mk 31/07/23

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

In [185]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\1_fiat500_VehicleSelection_Dataset.csv")
a
```

Out[185]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700
...	...	...	...	...	...	...	...	...	...
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	5
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lonprice
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	NO
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	1
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	1

1549 rows × 11 columns

In [147]:
```
a=a.head(10)
a
```

Out[147]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	Uni
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900	
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800	
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200	
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000	
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700	
5	6.0	pop	74.0	3623.0	70225.0	1.0	45.000702	7.68227005	7900	
6	7.0	lounge	51.0	731.0	11600.0	1.0	44.907242	8.611559868	10750	
7	8.0	lounge	51.0	1521.0	49076.0	1.0	41.903221	12.49565029	9190	
8	9.0	sport	73.0	4049.0	76000.0	1.0	45.548000	11.54946995	5600	
9	10.0	sport	51.0	3653.0	89000.0	1.0	45.438301	10.99170017	6000	

In [148]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               10 non-null     float64
 1   model            10 non-null     object 
 2   engine_power     10 non-null     float64
 3   age_in_days      10 non-null     float64
 4   km               10 non-null     float64
 5   previous_owners  10 non-null     float64
 6   lat              10 non-null     float64
 7   lon              10 non-null     object 
 8   price            10 non-null     object 
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      0 non-null      object 
dtypes: float64(7), object(4)
memory usage: 1008.0+ bytes
```

In [149]:
```
a.columns
```

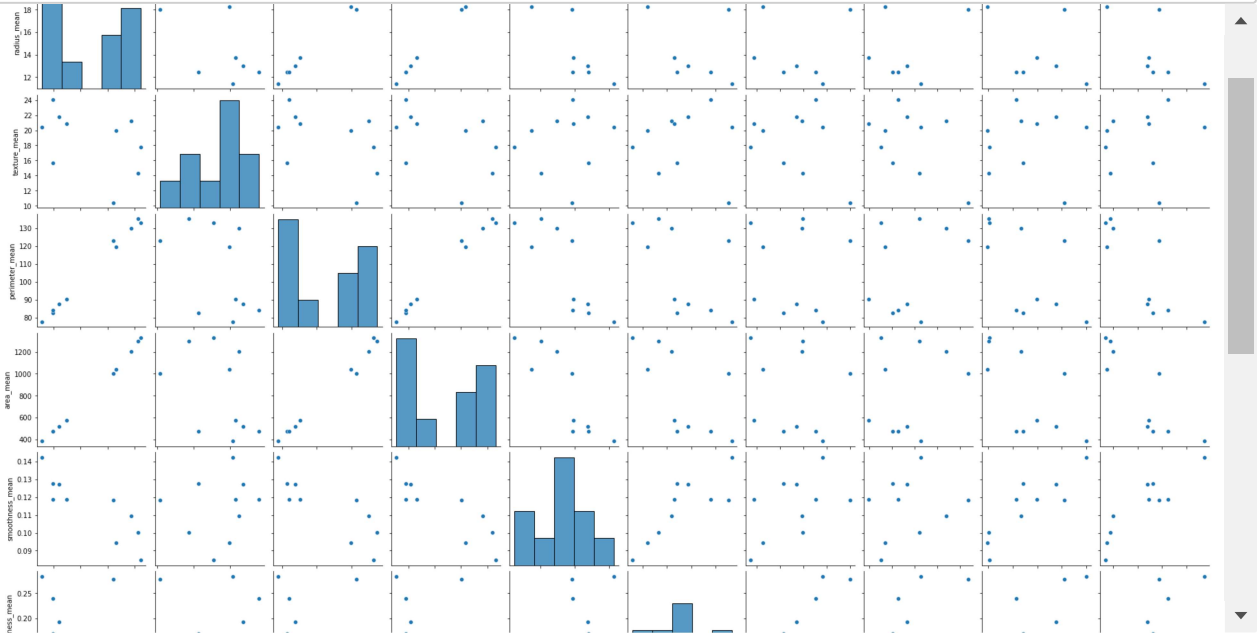
Out[149]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10'],
      dtype='object')
```

In [150]:
```
a.describe()
```

Out[150]:

	ID	engine_power	age_in_days	km	previous_owners	lat	Unnamed: 9
count	10.00000	10.000000	10.000000	10.000000	10.0	10.000000	0.0
mean	5.50000	60.000000	2611.600000	76250.900000	1.0	44.141076	NaN
std	3.02765	11.623731	1427.557214	49399.679798	0.0	1.887936	NaN
min	1.00000	51.000000	731.000000	11600.000000	1.0	40.633171	NaN
25%	3.25000	51.000000	1269.750000	36644.000000	1.0	42.654226	NaN
50%	5.50000	51.000000	2906.500000	73112.500000	1.0	44.953972	NaN
75%	7.75000	73.000000	3645.500000	102410.000000	1.0	45.487050	NaN
max	10.00000	74.000000	4658.000000	160000.000000	1.0	45.666359	NaN

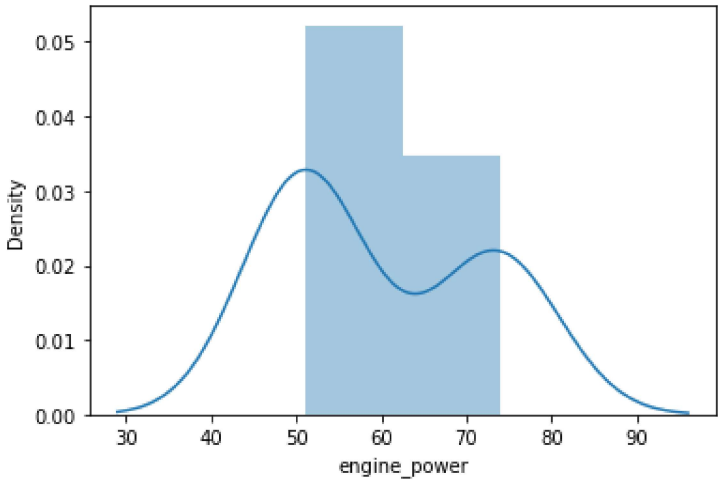
In [151]:
```
sns.pairplot(d)
```



In [152]:
```
sns.distplot(a['engine_power'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

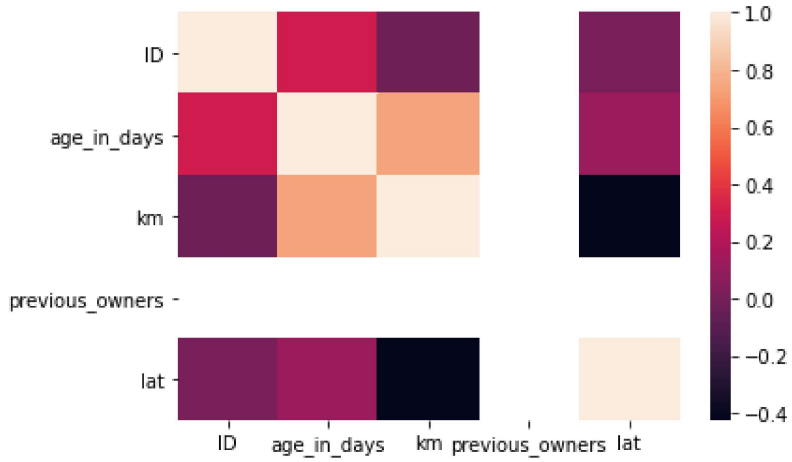
Out[152]:
```
<AxesSubplot:xlabel='engine_power', ylabel='Density'>
```



In [160]:
```
x1=a[['ID', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
```

In [161]:
```
sns.heatmap(x1.corr())
```

Out[161]: <AxesSubplot:>



In [162]:
```
x=a[['ID',  'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
y=a['engine_power']
```

In [163]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [164]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[164]: LinearRegression()

In [165]:
```
print(lr.intercept_)
```

```
12.805330256601351
```

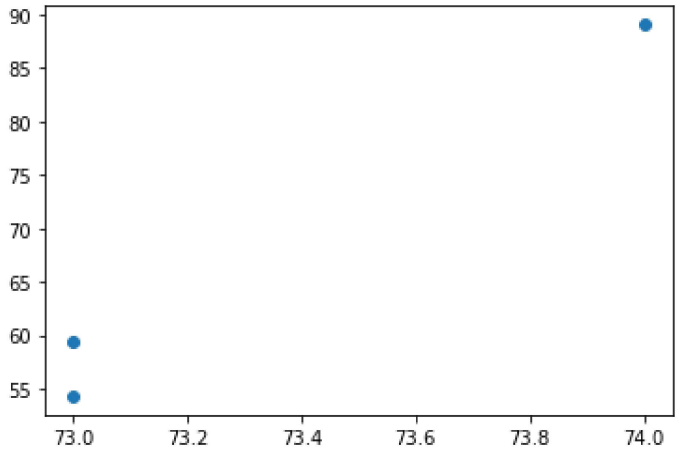
In [166]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[166]:

	Co-efficient
ID	-3.212770e+00
age_in_days	2.481704e-02
km	-4.344668e-05
previous_owners	-1.134110e-13
lat	-1.872889e+00
lon	-3.776369e-01
price	1.213071e-02

In [167]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[167]:
```
<matplotlib.collections.PathCollection at 0x190b4727a90>
```



In [168]:
```
print(lr.score(x_test,y_test))
```

```
-1141.0902643146762
```

In [169]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [170]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[170]:
```
Ridge(alpha=10)
```

In [171]:
```
rr.score(x_test,y_test)
```

Out[171]:
```
-948.517696237313
```

In [172]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[172]:
```
Lasso(alpha=10)
```

In [173]:
```
la.score(x_test,y_test)
```

Out[173]:
```
-700.233967798521
```

In [174]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[174]:
```
ElasticNet()
```

In [175]:
```
print(en.coef_)
```

```
[-2.30993644e+00  1.86610900e-02  4.08050256e-05  0.00000000e+00
 -0.00000000e+00 -1.85408711e-01  1.02213674e-02]
```

In [176]:
```
print(en.intercept_)
```

```
-54.553348066026395
```

In [177]:
```
print(en.predict(x_test))
```

```
[51.56739341 81.38613809 58.4154446 ]
```

In [178]:
```
en.score(x_test,y_test)
```

Out[178]:
```
-1088.9313759284828
```

In [179]:
```
from sklearn import metrics
```

In [180]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 15.790532559434105
```

In [181]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 253.7978365143725
```

In [182]:
```
print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
 Root Mean Squared Error 15.931033755358518
```