Linear regression

In [2]:

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

Out[2]:

	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 [3]:
```
data=data.drop('model',axis=1)
data
```

Out[3]:

	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 [4]:
```
cor=data.corr()
cor
```

Out[4]:

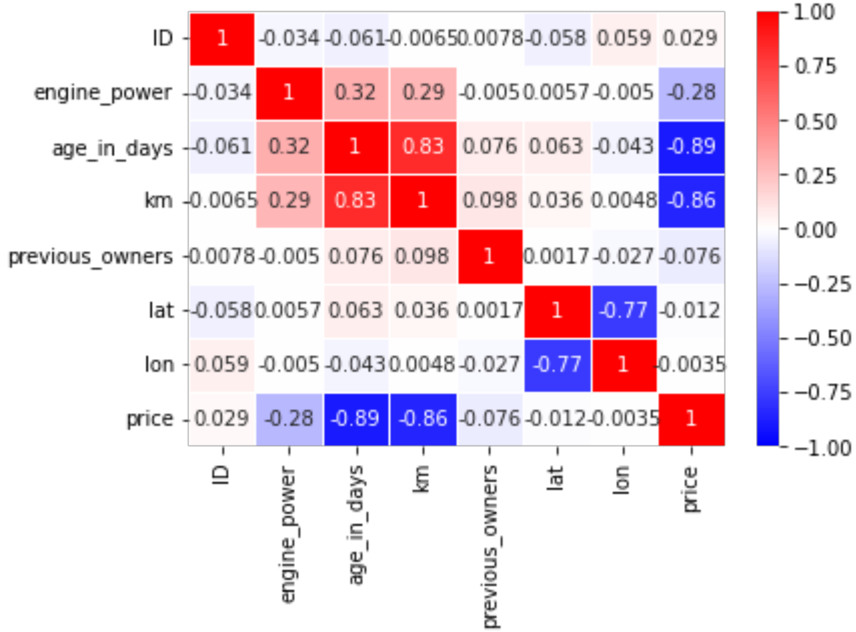
	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
ID	1.000000	-0.034059	-0.060753	-0.006537	0.007803	-0.058207	0.058941	0.028516
engine_power	-0.034059	1.000000	0.319190	0.285495	-0.005030	0.005721	-0.005032	-0.277235
age_in_days	-0.060753	0.319190	1.000000	0.833890	0.075775	0.062982	-0.042667	-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 [5]:

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

Out[5]:

```
<Axes: >
```



In [6]:

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

In [7]:
```
y=data['price']
x=data.drop("price",axis=1)
y
```

Out[7]:
```
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 [8]:
```
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 [9]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[9]:
```
▾ LinearRegression

LinearRegression()
```

In [ ]:
```

```

In [10]:
```
ypred=reg.predict(x_test)
```

In [11]:
```
from sklearn.metrics import r2_score
lE=r2_score(y_test,ypred)
lE
```

Out[11]:
```
0.8401365357197939
```

Ridgeregression

In [12]:
```
li=pd.read_csv("/home/placement/Downloads/fiat500 (2).csv")
li
```

Out[12]:

	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 [13]:
```
li=li.drop("model",axis=1)
li
```

Out[13]:

	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

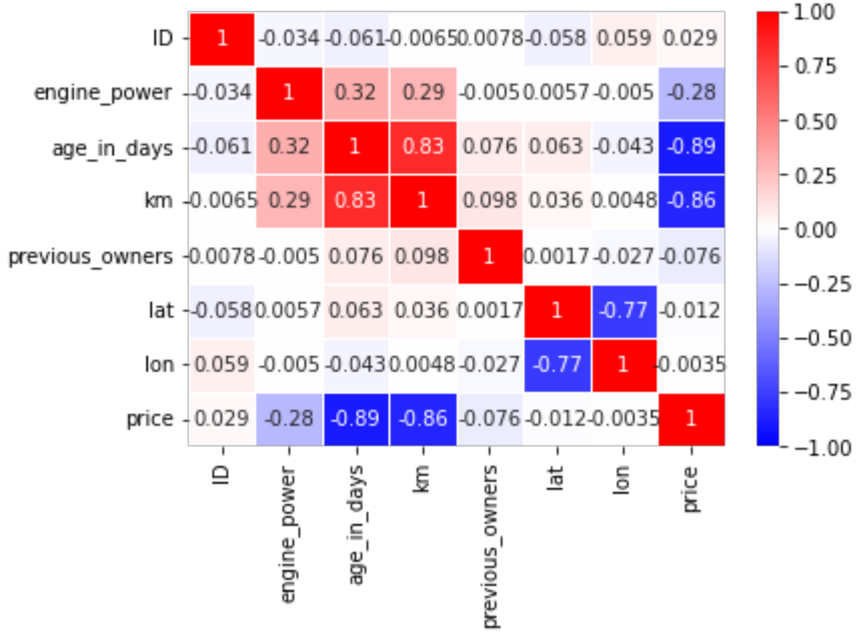
In [14]:
```
cor=li.corr()
cor
```

Out[14]:

	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 [15]:
```
import seaborn as s
s.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap='bwr')
```

Out[15]: <Axes: >



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

In [16]:
```
y=li['price']
x=li.drop("price",axis=1)
y
```

Out[16]:
```
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 [17]:
```
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 [18]:
```
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)
```

Out[18]:

▸ **GridSearchCV**

▸ **estimator: Ridge**

▸ Ridge

In [19]:
```
ridge_regressor.best_params_
```

Out[19]:
```
{'alpha': 30}
```

In [20]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [21]:
```
from sklearn.metrics import r2_score
RE=r2_score(y_test,y_pred_ridge)
RE
```

Out[21]: 0.8415256179582116

Elastic regression

In [22]:
```
re=pd.read_csv("/home/placement/Downloads/fiat500 (2).csv")
re
```

Out[22]:

	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 [23]:
```
re=re.drop("model",axis=1)
re
```

Out[23]:

	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 [24]:
```
#re=re.drop(['ID','lat','lon'],axis=1)
#re
```

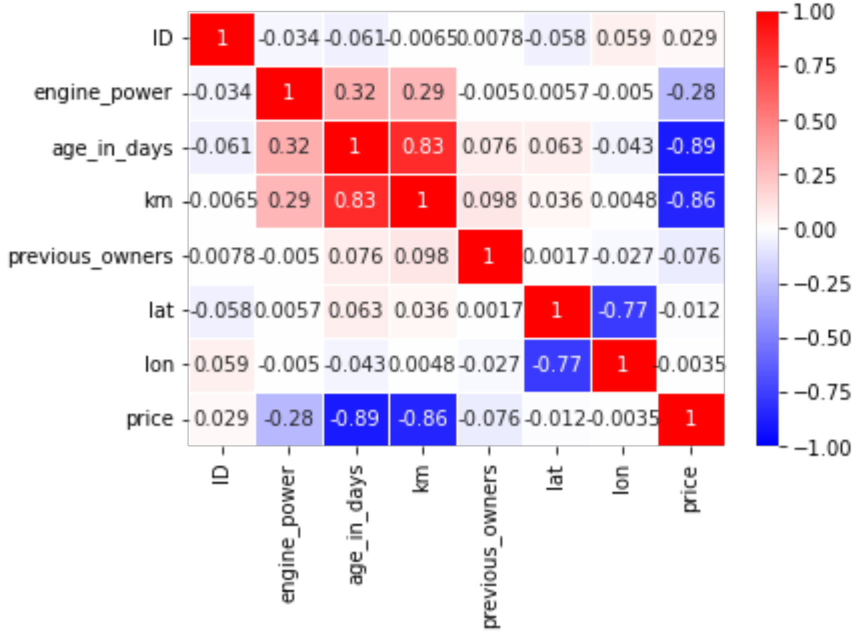
In [25]:
```
cor=re.corr()
cor
```

Out[25]:

	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 [26]:
```
import seaborn as s
s.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap='bwr')
```

Out[26]: <Axes: >



In [27]:
```
y=re['price']
x=re.drop("price",axis=1)
y
```

Out[27]:
```
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 [28]:
```
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import ElasticNet

elastic = ElasticNet()

parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]}

elastic_regressor = GridSearchCV(elastic, parameters)

elastic_regressor.fit(x_train,y_train)
```

Out[28]:
```
▸      GridSearchCV
▸ estimator: ElasticNet
     ▸ ElasticNet
```

In [29]:
```
elastic_regressor.best_params_
```
Out[29]:
```
{'alpha': 1}
```

In [35]:
```
elastic=ElasticNet(alpha=30)
elastic.fit(x_train,y_train)
y_pred_ridge=elastic.predict(x_test)
```

In [36]:
```
from sklearn.metrics import r2_score
EE=r2_score(y_test,y_pred_ridge)
EE
```
Out[36]:
```
0.841507172811023
```

Efficiencies

In [37]:
```
lE#liner regresssion efficiency
```
Out[37]:
```
0.8401365357197939
```

In [33]:
```
RE#ridge regresssion efficiency
```
Out[33]:
```
0.8415256179582116
```

In [34]:
```
EE#elastic regresssion efficiency
```
Out[34]:
```
0.8415256179582116
```

In [ ]: