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\Downloads\fiat500_VehicleSelection_Dataset.csv")
df
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870
1535	1536	pop	51	2223	60457	1	45.481541	9.413480
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270
1537	1538	pop	51	1766	54276	1	40.323410	17.568270

1538 rows × 9 columns

In [3]:
```
df.head(10)
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pric
0	1	lounge	51	882	25000	1	44.907242	8.611560	89(
1	2	pop	51	1186	32500	1	45.666359	12.241890	88(
2	3	sport	74	4658	142228	1	45.503300	11.417840	42(
3	4	lounge	51	2739	160000	1	40.633171	17.634609	60(
4	5	pop	73	3074	106880	1	41.903221	12.495650	57(
5	6	pop	74	3623	70225	1	45.000702	7.682270	79(
6	7	lounge	51	731	11600	1	44.907242	8.611560	1075
7	8	lounge	51	1521	49076	1	41.903221	12.495650	919
8	9	sport	73	4049	76000	1	45.548000	11.549470	56(
9	10	sport	51	3653	89000	1	45.438301	10.991700	60(

In [4]:
```
df.info
```

Out[4]:
```
<bound method DataFrame.info of         ID   model  engine_power  age_in_days
km  previous_owners  \
0        1  lounge            51          882   25000                1
1        2     pop            51         1186   32500                1
2        3   sport            74         4658  142228                1
3        4  lounge            51         2739  160000                1
4        5     pop            73         3074  106880                1
...    ...     ...           ...          ...     ...              ...
1533  1534   sport            51         3712  115280                1
1534  1535  lounge            74         3835  112000                1
1535  1536     pop            51         2223   60457                1
1536  1537  lounge            51         2557   80750                1
1537  1538     pop            51         1766   54276                1

            lat        lon  price
0     44.907242   8.611560   8900
1     45.666359  12.241890   8800
2     45.503300  11.417840   4200
3     40.633171  17.634609   6000
4     41.903221  12.495650   5700
...         ...        ...    ...
1533  45.069679   7.704920   5200
1534  45.845692   8.666870   4600
1535  45.481541   9.413480   7500
1536  45.000702   7.682270   5990
1537  40.323410  17.568270   7900

[1538 rows x 9 columns]>
```

In [5]:
```
df.describe
```

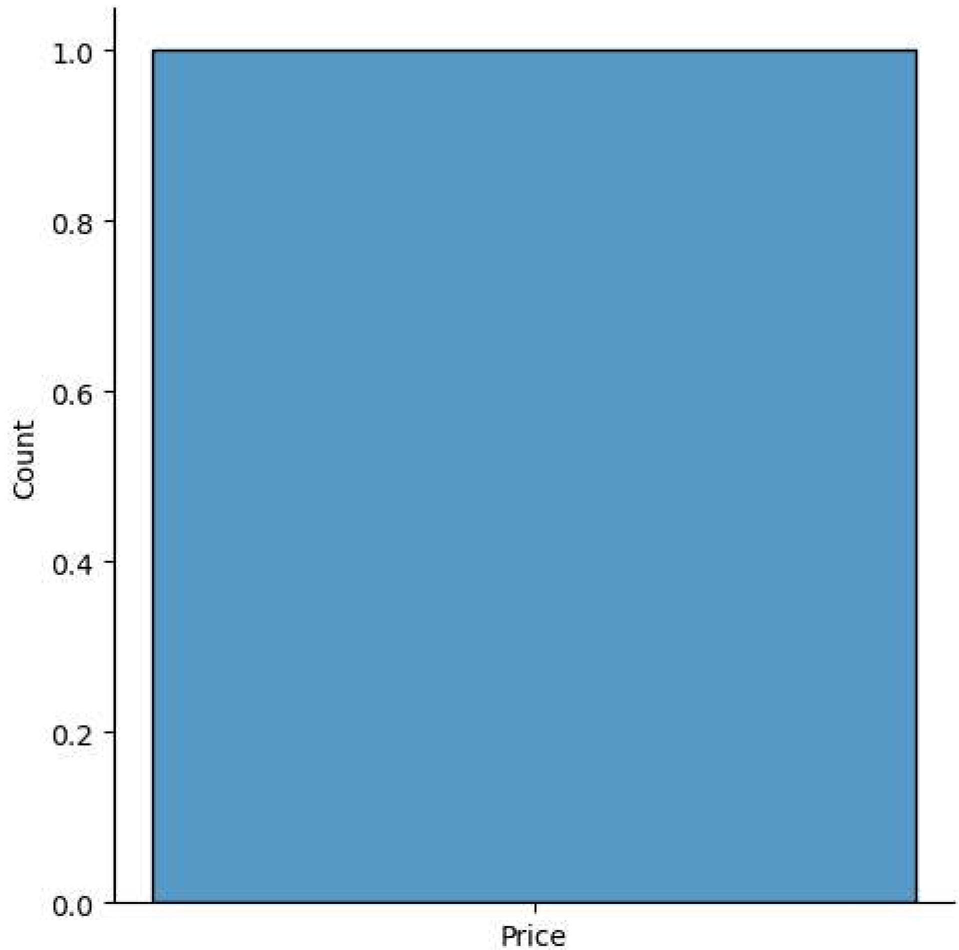
Out[5]:
```
<bound method NDFrame.describe of         ID   model  engine_power  age_in_da
ys      km  previous_owners  \
0        1  lounge           51        882   25000                1
1        2     pop           51       1186   32500                1
2        3   sport           74       4658  142228                1
3        4  lounge           51       2739  160000                1
4        5     pop           73       3074  106880                1
...    ...     ...          ...        ...     ...              ...
1533  1534   sport           51       3712  115280                1
1534  1535  lounge           74       3835  112000                1
1535  1536     pop           51       2223   60457                1
1536  1537  lounge           51       2557   80750                1
1537  1538     pop           51       1766   54276                1

            lat        lon  price
0     44.907242   8.611560   8900
1     45.666359  12.241890   8800
2     45.503300  11.417840   4200
3     40.633171  17.634609   6000
4     41.903221  12.495650   5700
...         ...        ...    ...
1533  45.069679   7.704920   5200
1534  45.845692   8.666870   4600
1535  45.481541   9.413480   7500
1536  45.000702   7.682270   5990
1537  40.323410  17.568270   7900

[1538 rows x 9 columns]>
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

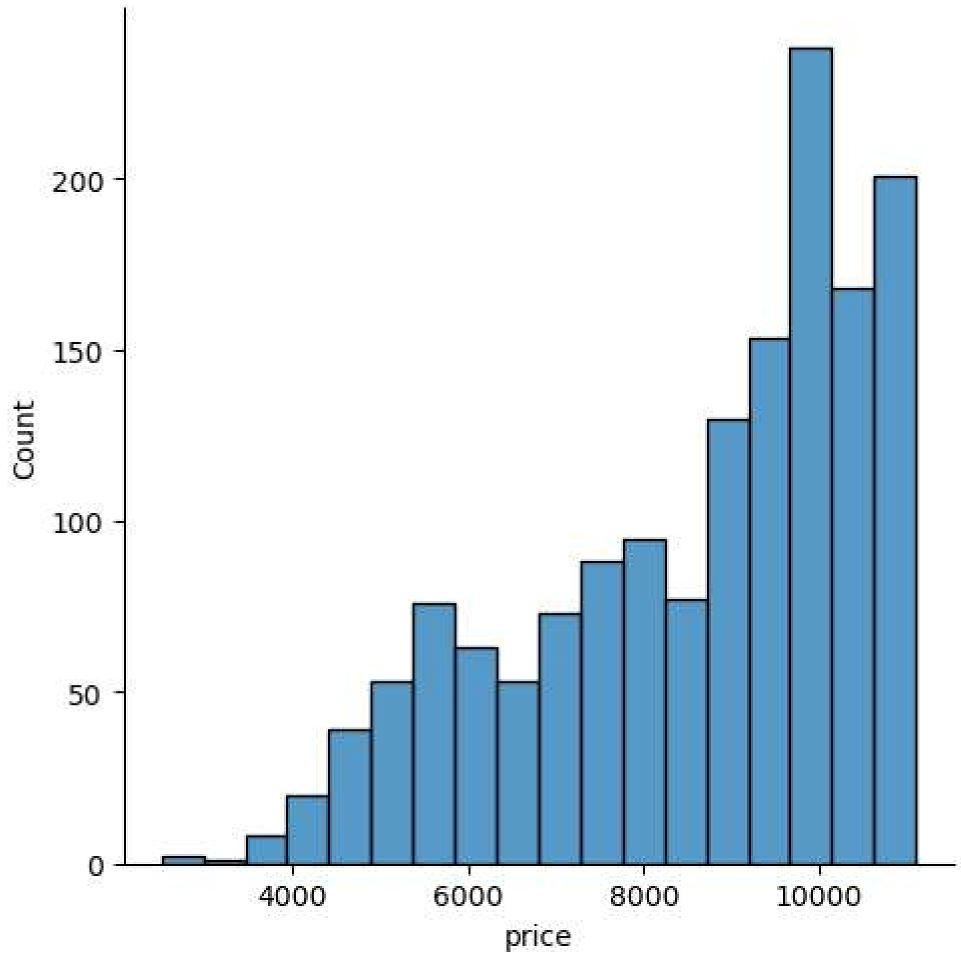
In [7]:
```
sns.displot(['Price'])
```

Out[7]: <seaborn.axisgrid.FacetGrid at 0x23fc6794490>



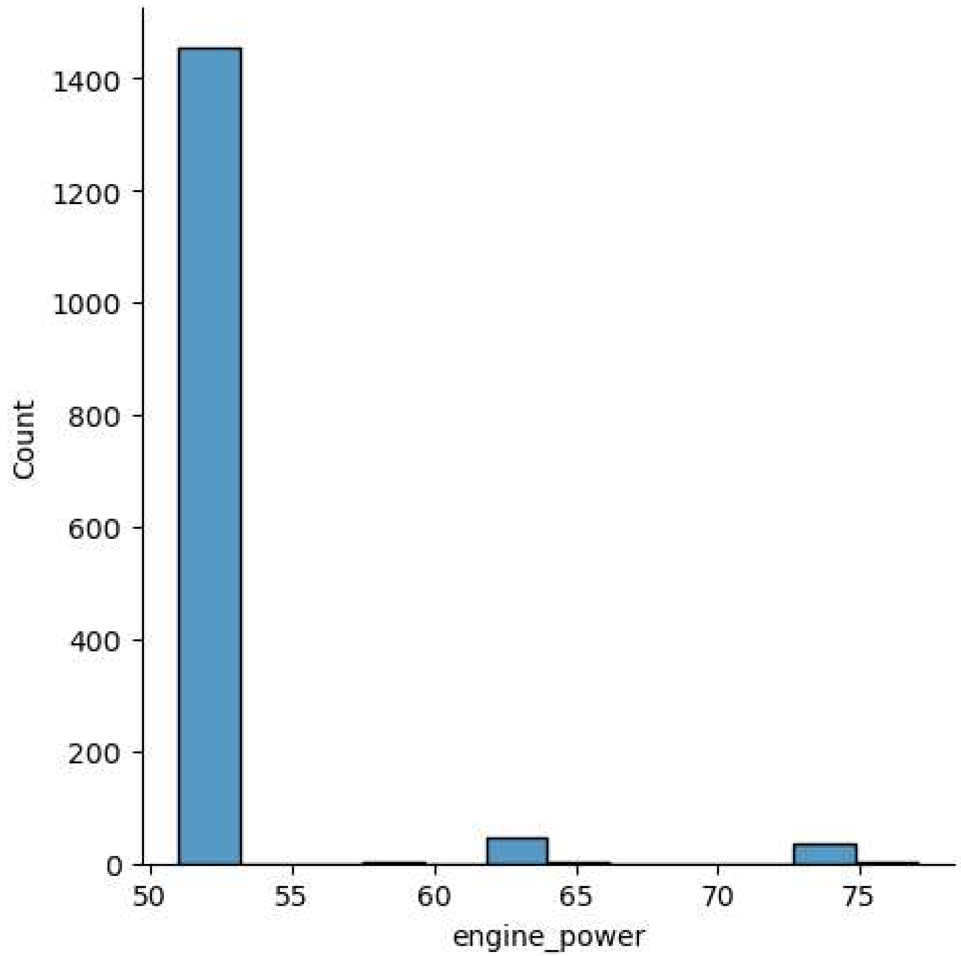
In [8]:
```
sns.displot(df['price'])
```

Out[8]: <seaborn.axisgrid.FacetGrid at 0x23fcca264d0>



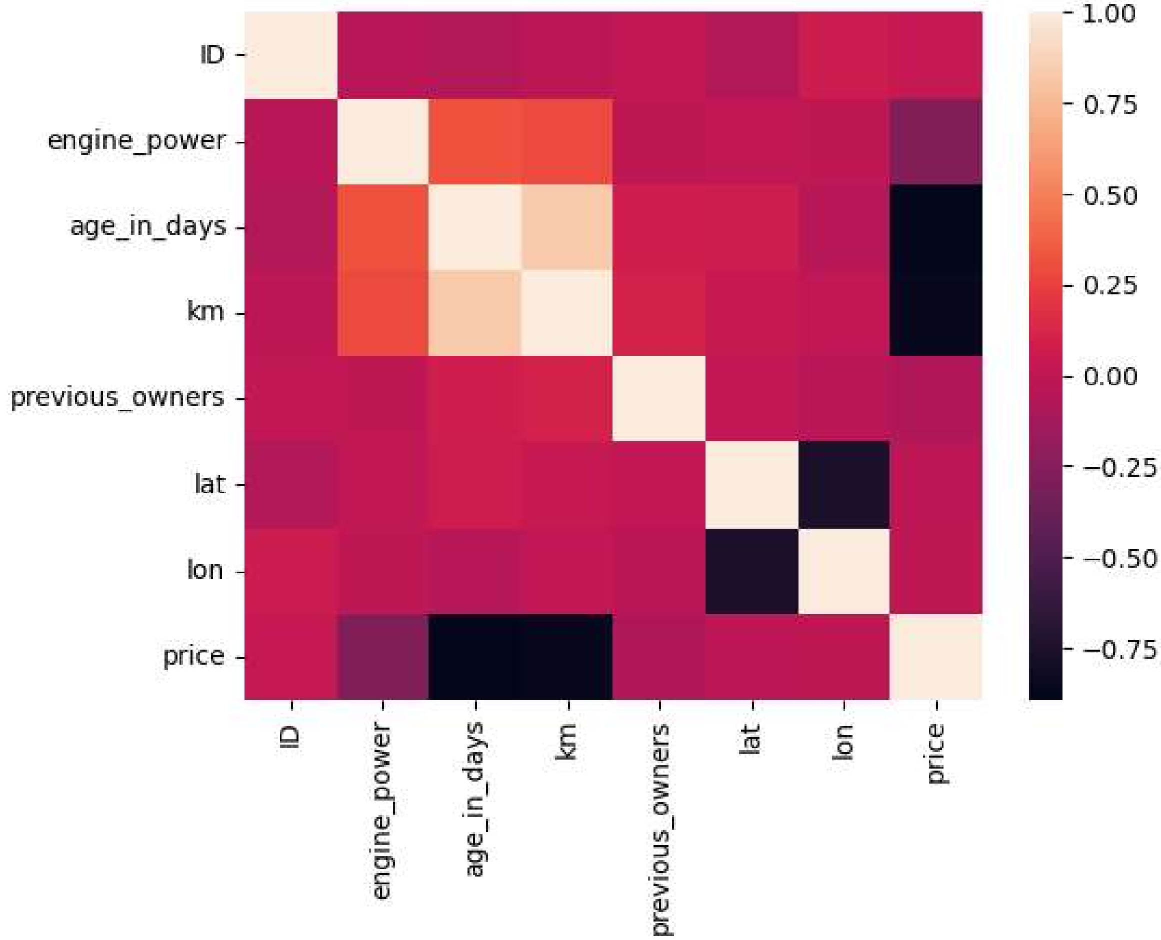
In [9]:
```
sns.displot(df['engine_power'])
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x23fcd165f30>



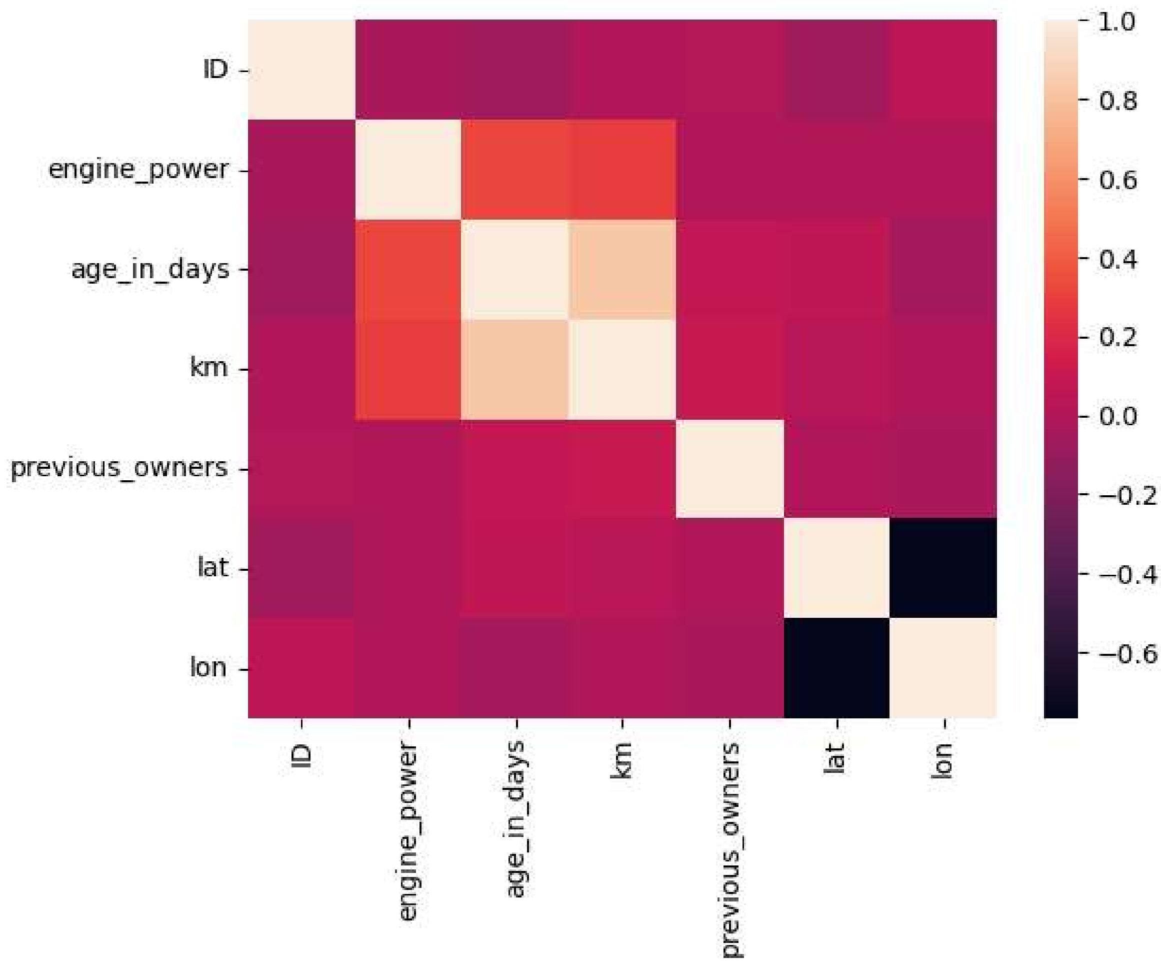
In [10]:
```
fiatdf=df[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
'lat', 'lon','price']]
sns.heatmap(fiatdf.corr())#with price
```

Out[10]: <Axes: >



In [11]:
```
fiatdf=df[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
'lat', 'lon']]
sns.heatmap(fiatdf.corr())#without price
```

Out[11]: <Axes: >



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=
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.intercept_)
```

```
8971.195683499936
```

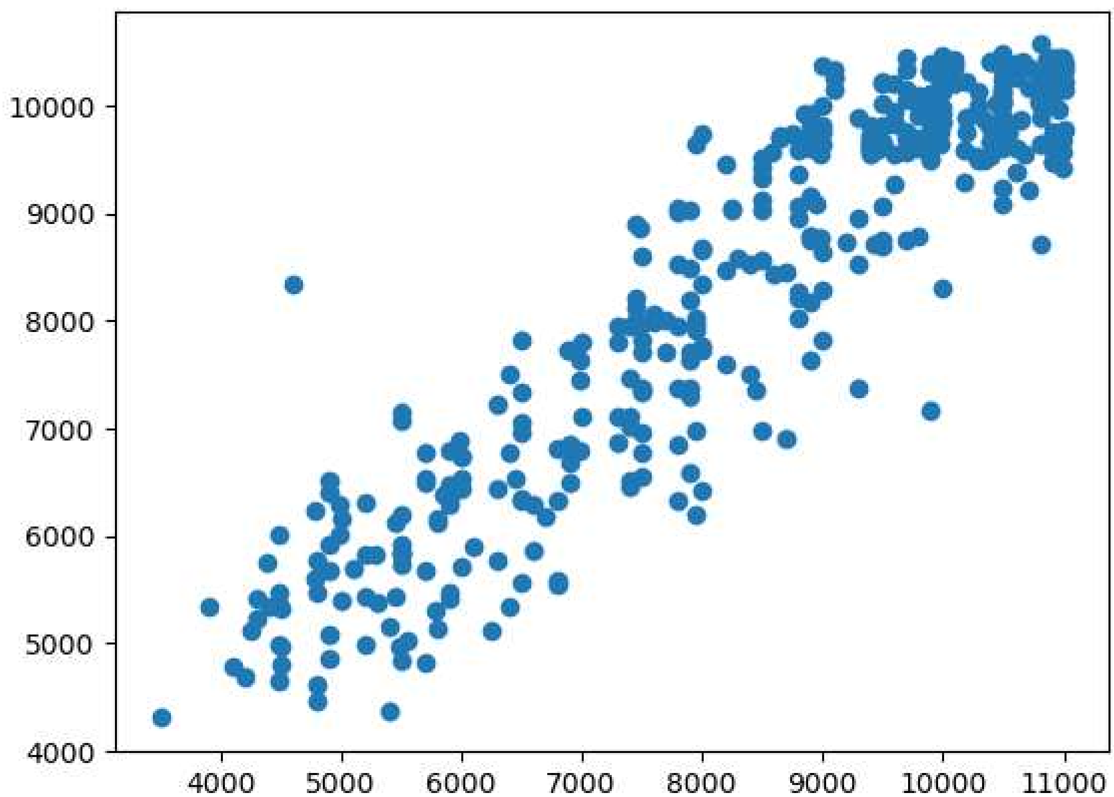
In [14]:
```
coeff_df=pd.DataFrame(regr.coef_,X.columns,columns=['coefficient'])
coeff_df
```

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

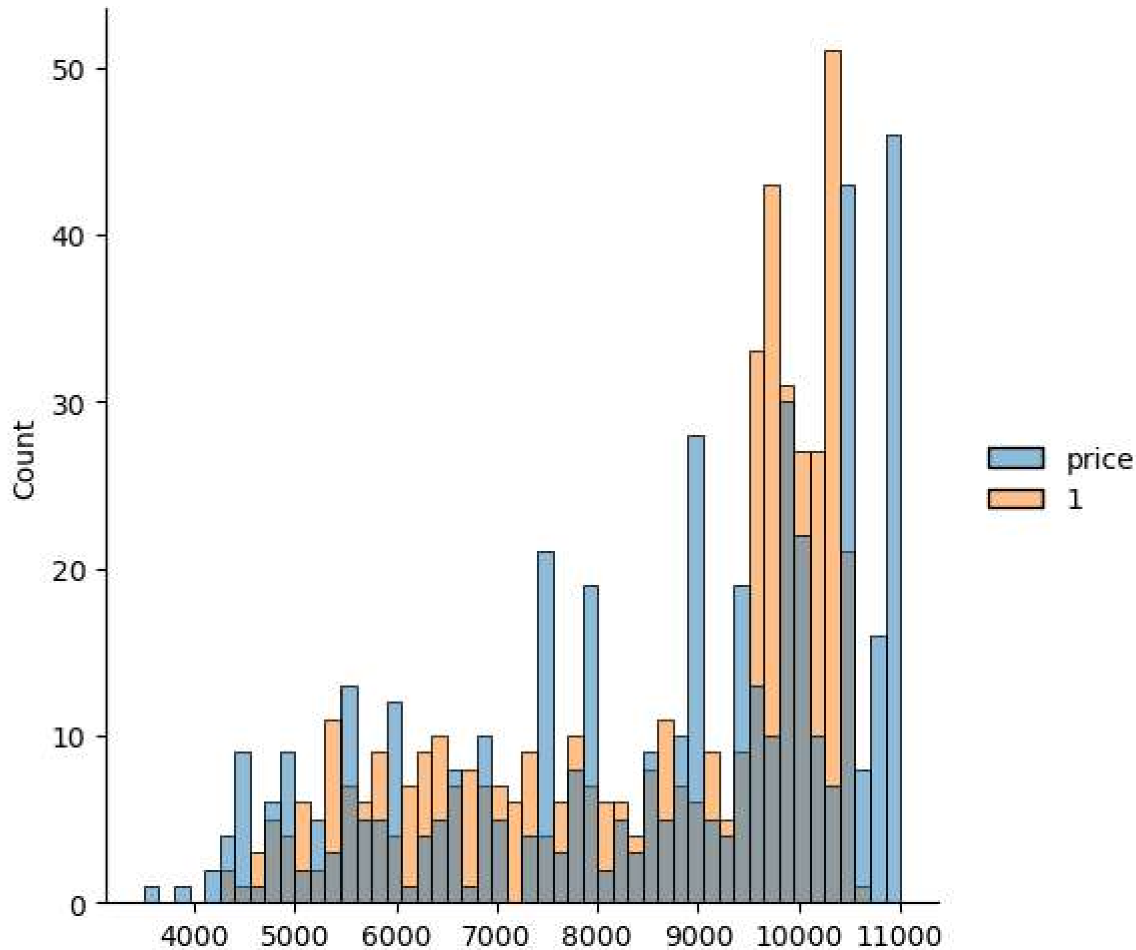
In [15]:
```
predictions=regr.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[15]: <matplotlib.collections.PathCollection at 0x23fce477ac0>



In [16]:
```
sns.displot((y_test,predictions),bins=50)
```

Out[16]: <seaborn.axisgrid.FacetGrid at 0x23fce36ace0>



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)))
```

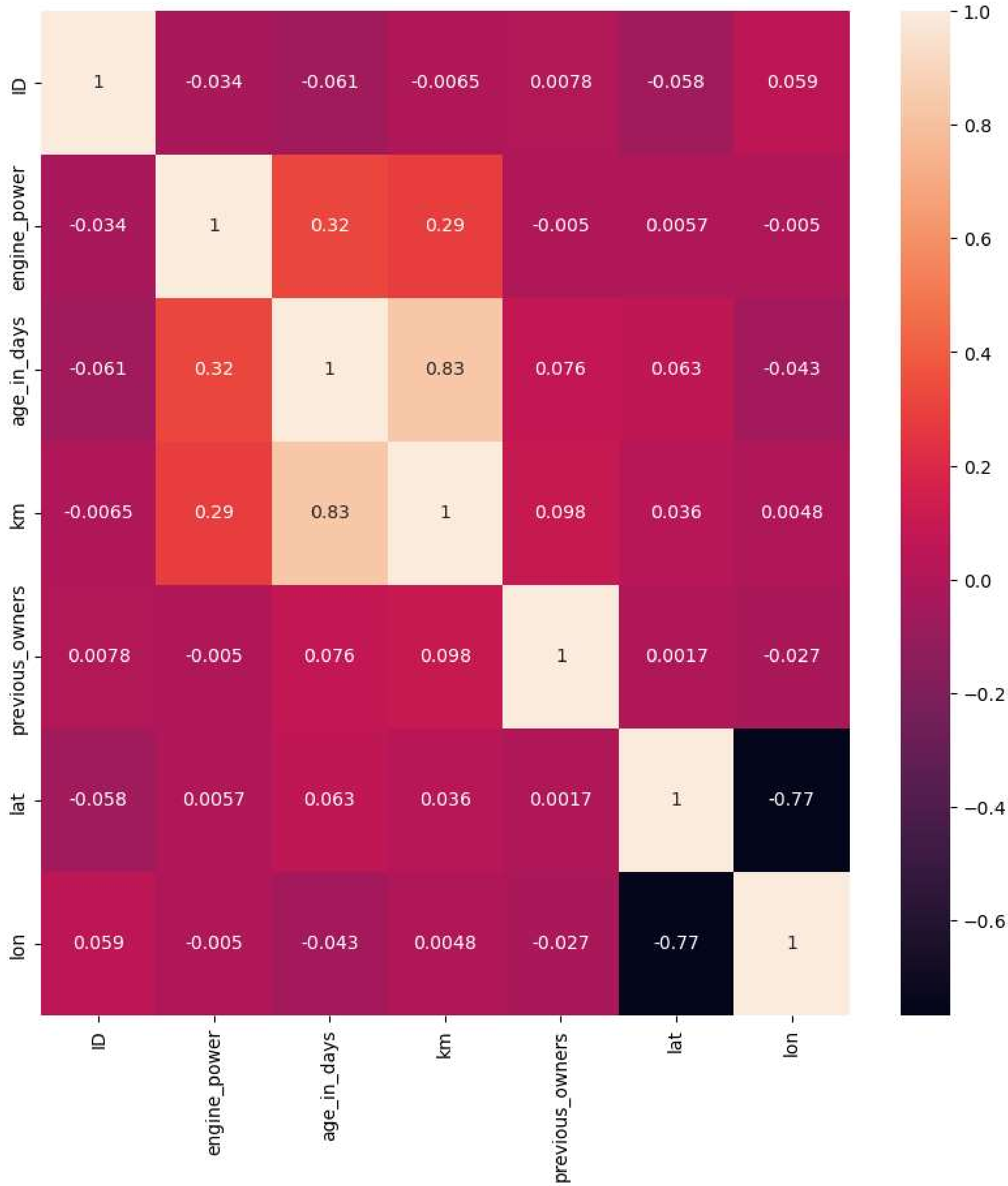
```
MAE: 593.0876179519935
MSE: 551442.6799691805
MAE: 742.5918663500029
```

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
```

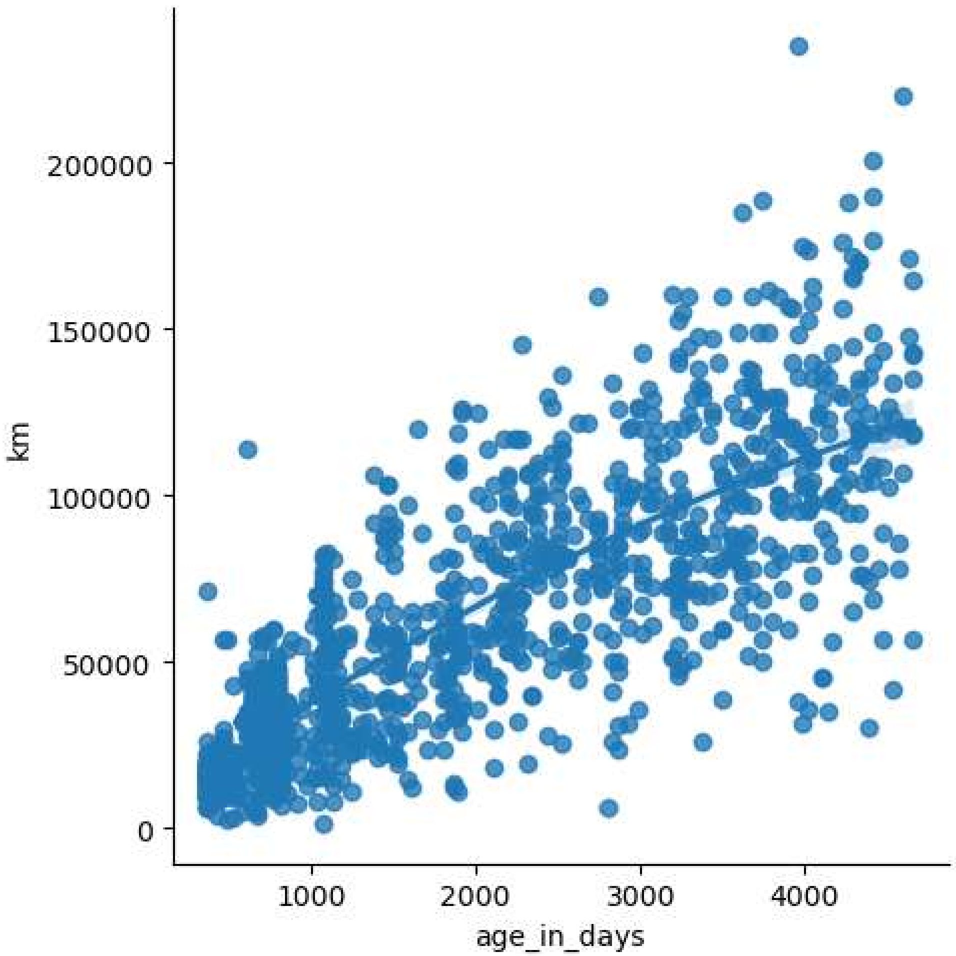
In [19]:
```
plt.figure(figsize=(10,10))
sns.heatmap(fiatdf.corr(),annot=True)
```

Out[19]: <Axes: >



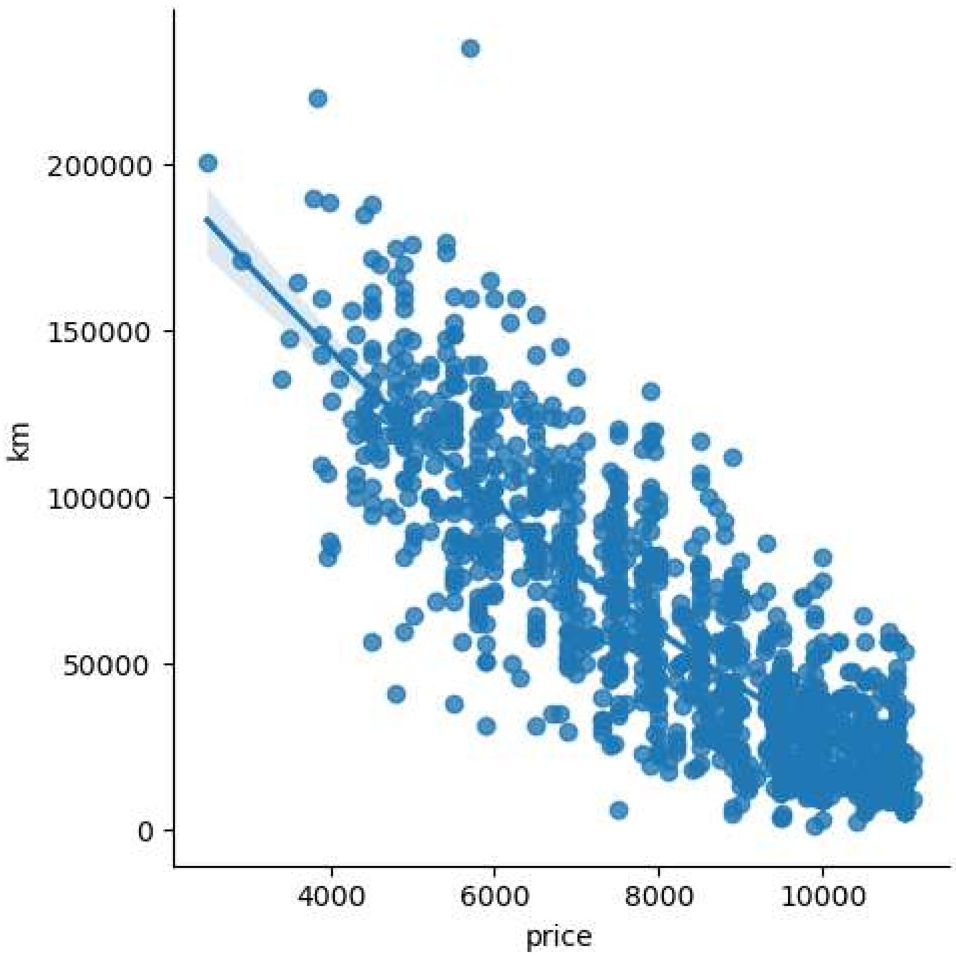
In [20]:
```
sns.lmplot(x="age_in_days",y="km",data=fiatdf,order=2)
```

Out[20]: <seaborn.axisgrid.FacetGrid at 0x23fceaa20b0>



In [21]:
```
sns.lmplot(x="price",y="km",data=df,order=2)
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x23fceb02bf0>



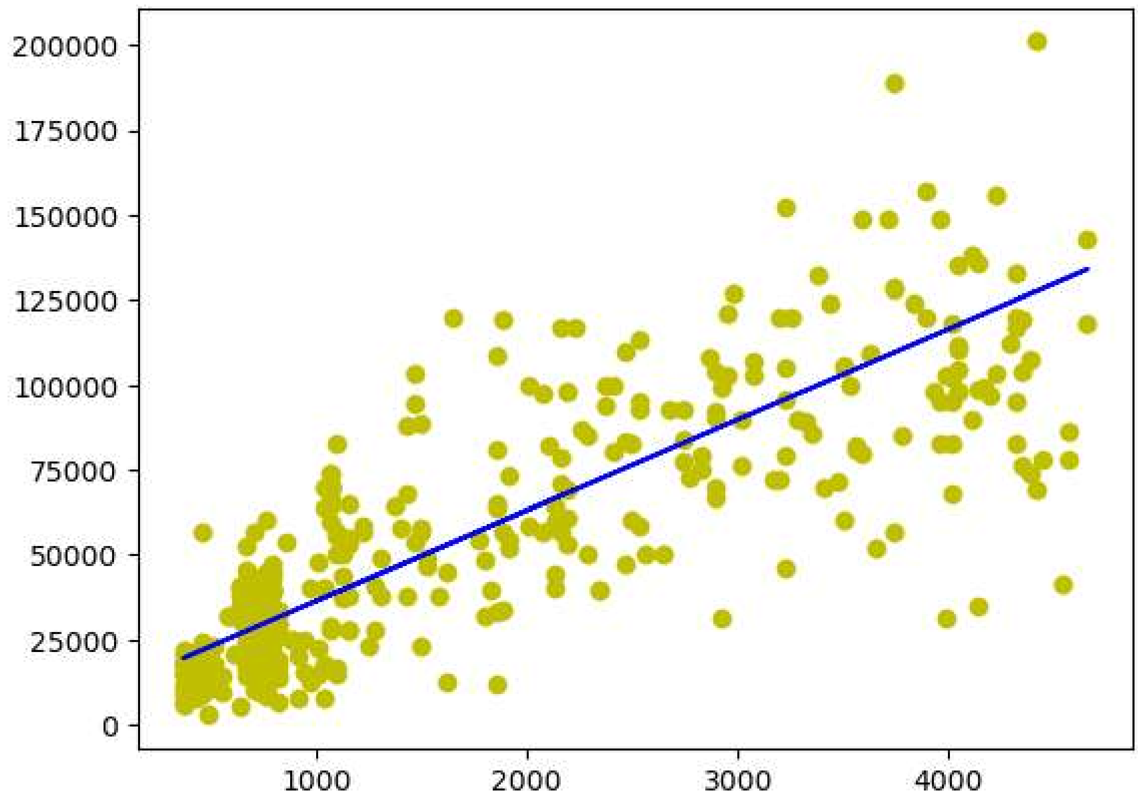
In [22]:
```
df.fillna(method='ffill',inplace=True)
x=np.array(df['age_in_days']).reshape(-1,1)
y=np.array(df['km']).reshape(-1,1)
df.dropna(inplace=True)
```

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 a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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 0.698875263814575
The test score for Linear model is 0.6788811761009449
```

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 0.698875263814575
The test score for ridge model is 0.6788811770230736
```

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 0.6988752638145399
The test score for lasso model is 0.6788812133066009
```

In [28]:
```
#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))
```

```
0.6988752638145734
0.6788811836133852
```

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_tra
#score
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.698875263814575
0.6788811761013169

C:\Users\Y.Saranya\anaconda3\lib\site-packages\sklearn\linear_model\_coordina
te_descent.py:1568: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for e
xample using ravel().
  y = column_or_1d(y, warn=True)
```

In [ ]: