In [1]:

```
pip install numpy
```

```
Collecting numpyNote: you may need to restart the kernel to use updated pack
ages.

  Downloading numpy-1.24.3-cp311-cp311-win_amd64.whl (14.8 MB)
     ---------------------------------------- 14.8/14.8 MB 789.5 kB/s eta 0:0
0:00
Installing collected packages: numpy
Successfully installed numpy-1.24.3


[notice] A new release of pip available: 22.3.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:

```
pip install pandas
```

```
Collecting pandas
  Downloading pandas-2.0.1-cp311-cp311-win_amd64.whl (10.6 MB)
     ---------------------------------------- 10.6/10.6 MB 849.5 kB/s eta 0:0
0:00
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\venka\appd
ata\local\programs\python\python311\lib\site-packages (from pandas) (2.8.2)
Collecting pytz>=2020.1
  Downloading pytz-2023.3-py2.py3-none-any.whl (502 kB)
     -------------------------------------- 502.3/502.3 kB 955.5 kB/s eta 0:0
0:00
Collecting tzdata>=2022.1
  Downloading tzdata-2023.3-py2.py3-none-any.whl (341 kB)
     -------------------------------------- 341.8/341.8 kB 559.2 kB/s eta 0:0
0:00
Requirement already satisfied: numpy>=1.21.0 in c:\users\venka\appdata\local
\programs\python\python311\lib\site-packages (from pandas) (1.24.3)
Requirement already satisfied: six>=1.5 in c:\users\venka\appdata\local\prog
rams\python\python311\lib\site-packages (from python-dateutil>=2.8.2->panda
s) (1.16.0)
Installing collected packages: pytz, tzdata, pandas
Successfully installed pandas-2.0.1 pytz-2023.3 tzdata-2023.3
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip available: 22.3.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [58]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [59]:
```
df=pd.read_csv(r"C:\Users\venka\OneDrive\Documents\fiat500_VehicleSelection_Da
print(df)
```

```
        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 [60]:
```
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 [61]:
```
df.head(10)
```

Out[61]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pri
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	107
7	8	lounge	51	1521	49076	1	41.903221	12.495650	91
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 [62]:
```
df.tail()
```

Out[62]:

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

In [63]:
```
df.describe()
```

Out[63]:

	ID	engine_power	age_in_days	km	previous_owners	lat
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
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 [64]:

```
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 [65]:

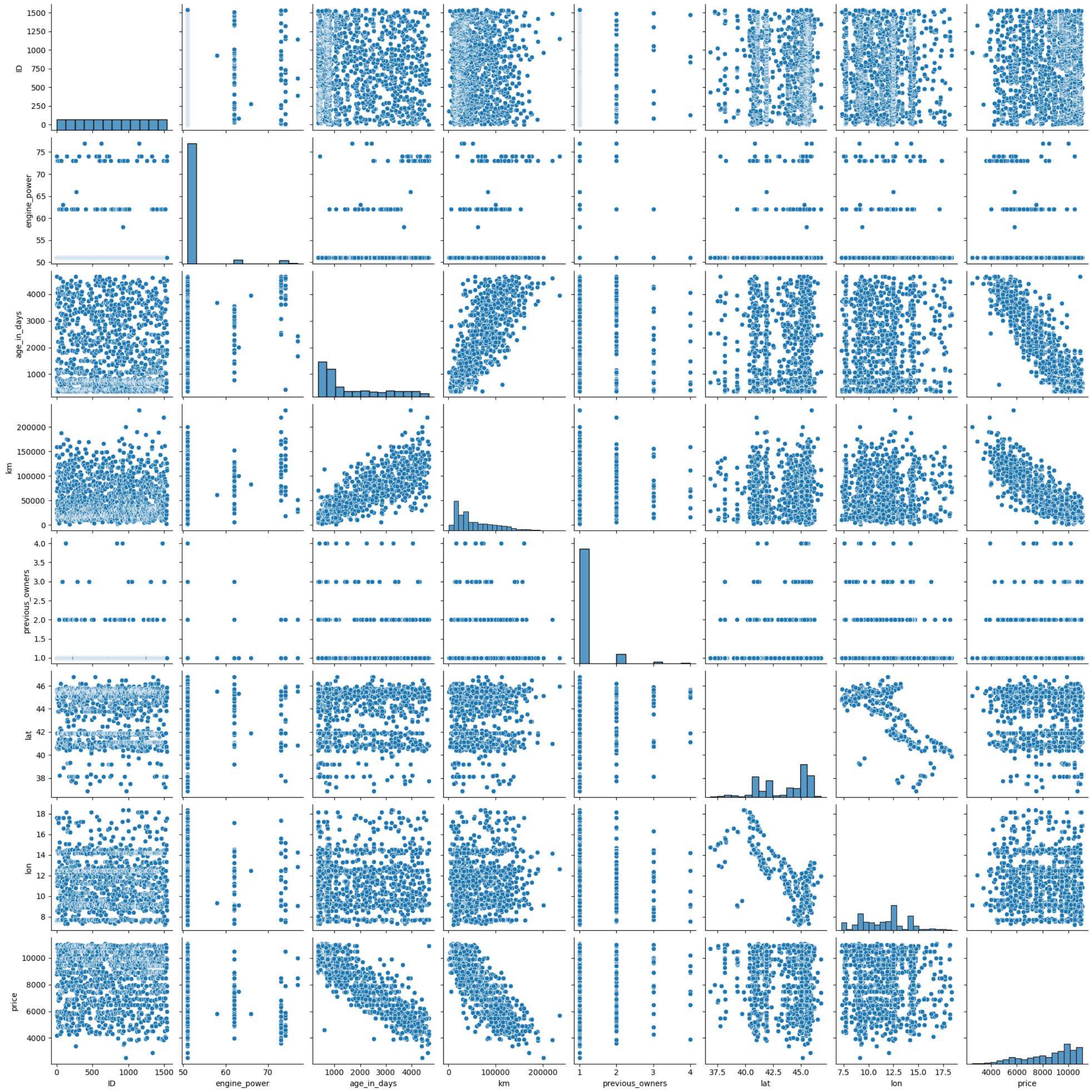
```
df.columns
```

Out[65]:

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

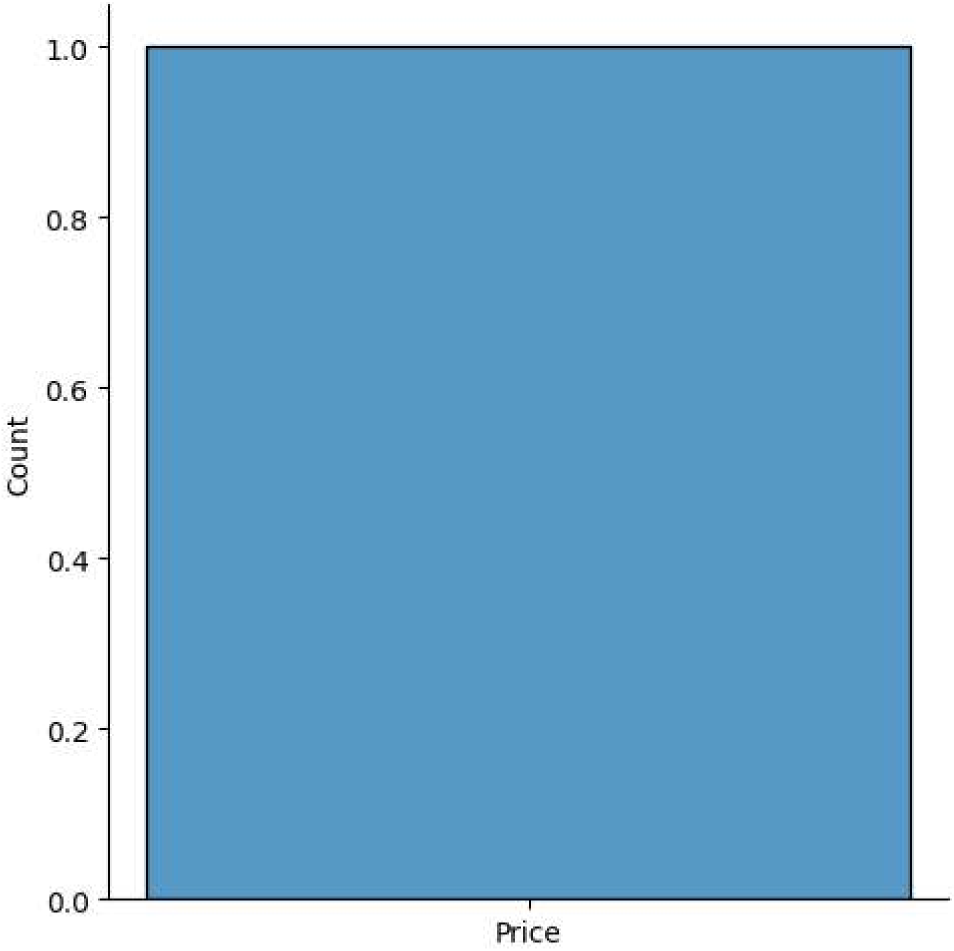
In [66]:
```
sns.pairplot(df)
```

Out[66]:
```
<seaborn.axisgrid.PairGrid at 0x16973151f50>
```



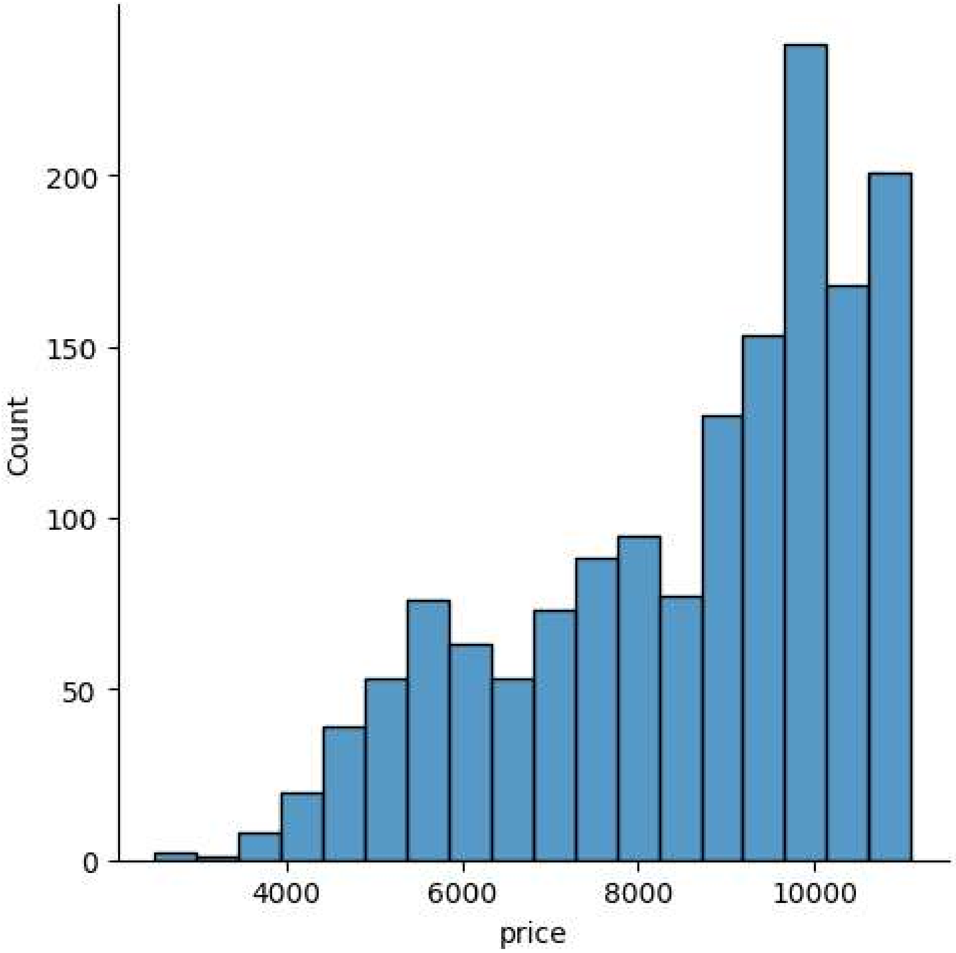
In [67]:
```
sns.displot(['Price'])
```

Out[67]:
```
<seaborn.axisgrid.FacetGrid at 0x1697334d190>
```



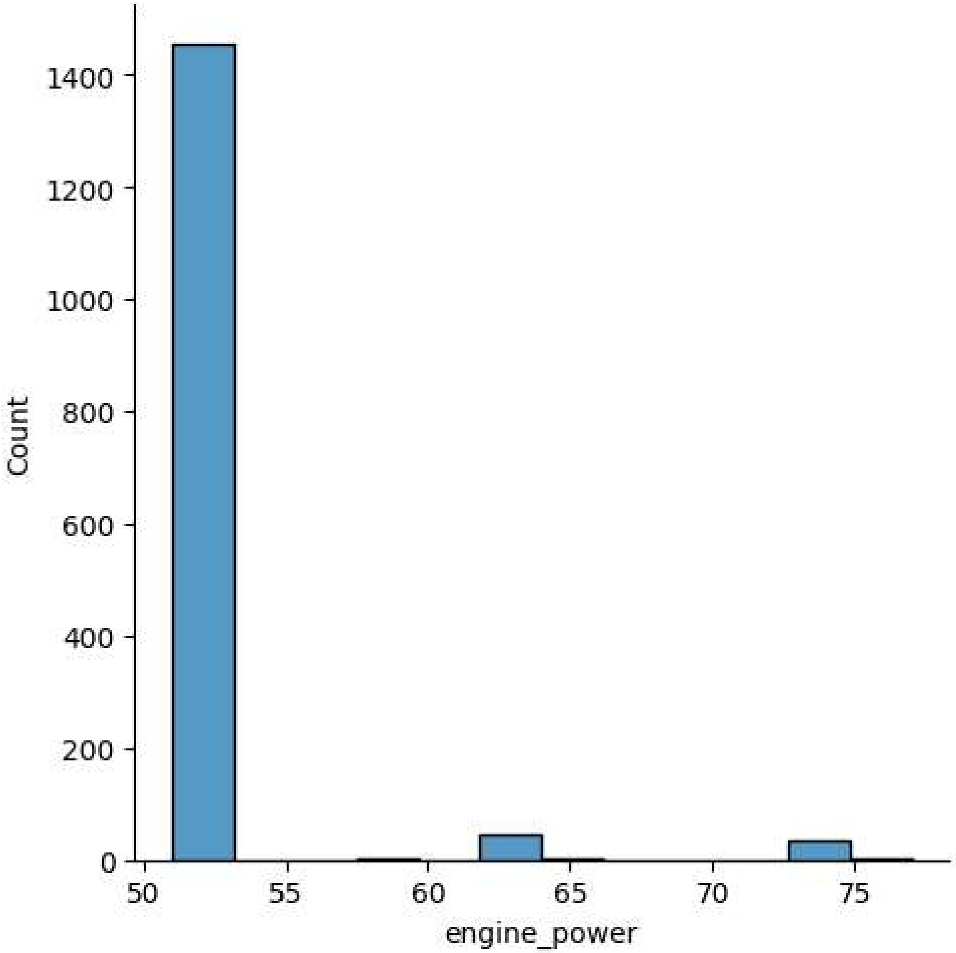
```
In [68]: sns.displot(df['price'])
```

```
Out[68]: <seaborn.axisgrid.FacetGrid at 0x1696f199c50>
```



In [69]:
```
sns.displot(df['engine_power'])
```

Out[69]:
```
<seaborn.axisgrid.FacetGrid at 0x169788e62d0>
```

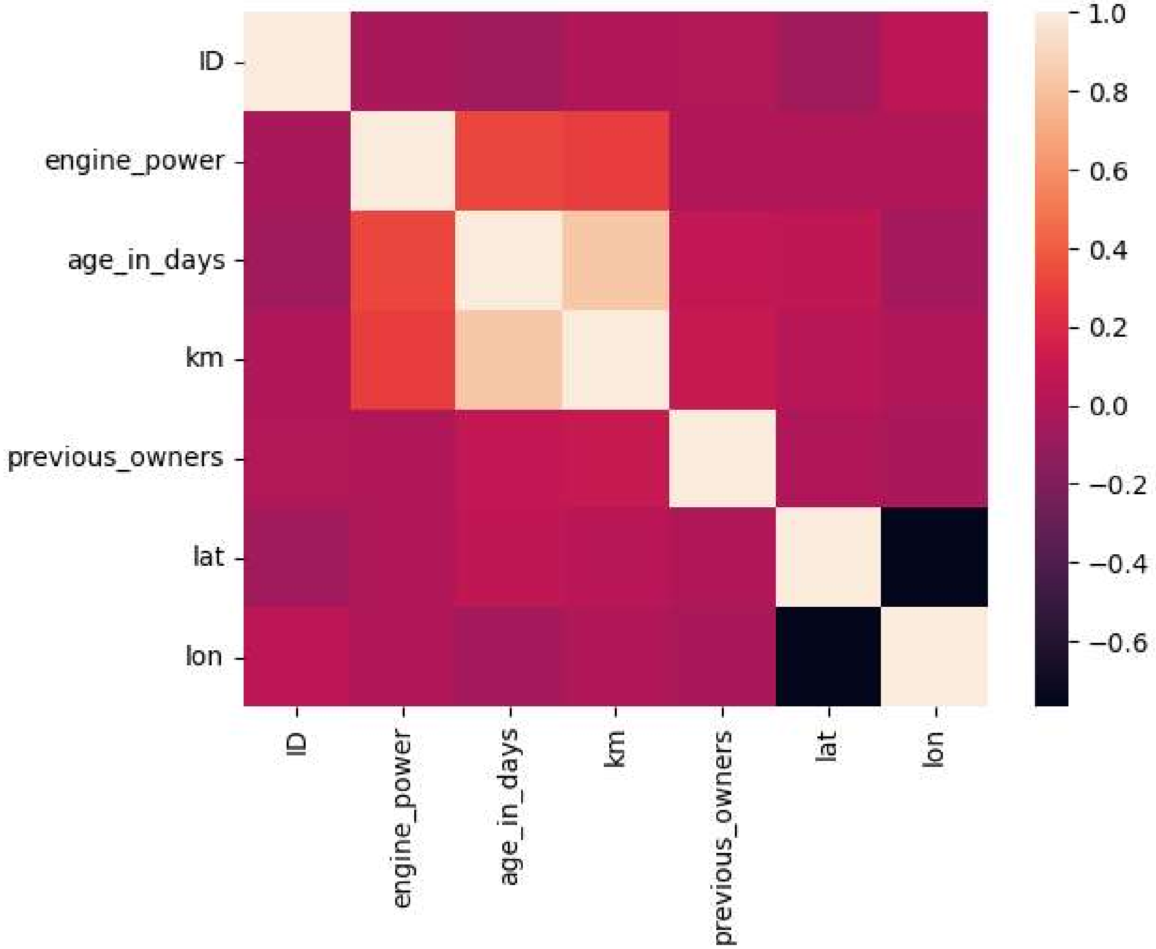


In [70]:

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

Out[70]:

```
<Axes: >
```



In [45]:

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

In [71]:

```
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
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.intercept_)
```

```
8971.19568349988
```

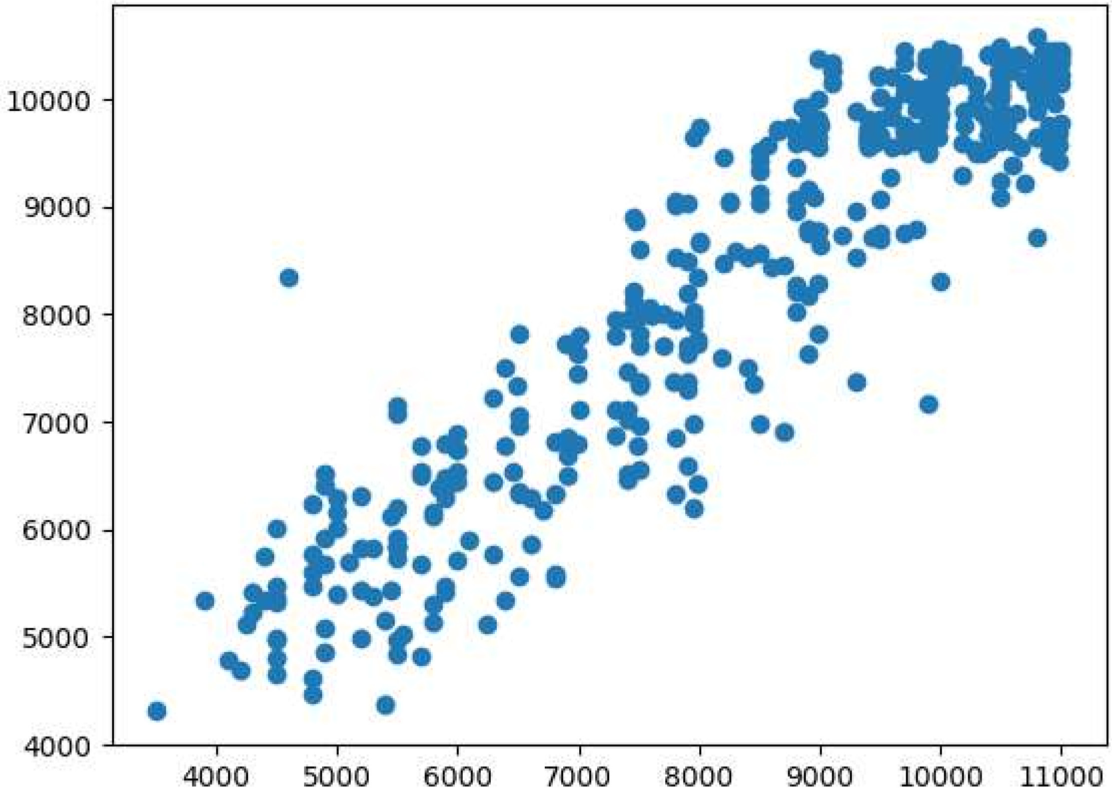
In [72]:
```
coeff_df=pd.DataFrame(regr.coef_,X.columns,columns=['coefficient'])
coeff_df
```

Out[72]:

	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

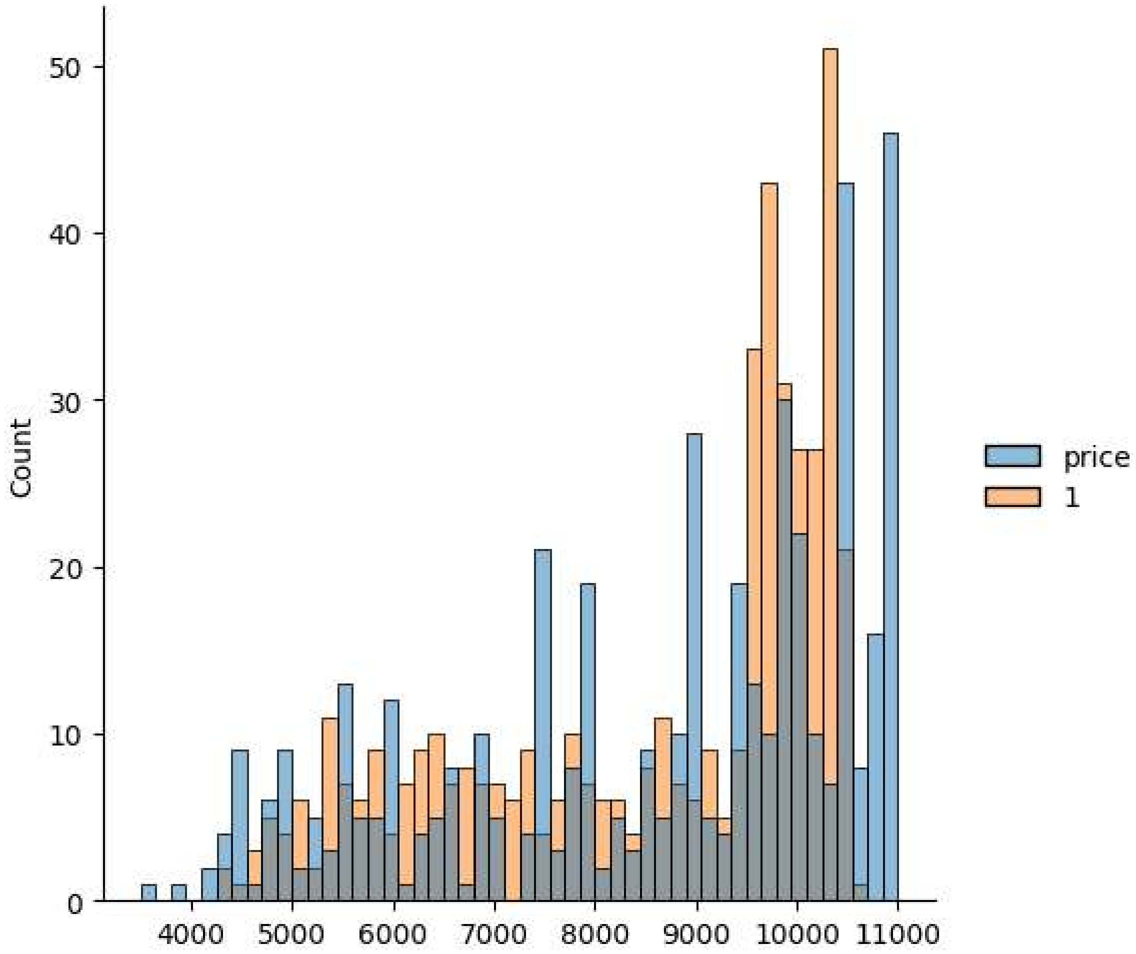
In [73]:
```
predictions=regr.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[73]:
```
<matplotlib.collections.PathCollection at 0x1697abcf590>
```



In [74]:
```
sns.displot((y_test,predictions),bins=50)
```

Out[74]:
```
<seaborn.axisgrid.FacetGrid at 0x1697ab66a50>
```



In [75]:
```
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: 593.0876179519931
MSE: 551442.6799691801
MAE: 742.5918663500026
```

In [76]:
```
#accuracy
regr=LinearRegression()
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
print(regr.score(X_test,y_test))
```

```
0.8597136704308868
```

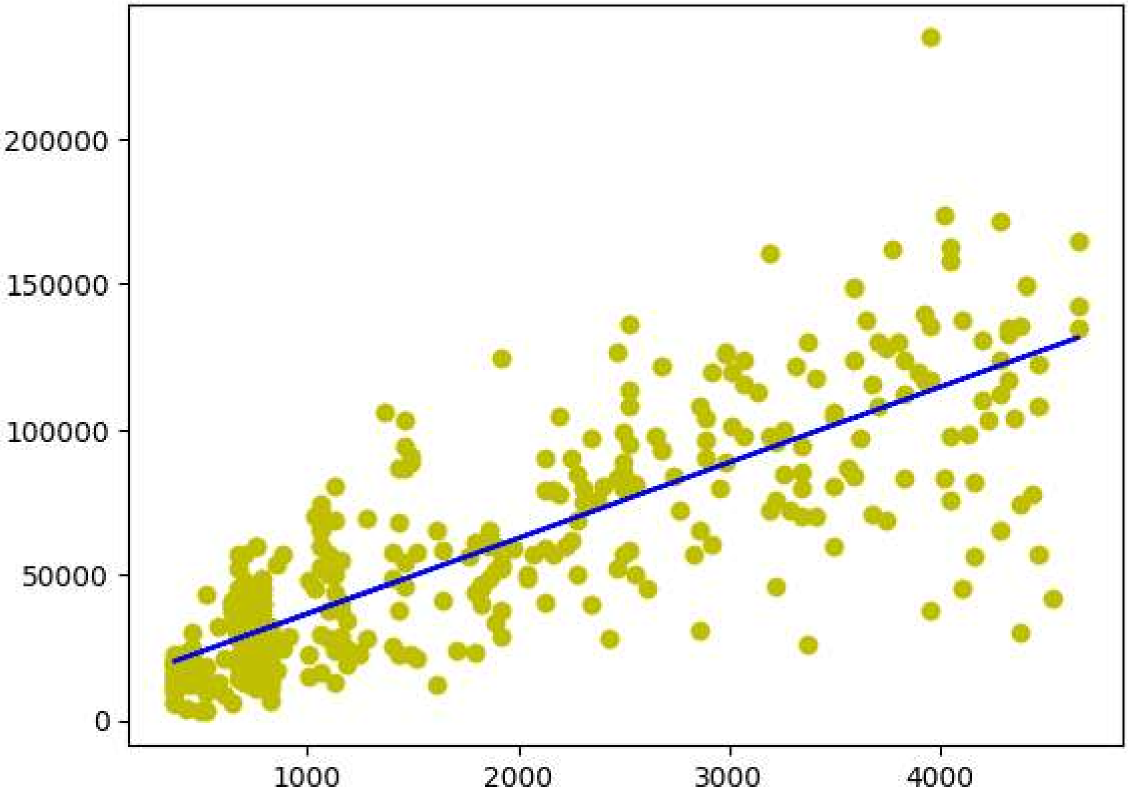
In [77]:
```
df.fillna(method='ffill',inplace=True)
```

In [78]:
```
x=np.array(df['age_in_days']).reshape(-1,1)
y=np.array(df['km']).reshape(-1,1)
df.dropna(inplace=True)
```

In [79]:
```
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[79]:
```
▾ LinearRegression
LinearRegression()
```

In [80]:
```
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 [82]:

```
#elasticnet
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
y_pred_elastic=regr.predict(X_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean Squared Error on test set",mean_squared_error)
```

```
[25.89689696]
[10640.73996329]
Mean Squared Error on test set 2769977842.8600845
```

In [ ]: