In [131]:
```
import pandas as pd            #importing pandas

y=pd.read_csv('/home/placement/Desktop/college trainning/fiat500.csv')  #reading csv using path
```

In [184]:
```
y.groupby(['price']).head()                    #grouping by price
```

Out[184]:

	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
...	...	...	...	...	...	...	...	...	...
1515	1516	lounge	51	1917	124999	1	45.564491	10.115610	6999
1521	1522	lounge	51	3774	85000	1	44.294300	9.674440	4000
1523	1524	pop	51	2251	79800	1	45.512051	10.427010	6450
1531	1532	sport	73	4505	127000	1	45.528511	9.593230	4750
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600

657 rows × 9 columns

In [87]:
```
h=y.rename(columns={'model':"model_name"})              # TO RENAME THE DATA
h
```

Out[87]:

	ID	model_name	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 [88]:
```
y.describe()
```

Out[88]:

	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 [185]:
```
y.head()
```

Out[185]:

	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 [90]:
```
x=y.drop("model",axis=1)                # to delete the data
x
```

Out[90]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	51	882	25000	1	44.907242	8.611560	8900
1	2	51	1186	32500	1	45.666359	12.241890	8800
2	3	74	4658	142228	1	45.503300	11.417840	4200
3	4	51	2739	160000	1	40.633171	17.634609	6000
4	5	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...
1533	1534	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 8 columns

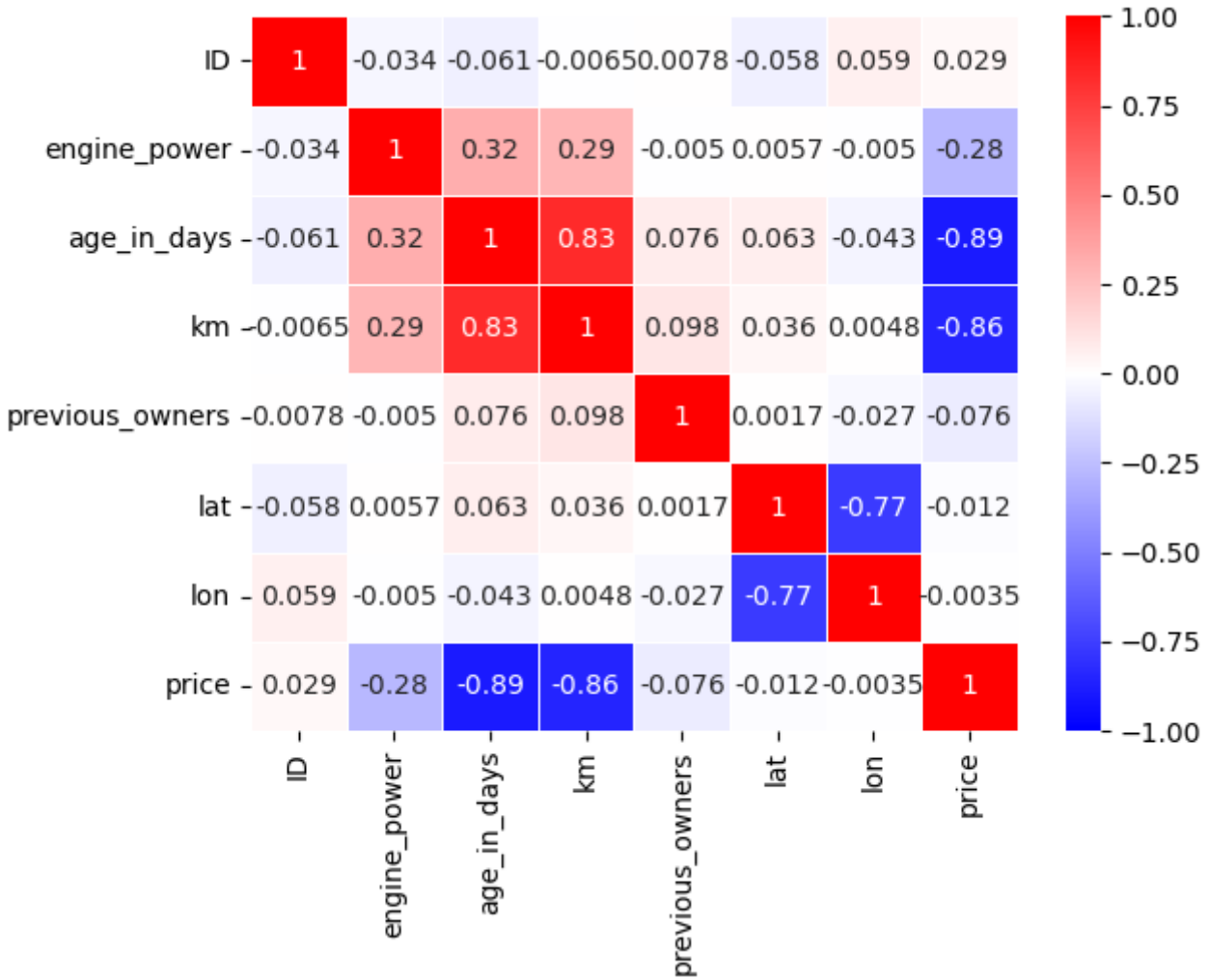
In [91]:
```
cor=x.corr()
cor                                    # to find the correlation
```

Out[91]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
ID	1.000000	-0.034059	-0.060753	-0.006537	0.007803	-0.058207	0.058941	0.028516
engine_power	-0.034059	1.000000	0.319190	0.285495	-0.005030	0.005721	-0.005032	-0.277235
age_in_days	-0.060753	0.319190	1.000000	0.833890	0.075775	0.062982	-0.042667	-0.893328
km	-0.006537	0.285495	0.833890	1.000000	0.097539	0.035519	0.004839	-0.859373
previous_owners	0.007803	-0.005030	0.075775	0.097539	1.000000	0.001697	-0.026836	-0.076274
lat	-0.058207	0.005721	0.062982	0.035519	0.001697	1.000000	-0.766646	-0.011733
lon	0.058941	-0.005032	-0.042667	0.004839	-0.026836	-0.766646	1.000000	-0.003541
price	0.028516	-0.277235	-0.893328	-0.859373	-0.076274	-0.011733	-0.003541	1.000000

In [92]:
```
import seaborn as sns
sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap="bwr")    # to print theheatmap
```

Out[92]: <Axes: >



In [93]:
```
cor.groupby(["price","km"]).count()      #correlation of the price and km
```

Out[93]:

price	km	ID	engine_power	age_in_days	previous_owners	lat	lon
-0.893328	**0.833890**	1	1	1	1	1	1
-0.859373	**1.000000**	1	1	1	1	1	1
-0.277235	**0.285495**	1	1	1	1	1	1
-0.076274	**0.097539**	1	1	1	1	1	1
-0.011733	**0.035519**	1	1	1	1	1	1
-0.003541	**0.004839**	1	1	1	1	1	1
0.028516	**-0.006537**	1	1	1	1	1	1
1.000000	**-0.859373**	1	1	1	1	1	1

In [94]:
```
x['model']=y['model'].map({'lounge':1,'pop':2,'sport':4})    #renaming the terms
x
```

Out[94]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price	model
0	1	51	882	25000	1	44.907242	8.611560	8900	1
1	2	51	1186	32500	1	45.666359	12.241890	8800	2
2	3	74	4658	142228	1	45.503300	11.417840	4200	4
3	4	51	2739	160000	1	40.633171	17.634609	6000	1
4	5	73	3074	106880	1	41.903221	12.495650	5700	2
...	...	...	...	...	...	...	...	...	...
1533	1534	51	3712	115280	1	45.069679	7.704920	5200	4
1534	1535	74	3835	112000	1	45.845692	8.666870	4600	1
1535	1536	51	2223	60457	1	45.481541	9.413480	7500	2
1536	1537	51	2557	80750	1	45.000702	7.682270	5990	1
1537	1538	51	1766	54276	1	40.323410	17.568270	7900	2

1538 rows × 9 columns

In [95]:
```
y.groupby(["model","ID"]).count()            # groupby the data
```

Out[95]:

		engine_power	age_in_days	km	previous_owners	lat	lon	price
model	**ID**							
lounge	**1**	1	1	1	1	1	1	1
	4	1	1	1	1	1	1	1
	7	1	1	1	1	1	1	1
	8	1	1	1	1	1	1	1
	12	1	1	1	1	1	1	1
...	**...**	...	...	...	...	...	...	...
sport	**1499**	1	1	1	1	1	1	1
	1502	1	1	1	1	1	1	1
	1505	1	1	1	1	1	1	1
	1532	1	1	1	1	1	1	1
	1534	1	1	1	1	1	1	1

1538 rows × 7 columns

In [96]:
```
a=x.sort_values("model")      # to sort the data
a
```

Out[96]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price	model
0	1	51	882	25000	1	44.907242	8.61156	8900	1
952	953	51	2009	35552	1	42.562832	12.64220	8900	1
951	952	51	3684	78000	1	43.552910	10.31998	6500	1
950	951	51	762	36463	1	41.107880	14.20881	10900	1
949	950	51	790	32000	1	41.572239	13.33369	9490	1
...	...	...	...	...	...	...	...	...	...
1057	1058	51	3500	60000	2	45.438301	10.99170	6900	4
77	78	51	2739	77149	3	44.754890	8.03190	7800	4
172	173	51	4077	124000	1	41.152061	15.08309	4250	4
33	34	51	3927	140000	2	40.755932	14.69019	5200	4
775	776	51	2588	51000	1	45.536591	10.23204	7900	4

1538 rows × 9 columns

In [97]:
```
y.groupby(['model']).count()
```

Out[97]:

model	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
lounge	1094	1094	1094	1094	1094	1094	1094	1094
pop	358	358	358	358	358	358	358	358
sport	86	86	86	86	86	86	86	86

In [98]:
```
a.tail(5)
```

Out[98]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price	model
1057	1058	51	3500	60000	2	45.438301	10.99170	6900	4
77	78	51	2739	77149	3	44.754890	8.03190	7800	4
172	173	51	4077	124000	1	41.152061	15.08309	4250	4
33	34	51	3927	140000	2	40.755932	14.69019	5200	4
775	776	51	2588	51000	1	45.536591	10.23204	7900	4

In [99]:
```
a.head(5)
```

Out[99]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price	model
0	1	51	882	25000	1	44.907242	8.61156	8900	1
952	953	51	2009	35552	1	42.562832	12.64220	8900	1
951	952	51	3684	78000	1	43.552910	10.31998	6500	1
950	951	51	762	36463	1	41.107880	14.20881	10900	1
949	950	51	790	32000	1	41.572239	13.33369	9490	1

In [100]:
```
data2=y
data2
```

Out[100]:

	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 [101]:
```
from sklearn.model_selection import train_test_split                                    #to import the sklearn
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.33,random_state=42)
```

In [166]:
```
from sklearn.linear_model import LinearRegression
reg= LinearRegression () #creatuing objects of linear regression
reg.fit(x_train,y_train)  #trainning and fitting lr objects trainning data
```

Out[166]:

▾ LinearRegression

LinearRegression()

In [167]:
```
pred=reg.predict(x_test)
pred
```

```
array([ 8426.37409382,  5447.47569851,  9914.20077691,  4687.39013431,
        7885.32100747,  5431.00822998,  9911.86294348, 10390.16991322,
        9680.84745901,  8844.57815539,  7764.08471024,  4257.54640953,
        9882.76503303, 10341.35258769,  5736.4484335 , 10179.87154436,
        9501.423448  ,  7997.3181334 ,  5532.33458288,  9894.57834738,
       10437.97459358,  6381.35845844,  9591.23555726,  9574.27908517,
       10322.30715736,  9501.22785499,  9789.955758  ,  9593.26549752,
        6775.82788536,  7915.34831306, 10389.98590521, 10351.58343315,
        7381.32686464,  9966.53983093, 10430.87188433, 10554.43156462,
       10285.85574963, 10035.88086558,  9526.63034431,  7742.78157141,
        9297.64938364, 10051.42272678, 10004.81256571,  9985.84167026,
        9374.6573594 ,  9561.57499854,  9754.94184269,  9819.85893758,
        8780.31447831,  6255.99008069,  6281.53627686,  8190.88781577,
        8588.91394592,  6566.97963218,  6850.70237466,  5511.29438169,
        8119.97866315,  9847.74830838,  7775.93862032,  9875.05509733,
       10121.29366536,  5791.92464084,  9835.42728501, 10043.91426822,
        8027.28015259,  4527.22080416, 10609.02444098,  3808.29240951,
        9952.37340054, 10511.20945172,  5746.34019592,  5486.40214756,
       10395.91036208,  6788.47519216,  8953.20120295, 10442.24187982,
        9455.6934072 ,  9976.26574762,  8528.35753837,  7960.77147517,
```

In [102]:
```
x_test.head(10)
```

Out[102]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price	model
481	482	51	3197	120000	2	40.174702	18.167629	7900	2
76	77	62	2101	103000	1	45.797859	8.644440	7900	2
1502	1503	51	670	32473	1	41.107880	14.208810	9400	1
669	670	51	913	29000	1	45.778591	8.946250	8500	1
1409	1410	51	762	18800	1	45.538689	9.928310	9700	1
1414	1415	51	762	39751	1	41.903221	12.495650	9900	1
1089	1090	51	882	33160	1	45.778999	12.997090	9900	1
1507	1508	51	701	17324	1	45.556549	9.534470	9950	1
970	971	51	701	29000	1	36.855839	14.760470	10700	1
1198	1199	51	1155	38000	1	41.239281	13.933020	8999	1

In [103]:
```
a=y.drop(["ID","lat","lon"],axis=1)                    #droping the data
a
```

Out[103]:

	model	engine_power	age_in_days	km	previous_owners	price
0	lounge	51	882	25000	1	8900
1	pop	51	1186	32500	1	8800
2	sport	74	4658	142228	1	4200
3	lounge	51	2739	160000	1	6000
4	pop	73	3074	106880	1	5700
...	...	...	...	...	...	...
1533	sport	51	3712	115280	1	5200
1534	lounge	74	3835	112000	1	4600
1535	pop	51	2223	60457	1	7500
1536	lounge	51	2557	80750	1	5990
1537	pop	51	1766	54276	1	7900

1538 rows × 6 columns

In [104]:
```
a=pd.get_dummies(a)            # to print the dummies
a
```

Out[104]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1538 rows × 8 columns

In [105]:
```
b=pd.get_dummies(a)
b
```

Out[105]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1538 rows × 8 columns

In [106]:
```
b.shape                                  # to find the shape in row and columns
```

Out[106]:
```
(1538, 8)
```

In [107]:
```
y=b["price"]
x=b.drop('price',axis=1)            # to delete the rows
```

In [108]:
```
y
```

Out[108]:
```
0       8900
1       8800
2       4200
3       6000
4       5700
        ...
1533    5200
1534    4600
1535    7500
1536    5990
1537    7900
Name: price, Length: 1538, dtype: int64
```

In [109]:
```
from sklearn.model_selection import train_test_split                                    #to import the sklearn
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.33,random_state=42)
```

In [110]:
```
x_test.head(10)
```

Out[110]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
481	51	3197	120000	2	0	1	0
76	62	2101	103000	1	0	1	0
1502	51	670	32473	1	1	0	0
669	51	913	29000	1	1	0	0
1409	51	762	18800	1	1	0	0
1414	51	762	39751	1	1	0	0
1089	51	882	33160	1	1	0	0
1507	51	701	17324	1	1	0	0
970	51	701	29000	1	1	0	0
1198	51	1155	38000	1	1	0	0

In [111]:
```
y_test.head(10)
```

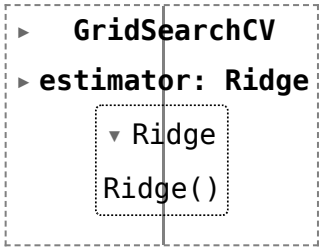
Out[111]:
```
481      7900
76       7900
1502     9400
669      8500
1409     9700
1414     9900
1089     9900
1507     9950
970     10700
1198     8999
Name: price, dtype: int64
```

In [112]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge

alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]

ridge = Ridge()

parameters = {'alpha': alpha}

ridge_regressor = GridSearchCV(ridge, parameters)

ridge_regressor.fit(x_train, y_train)
```

```
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=5.56109e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.70876e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.91585e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.08003e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.01022e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57959e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.24161e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.92759e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.09091e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
```

```
Ill-conditioned matrix (rcond=7.02112e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57414e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.23284e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.9277e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.09099e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.02123e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57407e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.23274e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Out[112]:



In [113]:
```
ridge_regressor.best_params_
```

Out[113]:
```
{'alpha': 30}
```

In [116]:
```
from sklearn.metrics import mean_squared_error          #calculating the mse
```

In [114]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
pred_ridge=ridge.predict(x_test)
```

In [119]:
```
ridge
```

Out[119]:
```
Ridge(alpha=30)
```

In [117]:
```
Ridge_Error=mean_squared_error(pred_ridge,y_test)
Ridge_Error                                                    #calculation of ridg error
```

Out[117]:
```
579521.7970897449
```

In [182]:
```
from sklearn.metrics import r2_score
r2_score(y_test,pred_ridge)
```

Out[182]:
```
0.8421969385523054
```

NEXT

In [129]:
```
y.groupby(['price']).count()
```

Out[129]:

price	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
2500	1	1	1	1	1	1	1	1
2900	1	1	1	1	1	1	1	1
3390	1	1	1	1	1	1	1	1
3500	1	1	1	1	1	1	1	1
3600	1	1	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...
10990	9	9	9	9	9	9	9	9
10999	5	5	5	5	5	5	5	5
11000	13	13	13	13	13	13	13	13
11090	2	2	2	2	2	2	2	2
11100	1	1	1	1	1	1	1	1

222 rows × 8 columns

In [142]:
```
b=y.loc[(y.model=="lounge")]
```

In [143]:
```
b
```

Out[143]:

	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
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
6	7	lounge	51	731	11600	1	44.907242	8.611560	10750
7	8	lounge	51	1521	49076	1	41.903221	12.495650	9190
11	12	lounge	51	366	17500	1	45.069679	7.704920	10990
...	...	...	...	...	...	...	...	...	...
1528	1529	lounge	51	2861	126000	1	43.841980	10.515310	5500
1529	1530	lounge	51	731	22551	1	38.122070	13.361120	9900
1530	1531	lounge	51	670	29000	1	45.764648	8.994500	10800
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990

1094 rows × 9 columns

In [145]:
```
c=l.drop("model",axis=1)                # to delete the data
c
```

Out[145]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	51	882	25000	1	44.907242	8.611560	8900
3	4	51	2739	160000	1	40.633171	17.634609	6000
6	7	51	731	11600	1	44.907242	8.611560	10750
7	8	51	1521	49076	1	41.903221	12.495650	9190
11	12	51	366	17500	1	45.069679	7.704920	10990
...	...	...	...	...	...	...	...	...
1528	1529	51	2861	126000	1	43.841980	10.515310	5500
1529	1530	51	731	22551	1	38.122070	13.361120	9900
1530	1531	51	670	29000	1	45.764648	8.994500	10800
1534	1535	74	3835	112000	1	45.845692	8.666870	4600
1536	1537	51	2557	80750	1	45.000702	7.682270	5990

1094 rows × 8 columns

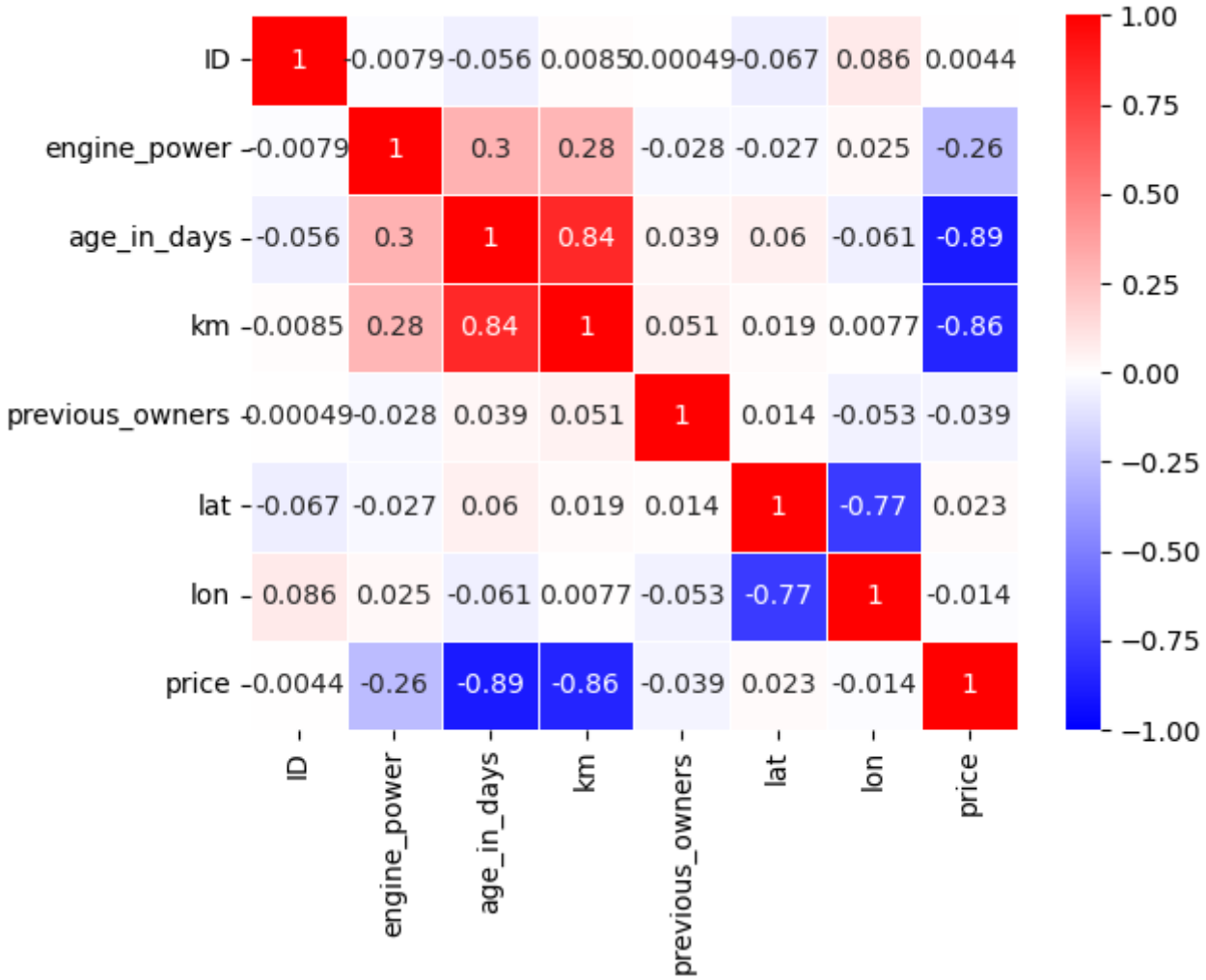
In [149]:
```
cory=c.corr()
cory                                    # to find the correlation
```

Out[149]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
ID	1.000000	-0.007859	-0.055600	0.008471	0.000486	-0.067365	0.085564	0.004406
engine_power	-0.007859	1.000000	0.300144	0.281481	-0.027861	-0.027428	0.025088	-0.260315
age_in_days	-0.055600	0.300144	1.000000	0.837210	0.038825	0.059653	-0.060897	-0.892417
km	0.008471	0.281481	0.837210	1.000000	0.050958	0.019335	0.007749	-0.859062
previous_owners	0.000486	-0.027861	0.038825	0.050958	1.000000	0.014027	-0.052962	-0.039448
lat	-0.067365	-0.027428	0.059653	0.019335	0.014027	1.000000	-0.769521	0.022973
lon	0.085564	0.025088	-0.060897	0.007749	-0.052962	-0.769521	1.000000	-0.014498
price	0.004406	-0.260315	-0.892417	-0.859062	-0.039448	0.022973	-0.014498	1.000000

In [164]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cory,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap="bwr")    # to print theheatmap
```

Out[164]: <Axes: >



In [181]:
```
re=pd.DataFrame(columns=['actual','predicted'])          #to compare the values
re['actual']=y_test
re['predicted']=pred
re=re.reset_index()
re['Id']=re.index                # to arrange the indexing

re.head(5)
```

Out[181]:

	index	actual	predicted	Id
0	481	7900	5867.650338	0
1	76	7900	7133.701423	1
2	1502	9400	9866.357762	2
3	669	8500	9723.288745	3
4	1409	9700	10039.591012	4

In [188]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.lineplot(x='Id',y='actual',data=re.head(100))
sns.lineplot(x='Id',y='predicted',data=re.head(100))       #to plot the line graph
plt.plot()
```

Out[188]: []