In [74]:
```
import pandas as pd
import numpy as np
```

In [75]:
```
df=pd.read_csv(r"C:\Users\venky\Downloads\Advertising.csv")
df
```

Out[75]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

In [76]:
```
df.head()
```

Out[76]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

In [77]:
```
df.tail()
```

Out[77]:

	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

In [78]:
```
df.describe()
```

Out[78]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

In [79]:
```
df.columns
```

Out[79]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [80]:
```
df.shape
```

Out[80]:
```
(200, 4)
```

In [81]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

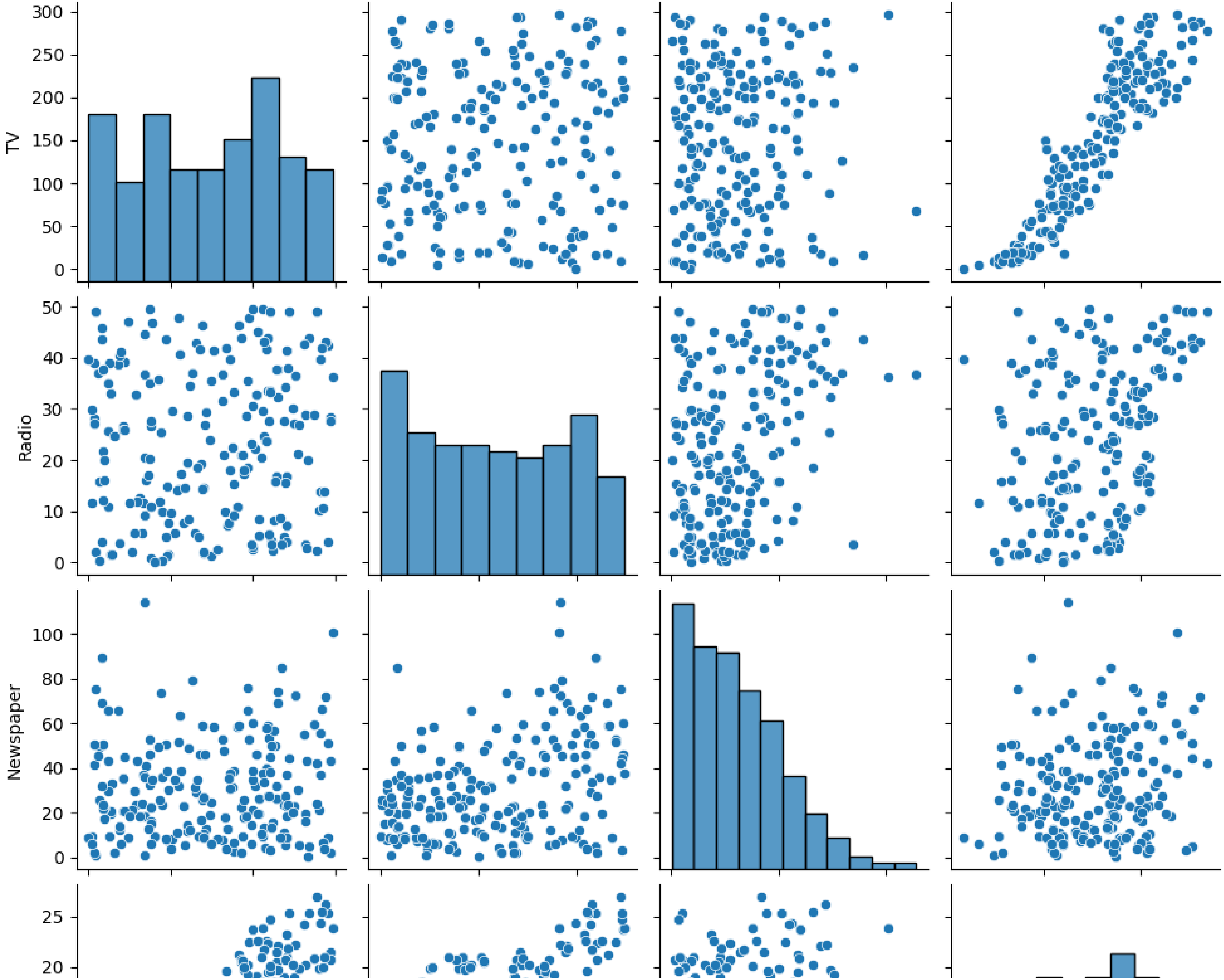
In [82]:
```
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

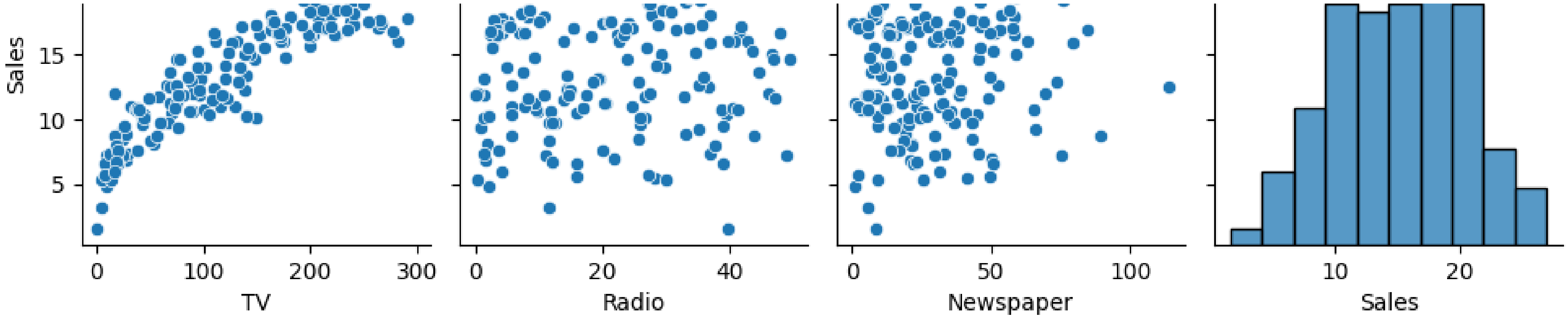
In [83]:
```
df=df[['TV','Radio','Newspaper','Sales']]
```

In [84]:
```
df.columns=['TV','Radio','Newspaper','Sales']
```

In [85]:
```
sns.pairplot(df)
```

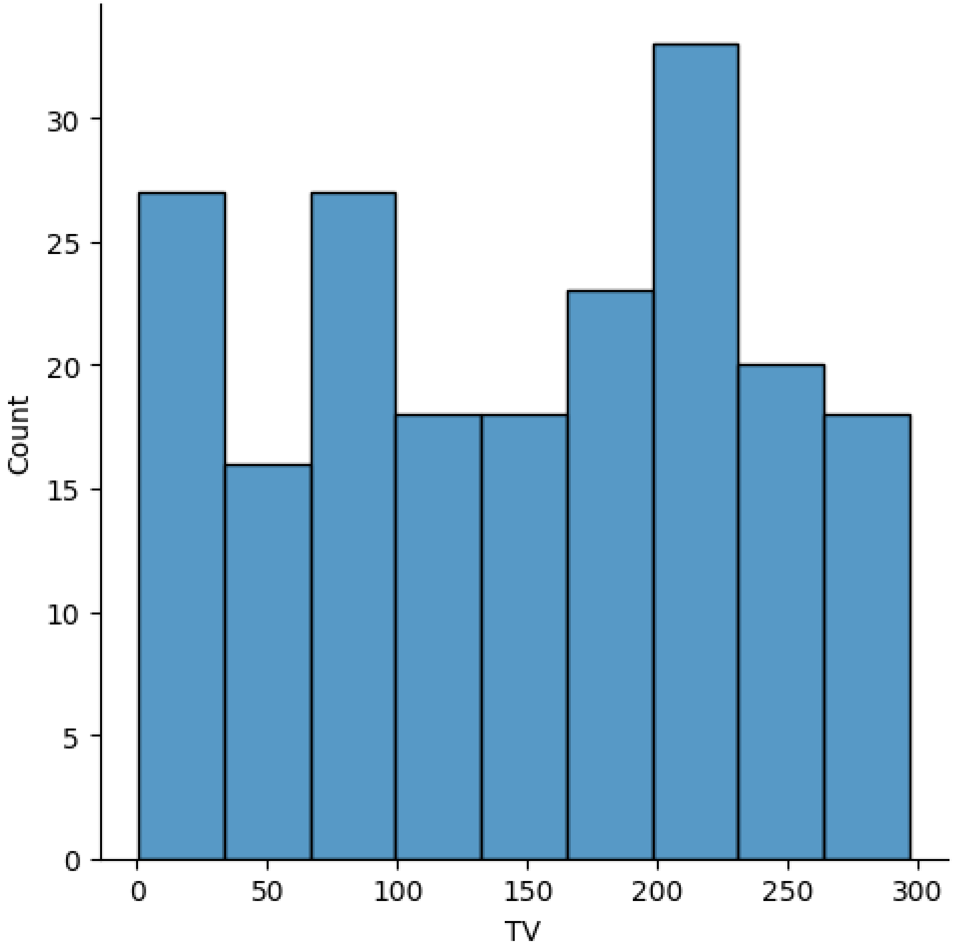
Out[85]:
```
<seaborn.axisgrid.PairGrid at 0x27fd568aa50>
```


TV
Radio
Newspaper
300
250
200
150
100
50
0
50
40
30
20
10
0
100
80
60
40
20
0
25
20


Sales
15
10
5
0
100
200
300
TV
0
20
40
Radio
0
50
100
Newspaper
10
20
Sales

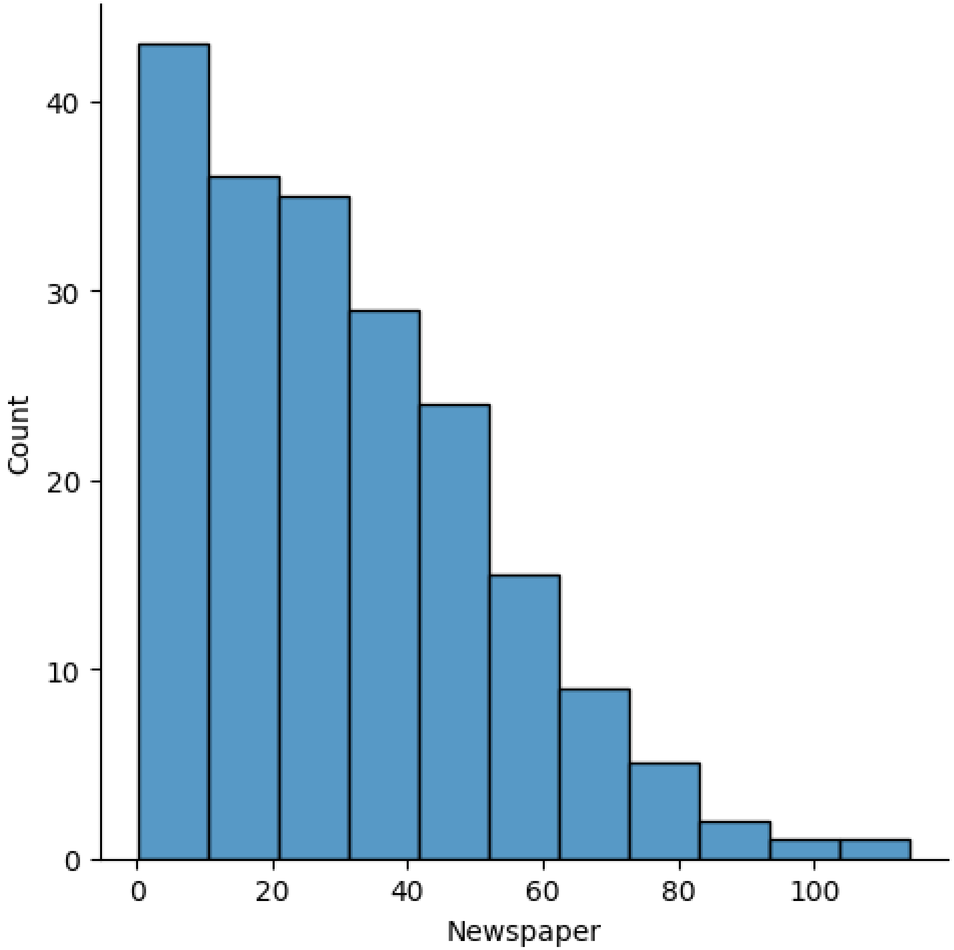
In [86]:
```
sns.displot(df['TV'])
```

Out[86]: <seaborn.axisgrid.FacetGrid at 0x27fd564ba90>



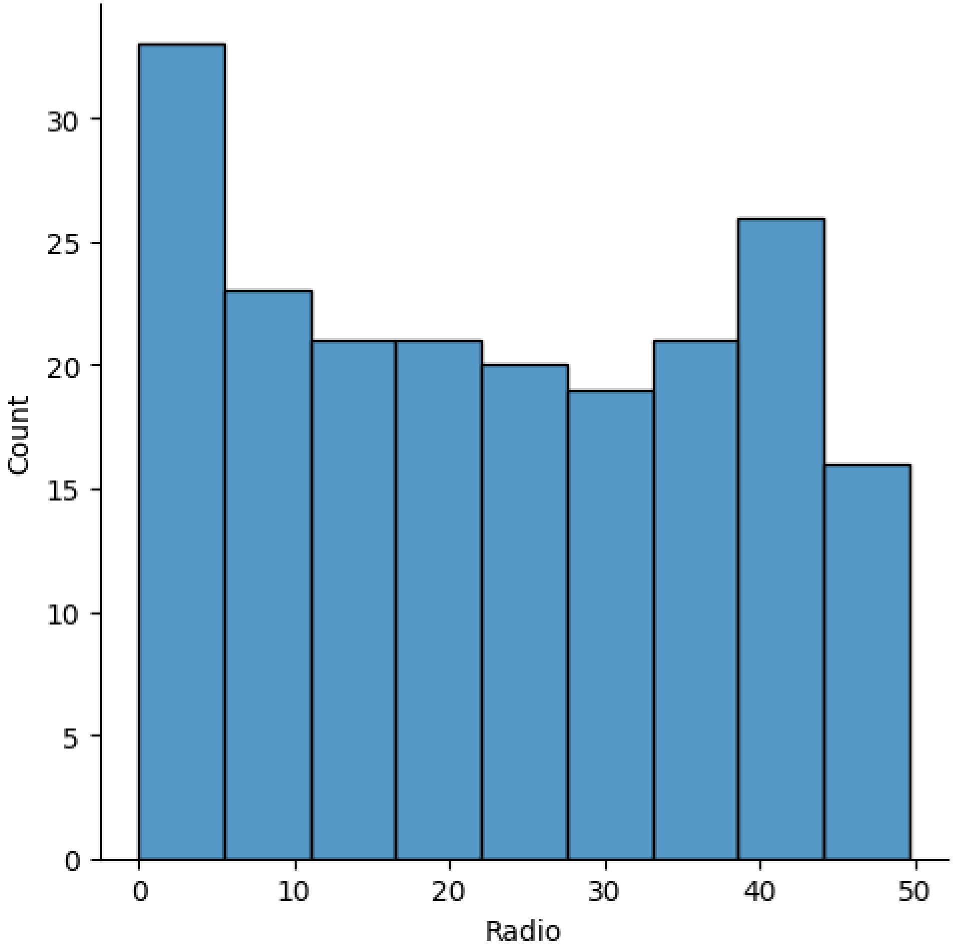
In [87]:
```
sns.displot(df['Newspaper'])
```

Out[87]: <seaborn.axisgrid.FacetGrid at 0x27fd7258910>



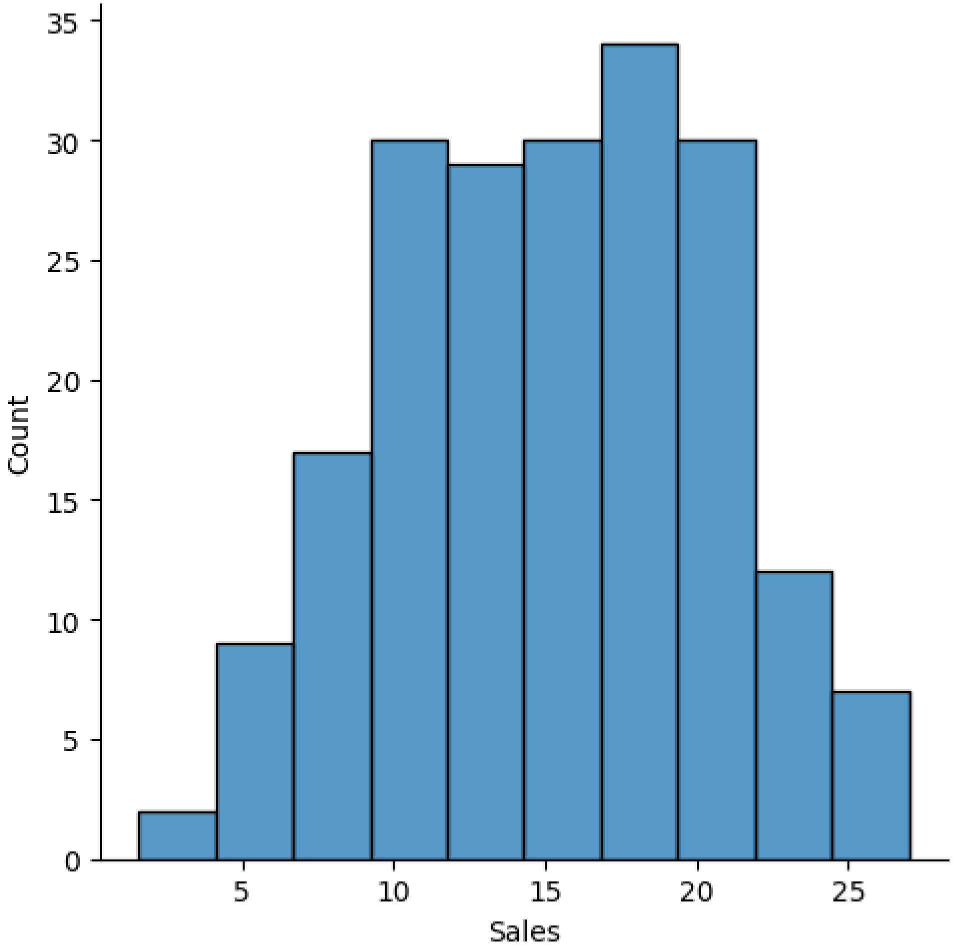
In [88]:
```
sns.displot(df['Radio'])
```

Out[88]: <seaborn.axisgrid.FacetGrid at 0x27fd727db50>



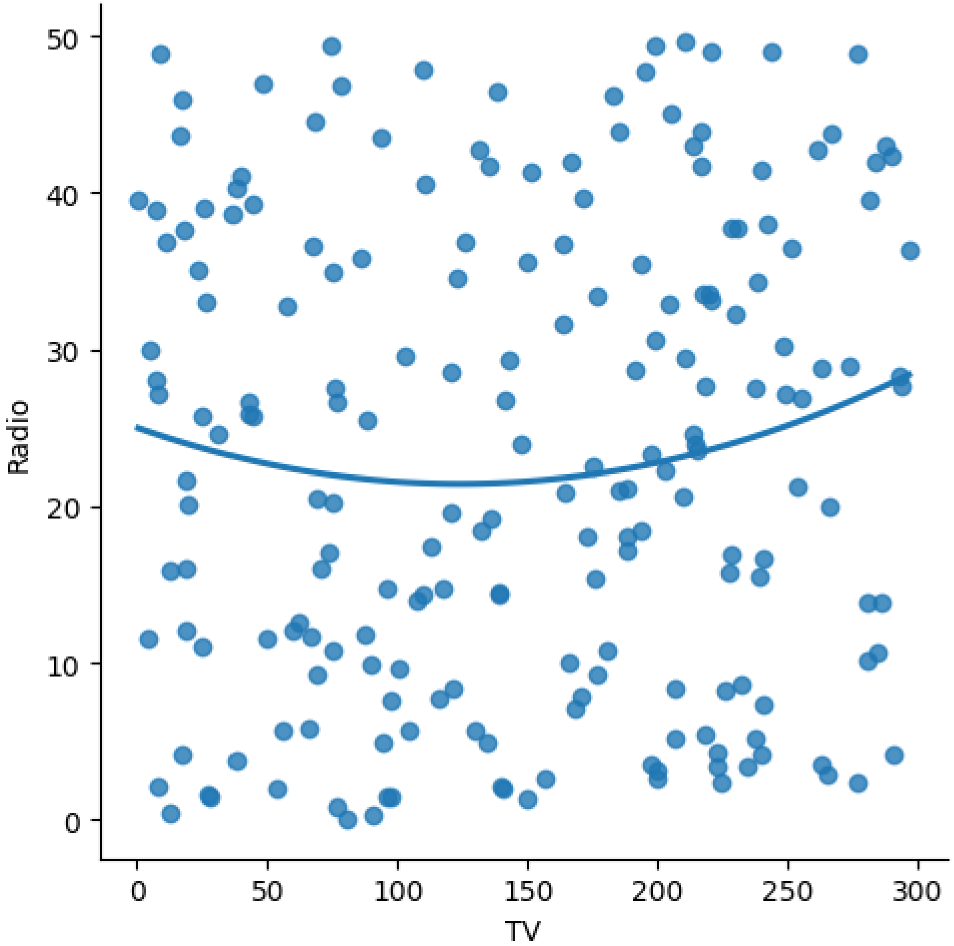
In [89]:
```
sns.displot(df['Sales'])
```

Out[89]: <seaborn.axisgrid.FacetGrid at 0x27fd6d8ba90>



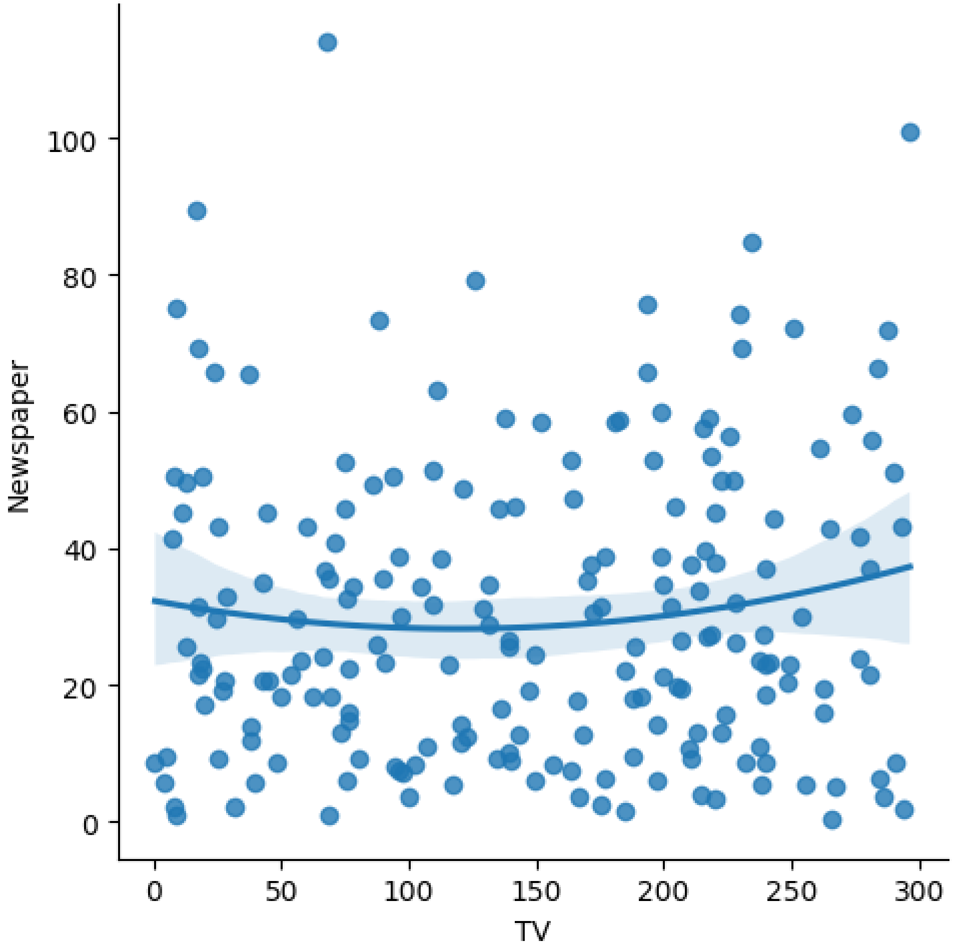
In [90]:
```
sns.lmplot(x="TV",y="Radio",data=df,order=2,ci=None)
```

Out[90]: <seaborn.axisgrid.FacetGrid at 0x27fd6df5d50>



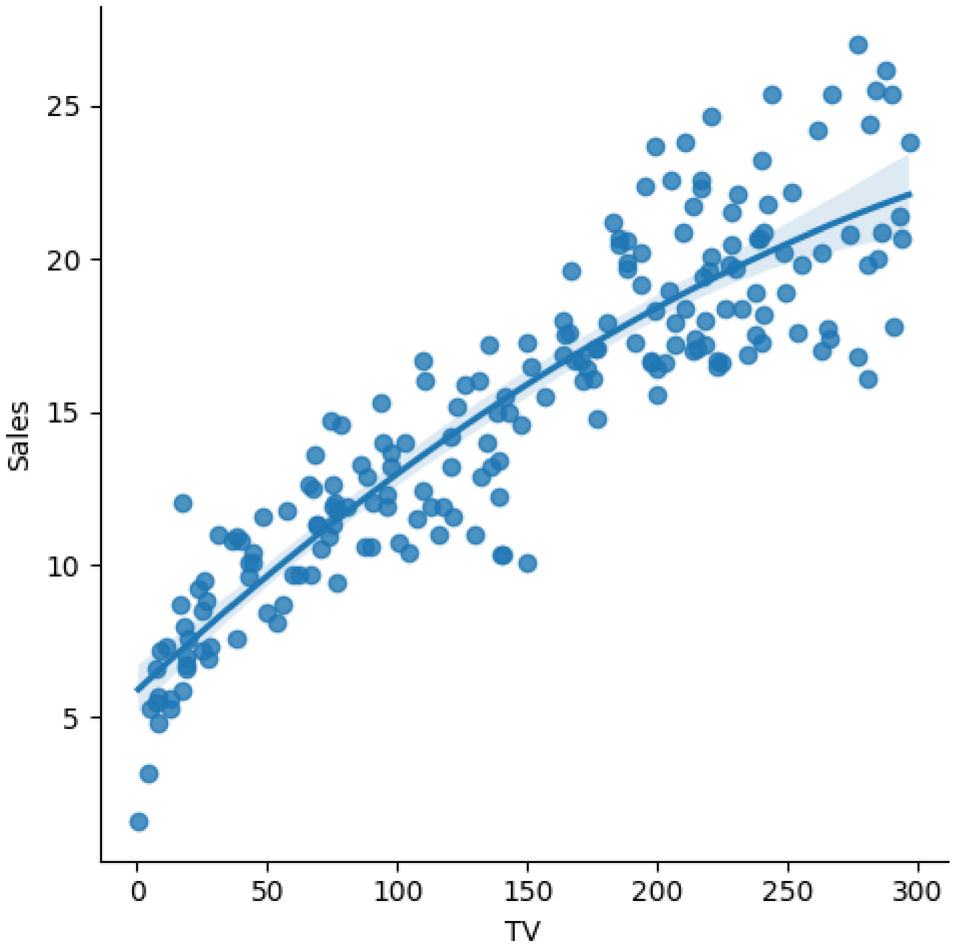
In [91]:
```
sns.lmplot(x="TV",y="Newspaper",data=df,order=2)
```

Out[91]: <seaborn.axisgrid.FacetGrid at 0x27fd6d91510>



In [92]:
```
sns.lmplot(x="TV",y="Sales",data=df,order=2)
```

Out[92]: <seaborn.axisgrid.FacetGrid at 0x27fd8423390>



In [93]:
```
x=np.array(df['TV']).reshape(-1,1)
y=np.array(df['Radio']).reshape(-1,1)
```

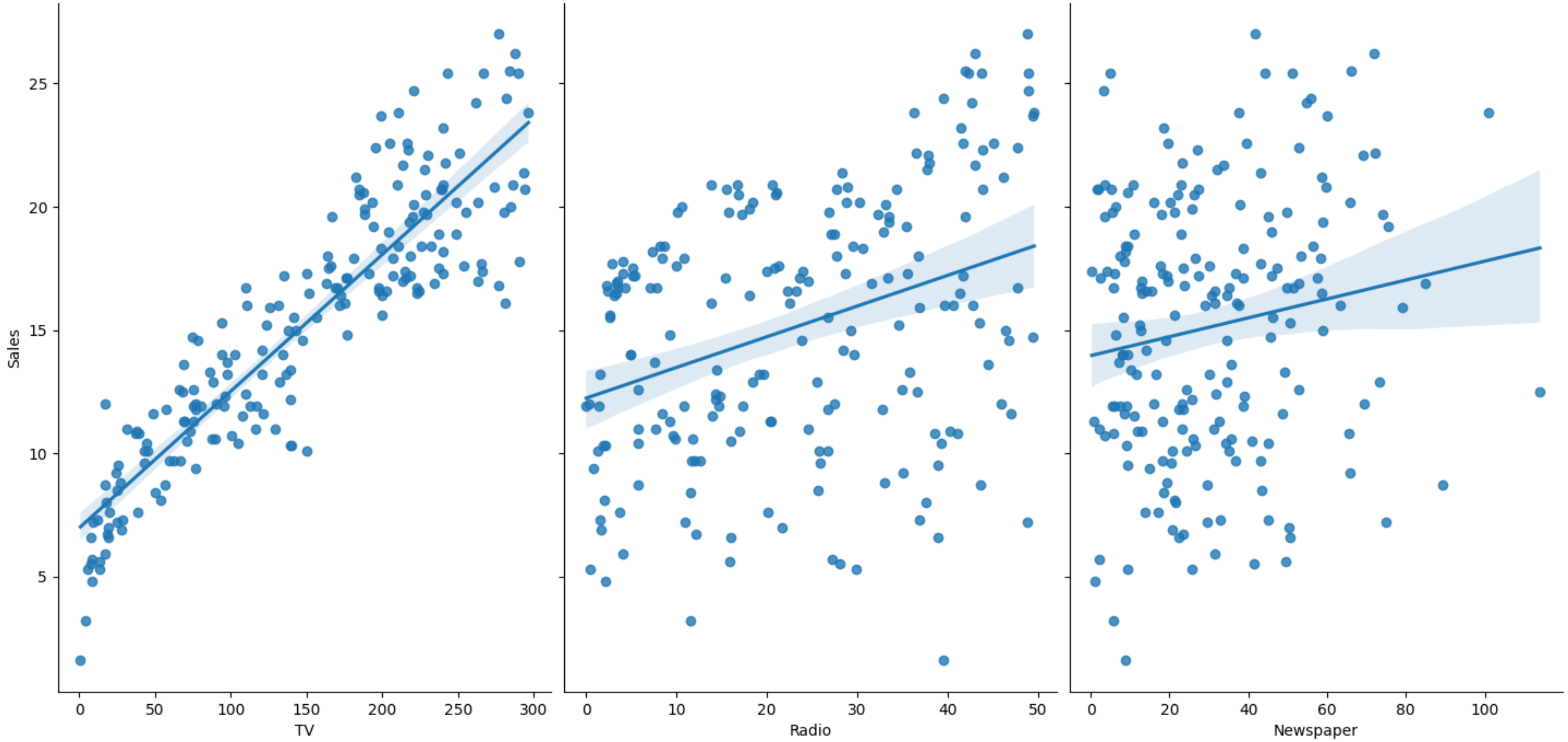
In [94]:
```
df.dropna(inplace=True)
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr=LinearRegression()
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[94]: 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 [95]:
```
sns.pairplot(df,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=7,aspect=0.7,kind='reg')
```

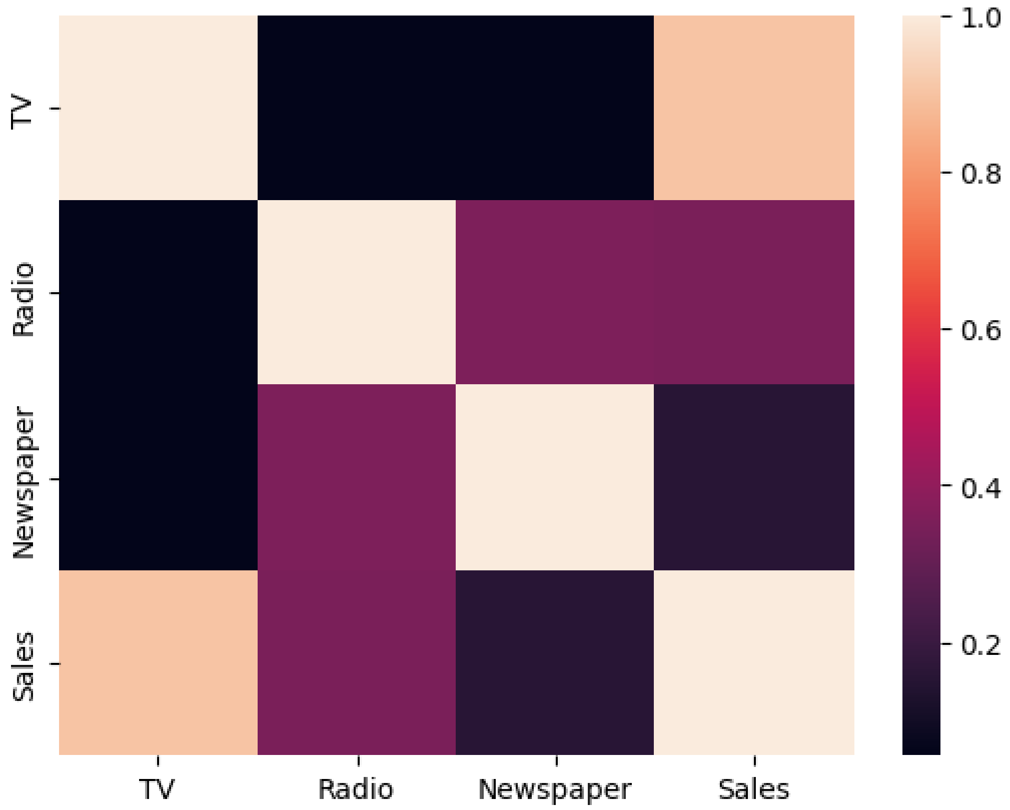
Out[95]: <seaborn.axisgrid.PairGrid at 0x27fd84212d0>



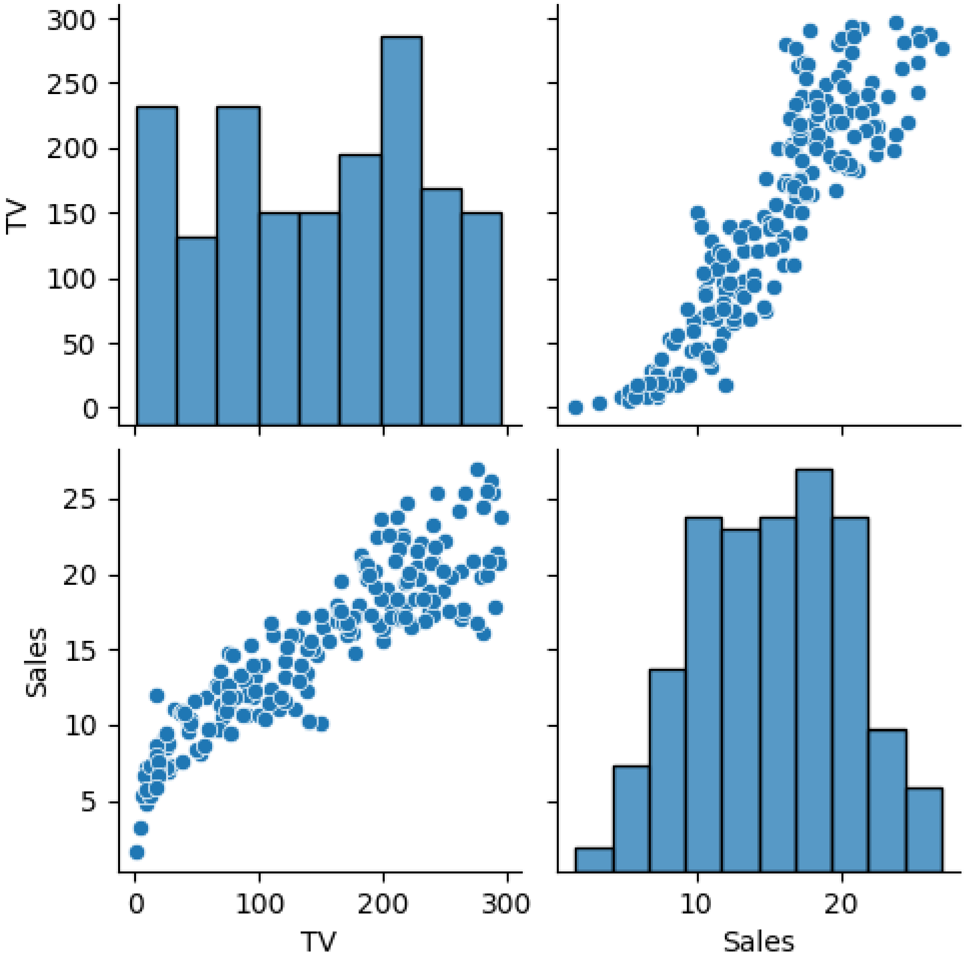
In [96]:
```
hk=df[['TV','Radio','Newspaper','Sales']]
```

In [97]:
```
sns.heatmap(hk.corr())
```

Out[97]: <Axes: >



In [98]:
```
df.drop(columns=['Radio','Newspaper'],inplace=True)
sns.pairplot(df)
df.Sales=np.log(df.Sales)
```



In [99]:
```
features=df.columns[0:2]
target=df.columns[-1]
X=df[features].values
y=df[target].values
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=17)
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

```
The dimension of X_train is (140, 2)
The dimension of X_test is (60, 2)
```

In [100]:
```
from sklearn.linear_model import Lasso,Ridge
```

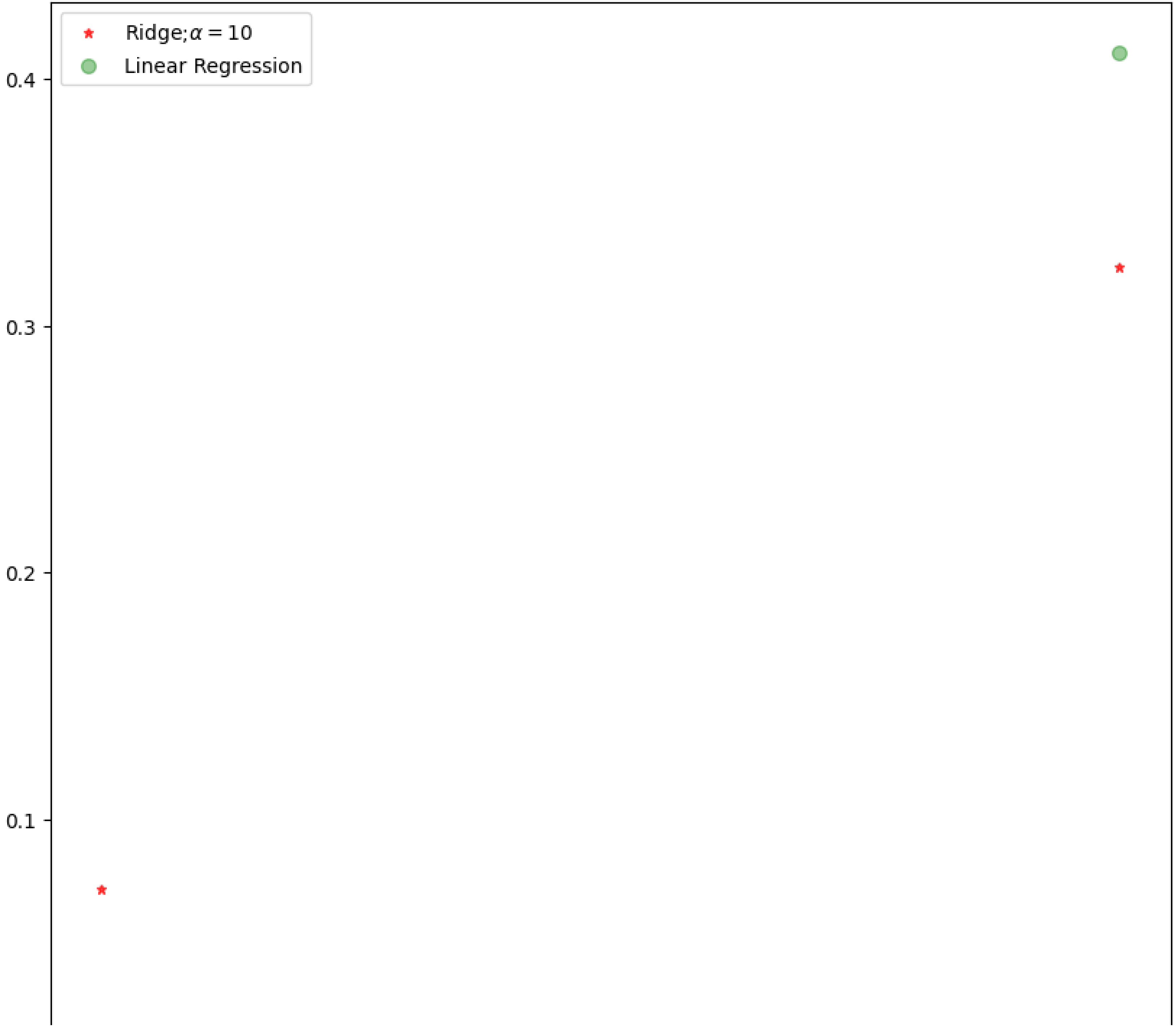
In [101]:
```
lr=LinearRegression()
lr.fit(X_train,y_train)
actual=y_test
train_score_lr=lr.score(X_train,y_train)
test_score_lr=lr.score(X_test,y_test)
print("\nLinear Regression Model:\n" )
print("The train score for lr model is {}".format(train_score_lr))
print("The train score lr model is {}".format(test_score_lr))
```

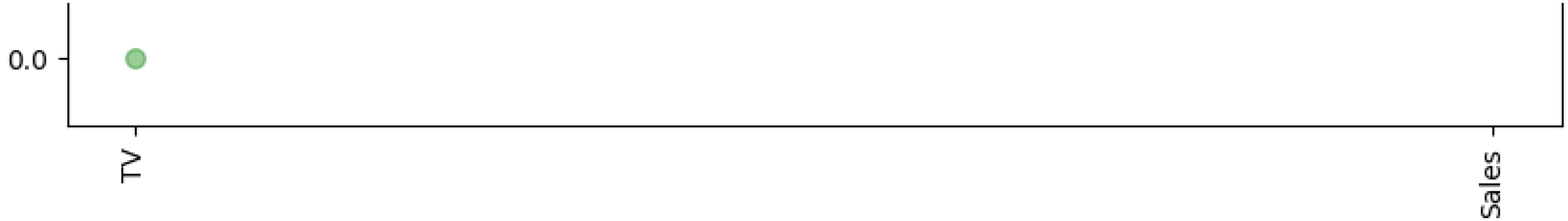
```

Linear Regression Model:

The train score for lr model is 1.0
The train score lr model is 1.0
```

In [102]:
```
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
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 train score for ridge model is {}".format(test_score_ridge))
```

```
Ridge model\:

The train score for ridge model is 0.990287139194161
The train score for ridge model is 0.9844266285141221
```

In [103]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;$\alpha=
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge; $\alpha = 10$
Linear Regression
0.4
0.3
0.2
0.1

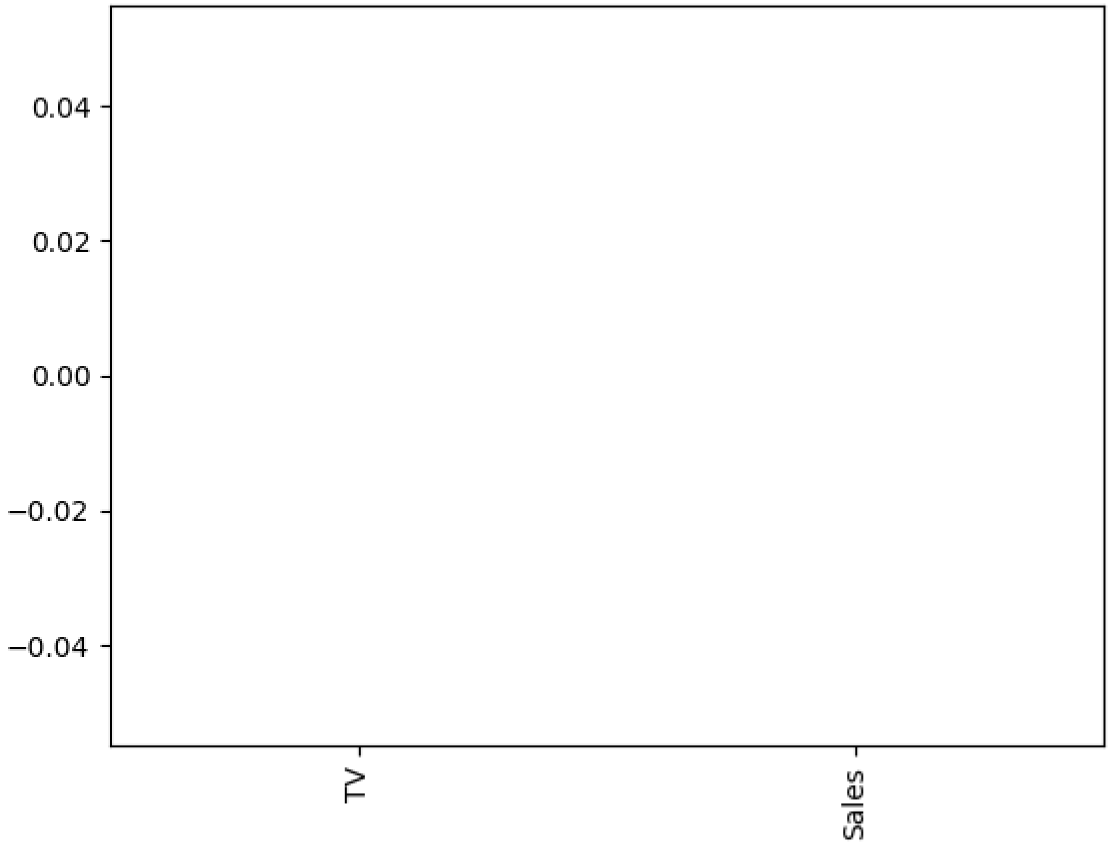


In [104]:
```
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(X_test,y_test)
print("\nRidge model\:\n")
print("The train score for lasso model is {}".format(train_score_ridge))
print("The train score for lasso model is {}".format(test_score_ridge))
```

```
Ridge model\:

The train score for lasso model is 0.990287139194161
The train score for lasso model is 0.9844266285141221
```

In [105]:
```
pd.Series(lassoReg.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

Out[105]: <Axes: >



In [106]:
```
from sklearn.linear_model import LassoCV
lasso_CV=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(X_train,y_train)
print("The train score for lasso model is{}".format(lasso_CV.score(X_train,y_train)))
print("The test score for lasso model is{}".format(lasso_CV.score(X_test,y_test)))
```

```
The train score for lasso model is0.9999999343798134
The test score for lasso model is0.9999999152638072
```

In [107]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;$\alpha=
plt.plot(features,lasso_CV.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso,$\alpha
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.title("comparision plot of Ridge,Lasso and LinearRegression model")
plt.show()
```


comparision plot of Ridge,Lasso and LinearRegression model
Ridge;$\alpha = 10$
lasso,$\alpha = grid$
LinearRegression
0.4
0.3
0.2
0.1

In [108]:
```
from sklearn.linear_model import RidgeCV
ridge_CV=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(X_train,y_train)
print("The train score for ridge model is{}".format(ridge_CV.score(X_train,y_train)))
print("The test score for ridge model is{}".format(ridge_CV.score(X_test,y_test)))
```

```
The train score for ridge model is0.9999999999976281
The test score for ridge model is0.9999999999962489
```

In [109]:
```
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 the tset set",mean_squared_error)
```

```
[0.00417976 0.        ]
2.0263839193110043
mean Squared Error on the tset set 0.5538818050142152
```

VEHICLE-SELECTION ¶

In [110]:
```
df=pd.read_csv(r"C:\Users\venky\Downloads\fiat500_VehicleSelection_Dataset.csv")
df
```

Out[110]:

	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
...	...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [111]:
```
df.head()
```

Out[111]:

	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 [112]:
```
df.tail()
```

Out[112]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7900

In [113]:
```
df.describe()
```

Out[113]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
count	1538.000000	1538.000000	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	11.563428	8576.003901
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

In [114]:
```
df.shape
```

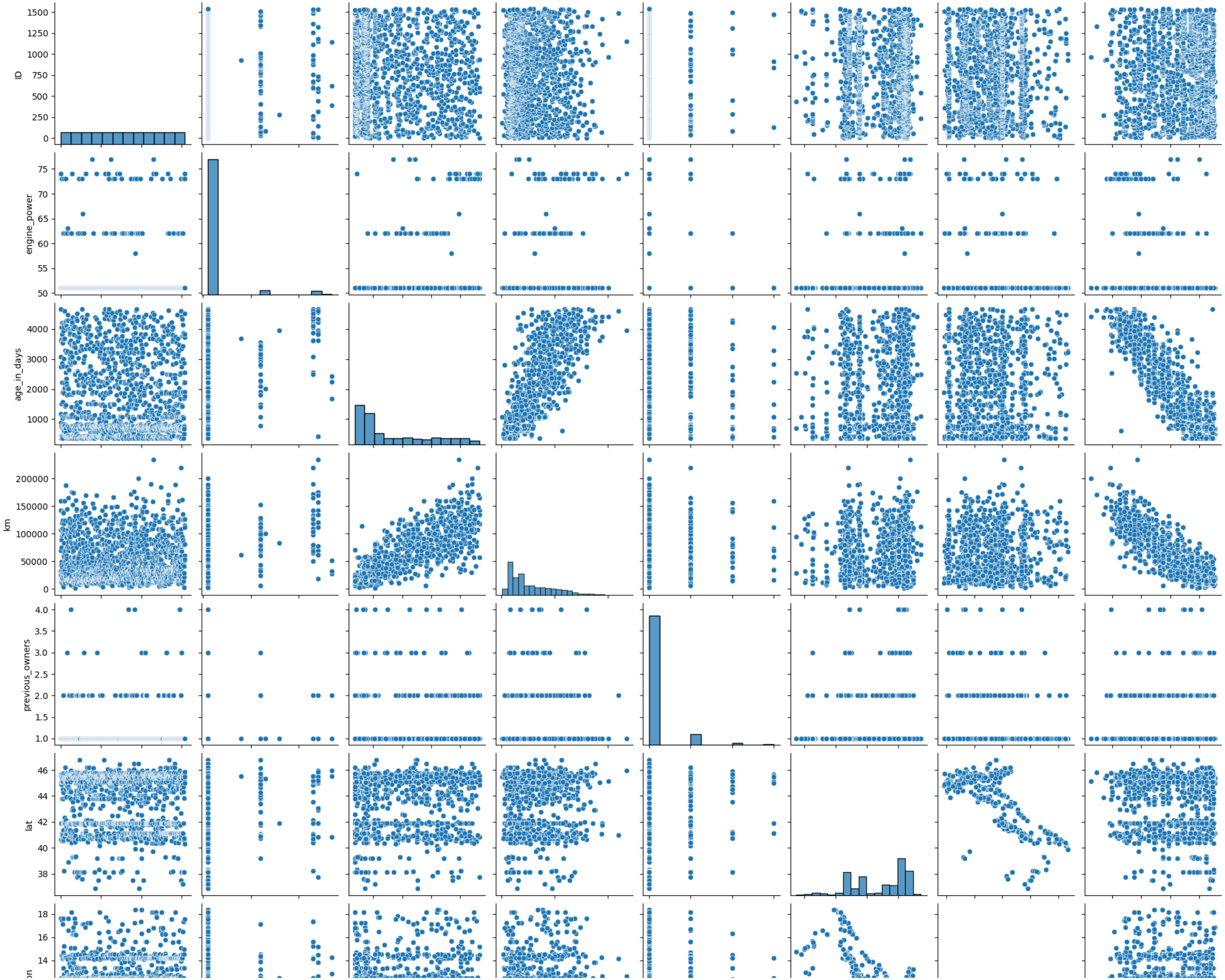
Out[114]:
```
(1538, 9)
```

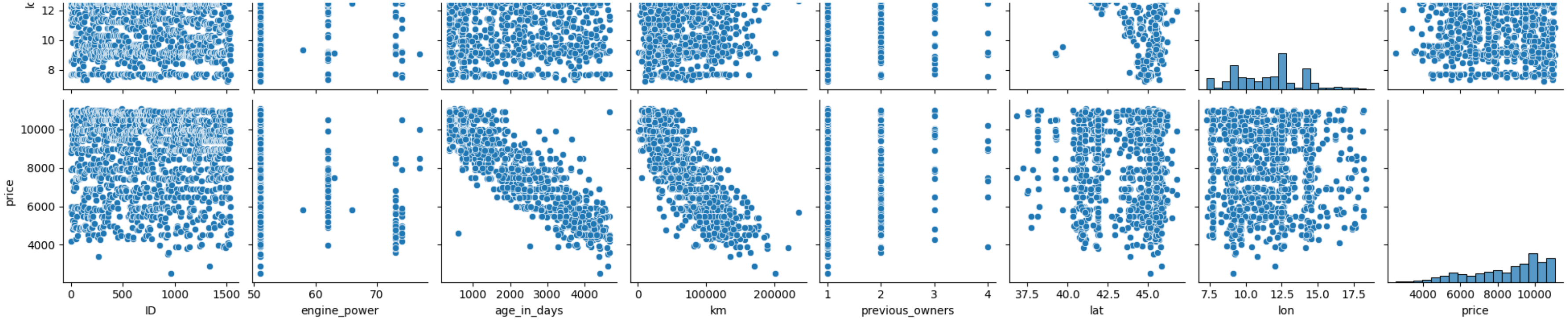
In [115]:
```
df.columns
```

Out[115]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [116]:
```
sns.pairplot(df)
```

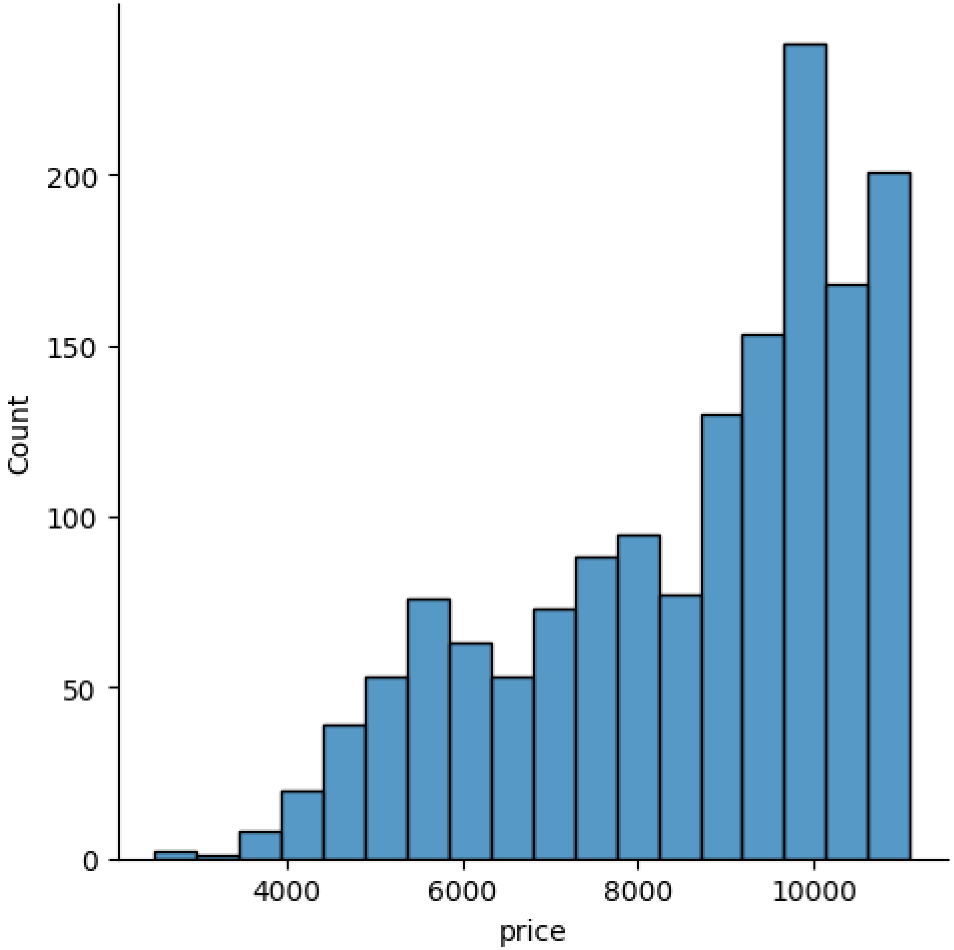
Out[116]:
```
<seaborn.axisgrid.PairGrid at 0x27fd8f75a50>
```


ID
engine_power
age_in_days
km
previous_owners
lat


price
ID
engine_power
age_in_days
km
previous_owners
lat
lon
price

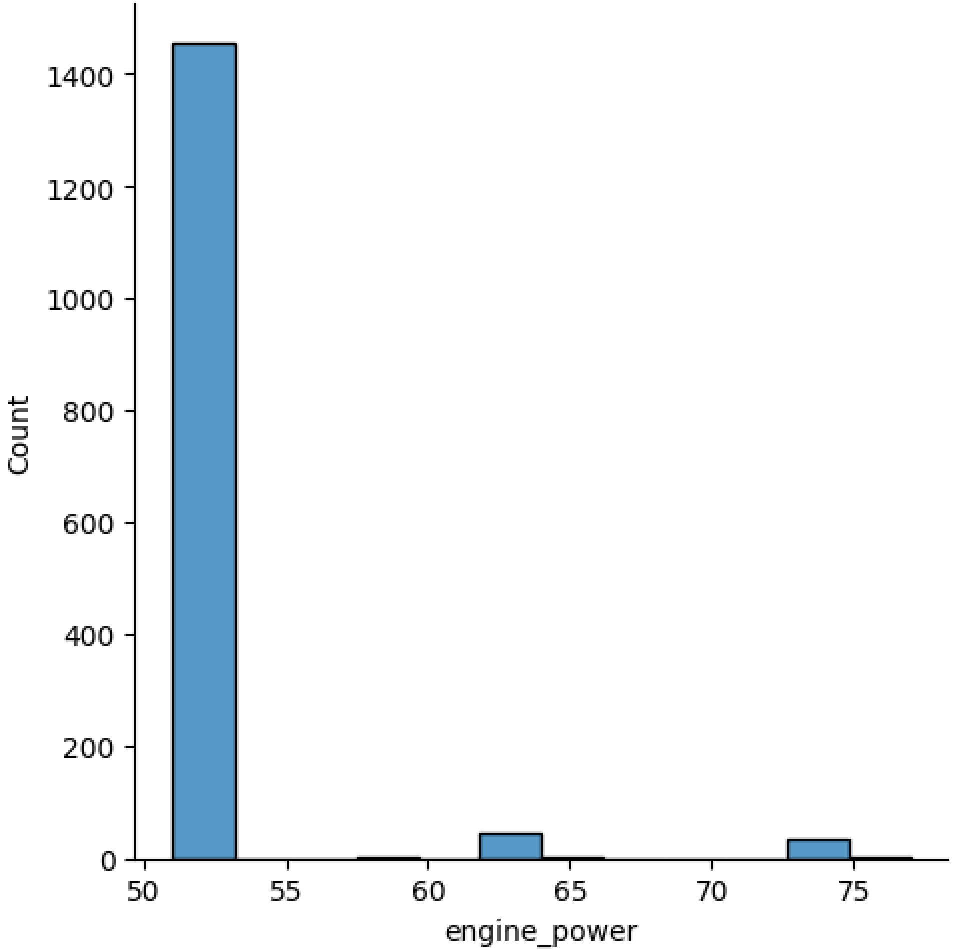
In [117]:
```
sns.displot(df['price'])
```

Out[117]: <seaborn.axisgrid.FacetGrid at 0x27fdc7dba90>



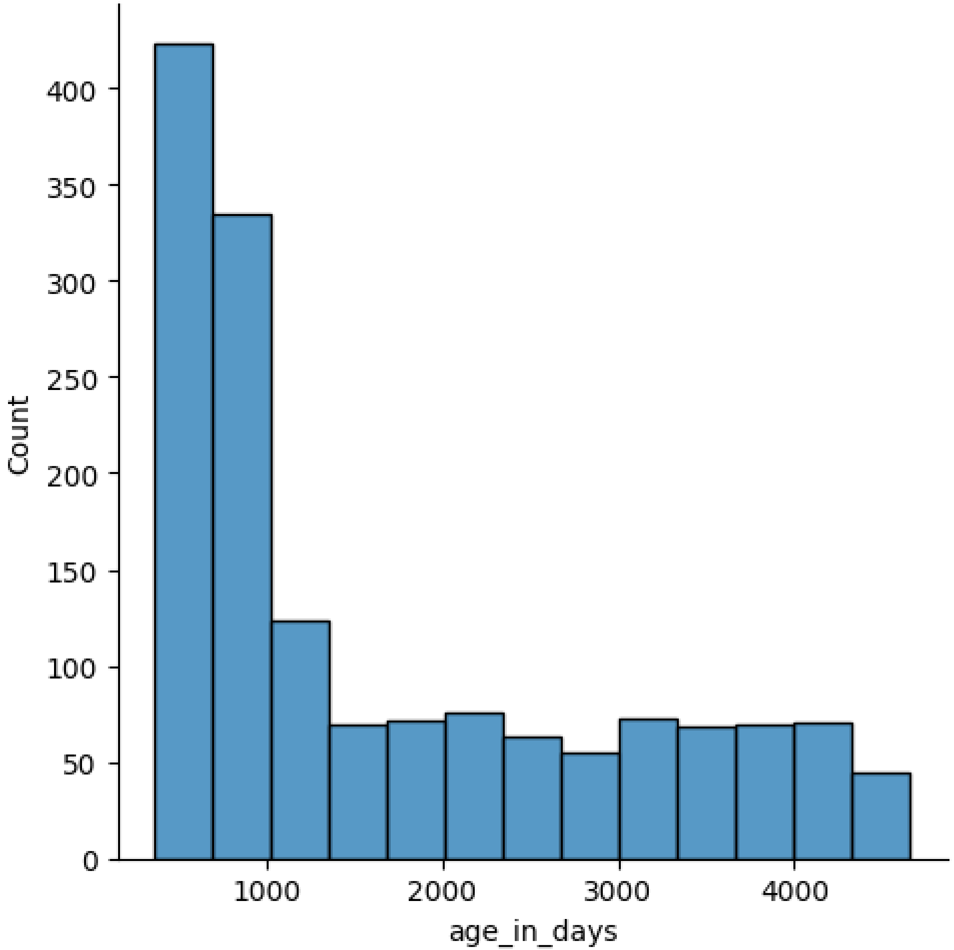
In [118]:
```
sns.displot(df['engine_power'])
```

Out[118]: <seaborn.axisgrid.FacetGrid at 0x27fddb1e790>



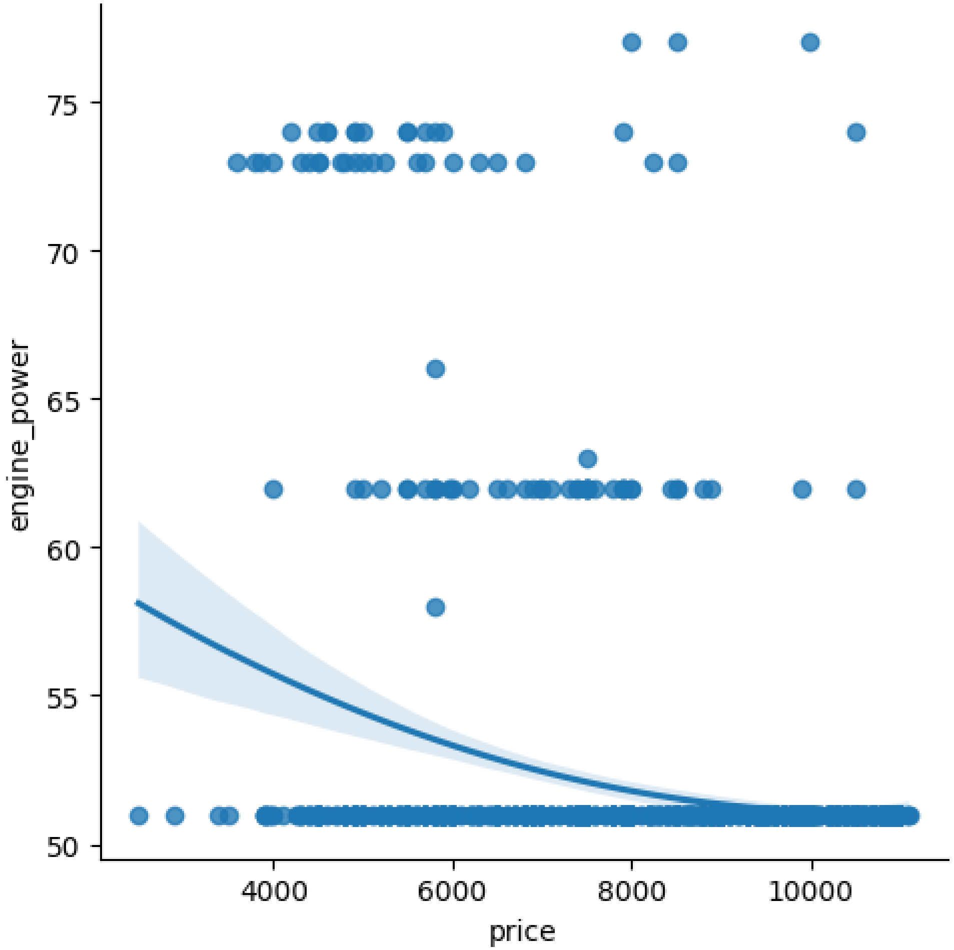
In [119]:
```
sns.displot(df['age_in_days'])
```

Out[119]: <seaborn.axisgrid.FacetGrid at 0x27fdebb3a90>



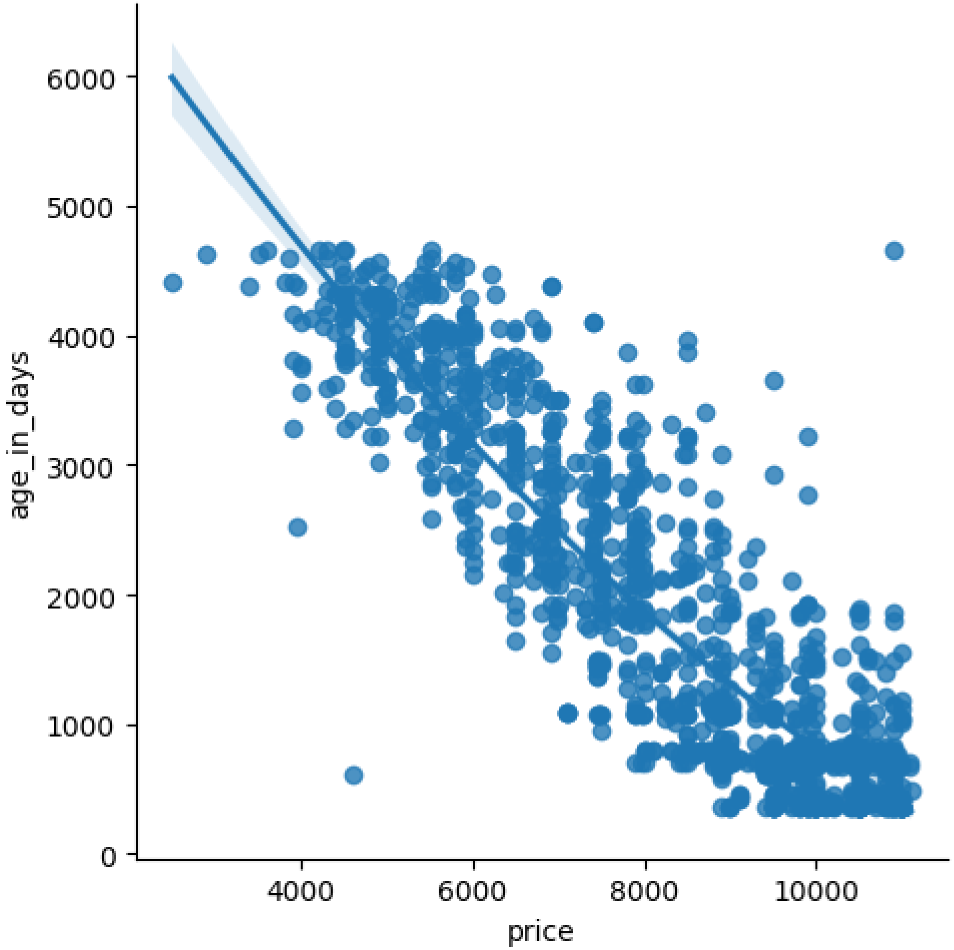
In [120]:
```
sns.lmplot(x="price",y="engine_power",data=df,order=2)
```

Out[120]: <seaborn.axisgrid.FacetGrid at 0x27fddc01e50>



In [121]:
```
sns.lmplot(x="price",y="age_in_days",data=df,order=2)
```

Out[121]: <seaborn.axisgrid.FacetGrid at 0x27fdec3fc10>



In [122]:
```
x=np.array(df['price']).reshape(-1,1)
y=np.array(df['age_in_days']).reshape(-1,1)
```

In [123]:
```
df.dropna(inplace=True)
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr=LinearRegression()
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

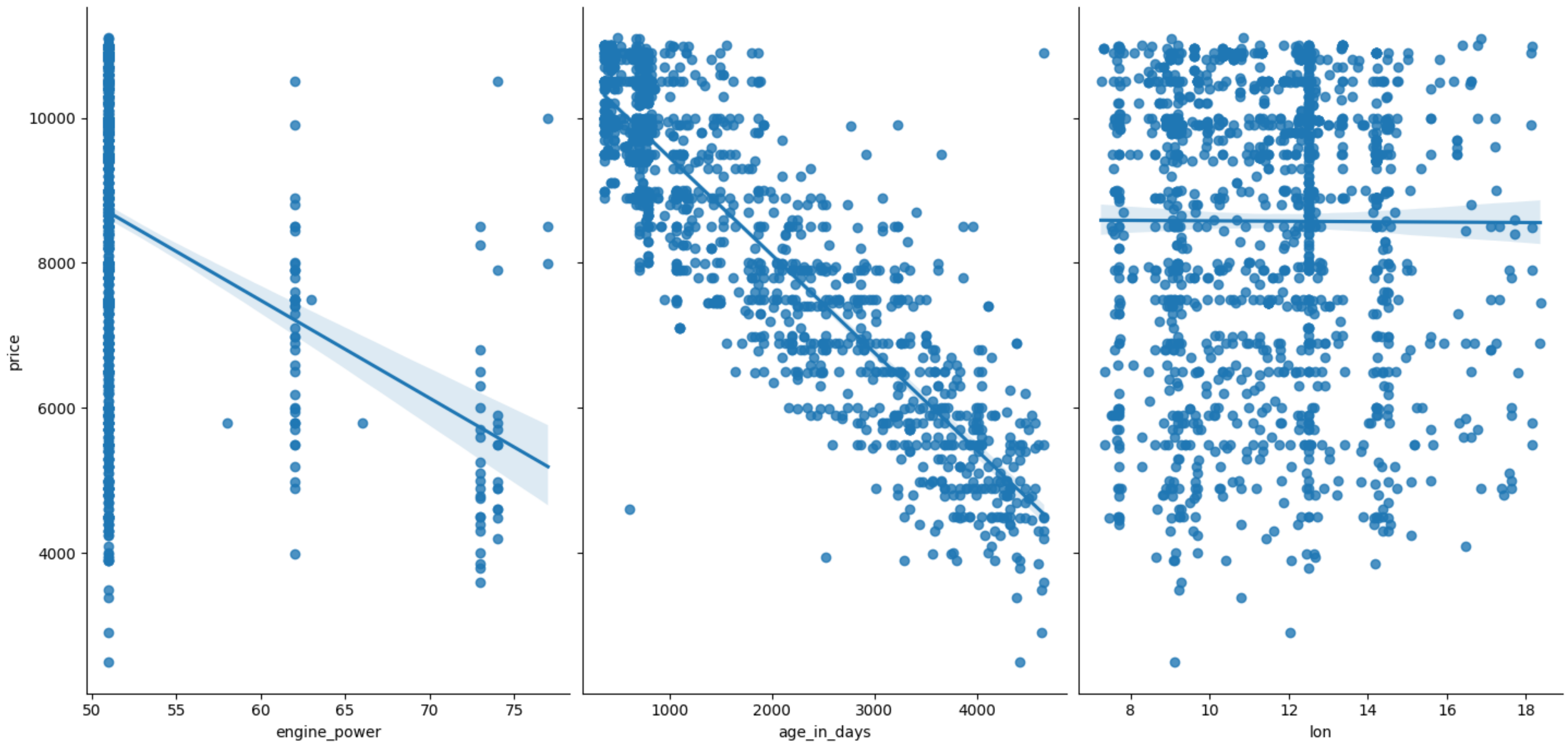
Out[123]:
```
LinearRegression()
```

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 [125]:
```
df.drop(columns=["model"],inplace=True)
```

In [126]:
```
sns.pairplot(df,x_vars=['engine_power','age_in_days','lon'],y_vars='price',height=7,aspect=0.7,kind='reg')
```

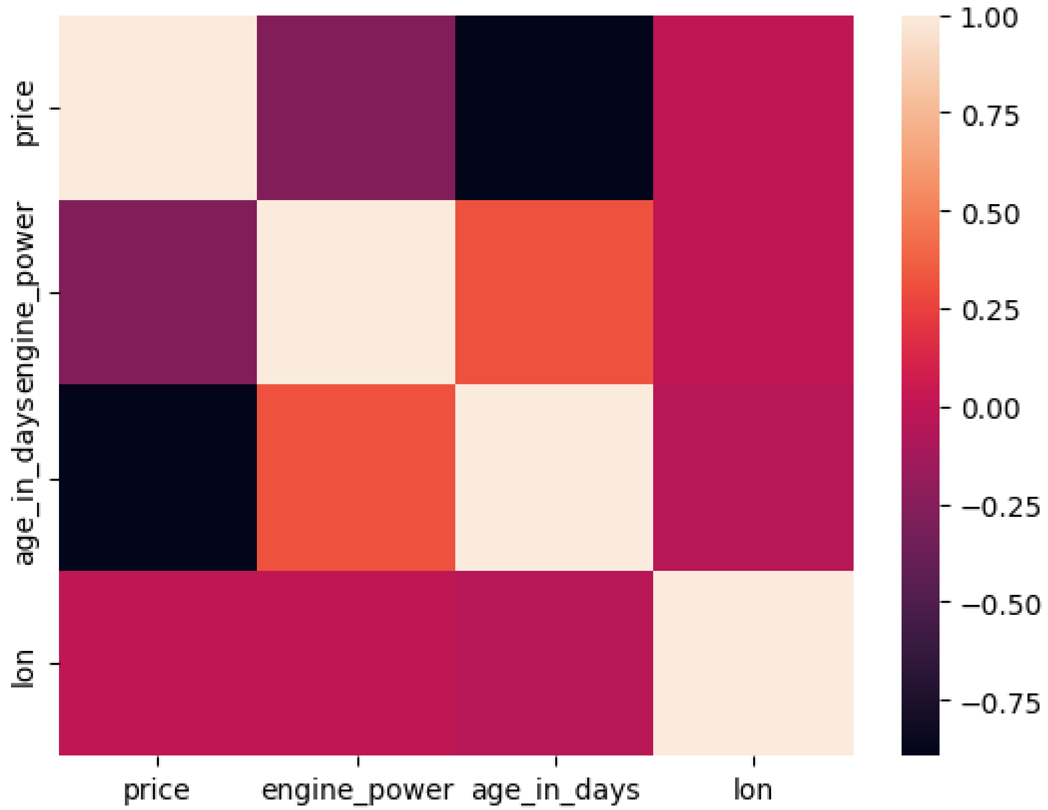
Out[126]: <seaborn.axisgrid.PairGrid at 0x27fdecc5350>



In [127]:
```
hk=df[['price','engine_power','age_in_days','lon']]
```

In [128]:
```
sns.heatmap(hk.corr())
```

Out[128]: <Axes: >



In [129]:
```
features=df.columns[0:2]
target=df.columns[-1]
X=df[features].values
y=df[target].values
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=17)
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

```
The dimension of X_train is (1076, 2)
The dimension of X_test is (462, 2)
```

In [130]:
```
from sklearn.linear_model import Lasso,Ridge
```

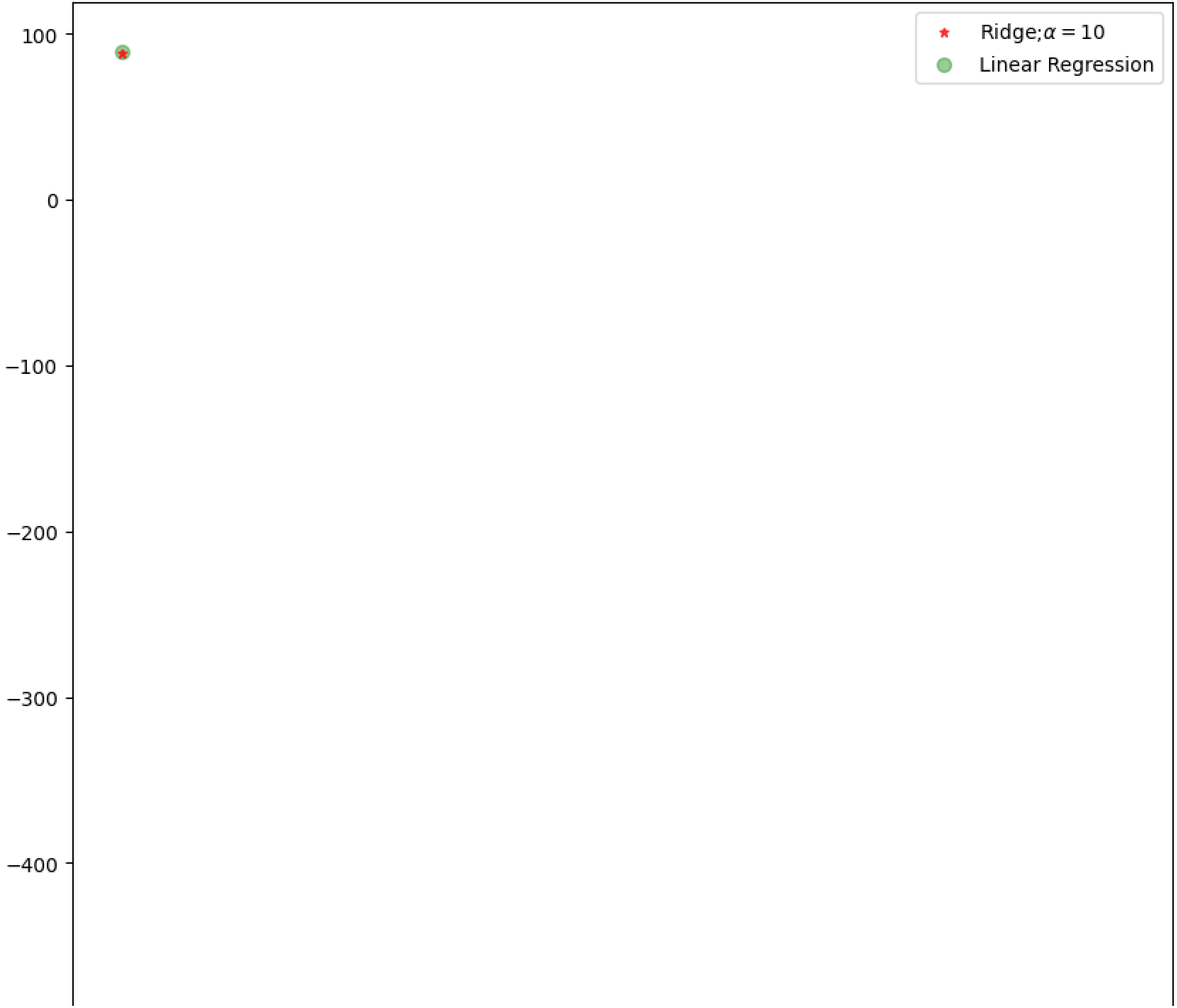
In [131]:
```
lr=LinearRegression()
lr.fit(X_train,y_train)
actual=y_test
train_score_lr=lr.score(X_train,y_train)
test_score_lr=lr.score(X_test,y_test)
print("\nLinear Regression Model:\n" )
print("The train score for lr model is {}".format(train_score_lr))
print("The train score lr model is {}".format(test_score_lr))
```

```

Linear Regression Model:

The train score for lr model is 0.07448634159905865
The train score lr model is 0.07913288661070894
```

In [132]:
```
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
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 train score for ridge model is {}".format(test_score_ridge))
```

```
Ridge model\:

The train score for ridge model is 0.07448028989896427
The train score for ridge model is 0.07885996726883082
```

In [133]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;$\alpha=
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

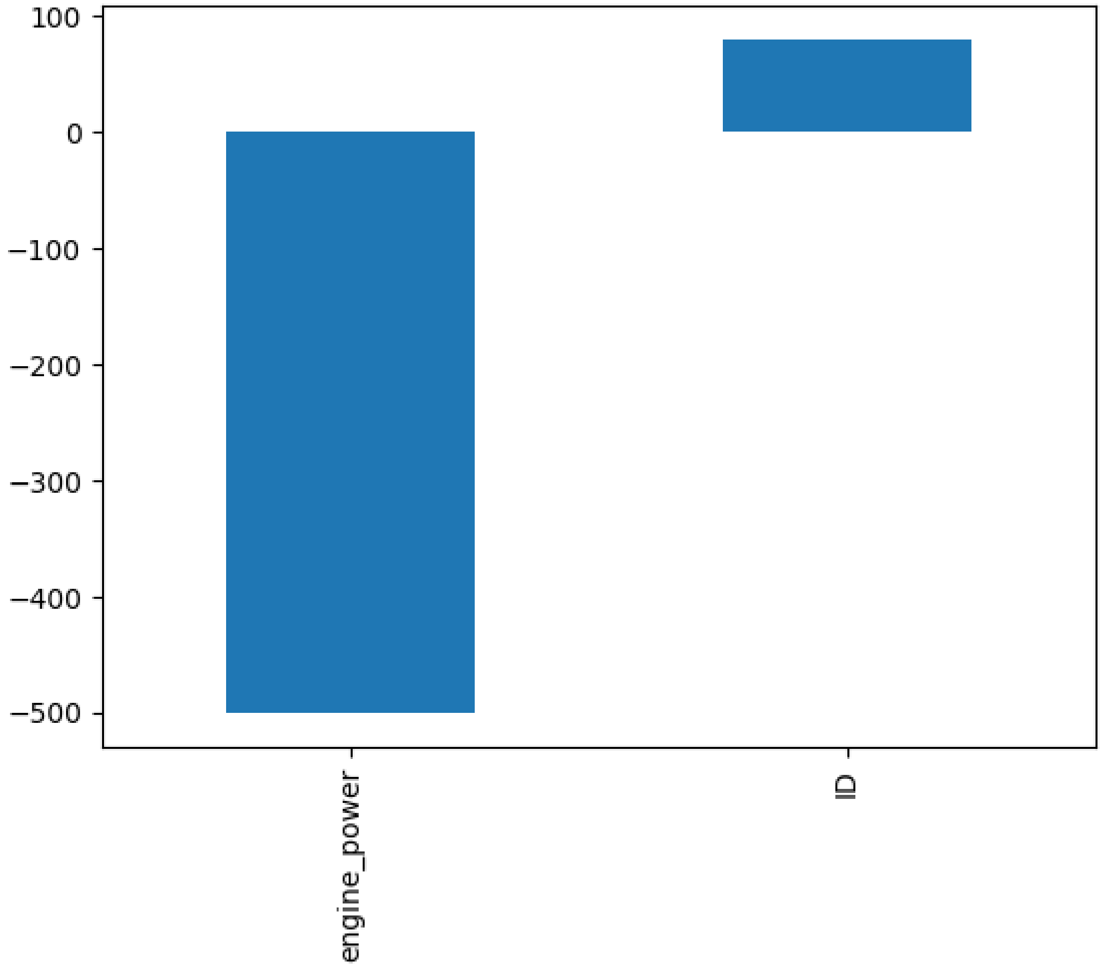

100
0
−100
−200
−300
−400
Ridge; $\alpha = 10$
Linear Regression

In [134]:
```
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(X_test,y_test)
print("\nRidge model\:\n")
print("The train score for lasso model is {}".format(train_score_ridge))
print("The train score for lasso model is {}".format(test_score_ridge))
```

```
Ridge model\:

The train score for lasso model is 0.07448028989896427
The train score for lasso model is 0.07885996726883082
```

In [135]:
```
pd.Series(lassoReg.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

Out[135]: <Axes: >



In [136]:

```
from sklearn.linear_model import LassoCV
lasso_CV=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(X_train,y_train)
print("The train score for lasso model is{}".format(lasso_CV.score(X_train,y_train)))
print("The test score for lasso model is{}".format(lasso_CV.score(X_test,y_test)))
```

```
The train score for lasso model is0.07448634159905387
The test score for lasso model is0.07913288806451946
```

In [137]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;$\alpha=
plt.plot(features,lasso_CV.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso,$\alpha
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.title("comparision plot of Ridge,Lasso and LinearRegression model")
plt.show()
```


comparision plot of Ridge,Lasso and LinearRegression model
100
0
−100
−200
−300
−400
Ridge;$\alpha = 10$
lasso,$\alpha = grid$
LinearRegression

In [138]:
```
from sklearn.linear_model import RidgeCV
ridge_CV=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(X_train,y_train)
print("The train score for ridge model is{}".format(ridge_CV.score(X_train,y_train)))
print("The test score for ridge model is{}".format(ridge_CV.score(X_test,y_test)))
```

```
The train score for ridge model is0.07448028989896427
The test score for ridge model is0.07885996726883171
```

In [139]:
```
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 the tset set",mean_squared_error)
```

```
[ 8.46751882e-02 -1.30405006e+02]
15279.442735227916
mean Squared Error on the tset set 48390222.80186546
```

In [ ]: