In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
from sklearn.linear_model import Ridge, RidgeCV, Lasso
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
df=pd.read_csv(r"C:\Users\shaik\Downloads\fiat500_VehicleSelection_Dataset (2).csv")
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1	lounge	51	882	25000	1	44.907242	8.611
1	2	pop	51	1186	32500	1	45.666359	12.241
2	3	sport	74	4658	142228	1	45.503300	11.417
3	4	lounge	51	2739	160000	1	40.633171	17.634
4	5	pop	73	3074	106880	1	41.903221	12.495
...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.704
1534	1535	lounge	74	3835	112000	1	45.845692	8.666
1535	1536	pop	51	2223	60457	1	45.481541	9.413
1536	1537	lounge	51	2557	80750	1	45.000702	7.682
1537	1538	pop	51	1766	54276	1	40.323410	17.568

1538 rows × 9 columns

In [3]:

```
df.head(10)
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1	lounge	51	882	25000	1	44.907242	8.611560
1	2	pop	51	1186	32500	1	45.666359	12.241890
2	3	sport	74	4658	142228	1	45.503300	11.417840
3	4	lounge	51	2739	160000	1	40.633171	17.634609
4	5	pop	73	3074	106880	1	41.903221	12.495650
5	6	pop	74	3623	70225	1	45.000702	7.682270
6	7	lounge	51	731	11600	1	44.907242	8.611560
7	8	lounge	51	1521	49076	1	41.903221	12.495650
8	9	sport	73	4049	76000	1	45.548000	11.549470
9	10	sport	51	3653	89000	1	45.438301	10.991700

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   int64  
 1   model            1538 non-null   object 
 2   engine_power     1538 non-null   int64  
 3   age_in_days      1538 non-null   int64  
 4   km               1538 non-null   int64  
 5   previous_owners  1538 non-null   int64  
 6   lat              1538 non-null   float64
 7   lon              1538 non-null   float64
 8   price            1538 non-null   int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	l
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.00000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.54136
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.13351
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.85583
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.80299
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.39409
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.46796
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.79561

In [6]:

```
df.columns
```

Out[6]:

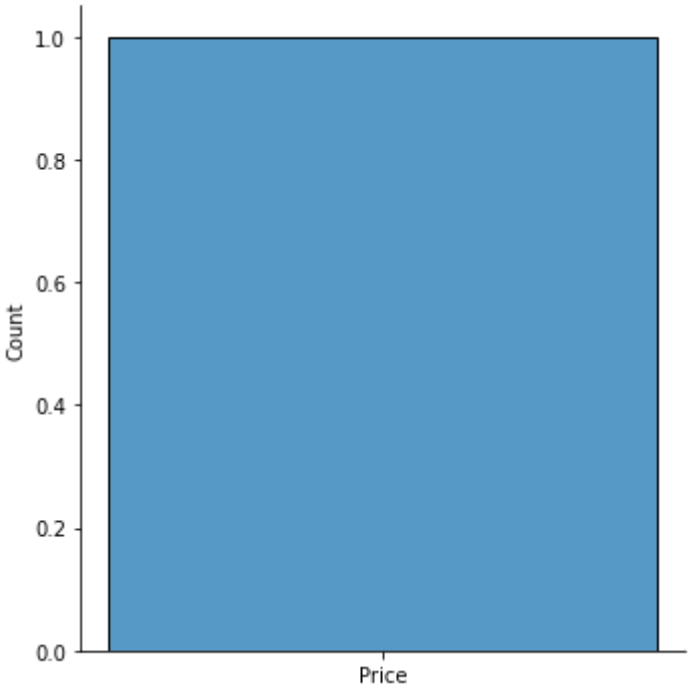
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owner
s',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [7]:

```
sns.displot(['Price'])
```

Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x24e86389bb0>
```

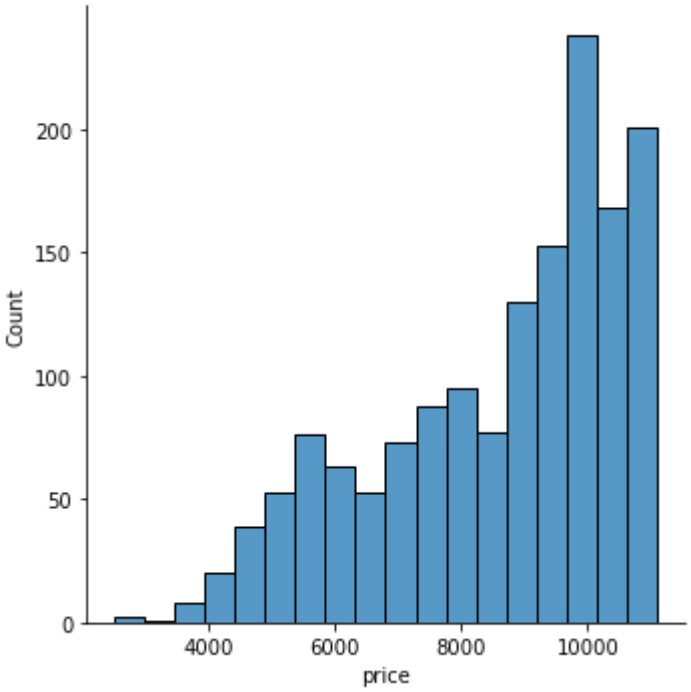


In [8]:

```
sns.displot(df['price'])
```

Out[8]:

```
<seaborn.axisgrid.FacetGrid at 0x24e8642e9d0>
```

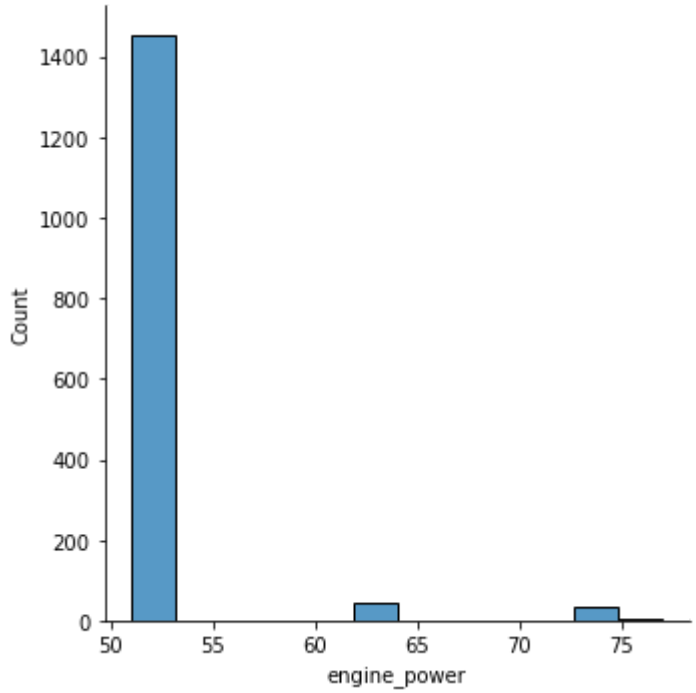


In [9]:

```
sns.displot(df['engine_power'])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x24e86bc3eb0>
```

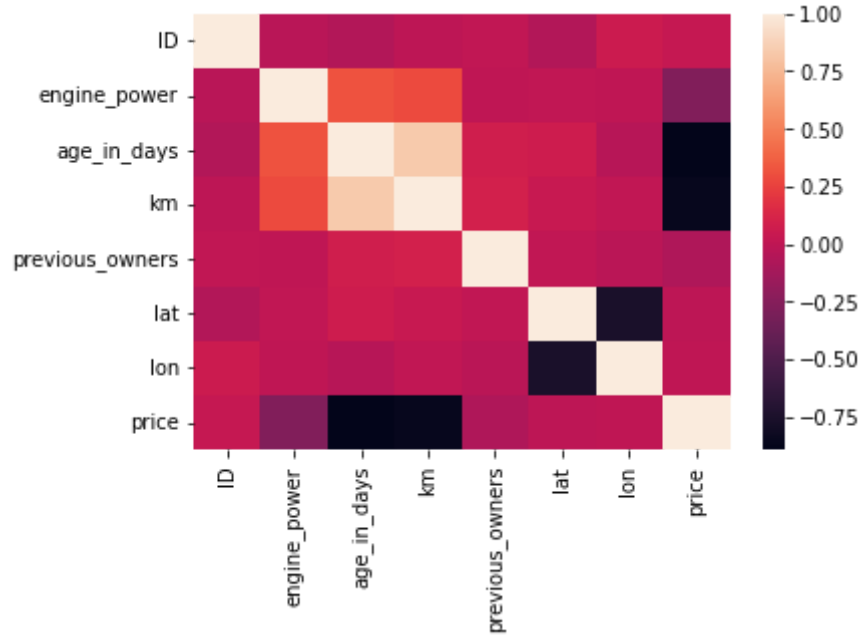


In [10]:

```
fiatdf=df[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
 'lat', 'lon','price']]
sns.heatmap(fiatdf.corr())#with price
```

Out[10]:

```
<AxesSubplot:>
```

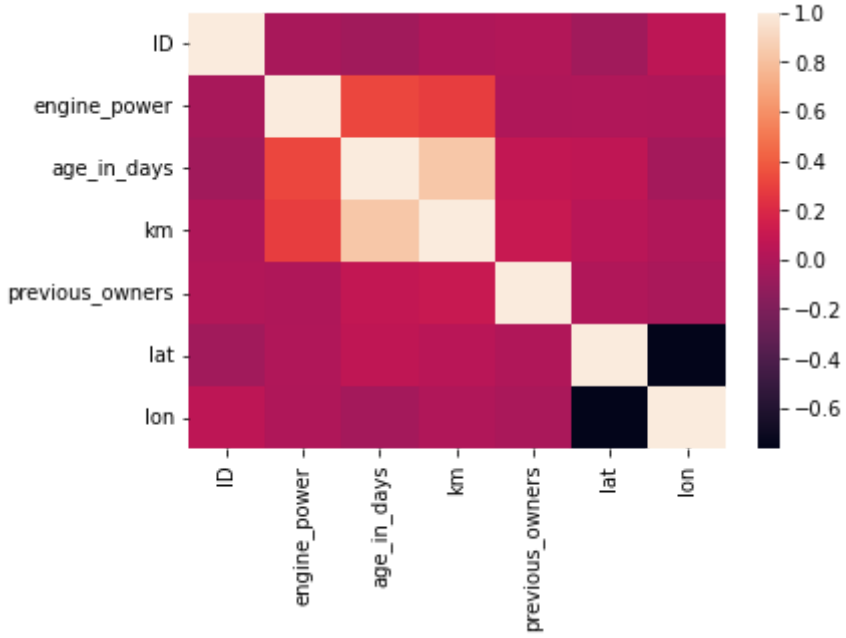


In [11]:

```
fiatdf=df[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
 'lat', 'lon']]
sns.heatmap(fiatdf.corr())#without price
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
X=fiatdf[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
 'lat', 'lon']]
y=df['price']
```

In [13]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=101)
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.intercept_)
```

```
8971.195683499936
```

In [14]:

```
coeff_df=pd.DataFrame(regr.coef_,X.columns,columns=['coefficient'])
coeff_df
```

Out[14]:

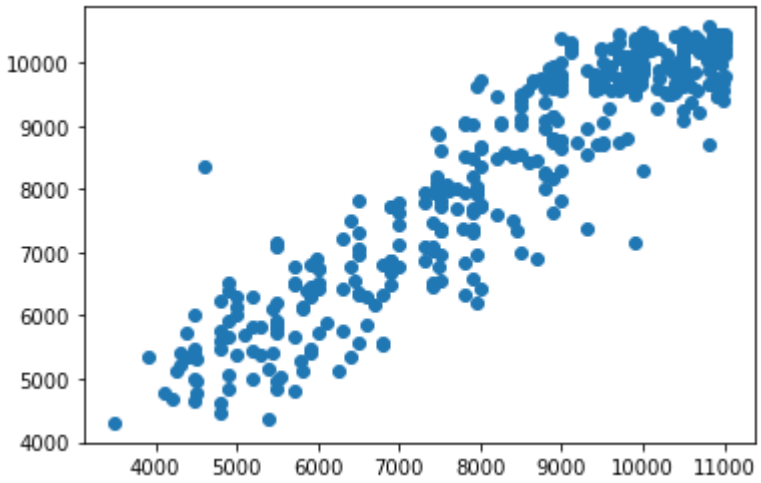
	coefficient
ID	-0.046704
engine_power	11.646408
age_in_days	-0.898018
km	-0.017232
previous_owners	26.400886
lat	32.189709
lon	0.161073

In [15]:

```
predictions=regr.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[15]:

```
<matplotlib.collections.PathCollection at 0x24e87e3ca00>
```

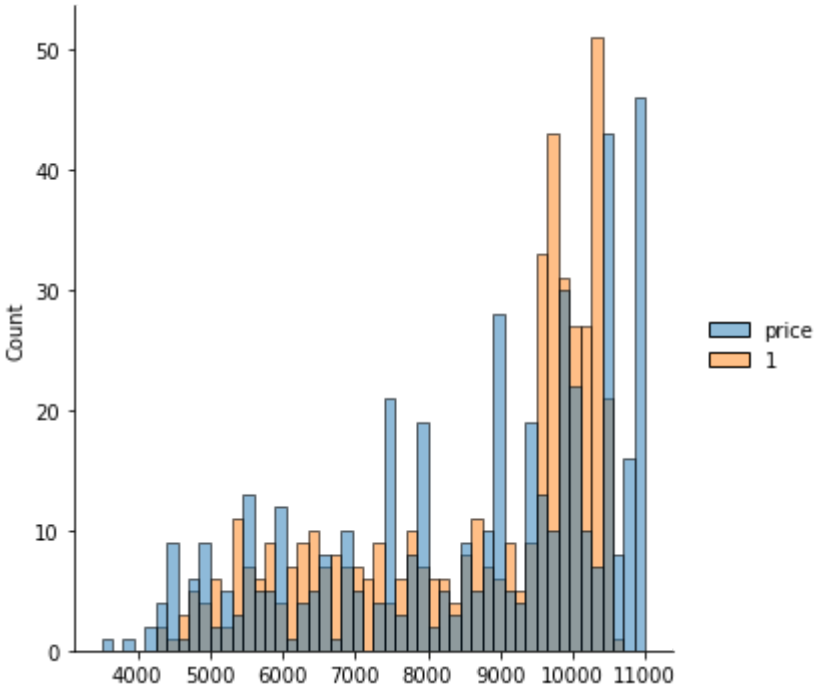


In [16]:

```
sns.displot((y_test,predictions),bins=50)
```

Out[16]:

```
<seaborn.axisgrid.FacetGrid at 0x24e86c02550>
```



In [17]:

```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,predictions))
print('MSE:',metrics.mean_squared_error(y_test,predictions))
print('MAE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE: 593.0876179519935
MSE: 551442.6799691805
MAE: 742.5918663500029
```

In [18]:

```
#accuracy
regr=LinearRegression()
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
print(regr.score(X_test,y_test))
```

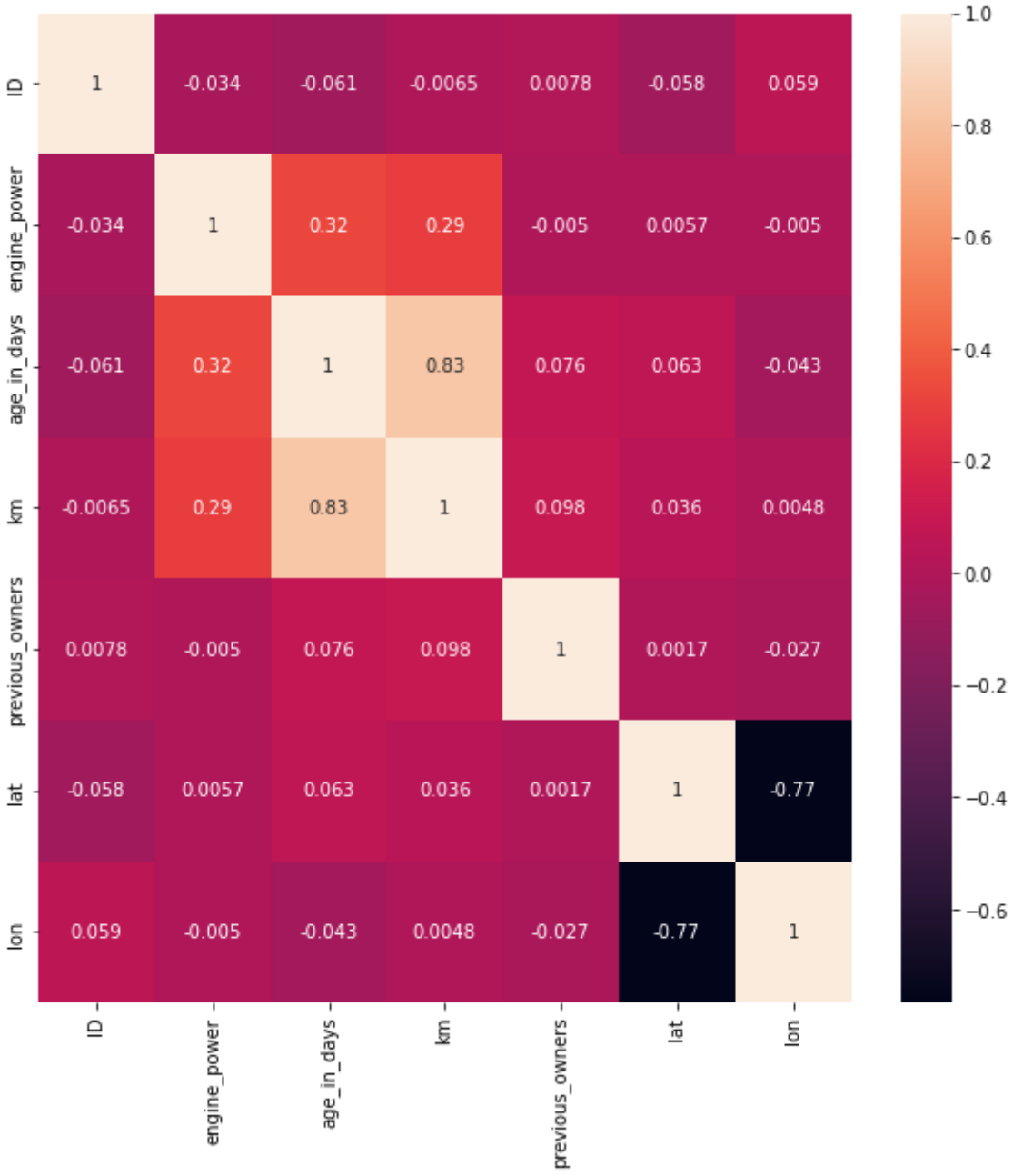
```
0.8597136704308866
```

In [19]:

```
plt.figure(figsize=(10,10))
sns.heatmap(fiatdf.corr(),annot=True)
```

Out[19]:

<AxesSubplot:>

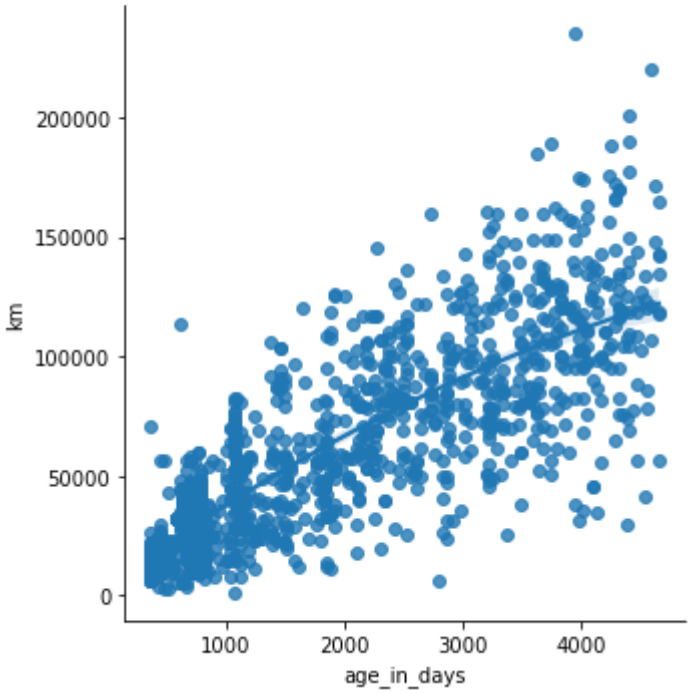


In [20]:

```
sns.lmplot(x="age_in_days",y="km",data=fiatdf,order=2)
```

Out[20]:

```
<seaborn.axisgrid.FacetGrid at 0x24e87eccf70>
```

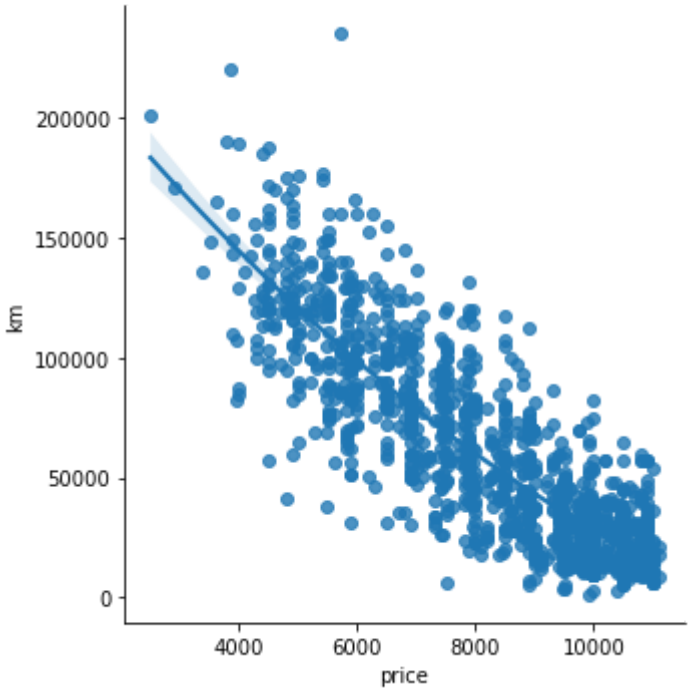


In [21]:

```
sns.lmplot(x="price",y="km",data=df,order=2)
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x24e87dda790>
```



In [22]:

```
df.fillna(method='ffill',inplace=True)
x=np.array(df['age_in_days']).reshape(-1,1)
y=np.array(df['km']).reshape(-1,1)
df.dropna(inplace=True)
```

In [23]:

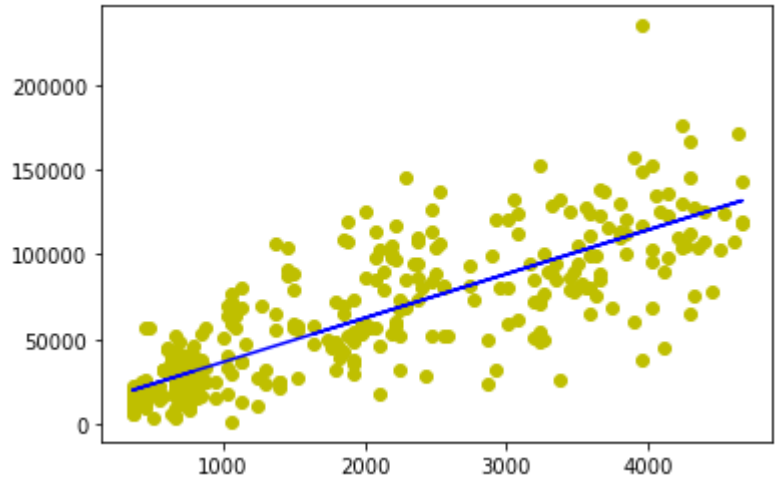
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[23]:

```
LinearRegression()
```

In [24]:

```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='y')
plt.plot(X_test,y_pred,color='b')
plt.show()
```



In [25]:

```
#Linear regression model
regr=LinearRegression()
regr.fit(X_train,y_train)
actual=y_test #actual value
train_score_regr=regr.score(X_train,y_train)
test_score_regr=regr.score(X_test,y_test)
print("\nLinear model:\n")
print("The train score for Linear model is {}".format(train_score_regr))
print("The test score for Linear model is {}".format(test_score_regr))
```

```
Linear model:

The train score for Linear model is 0.6980611050606098
The test score for Linear model is 0.6884389845200596
```

In [26]:

```
#ridge regression model
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test score for ridge regression
train_score_ridge=ridgeReg.score(X_train,y_train)
test_score_ridge=ridgeReg.score(X_test,y_test)
print("\nRidge model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
Ridge model:

The train score for ridge model is 0.6980611050606098
The test score for ridge model is 0.6884389846996535
```

In [27]:

```
#lasso regression model
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
#train and test score for ridge regression
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(X_test,y_test)
print("\nLasso model:\n")
print("The train score for lasso model is {}".format(train_score_lasso))
print("The test score for lasso model is {}".format(test_score_lasso))
```

```
Lasso model:

The train score for lasso model is 0.6980611050605732
The test score for lasso model is 0.6884389919329456
```

In [29]:

```
#using the linear cv model for ridge regression
from sklearn.linear_model import RidgeCV
#ridge cross validation
ridge_cv=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(X_train,y_train)
#score
print(ridge_cv.score(X_train,y_train))
print(ridge_cv.score(X_test,y_test))
```

```
0.6980611050606098
0.6884389846151601
```

In [30]:

```
#using the linear cv model for lasso regression
from sklearn.linear_model import LassoCV
#lasso cross validation
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(X_train,y_train
#score
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.6980611050606098
0.6884389845201336

C:\Users\shaik\anaconda3\lib\site-packages\sklearn\linear_model\_coordinat
e_descent.py:1571: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [ ]:

In [ ]: