In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

read a file

In [2]:
```
sai=pd.read_csv('/home/placement/Downloads/fiat500 (1).csv')
```

info data

In [3]:
```
sai.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
```

describe data

In [4]:
```
sai.describe()
```

Out[4]:

	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

column names

In [5]:
```
list(sai)
```

Out[5]:
```
['ID',
 'model',
 'engine_power',
 'age_in_days',
 'km',
 'previous_owners',
 'lat',
 'lon',
 'price']
```

mapping strings

In [6]:
```
sai['model']=sai['model'].map({'lounge':1,'pop':2,'sport':3})
sai
```

Out[6]:

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

1538 rows × 9 columns

checking nullvalues

In [26]:
```
sai.isnull()
```

Out[26]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
1533	False	False	False	False	False	False	False	False	False
1534	False	False	False	False	False	False	False	False	False
1535	False	False	False	False	False	False	False	False	False
1536	False	False	False	False	False	False	False	False	False
1537	False	False	False	False	False	False	False	False	False

1538 rows × 9 columns

searching for certain values

In [7]:
```
k=sai.loc[(sai.km<=50000)]
k
```

Out[7]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	1	51	882	25000	1	44.907242	8.61156	8900
1	2	2	51	1186	32500	1	45.666359	12.24189	8800
6	7	1	51	731	11600	1	44.907242	8.61156	10750
7	8	1	51	1521	49076	1	41.903221	12.49565	9190
10	11	2	51	790	43286	1	40.871429	14.43896	8950
...	...	...	...	...	...	...	...	...	...
1525	1526	1	51	790	41870	1	45.707249	11.47760	9500
1526	1527	1	51	1705	23600	1	38.122070	13.36112	9300
1527	1528	2	51	517	3000	1	40.748241	14.52835	9999
1529	1530	1	51	731	22551	1	38.122070	13.36112	9900
1530	1531	1	51	670	29000	1	45.764648	8.99450	10800

907 rows × 9 columns

groupby command

In [8]:
```
k=sai.groupby(['price']).count()
k
```

Out[8]:

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

rename columns

In [9]:
```
p=sai.rename(columns={'model_name':'model','age_in_days':'age','previous_owners':'pre'})
list(p)
```

Out[9]:
```
['ID', 'model', 'engine_power', 'age', 'km', 'pre', 'lat', 'lon', 'price']
```

eliminating columns

In [10]:
```
d=p.drop(['model','age','lat','lon','engine_power','ID'],axis=1)
d
```

Out[10]:

	km	pre	price
0	25000	1	8900
1	32500	1	8800
2	142228	1	4200
3	160000	1	6000
4	106880	1	5700
...	...	...	...
1533	115280	1	5200
1534	112000	1	4600
1535	60457	1	7500
1536	80750	1	5990
1537	54276	1	7900

1538 rows × 3 columns

In [11]:
```
k=d.groupby(['pre']).count()
k
```

Out[11]:

	km	price
pre		
1	1389	1389
2	117	117
3	23	23
4	9	9

correlaton for certain columns

In [21]:
```
c=d.corr()
c
```

Out[21]:

	km	pre	price
km	1.000000	0.097539	-0.859373
pre	0.097539	1.000000	-0.076274
price	-0.859373	-0.076274	1.000000

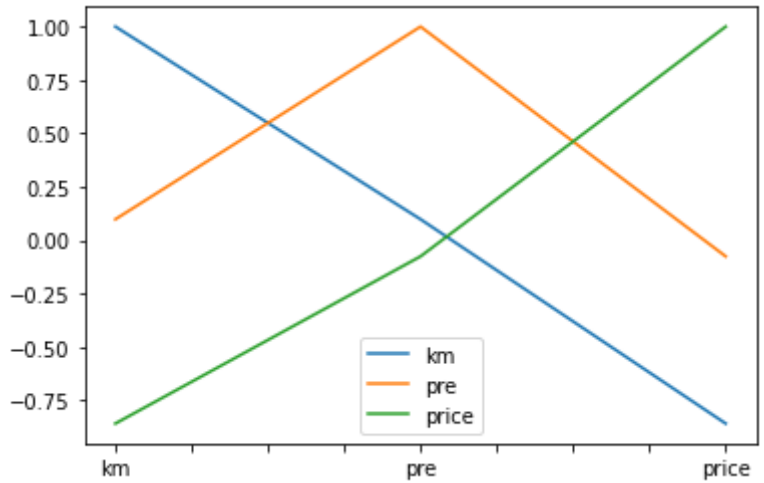
In [22]:
```
a=abs(c)
a
```

Out[22]:

	km	pre	price
km	1.000000	0.097539	0.859373
pre	0.097539	1.000000	0.076274
price	0.859373	0.076274	1.000000

In [23]:
```
c.plot()
```

Out[23]: <Axes: >

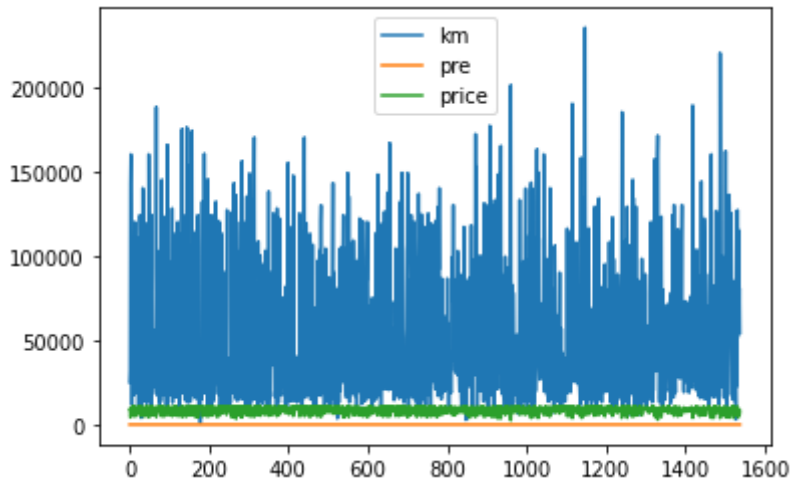


In [25]:
```
sns.heatmap(c,vmax=1,vmin=-1,annot=True,linewidth=5,cmap='bwr')
```

Out[25]: <Axes: >

In [24]:
```
#d.plot ()
```

Out[24]: <Axes: >



In [16]:
```
k=d.groupby(['price']).count()
k
```

Out[16]:

price	km	pre
2500	1	1
2900	1	1
3390	1	1
3500	1	1
3600	1	1
...	...	...
10990	9	9
10999	5	5
11000	13	13
11090	2	2
11100	1	1

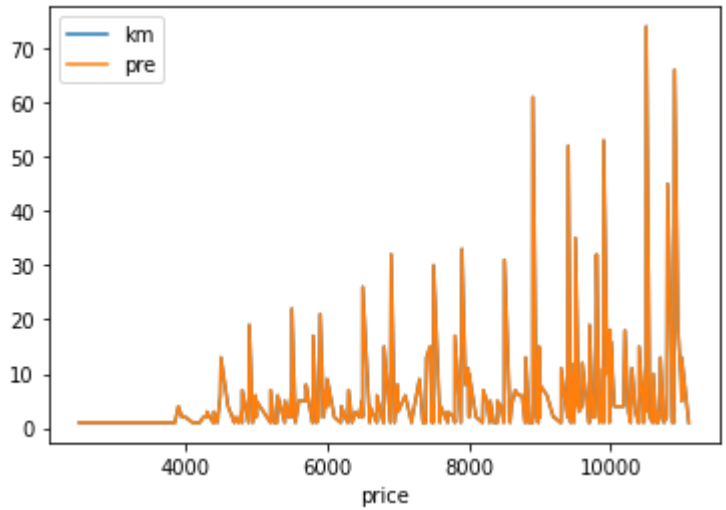
222 rows × 2 columns

save this file into csv

In [17]:
```
k.to_csv('ksp.csv')
```

In [18]:
```
k.plot()
```

Out[18]: <Axes: xlabel='price'>



correlation

In [19]:
```
cor=sai.corr()
cor
```

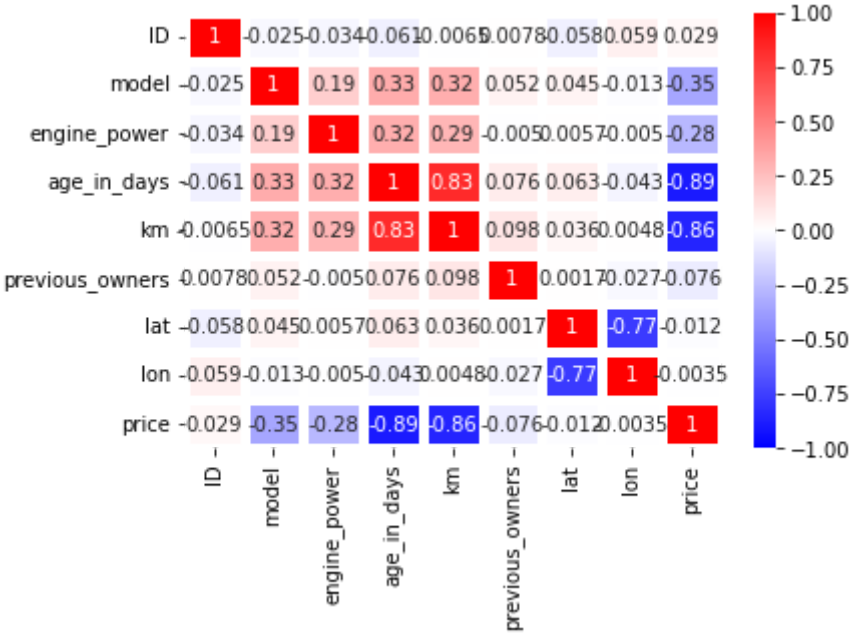
Out[19]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
ID	1.000000	-0.024740	-0.034059	-0.060753	-0.006537	0.007803	-0.058207	0.058941	0.028516
model	-0.024740	1.000000	0.189906	0.326508	0.319580	0.052480	0.044901	-0.013200	-0.349885
engine_power	-0.034059	0.189906	1.000000	0.319190	0.285495	-0.005030	0.005721	-0.005032	-0.277235
age_in_days	-0.060753	0.326508	0.319190	1.000000	0.833890	0.075775	0.062982	-0.042667	-0.893328
km	-0.006537	0.319580	0.285495	0.833890	1.000000	0.097539	0.035519	0.004839	-0.859373
previous_owners	0.007803	0.052480	-0.005030	0.075775	0.097539	1.000000	0.001697	-0.026836	-0.076274
lat	-0.058207	0.044901	0.005721	0.062982	0.035519	0.001697	1.000000	-0.766646	-0.011733
lon	0.058941	-0.013200	-0.005032	-0.042667	0.004839	-0.026836	-0.766646	1.000000	-0.003541
price	0.028516	-0.349885	-0.277235	-0.893328	-0.859373	-0.076274	-0.011733	-0.003541	1.000000

correlation plot

In [30]:
```
h=sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidth=5,cmap='bwr')
h
```

Out[30]: <Axes: >



In [ ]: