In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing,svm
from sklearn.linear_model import LinearRegression
```

In [3]:
```
df=pd.read_csv(r"C:\Users\Lenovo\OneDrive\Desktop\Data Sets\fiat500_VehicleSele
df
```

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
...	...	...	...	...	...	...	...	...
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 [4]:
```
df.head()
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700

In [5]:
```
df.tail()
```

Out[5]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	p
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7

In [6]:
```
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 [8]:
```
df.describe()
```

Out[8]:

	ID	engine_power	age_in_days	km	previous_owners	lat	
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	

In [10]:
```
df.shape
```

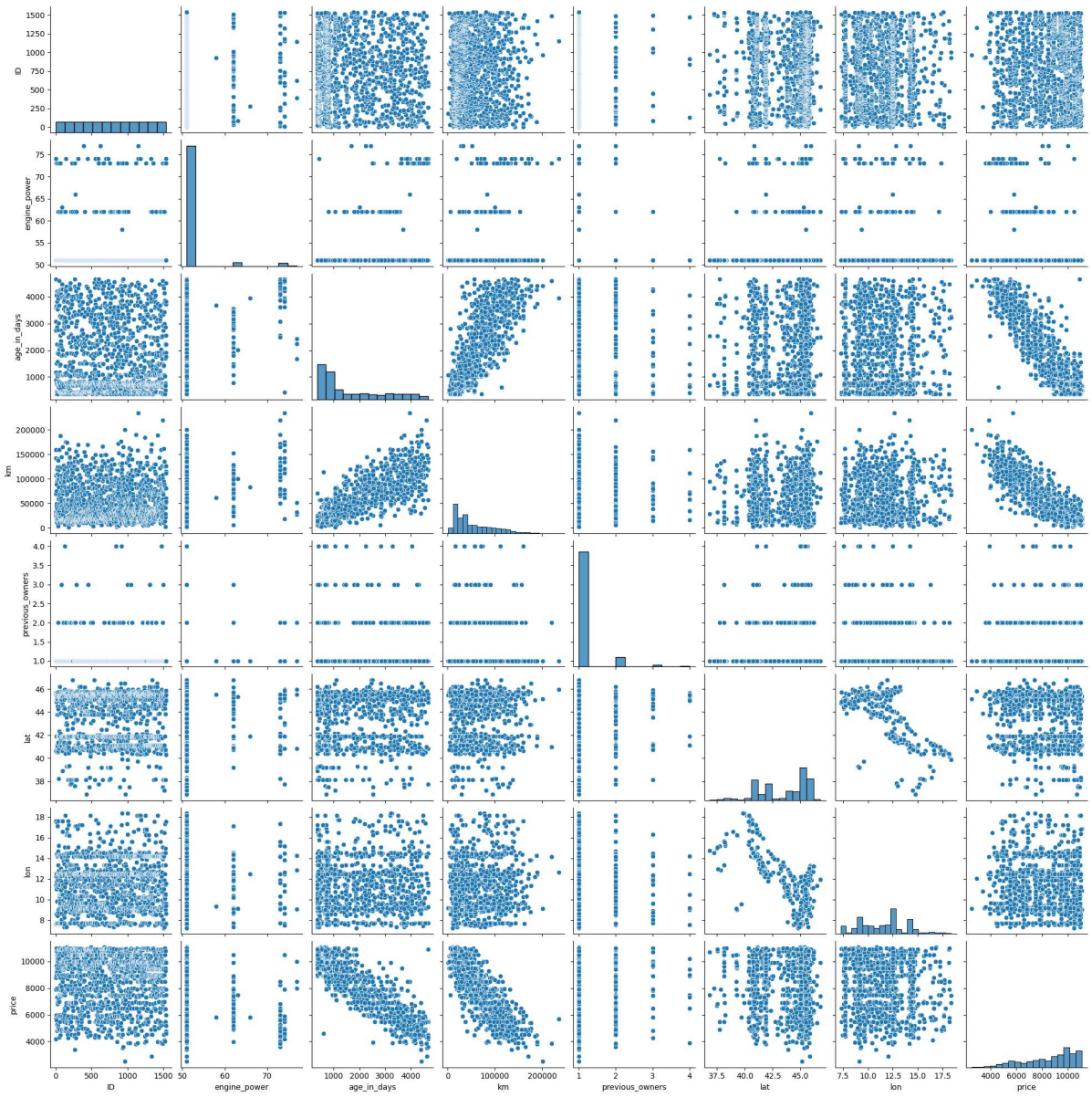
Out[10]:
```
(1538, 9)
```

In [12]:
```
df.columns
```

Out[12]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

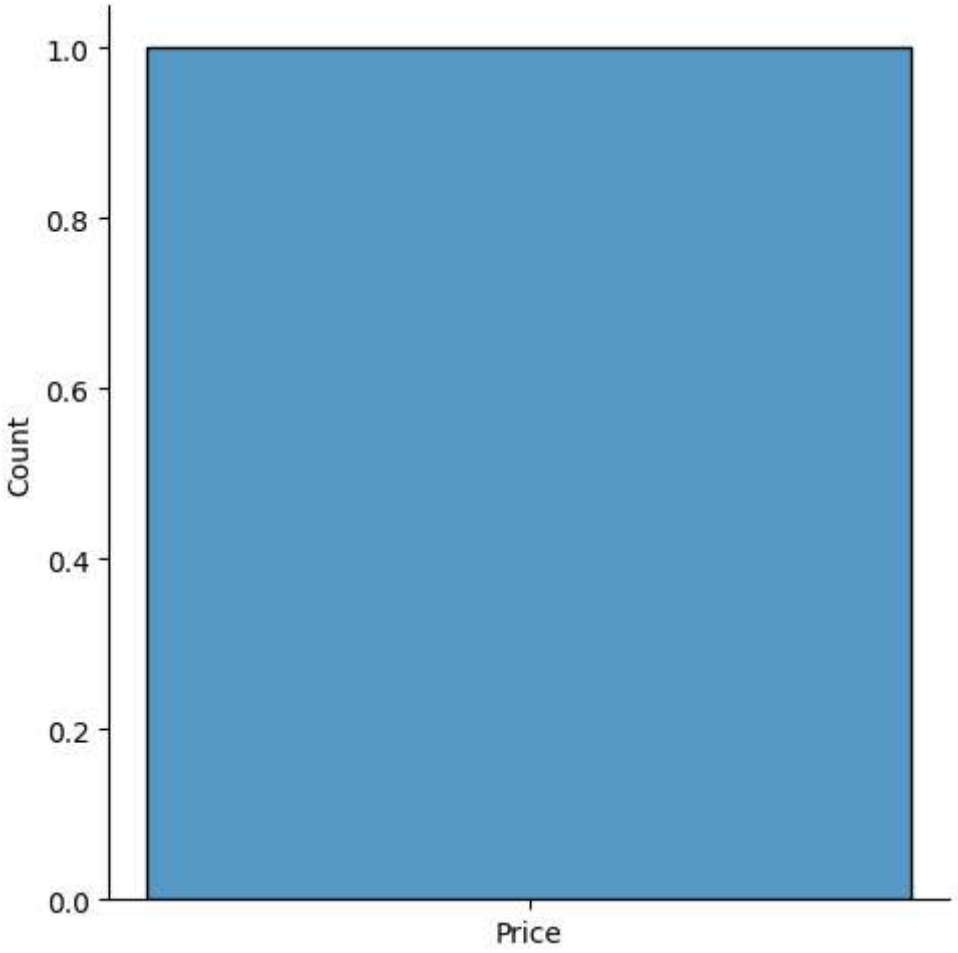
In [13]:
```
sns.pairplot(df)
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x2c2345047d0>
```



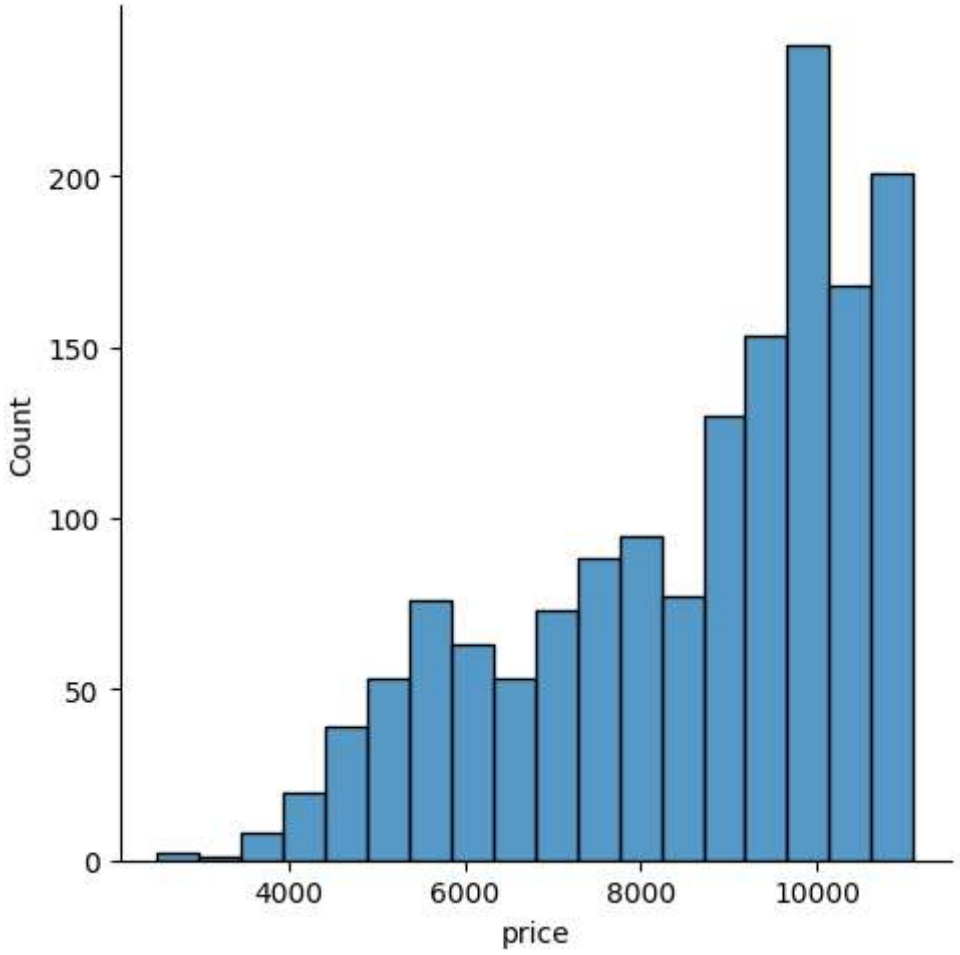
In [14]:
```
sns.displot(['Price'])
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x2c24bce64d0>
```



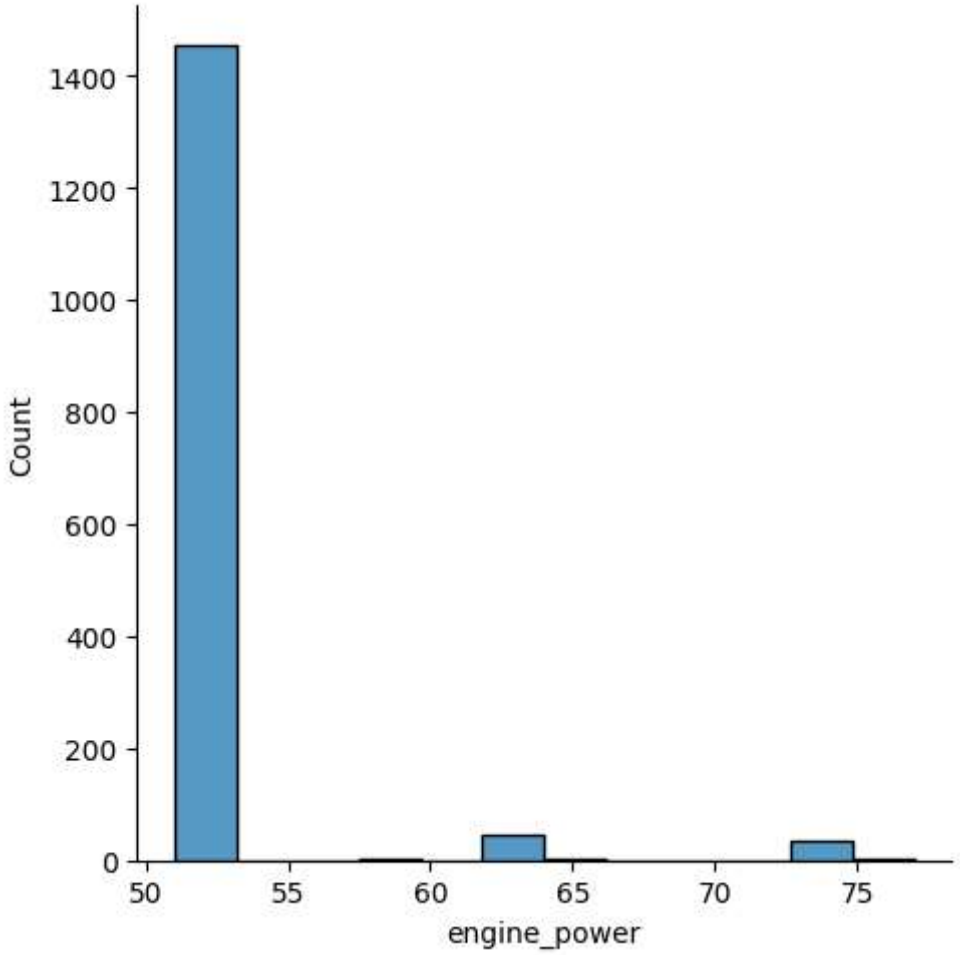
In [15]:
```
sns.displot(df['price'])
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x2c24bac3e10>
```



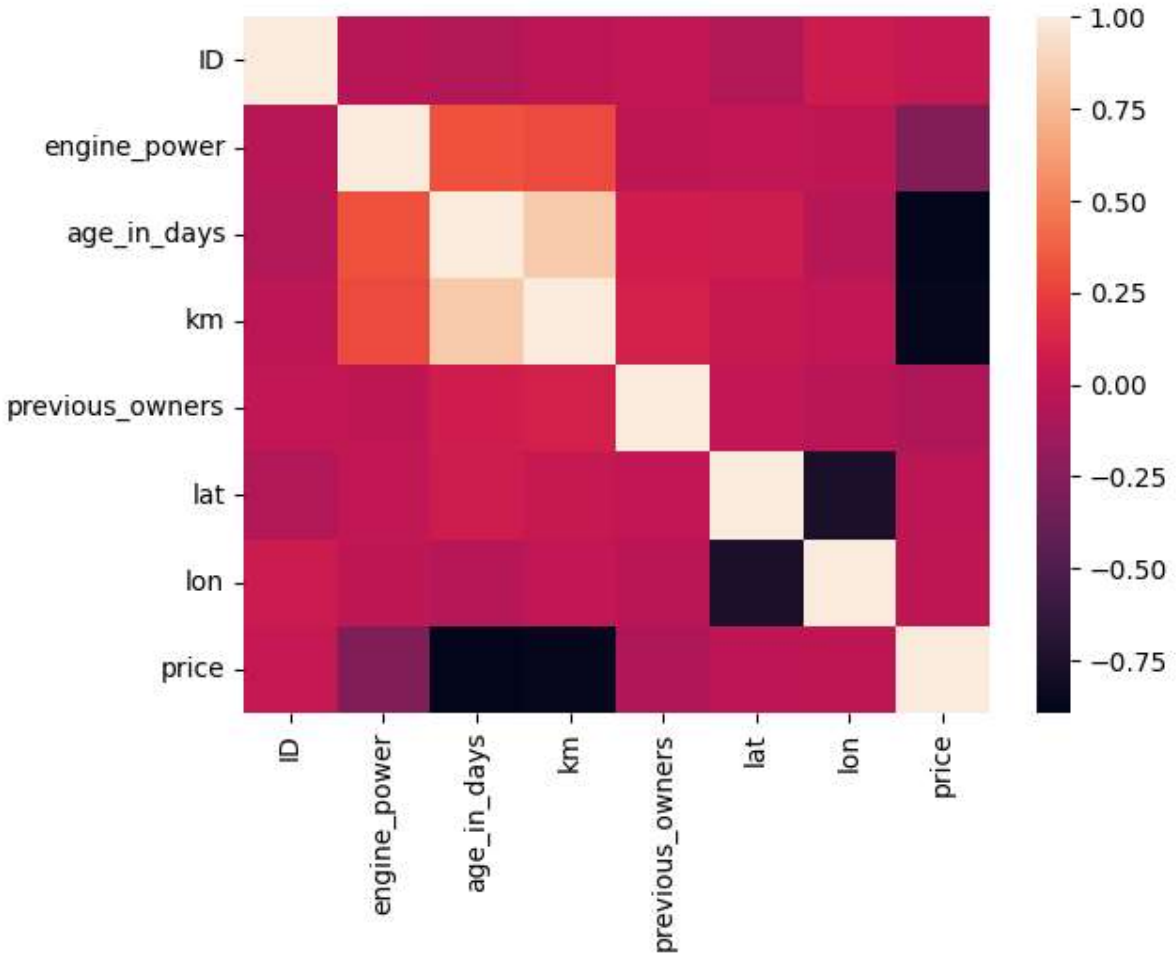
In [16]:
```
sns.displot(df['engine_power'])
```

Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x2c24bd28650>
```



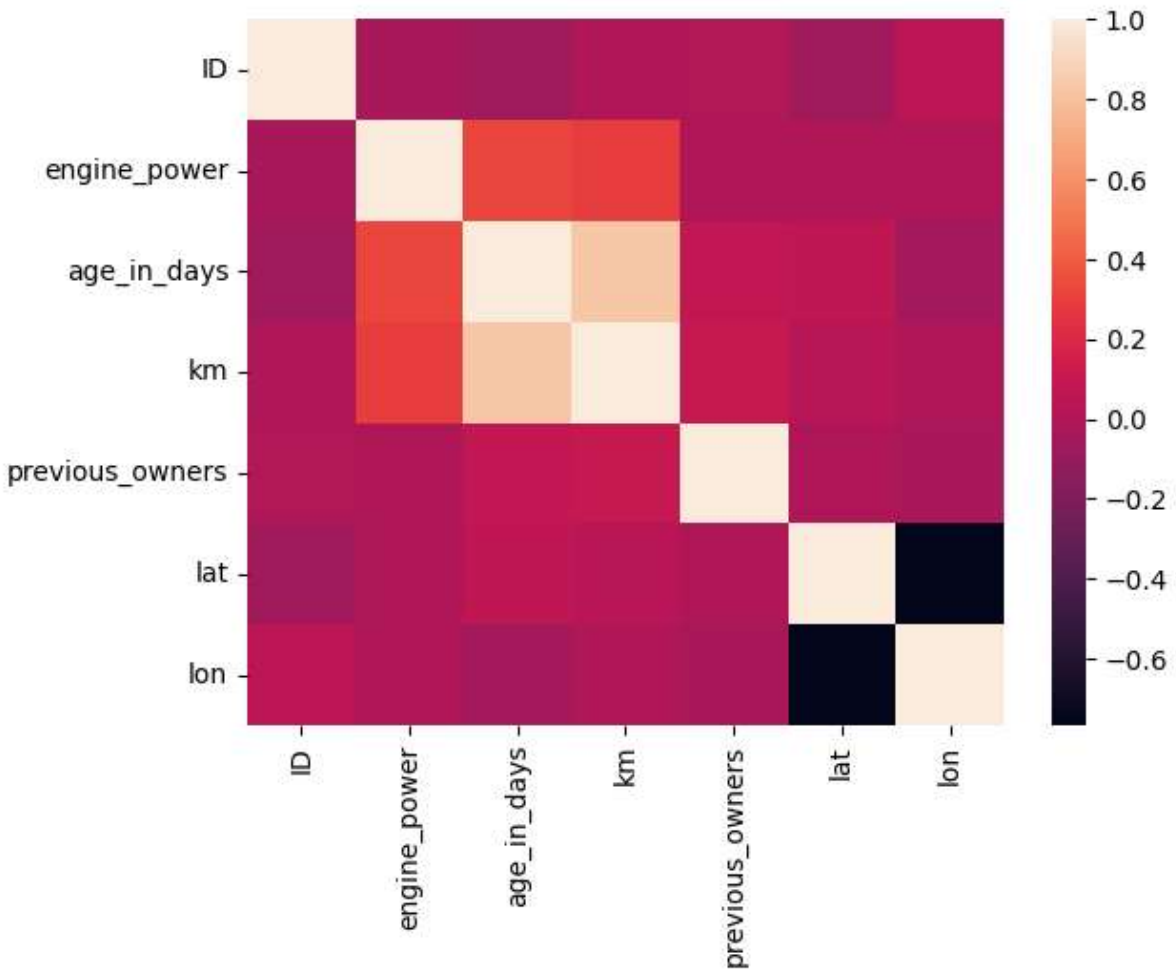
In [17]:
```
fiatdf=df[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
 'lat', 'lon','price']]
sns.heatmap(fiatdf.corr())
```

Out[17]: <Axes: >



In [18]:
```
fiatdf=df[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
 'lat', 'lon']]
sns.heatmap(fiatdf.corr())#without price
```

Out[18]: <Axes: >



In [19]:
```
X=fiatdf[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
 'lat', 'lon']]
y=df['price']
```

In [20]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.intercept_)
```

```
8971.195683500262
```

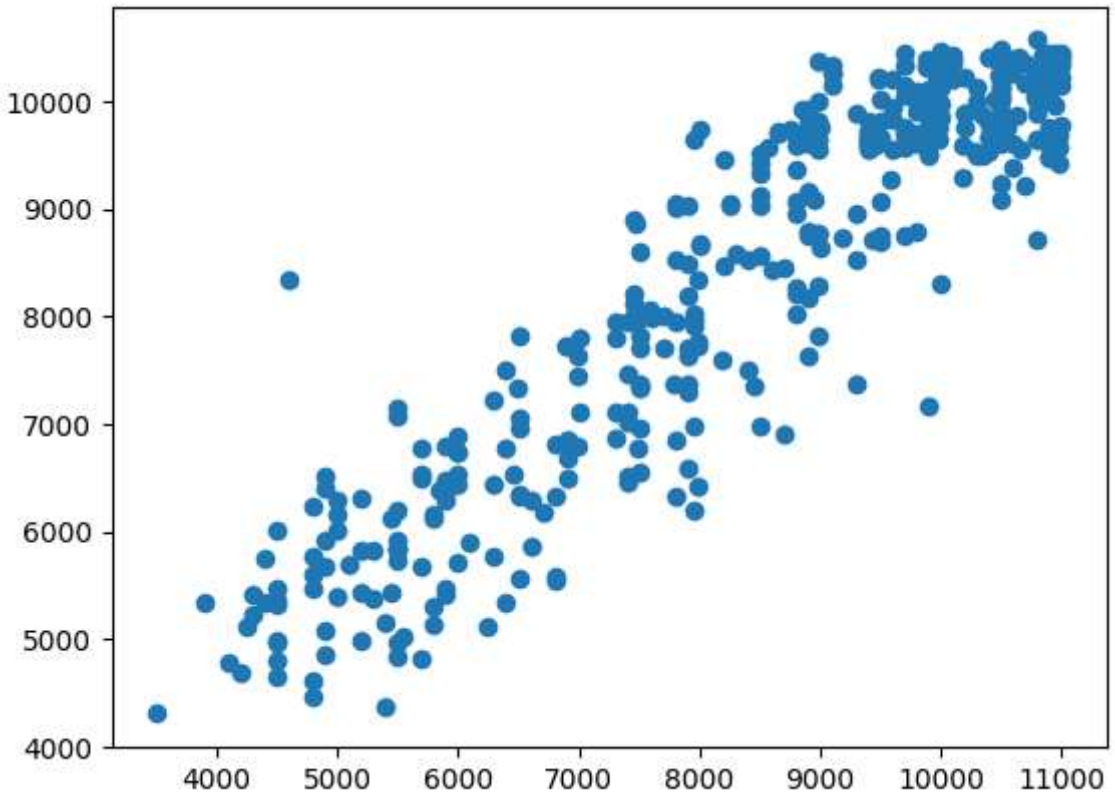
In [21]:
```
coeff_df=pd.DataFrame(regr.coef_,X.columns,columns=['coefficient'])
coeff_df
```

Out[21]:

	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

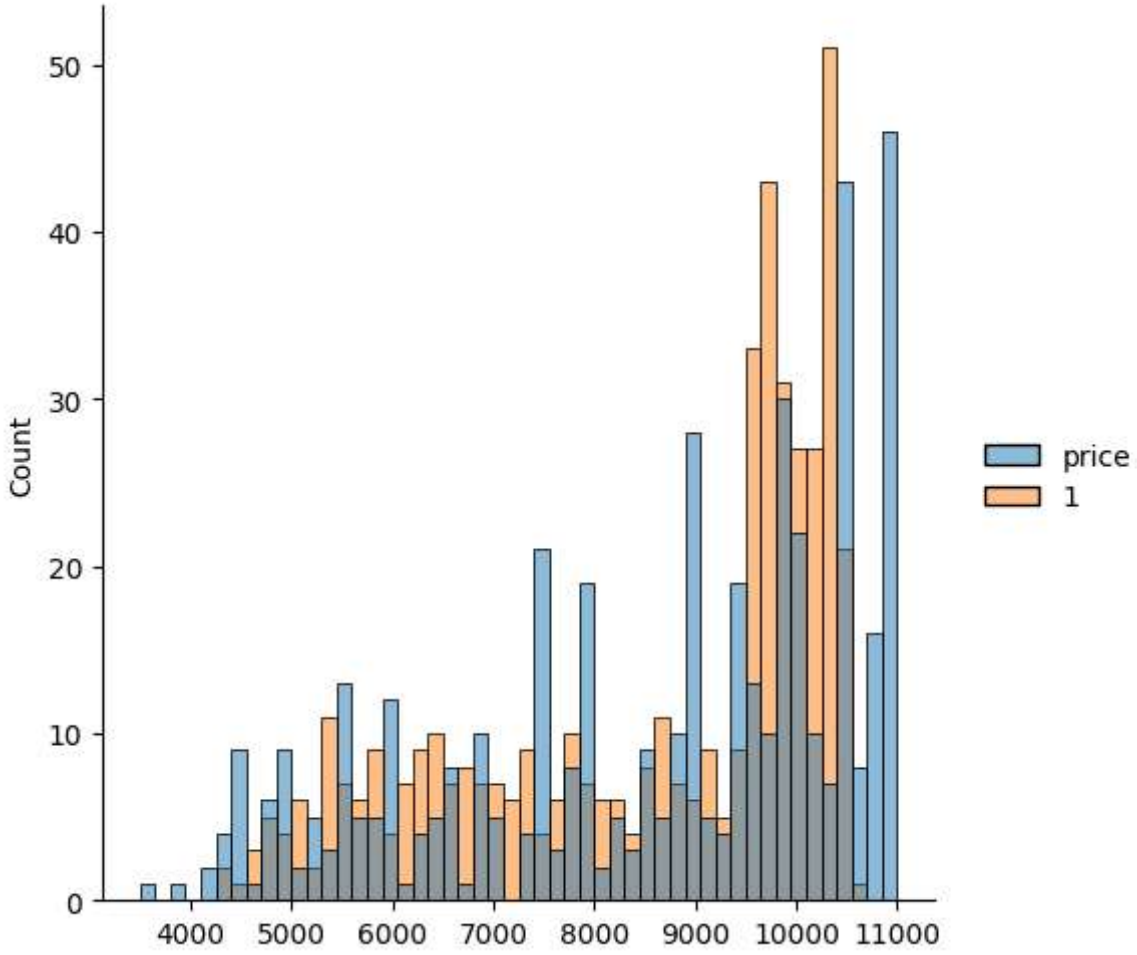
In [22]:
```
predictions=regr.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[22]: <matplotlib.collections.PathCollection at 0x2c24e17fa50>

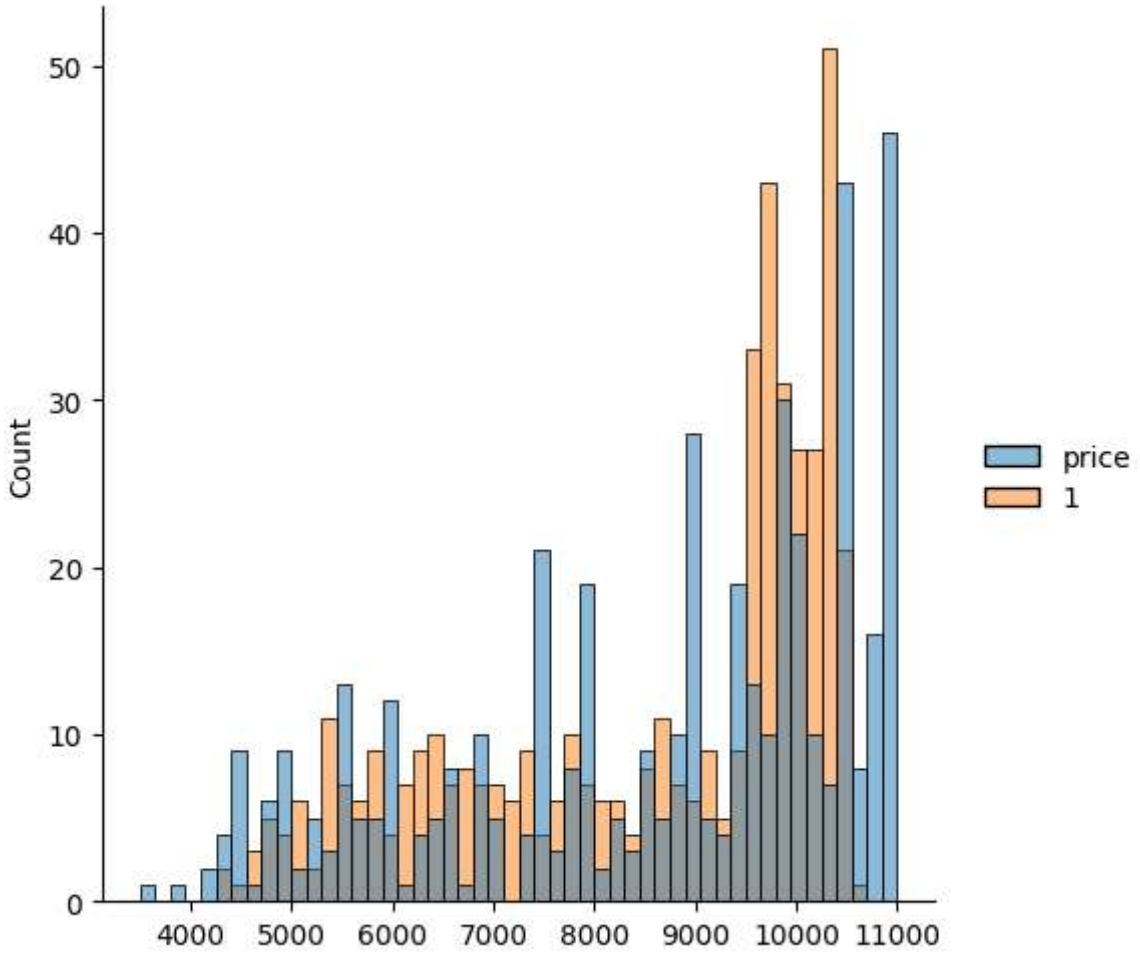


In [23]:
```
sns.displot((y_test,predictions),bins=50)#without semicolon
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x2c24de1c7d0>



In [24]:
```
sns.displot((y_test,predictions),bins=50);#with semicolon
```



In [25]:
```
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: 593.0876179519996
MSE: 551442.6799691911
MAE: 742.59186635001
```

In [26]:
```
#accuracy
regr=LinearRegression()
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
print(regr.score(X_test,y_test))
```

```
0.859713670430884
```

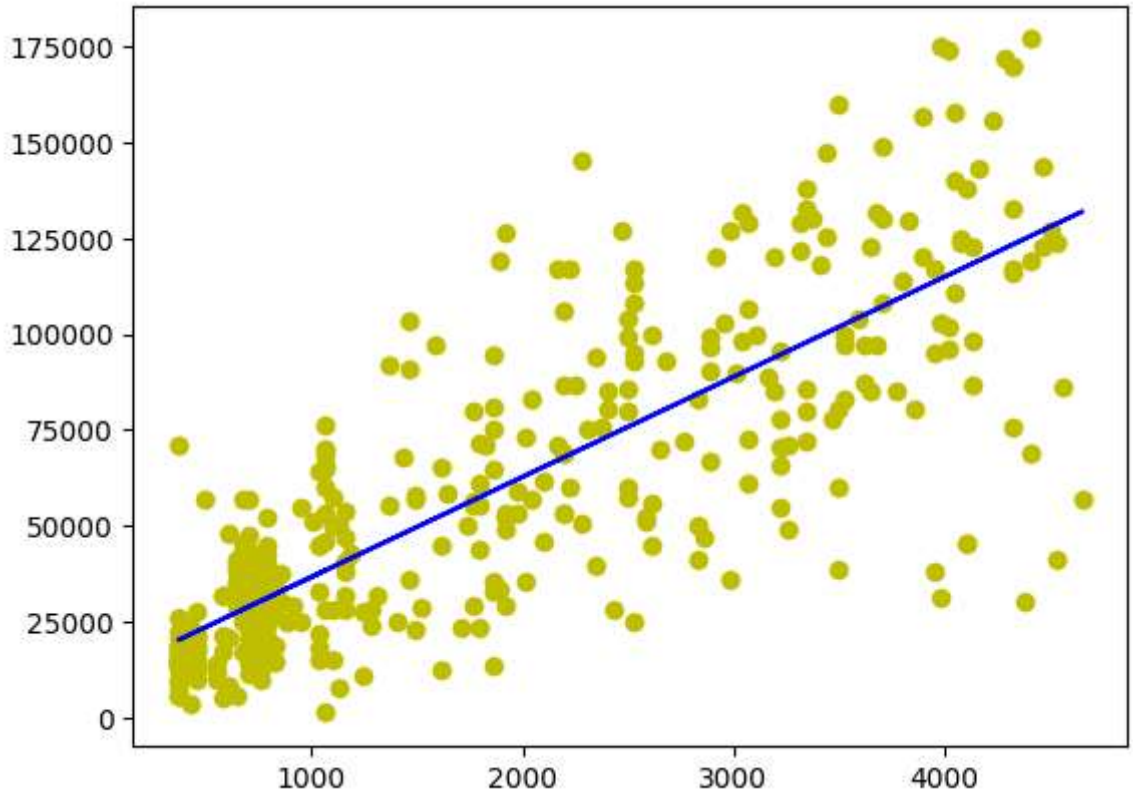
In [27]:
```
df.fillna(method='ffill',inplace=True)
```

In [28]:
```
x=np.array(df['age_in_days']).reshape(-1,1)
y=np.array(df['km']).reshape(-1,1)
df.dropna(inplace=True)
```

In [29]:
```
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[29]:
```
LinearRegression
LinearRegression()
```

In [30]:
```
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 [ ]:
```
```