In [1]:
```
import pandas as pd
```

In [2]:
```
data=pd.read_csv('fiat500.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	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 [4]:
```
data1=data.loc[(data.km<=50000)]
```

In [5]:
```
data1

```

Out[5]:

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

907 rows × 9 columns

In [6]:
```
data2=data1.groupby(['model']).count()
```

In [7]:
```
data2
```

Out[7]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
model								
lounge	734	734	734	734	734	734	734	734
pop	162	162	162	162	162	162	162	162
sport	11	11	11	11	11	11	11	11

In [8]:
```
data2=data2.rename(columns={'age_in_days':'ageindays'})
list(data2)
```

Out[8]:
```
['ID',
 'engine_power',
 'ageindays',
 'km',
 'previous_owners',
 'lat',
 'lon',
 'price']
```

In [9]:
```
data2.head()
```

Out[9]:

	ID	engine_power	ageindays	km	previous_owners	lat	lon	price
model								
lounge	734	734	734	734	734	734	734	734
pop	162	162	162	162	162	162	162	162
sport	11	11	11	11	11	11	11	11

In [10]:
```
print(data.to_string())
```

```
        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
10      11     pop            51          790   43286                1  40.871429  14.438960   8950
11      12  lounge            51          366   17500                1  45.069679   7.704920  10990
12      13  lounge            51          456   18450                1  45.426571  11.788130   9700
13      14     pop            51         3835  120000                1  40.531590  17.436159   4800
14      15  lounge            51         1035   40500                1  40.911362  14.211200   9300
15      16  lounge            51         1096   28200                1  45.697208   9.845970   9500
16      17  lounge            73         4200  110000                1  41.082352  14.254250   5250
17      18     pop            51         2223   96848                1  43.782372  11.254990   7990
```

In [11]:
```
data.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 [ ]:
```
#day2
```

In [24]:
```
data1=pd.read_csv('fiat500.csv')
```

In [26]:
```
list(data1)
```

Out[26]:
```
['ID',
 'model',
 'engine_power',
 'age_in_days',
 'km',
 'previous_owners',
 'lat',
 'lon',
 'price']
```

In [27]:
```
data1.head
```

Out[27]:
```
<bound method NDFrame.head of         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 [29]:
```
data1.corr()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[29], line 1
----> 1 data1.corr()

File ~/.local/lib/python3.8/site-packages/pandas/core/frame.py:10054, in DataFrame.corr(self, method, min
_periods, numeric_only)
  10052 cols = data.columns
  10053 idx = cols.copy()
> 10054 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10056 if method == "pearson":
  10057     correl = libalgos.nancorr(mat, minp=min_periods)

File ~/.local/lib/python3.8/site-packages/pandas/core/frame.py:1838, in DataFrame.to_numpy(self, dtype, c
opy, na_value)
   1836 if dtype is not None:
   1837     dtype = np.dtype(dtype)
-> 1838 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1839 if result.dtype is not dtype:
   1840     result = np.array(result, dtype=dtype, copy=False)

File ~/.local/lib/python3.8/site-packages/pandas/core/internals/managers.py:1732, in BlockManager.as_arra
y(self, dtype, copy, na_value)
   1730         arr.flags.writeable = False
   1731 else:
-> 1732     arr = self._interleave(dtype=dtype, na_value=na_value)
   1733     # The underlying data was copied within _interleave, so no need
   1734     # to further copy if copy=True or setting na_value
   1736 if na_value is not lib.no_default:

File ~/.local/lib/python3.8/site-packages/pandas/core/internals/managers.py:1794, in BlockManager._interl
eave(self, dtype, na_value)
   1792     else:
   1793         arr = blk.get_values(dtype)
-> 1794     result[rl.indexer] = arr
   1795     itemmask[rl.indexer] = 1
   1797 if not itemmask.all():
```

```
ValueError: could not convert string to float: 'lounge'
```

In [28]:
```
data2=data1.drop(['model'],axis=1) #it drops the coloumn
```

In [30]:
```
data2.head()
```

Out[30]:

	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

In [31]:
```
data2.corr()
```

Out[31]:

	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 [ ]:
```
#Last day
```

In [12]:
```
data.groupby(['model']).count()
```

Out[12]:

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 [13]:
```
'''num=int(input())
n=100
c=0
while num>0:
    for i in range(2,n):
        if n%i!=0:
            c=c+1
        if (c==n-2):
                print(n)
                num-=1
    n+=1
    c=0'''
```

Out[13]:
```
'num=int(input())\nn=100\nc=0\nwhile num>0:\n    for i in range(2,n):\n        if n%i!=0:\n            c=c+
1\n        if (c==n-2):\n                print(n)\n                num-=1\n    n+=1\n    c=0'
```

In [14]:
```
data['model']=data['model'].map({'lounge':1,'pop':2,'sport':3})
```

In [15]:
```
data.head()
```

Out[15]:

	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

In [16]:
```
data.describe()
```

Out[16]:

	ID	model	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	1538.000000
mean	769.500000	1.344603	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.563428	8576.003901
std	444.126671	0.581296	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
min	1.000000	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
25%	385.250000	1.000000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
50%	769.500000	1.000000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
75%	1153.750000	2.000000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
max	1538.000000	3.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

In [17]:
```
list(data)
```

Out[17]:
```
['ID',
 'model',
 'engine_power',
 'age_in_days',
 'km',
 'previous_owners',
 'lat',
 'lon',
 'price']
```

In [18]:
```
data2=data.rename(columns={'price':'cost'})
```

In [19]:
```
data2
```

Out[19]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	cost
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

In [20]:
```
cor=data.corr()
```

In [21]:
```
cor
```

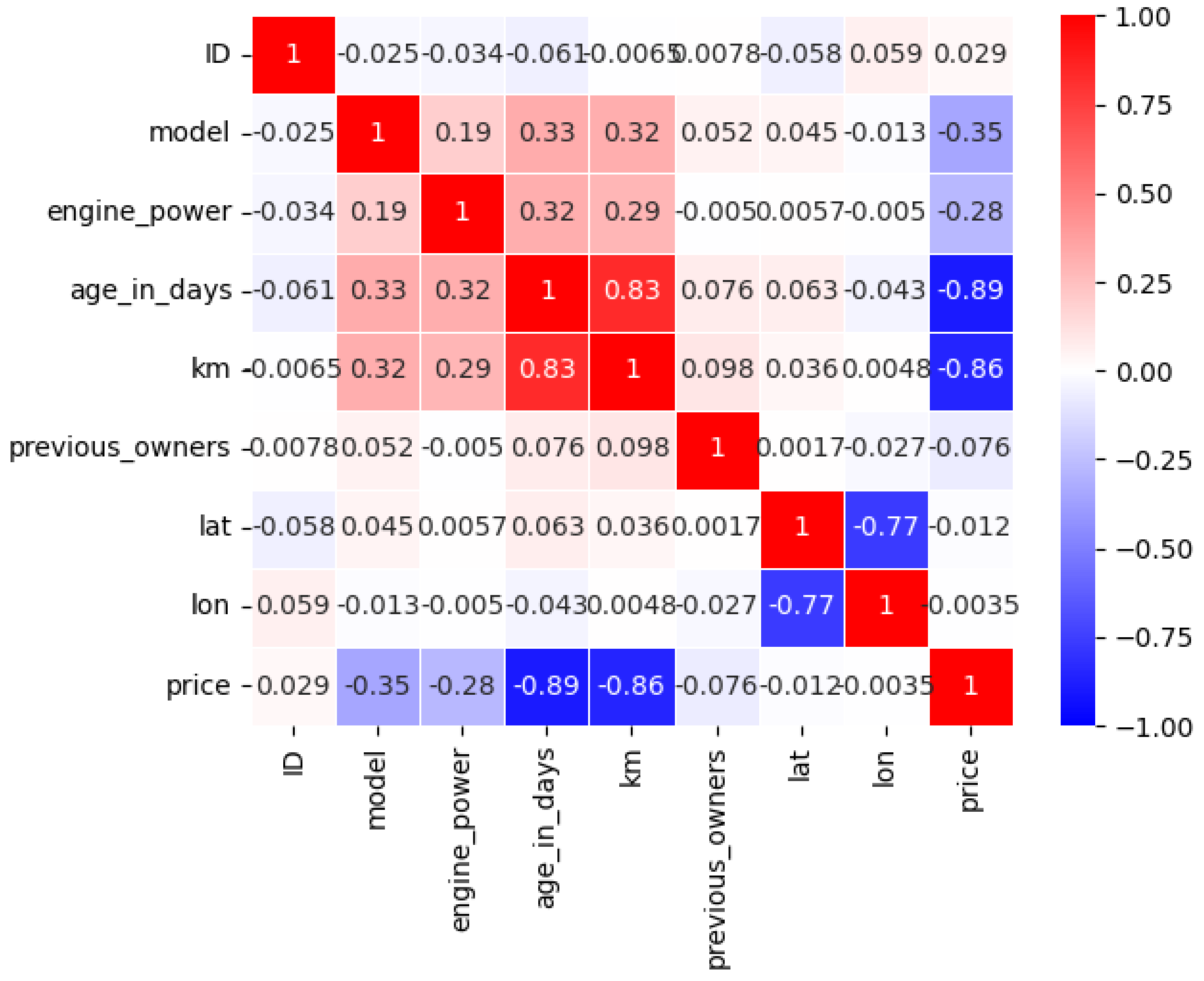
Out[21]:

	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

In [22]:
```
import seaborn as sns
```

In [23]:
```
sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidth=0.5,cmap='bwr')
```

Out[23]:
```
<Axes: >
```



	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
ID	1	-0.025	-0.034	-0.061	-0.0065	0.0078	-0.058	0.059	0.029
model	-0.025	1	0.19	0.33	0.32	0.052	0.045	-0.013	-0.35
engine_power	-0.034	0.19	1	0.32	0.29	-0.005	0.0057	-0.005	-0.28
age_in_days	-0.061	0.33	0.32	1	0.83	0.076	0.063	-0.043	-0.89
km	-0.0065	0.32	0.29	0.83	1	0.098	0.036	0.0048	-0.86
previous_owners	0.0078	0.052	-0.005	0.076	0.098	1	0.0017	-0.027	-0.076
lat	-0.058	0.045	0.0057	0.063	0.036	0.0017	1	-0.77	-0.012
lon	0.059	-0.013	-0.005	-0.043	0.0048	-0.027	-0.77	1	-0.0035
price	0.029	-0.35	-0.28	-0.89	-0.86	-0.076	-0.012	-0.0035	1

In [ ]:

```

```