fiat500 Data

In [4]:

```
import pandas as pd
import numpy as np
data=pd.read_csv("/home/placement/Downloads/fiat500 (2).csv")
data.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

Deleating the model column

In [5]:
```
data=data.drop('model',axis=1)
data
```

Out[5]:

	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

Correlation

In [6]:
```
cor=data.corr()
cor
```

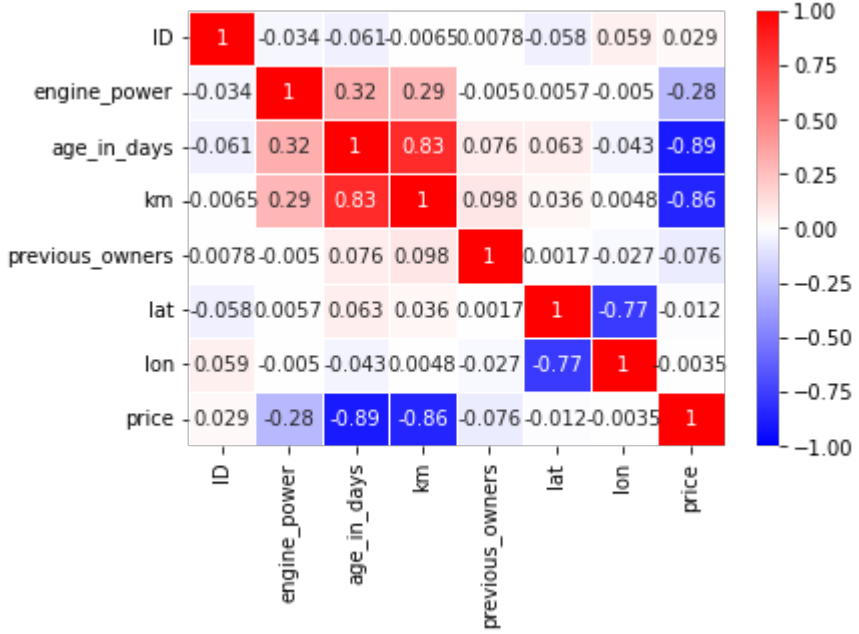
Out[6]:

	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

Graphical representation

In [7]:
```
import seaborn as s
s.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap='bwr')
```

Out[7]: <Axes: >



In [8]:
```
data=data.drop(['ID','lat','lon'],axis=1)
```

In [9]:
```
data
```

Out[9]:

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

1538 rows × 5 columns

In [10]:
```
y=data['price']
x=data.drop("price",axis=1)
y
```

Out[10]:
```
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 [11]:
```
x
```

Out[11]:

	engine_power	age_in_days	km	previous_owners
0	51	882	25000	1
1	51	1186	32500	1
2	74	4658	142228	1
3	51	2739	160000	1
4	73	3074	106880	1
...	...	...	...	...
1533	51	3712	115280	1
1534	74	3835	112000	1
1535	51	2223	60457	1
1536	51	2557	80750	1
1537	51	1766	54276	1

1538 rows × 4 columns

Training&Testing Data

In [12]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [13]:
```
x_test.head()
```

Out[13]:

	engine_power	age_in_days	km	previous_owners
481	51	3197	120000	2
76	62	2101	103000	1
1502	51	670	32473	1
669	51	913	29000	1
1409	51	762	18800	1

In [14]:
```
x_train.head()
```

Out[14]:

	engine_power	age_in_days	km	previous_owners
527	51	425	13111	1
129	51	1127	21400	1
602	51	2039	57039	1
331	51	1155	40700	1
323	51	425	16783	1

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

Out[15]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

In [16]:
```
y_train.head()
```

Out[16]:
```
527    9990
129    9500
602    7590
331    8750
323    9100
Name: price, dtype: int64
```

In [17]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[17]:
```
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 [18]:
```
ypred=reg.predict(x_test)
```

In [19]:
```
ypred
```

Out[19]:
```
array([ 6073.66164032,  7341.33056605,  9800.22745889,  9652.96481488,
         9974.18650172,  9584.74918024,  9602.66085856, 10054.79501723,
         9837.76145484,  9274.72564967, 10376.40925293,  7627.2708871 ,
         7586.13886127,  6777.61713754,  9594.3515114 , 10315.65507697,
         9741.59372697,  7572.9040847 ,  4992.69453982, 10398.20922435,
        10321.59089881, 10334.24308334,  7774.10509188,  9888.15483782,
         7227.85647519,  9252.73227807,  4979.11139564,  6828.17363834,
         7702.42621854,  9553.85486262,  7221.04780645,  5073.0165671 ,
         5556.4267214 ,  4970.49802742,  8882.94456337,  5519.6923554 ,
        10087.83968148,  8160.28949584,  6151.8911467 ,  8696.51762984,
         9705.09169354,  6977.32588859,  9321.40522585, 10481.43915706,
         8600.7328436 , 10269.7650493 ,  9318.5378423 ,  8786.37011809,
         6926.31348142,  8994.3404015 ,  9345.36114439, 10234.9879404 ,
        10007.10480998,  6962.45146182,  9719.34818612,  9614.70537838,
         9648.65369512, 10386.17797314,  9727.11367001,  7418.70343875,
        10043.96106656,  6886.34161907,  9786.2978823 ,  7033.53810698,
         6280.47449776,  9931.48504001,  9712.48972008,  8754.87304997,
         8365.19863387,  6367.14830383,  7659.17854481,  6924.82229663,
         8252.99687794, 10381.79601472,  7245.56863861,  8472.19600962,
```

Efficiency of Model

In [20]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[20]:
```
0.8401365357197939
```

In [21]:
```
from sklearn.metrics import mean_squared_error as me
kk=me(y_test,ypred)
kk
```

Out[21]:
```
587088.4966282183
```

Rms value

In [22]:
```
import math as m
k=m.sqrt(kk)
print(k)
```

```
766.2170036146538
```

In [23]:
```
results=pd.DataFrame(columns=["price","predicted"])
results["price"]=y_test
results["predicted"]=ypred
```

In [24]:
```
results.head(10)
```

Out[24]:

	price	predicted
481	7900	6073.661640
76	7900	7341.330566
1502	9400	9800.227459
669	8500	9652.964815
1409	9700	9974.186502
1414	9900	9584.749180
1089	9900	9602.660859
1507	9950	10054.795017
970	10700	9837.761455
1198	8999	9274.725650

In [25]:
```
results["actual price"]=results.apply(lambda column:column.price-column.predicted,axis=1)
```

In [26]:
```
results
```

Out[26]:

	price	predicted	actual price
481	7900	6073.661640	1826.338360
76	7900	7341.330566	558.669434
1502	9400	9800.227459	-400.227459
669	8500	9652.964815	-1152.964815
1409	9700	9974.186502	-274.186502
...	...	...	...
291	10900	9967.877499	932.122501
596	5699	6481.067824	-782.067824
1489	9500	10257.140033	-757.140033
1436	6990	8287.581490	-1297.581490
575	10900	10313.573220	586.426780

508 rows × 3 columns

In [ ]:
```

```