In [ ]:

In [ ]:

FIAT500_VEHICLE SELECTION USING ***ELASTIC NET***

In [ ]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv(r"/content/fiat500_VehicleSelection_Dataset (1).csv")
df
```

Out[ ]:

	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 [ ]:

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

Out[ ]:

	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
5	6	pop	74	3623	70225	1	45.000702	7.682270	7900
6	7	lounge	51	731	11600	1	44.907242	8.611560	10750
7	8	lounge	51	1521	49076	1	41.903221	12.495650	9190
8	9	sport	73	4049	76000	1	45.548000	11.549470	5600
9	10	sport	51	3653	89000	1	45.438301	10.991700	6000

In [ ]:

```
df.tail()
```

Out[ ]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pr
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 [ ]:

```
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 [ ]:

```
df.describe()
```

Out[ ]:

	ID	engine_power	age_in_days	km	previous_owners	lat	
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	153
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	1
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	1
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	1
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	1

In [ ]:

```
df.columns
```

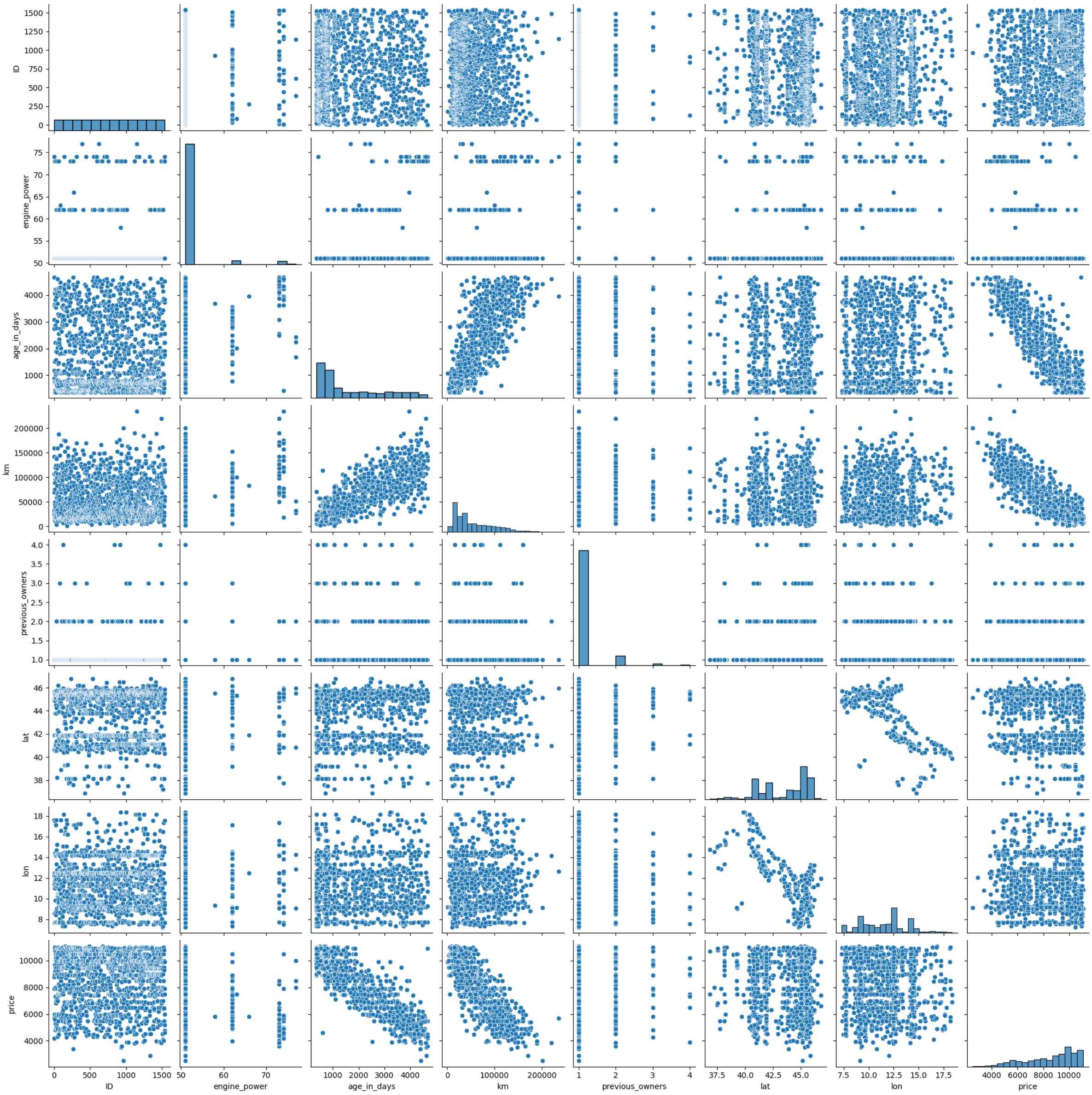
Out[ ]:

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

In [ ]:

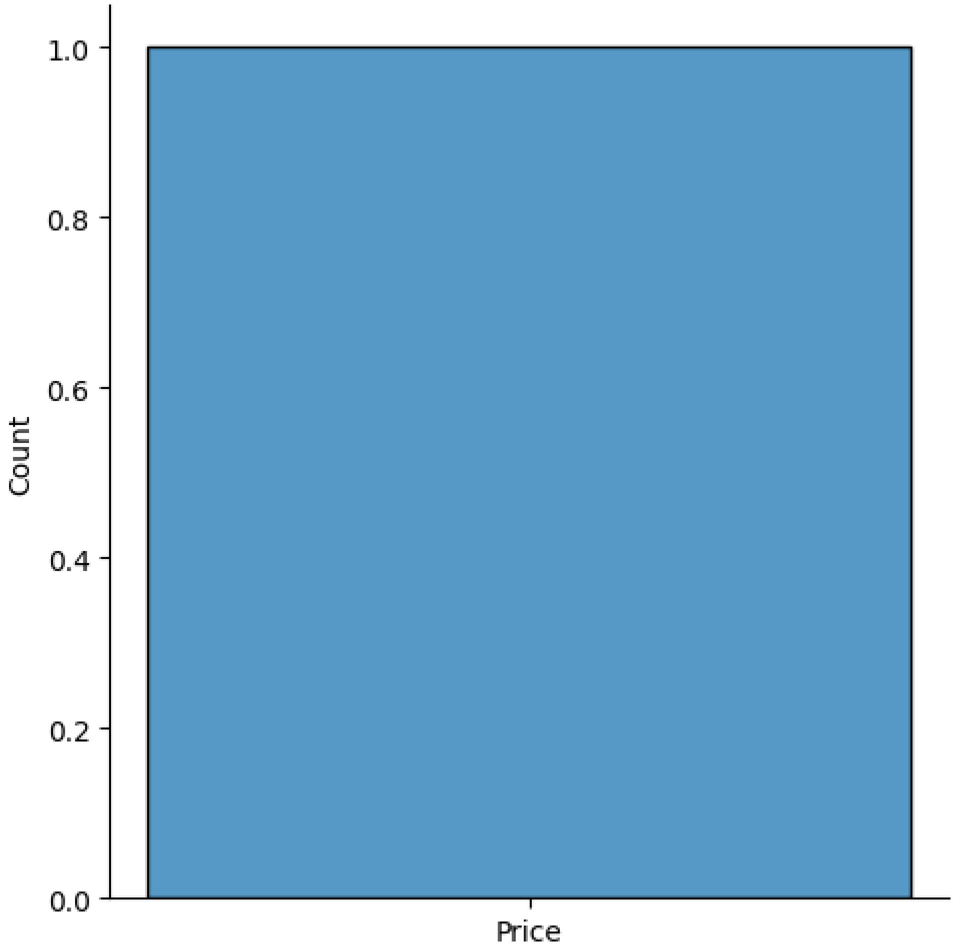
```
#EDA
sns.pairplot(df)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7fed1e05cb20>
```



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

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fed19247070>
```

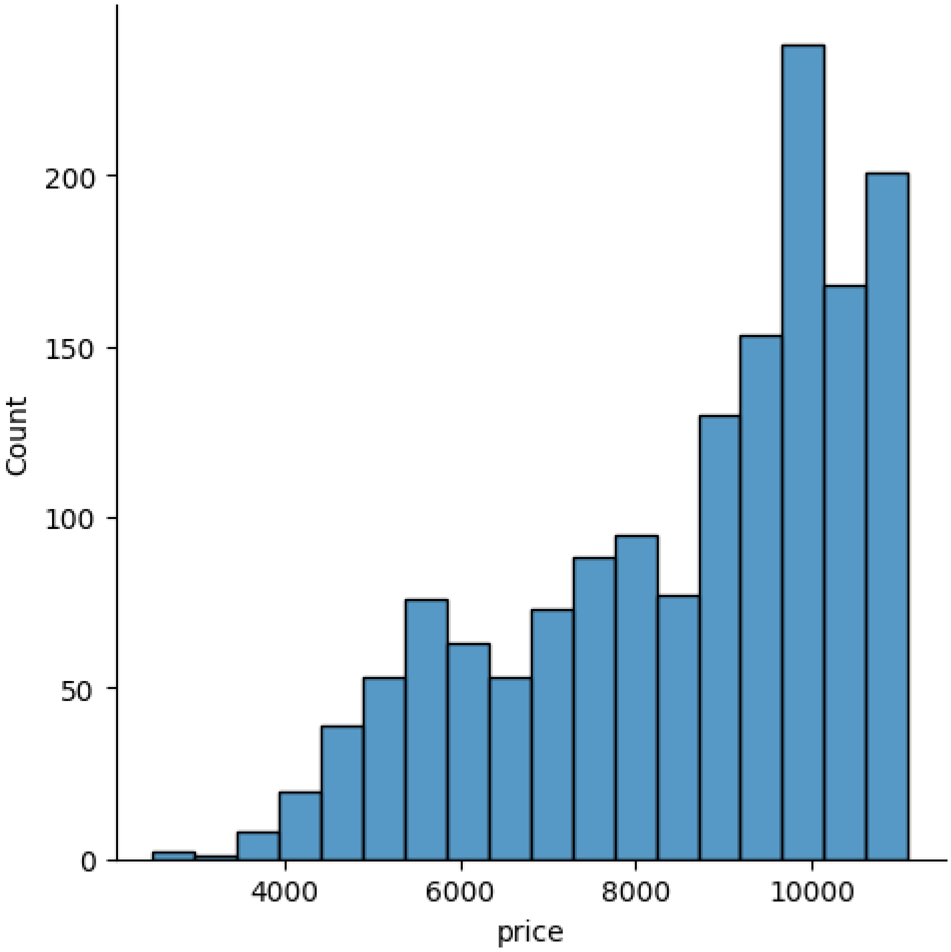


In [ ]:

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

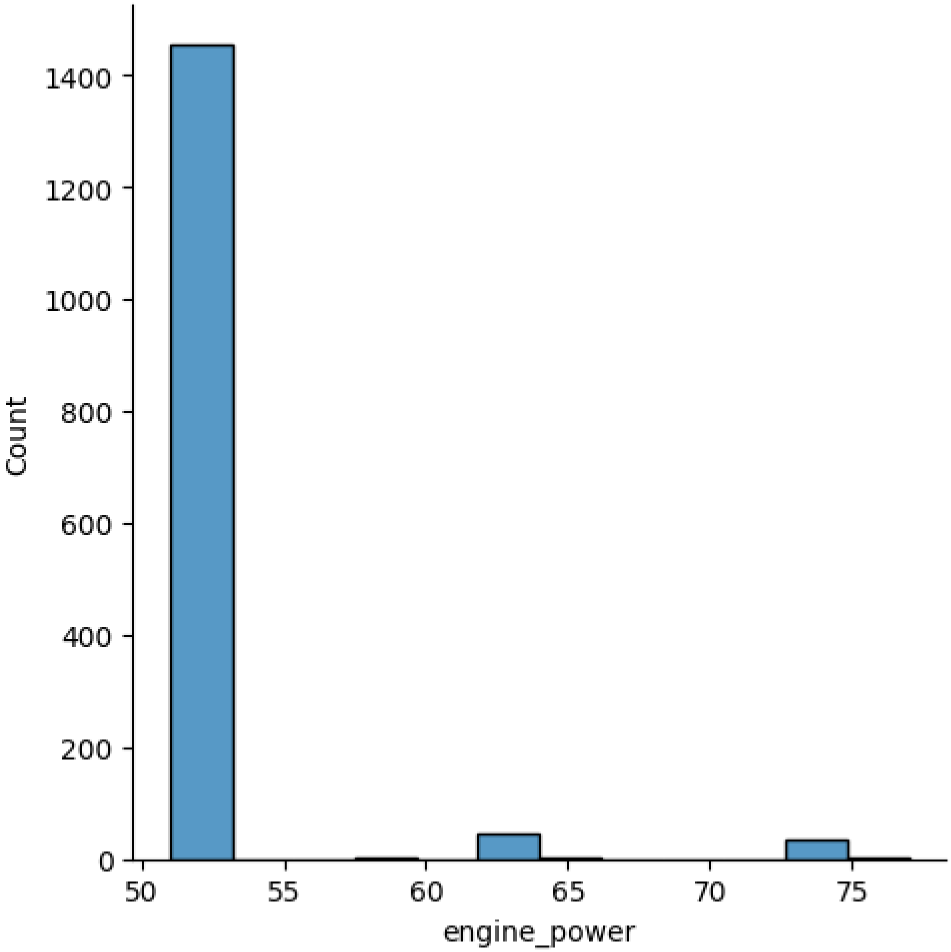
Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7fed19246b30>
```



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

```
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7fed16a63bb0>
```

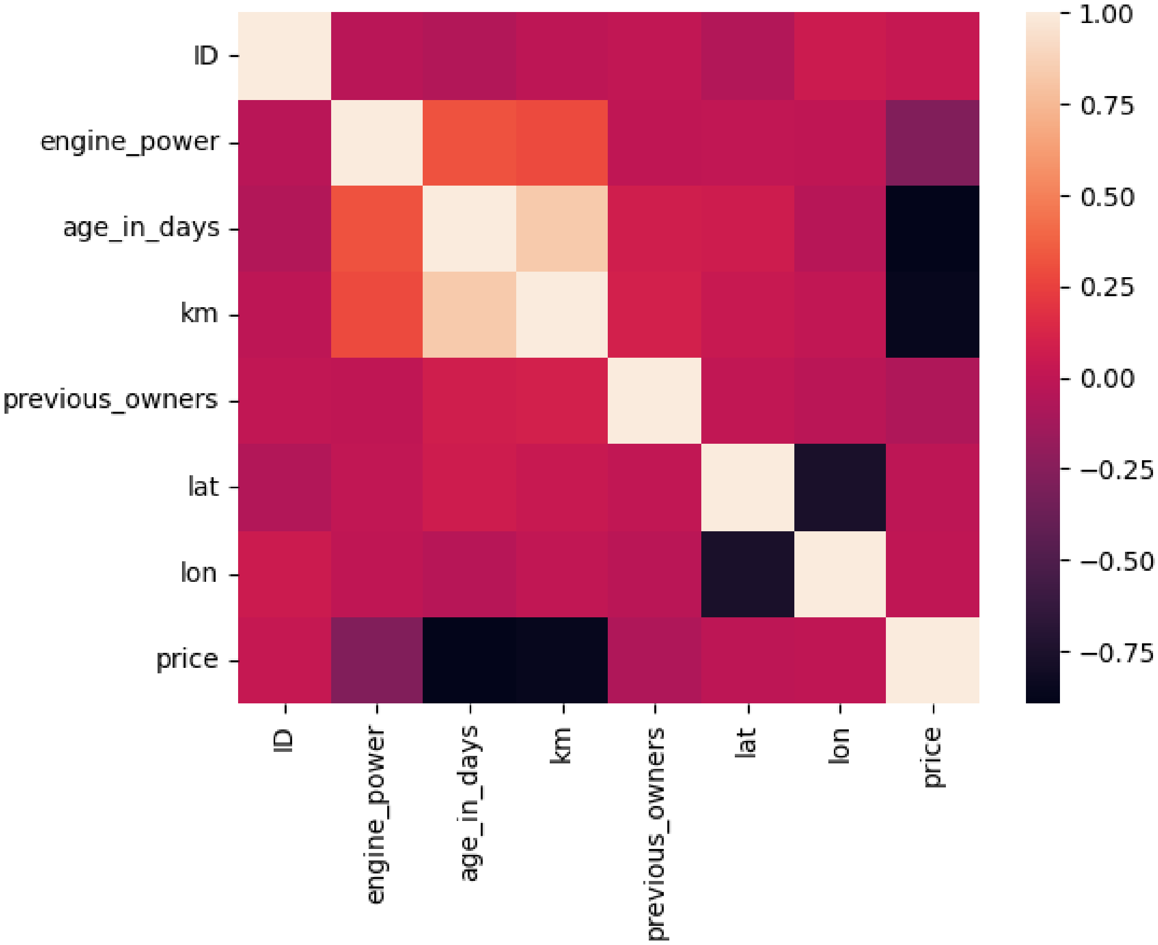


In [ ]:

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

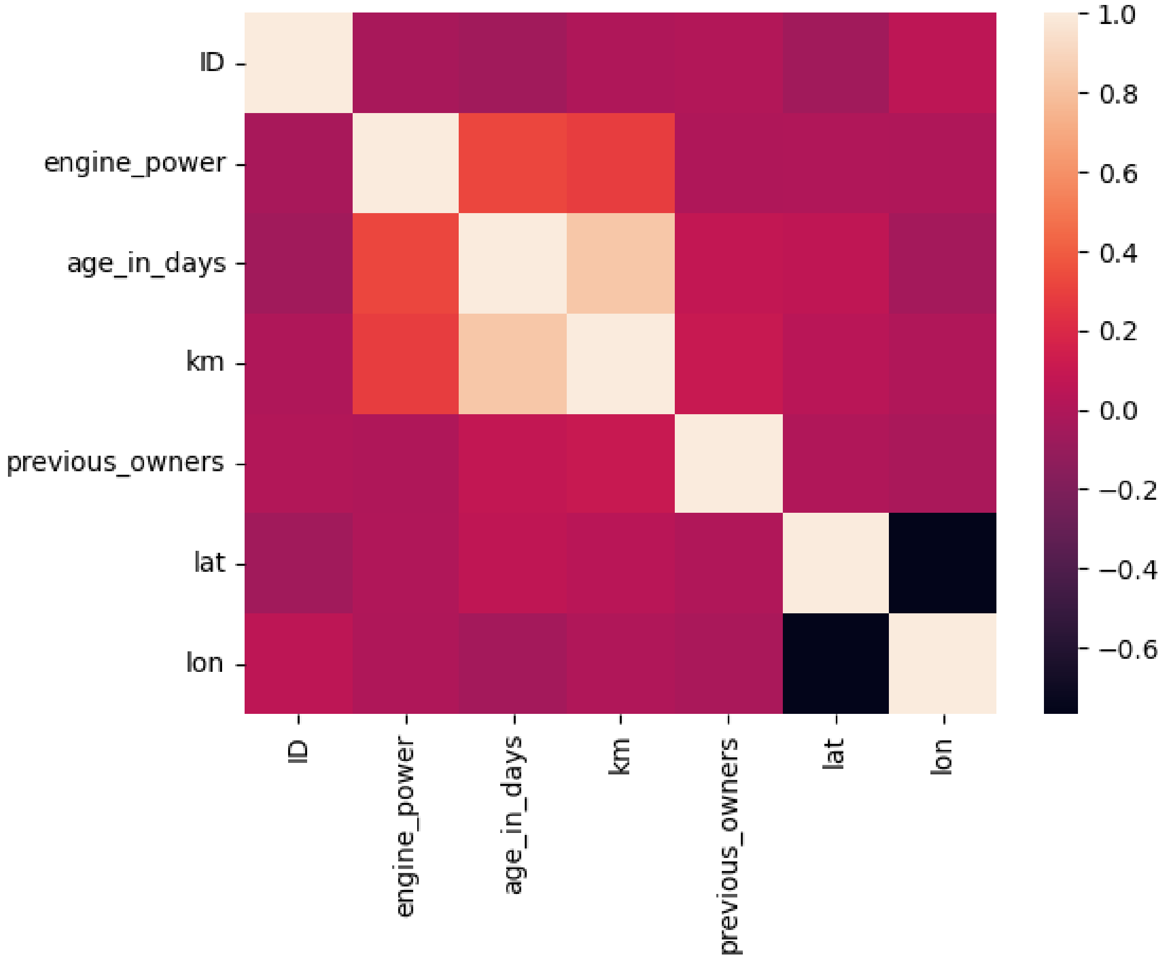
Out[ ]:

```
<Axes: >
```



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

Out[ ]: <Axes: >
```



In [ ]:

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

In [ ]:

```
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=101)
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.intercept_)
```

```
8971.195683500588
```

In [ ]:

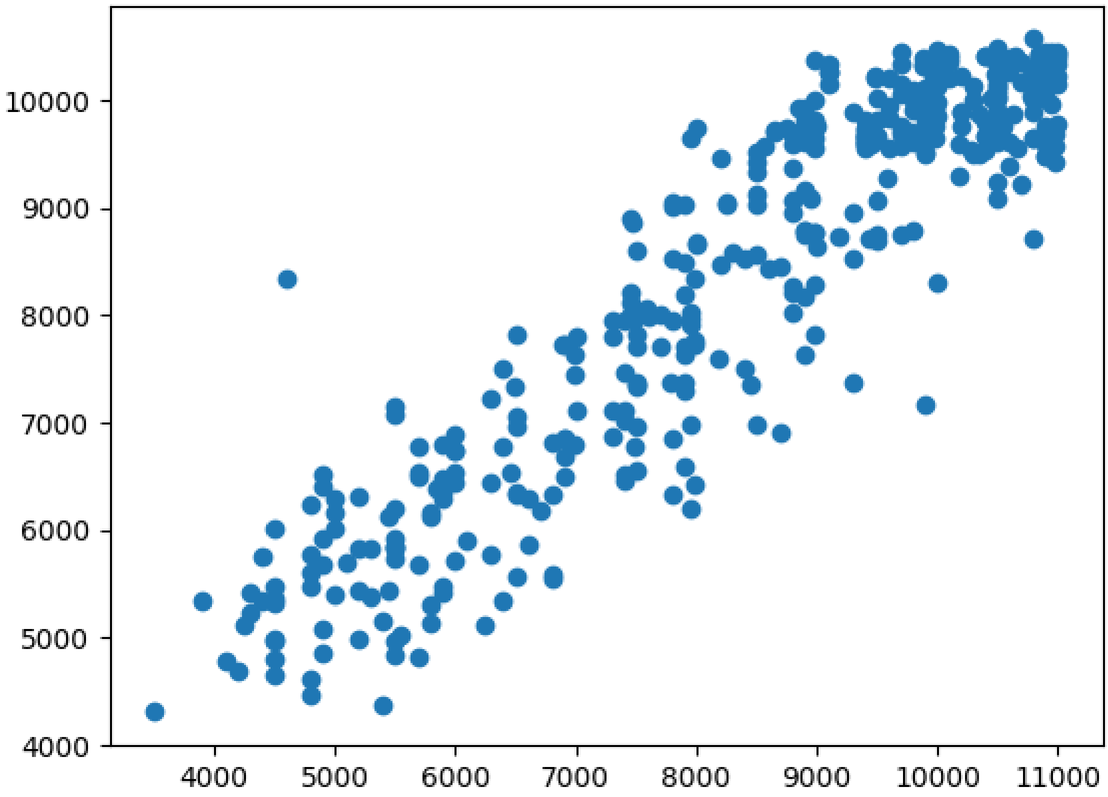
```
coeff_df=pd.DataFrame(regr.coef_,X.columns,columns=['coefficient'])
coeff_df
```

Out[ ]:

	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

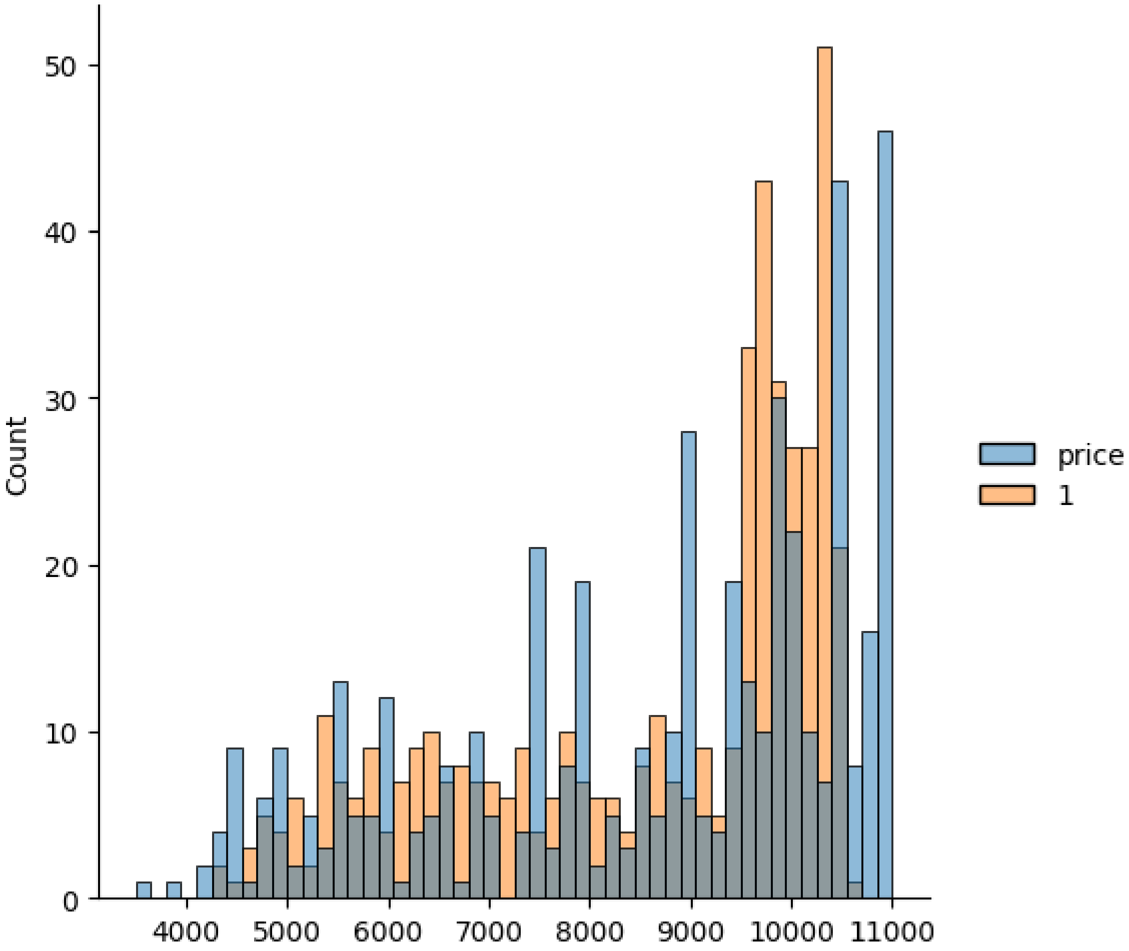
In [ ]:
```
predictions=regr.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x7fed138bd090>
```

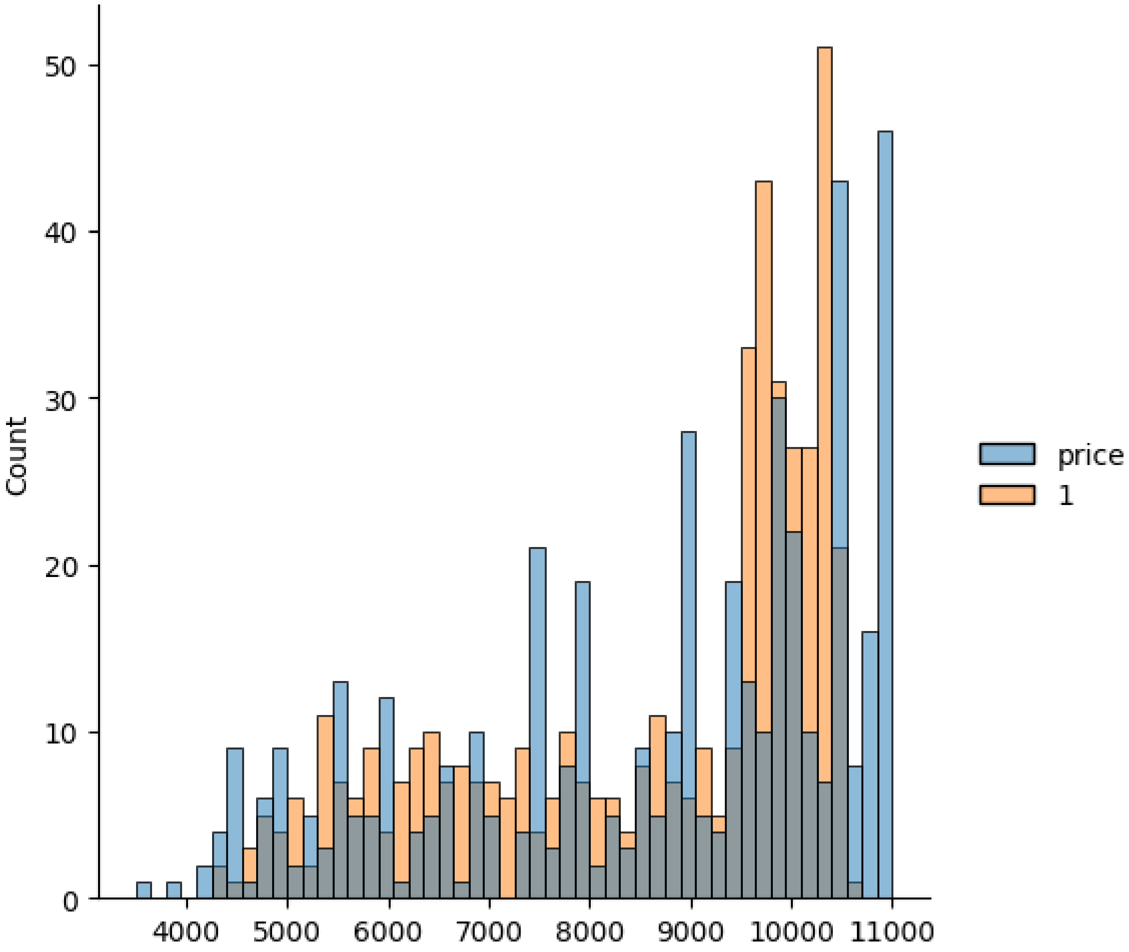


In [ ]:
```
sns.displot((y_test,predictions),bins=50)#without semicolon
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fed198fbaf0>
```



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



In [ ]:

```
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: 593.0876179519936
MSE: 551442.6799691812
MAE: 742.5918663500033
```

In [ ]:

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

```
0.8597136704308864
```

In [ ]:

```
df.fillna(method='ffill',inplace=True)
```

In [ ]:

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

In [ ]:

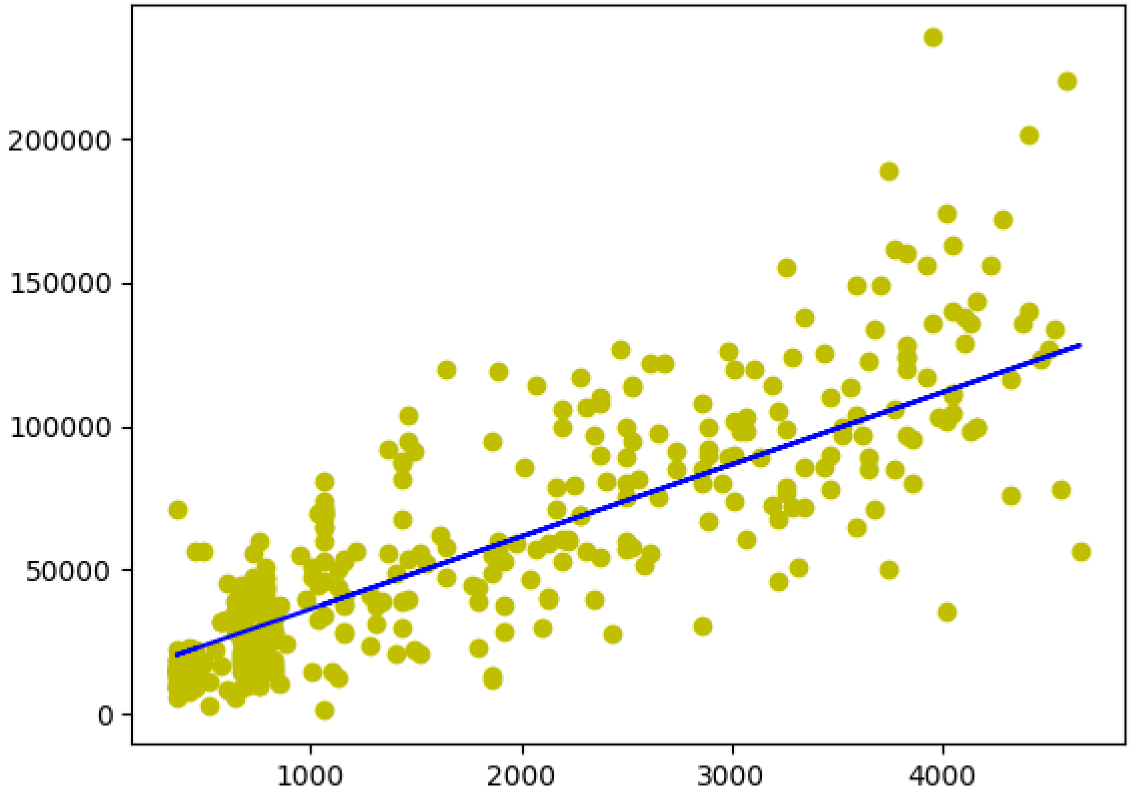
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[ ]:

```
▾ LinearRegression
LinearRegression()
```

In [ ]:

```
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 [ ]:

```
#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 2692062172.9534926
```