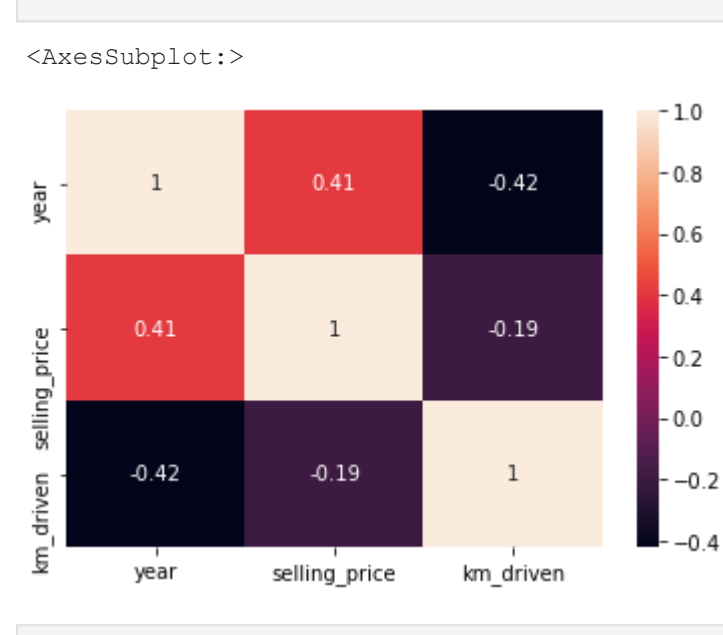
Out[29]:

	year	selling_price	km_driven	Car_Age	name_Ambassador Grand 1800 ISZ MPFI PW CL	name_Audi A6 2.7 TDI	name_Audi A6 2.8 FSI	name_BMW 5 Series 530i	name_BMW 7 Series 730Ld	nam… sD
year	1.000000	0.639216	-0.493062	-1.000000	-0.005392	-0.014060	-0.022727	-0.031395	-0.044405	-0
selling_price	0.639216	1.000000	-0.282218	-0.639216	0.002585	0.031548	0.017756	0.006033	0.064129	0
km_driven	-0.493062	-0.282218	1.000000	0.493062	-0.005752	-0.007540	0.003188	0.021068	-0.020780	0
Car_Age	-1.000000	-0.639216	0.493062	1.000000	0.005392	0.014060	0.022727	0.031395	0.044405	0
name_Ambassador Grand 1800 ISZ MPFI PW CL	-0.005392	0.002585	-0.005752	0.005392	1.000000	-0.000262	-0.000262	-0.000262	-0.000371	-0
...	...	...	...	...	...	...	...	...	...	
transmission_Manual	-0.087046	-0.179249	0.116311	0.087046	0.004132	-0.063487	-0.063487	-0.063487	-0.089796	-0
owner_Fourth & Above Owner	-0.131750	-0.108576	0.118843	0.131750	-0.002048	-0.002048	-0.002048	-0.002048	-0.002897	-0
owner_Second Owner	-0.300481	-0.216583	0.243864	0.300481	0.027128	-0.009670	-0.009670	0.027128	-0.013677	0
owner_Test Drive Car	0.092001	0.093820	-0.087382	-0.092001	-0.000871	-0.000871	-0.000871	-0.000871	-0.001232	-0
owner_Third Owner	-0.233022	-0.177025	0.196796	0.233022	-0.004263	-0.004263	0.061541	-0.004263	-0.006029	-0

1308 rows × 1308 columns

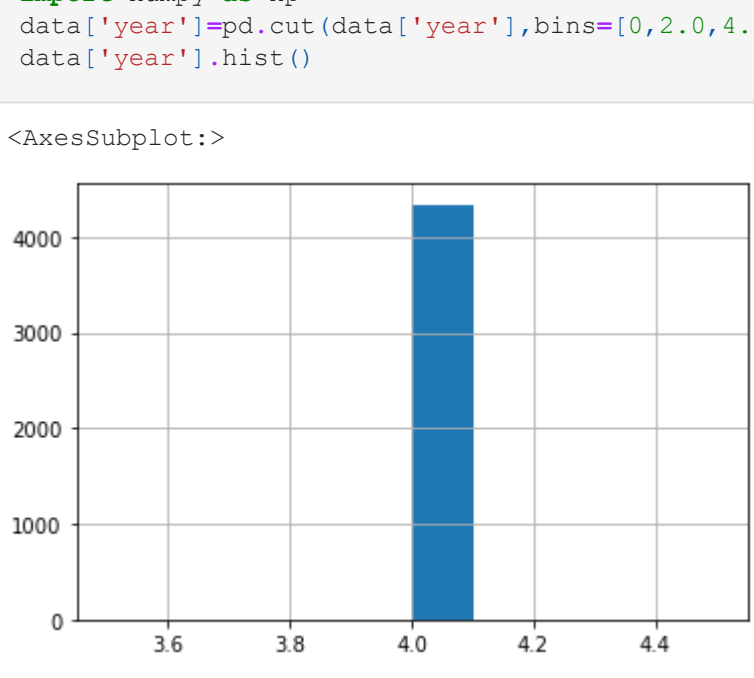
In [6]:
```python
corelation=data.corr()
sns.heatmap(corelation,xticklabels=corelation.columns,yticklabels=corelation.columns,annot=True)
```

Out[6]: <AxesSubplot:>



In [12]:
```python
import numpy as np
data['year']=pd.cut(data['year'],bins=[0,2.0,4.0,6.0,np.inf],labels=[1,2,3,4])
data['year'].hist()
```

Out[12]: <AxesSubplot:>



In [85]:
```python
data=pd.get_dummies(data,drop_first=True)
```

In [86]:
```python
data.head()
```

Out[86]:

	km_driven	transmission	name_Ambassador Classic 2000 Dsz	name_Ambassador Grand 1800 ISZ MPFI PW CL	name_Audi A4 1.8 TFSI	name_Audi A4 2.0 TDI	name_Audi A4 2.0 TDI 177 Bhp Premium Plus	name_Audi A4 3.0 TDI Quattro	name_Audi A4 30 TFSI Technology	name_Audi A4 35 TDI Premium	..
0	70000	0	0	0	0	0	0	0	0	0	..
1	50000	0	0	0	0	0	0	0	0	0	..
2	100000	0	0	0	0	0	0	0	0	0	..
3	46000	0	0	0	0	0	0	0	0	0	..
4	141000	0	0	0	0	0	0	0	0	0	..

5 rows × 1511 columns

In [87]:
```python
#independent and dependent feature
X=data.iloc[:,1:]
Y=data.iloc[:,0]
```

In [88]:
```python
X.head()
```

Out[88]:

	transmission	name_Ambassador Classic 2000 Dsz	name_Ambassador Grand 1800 ISZ MPFI PW CL	name_Audi A4 1.8 TFSI	name_Audi A4 2.0 TDI	name_Audi A4 2.0 TDI 177 Bhp Premium Plus	name_Audi A4 3.0 TDI Quattro	name_Audi A4 30 TFSI Technology	name_Audi A4 35 TDI Premium	name_Audi A4 35 TDI Premium Plus
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0

5 rows × 1510 columns

In [89]:
```python
Y.head()
```

Out[89]:
```
0     70000
1     50000
2    100000
3     46000
4    141000
Name: km_driven, dtype: int64
```

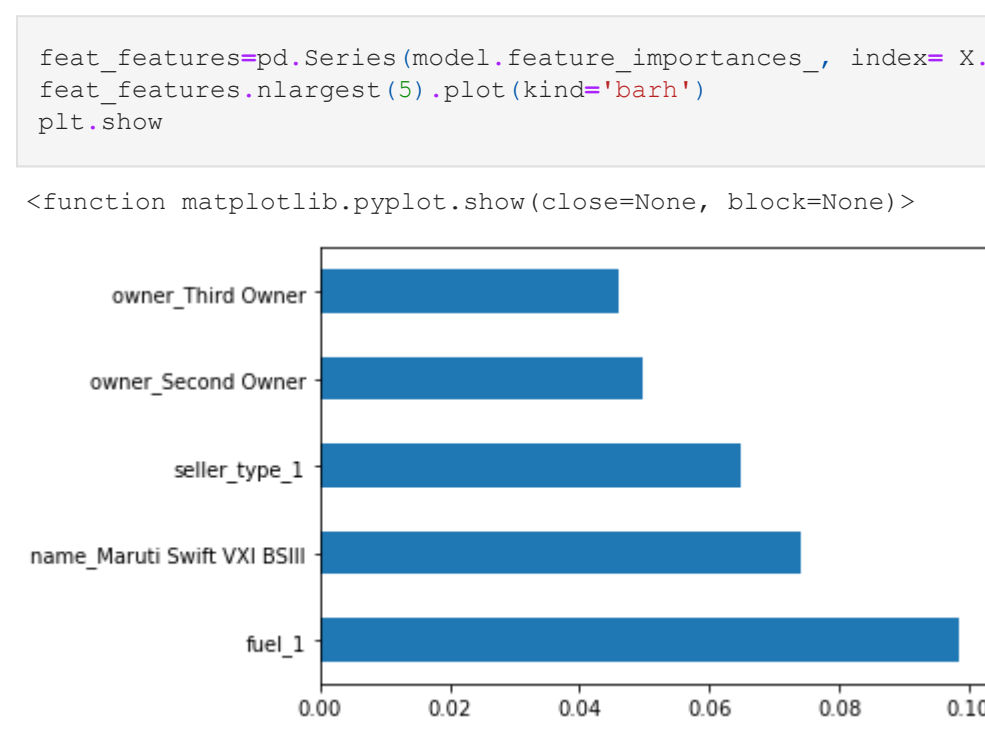
In [90]:
```python
### importance Feature
from sklearn.ensemble import ExtraTreesRegressor
model= ExtraTreesRegressor()
```

In [91]:
```python
model.fit(X,Y)
```

Out[91]: ExtraTreesRegressor()

In [92]:
```python
feat_features=pd.Series(model.feature_importances_, index= X.columns)
feat_features.nlargest(5).plot(kind='barh')
plt.show
```

Out[92]: <function matplotlib.pyplot.show(close=None, block=None)>



In [93]:
```python
from sklearn.model_selection import train_test_split
X_train,X_test, Y_train, Y_test= train_test_split(X,Y, test_size=0.2)
```

In [94]:
```python
X_train.head()
```

Out[94]:

	transmission	name_Ambassador Classic 2000 Dsz	name_Ambassador Grand 1800 ISZ MPFI PW CL	name_Audi A4 1.8 TFSI	name_Audi A4 2.0 TDI	name_Audi A4 2.0 TDI 177 Bhp Premium Plus	name_Audi A4 3.0 TDI Quattro	name_Audi A4 30 TFSI Technology	name_Audi A4 35 TDI Premium	name_Auc A4 35 TD Premiur Plu
4250	0	0	0	0	0	0	0	0	0	
3995	0	0	0	0	0	0	0	0	0	
643	0	0	0	0	0	0	0	0	0	
2537	0	0	0	0	0	0	0	0	0	
59	0	0	0	0	0	0	0	0	0	

5 rows × 1510 columns

In [95]:
```python
from sklearn.ensemble import RandomForestRegressor
re_random=RandomForestRegressor()
```

In [96]:
```python
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]

max_features = ['auto', 'sqrt']

max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]

min_samples_split = [2, 5, 10, 15, 100]

min_samples_leaf = [1, 2, 5, 10]
```

In [97]:
```python
from sklearn.model_selection import RandomizedSearchCV
```

In [98]:
```python
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

print(random_grid)
```

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_features': ['auto', 'sqr
t'], 'max_depth': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2,
5, 10]}
```

In [99]:
```python
rf = RandomForestRegressor()
```

In [100…
```python
rf_random=RandomizedSearchCV(estimator= rf , param_distributions=random_grid,
                             scoring='neg_mean_squared_error',n_iter= 10, cv= 5, verbose=2,random_state=42,n_jo
```

In [101…
```python
rf_random.fit(X_train,Y_train)
```

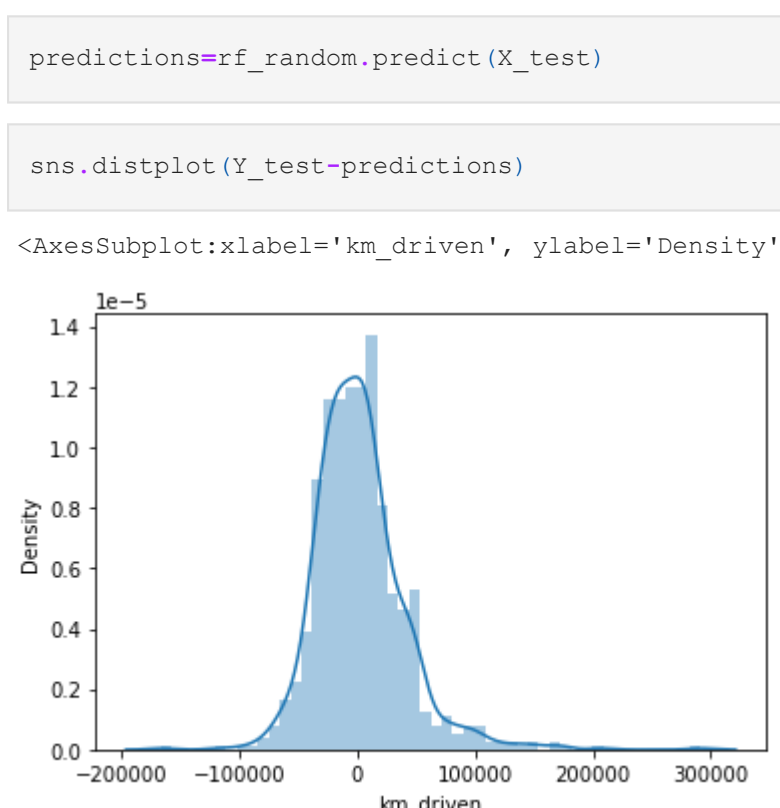
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time
=   4.3s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time
=   4.1s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time
=   5.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time
=   6.1s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time
=   4.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total ti
me=  10.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total ti
me=   8.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total ti
me=   8.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total ti
me=   7.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total ti
me=   8.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total ti
me=  17.3s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total ti
me=  17.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total ti
me=  17.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total ti
me=  17.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total ti
me=  17.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time
=  23.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time
=  24.3s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time
=  24.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time
=  24.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time
=  24.9s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total tim
e=  33.9s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total tim
e=  34.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total tim
e=  34.4s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total tim
e=  34.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total tim
e=  34.6s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total tim
e=  15.4s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total tim
e=  15.1s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total tim
e=  15.4s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total tim
e=  15.2s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total tim
e=  15.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total ti
me=   2.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total ti
me=   2.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total ti
me=   2.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total ti
me=   2.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total ti
me=   2.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total tim
e=   2.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total tim
e=   2.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total tim
e=   2.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total tim
e=   2.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total tim
e=   2.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time
=   2.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time
=   2.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time
=   2.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time
=   2.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time
=   2.4s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total tim
e=  58.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total tim
e=  58.7s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total tim
e= 1.0min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total tim
e= 1.0min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total tim
e= 1.0min
```

Out[101…
```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
                                                         900, 1000, 1100,
                                                         1200]},
                   random_state=42, scoring='neg_mean_squared_error',
                   verbose=2)
```

In [103…
```python
predictions=rf_random.predict(X_test)
```

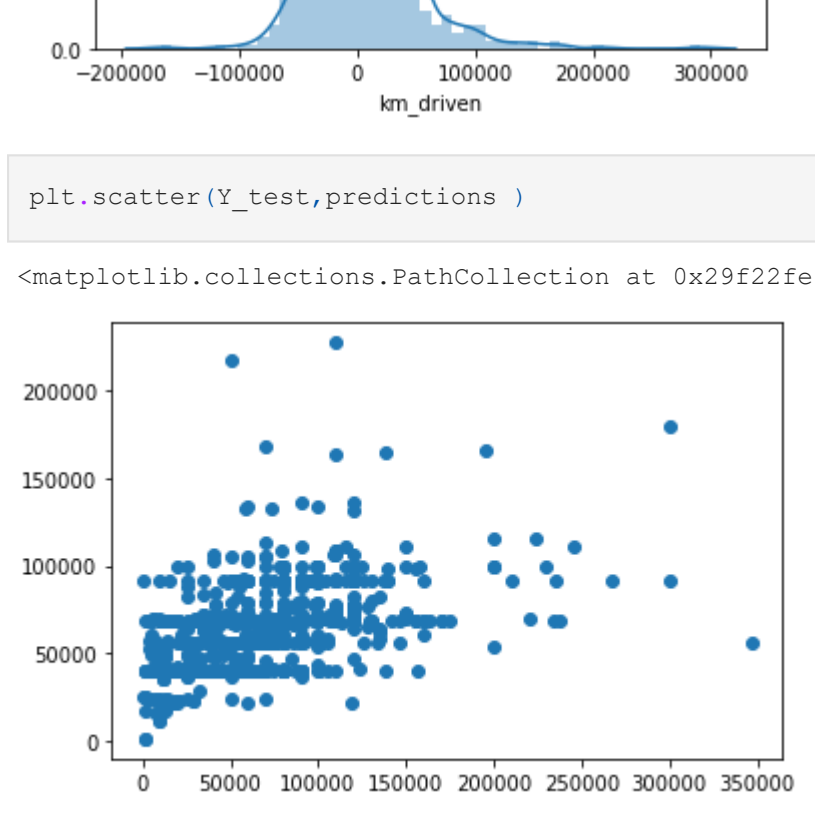
In [104…
```python
sns.distplot(Y_test-predictions)
```

Out[104… <AxesSubplot:xlabel='km_driven', ylabel='Density'>



In [105…
```python
plt.scatter(Y_test,predictions )
```

Out[105… <matplotlib.collections.PathCollection at 0x29f22fe05e0>



In [106…
```python
from sklearn import metrics
```

In [117…
```python
print('MAE:', metrics.mean_absolute_error(Y_test, predictions)/10000)

print('RMSE:', np.sqrt(metrics.mean_squared_error(Y_test, predictions))/10000)
```

```
MAE: 2.73215213957661
RMSE: 3.8521705952256498
```

In [118…
```python
import pickle

file = open('random_forest_regression_model.csv', 'wb')


pickle.dump(rf_random, file)
```

In [ ]: