In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df = pd.read_csv("1_fiat500_VehicleSelection_Dataset.csv")[0:1500].dropna(axis="columns
df
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...
1495	1496.0	pop	62.0	3347.0	80000.0	3.0	44.283878	11.88813972	7900
1496	1497.0	pop	51.0	1461.0	91055.0	3.0	44.508839	11.46907997	7450
1497	1498.0	lounge	51.0	397.0	15840.0	3.0	38.122070	13.36112022	10700
1498	1499.0	sport	51.0	1400.0	60000.0	1.0	45.802021	9.187789917	10800
1499	1500.0	pop	51.0	1066.0	53100.0	1.0	38.122070	13.36112022	8900

1500 rows × 9 columns

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

Out[3]:

	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

Data cleaning and pre processing

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1500 non-null   float64
 1   model            1500 non-null   object 
 2   engine_power     1500 non-null   float64
 3   age_in_days      1500 non-null   float64
 4   km               1500 non-null   float64
 5   previous_owners  1500 non-null   float64
 6   lat              1500 non-null   float64
 7   lon              1500 non-null   object 
 8   price            1500 non-null   object 
dtypes: float64(6), object(3)
memory usage: 105.6+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	lat
count	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000
mean	750.500000	51.875333	1641.629333	53074.900000	1.126667	43.545904
std	433.157015	3.911606	1288.091104	39955.013731	0.421197	2.112907
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839
25%	375.750000	51.000000	670.000000	20000.000000	1.000000	41.802990
50%	750.500000	51.000000	1035.000000	38720.000000	1.000000	44.360376
75%	1125.250000	51.000000	2616.000000	78170.250000	1.000000	45.467960
max	1500.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612

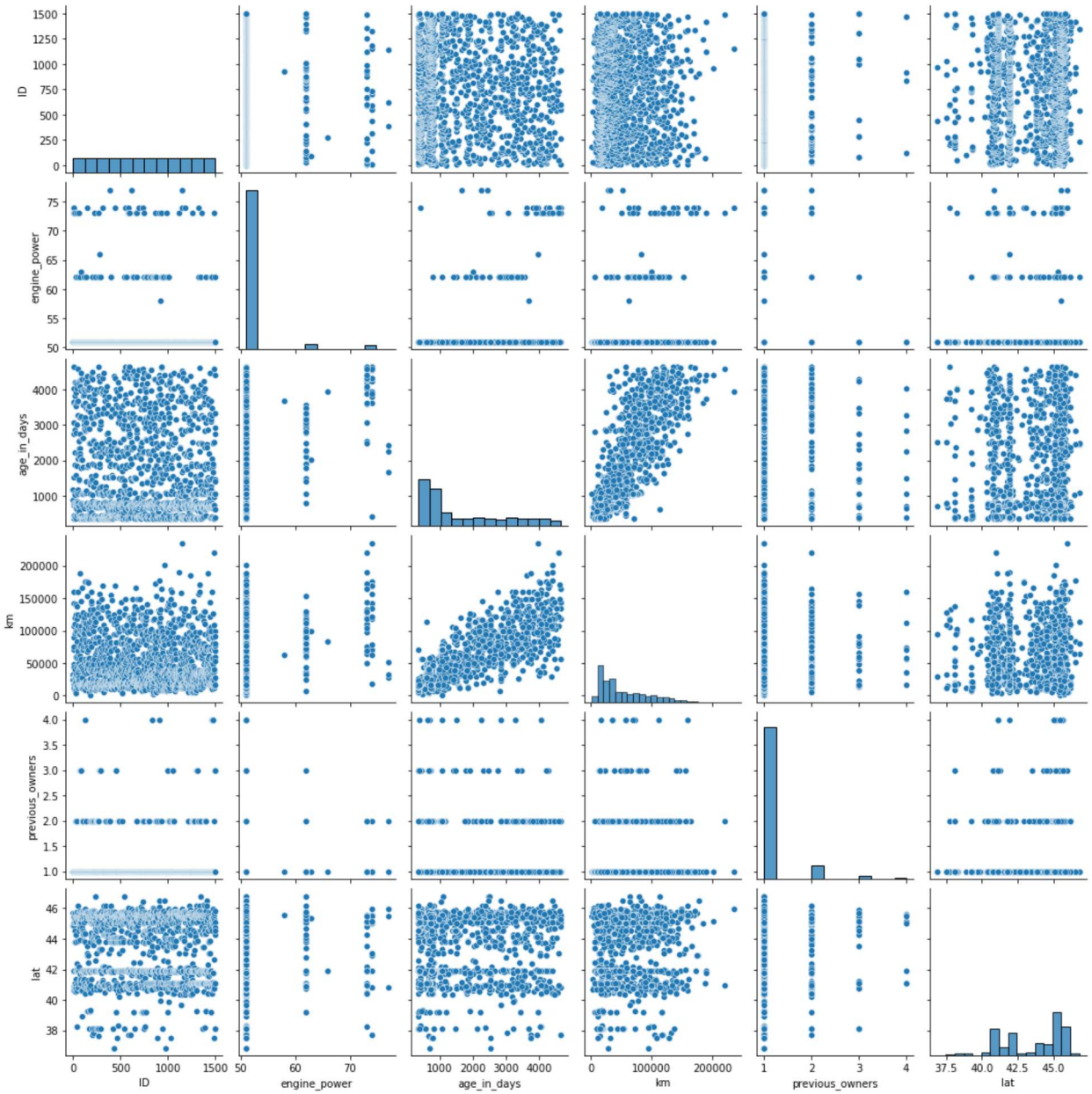
In [6]:
```
df.columns
```

Out[6]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

EDA and VISUALIZATION

In [7]:
```
sns.pairplot(df)
```

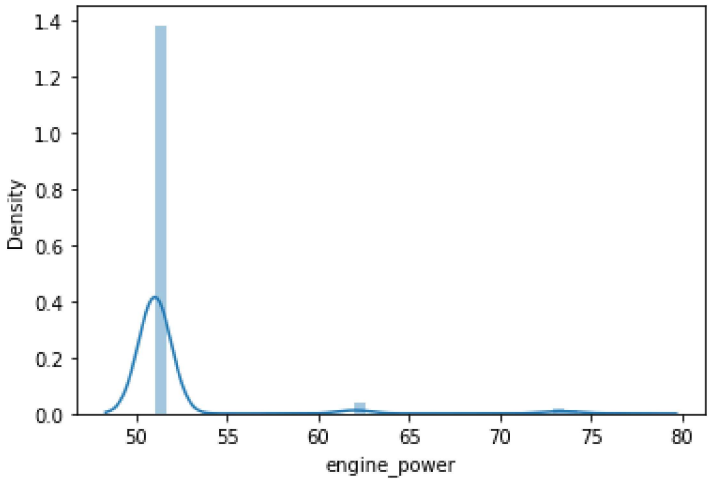
Out[7]: <seaborn.axisgrid.PairGrid at 0x13e8d38ea60>



In [8]:
```
sns.distplot(df["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)
```

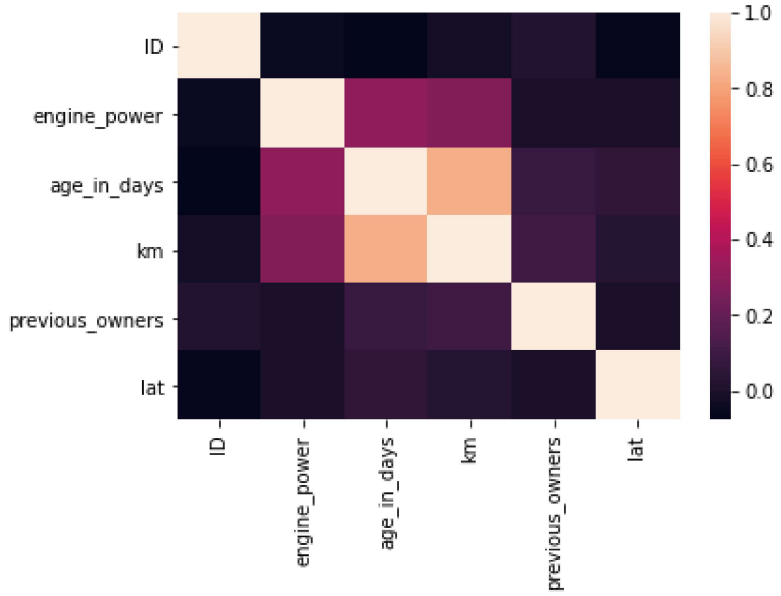
Out[8]: <AxesSubplot:xlabel='engine_power', ylabel='Density'>



In [9]:
```
df1 = df[['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
```

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>



In [11]:
```
x = df1[['ID', 'age_in_days', 'km', 'previous_owners',
       'lat']]
y = df1['engine_power']
```

split the data into training and test data

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
lr.intercept_
```

Out[14]:
```
52.701983657814935
```

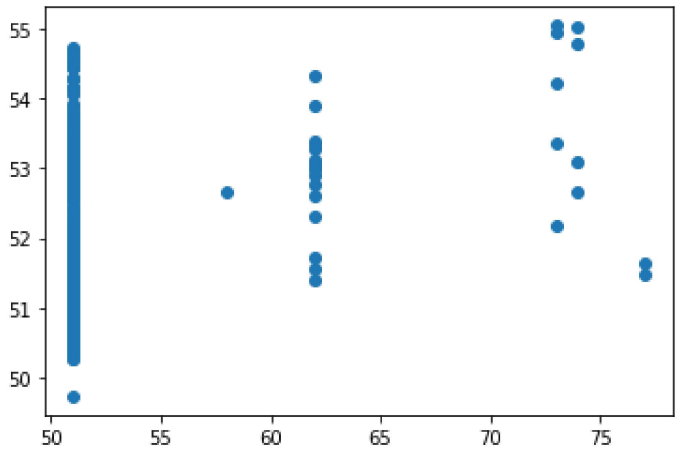
In [15]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
ID	-0.000312
age_in_days	0.000413
km	0.000018
previous_owners	-0.201347
lat	-0.047461

In [16]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x13e90284be0>
```



In [17]:
```
lr.score(x_test,y_test)
```

Out[17]:
```
0.0922569149071083
```

In [18]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[18]:
```
ElasticNet()
```

In [19]:
```
print(en.coef_)
```

```
[-3.03184056e-04  4.06678564e-04  1.74747153e-05 -0.00000000e+00
 -0.00000000e+00]
```

In [20]:
```
print(en.intercept_)
```

```
50.4206700808331
```

In [21]:

```
print(en.predict(x_test))
```

```
[50.73103787 50.93632915 50.66462291 54.2128627  50.91561487 52.57214706
 51.06650245 51.21592527 52.88669156 51.6794588  50.83406197 53.87115775
 50.64095564 52.11696653 51.3811121  53.21700911 50.75989083 51.21650726
 50.52225085 51.08726976 53.25511277 51.70875091 51.22391422 50.79441675
 53.39805424 50.87550961 50.72721915 50.51305626 53.99856918 52.82324083
 50.60741972 53.02716007 50.7281364  51.16921353 51.06693188 52.81998875
 50.82870026 51.67773295 50.82908514 53.29080566 51.10786263 51.9946137
 51.55241962 54.49398801 50.65558019 50.62356176 50.68686395 51.19393916
 51.11744571 51.01860562 53.3009708  51.28649858 50.92370379 52.95528705
 51.37028974 52.59145268 53.32440311 52.99975627 50.97598943 50.87168374
 51.20992104 51.62693604 51.10430369 50.7065716  50.69600139 51.57001399
 51.98952191 50.79078954 50.75735413 50.42097558 50.491092   50.85312562
 50.79708491 50.52093905 52.22977118 50.69113335 51.97012648 50.59307944
 50.81825359 51.34801072 50.90563686 53.17544091 53.71283035 51.26328174
 50.96385653 53.6286169  51.91595905 50.71958705 50.4055553  50.57756182
 54.31125864 51.978403   50.89963231 54.2367647  50.39922718 51.19679438
 53.60801224 50.75394806 51.25577713 51.00688714 50.7920553  50.64817654
 50.9070437  50.89918215 50.81692702 50.59448925 50.69396649 50.5674132
 50.55154421 52.56416014 52.78465492 50.91853652 51.27634399 52.41596063
 50.61833243 51.0753915  53.10489718 51.08312262 50.78890543 51.1550434
 50.83158084 51.88035951 51.05450908 51.06387051 51.32492361 51.70344028
 51.4275245  50.94797831 54.46651642 52.73165902 50.65153433 50.83899654
 50.76525059 51.65802669 53.47138847 51.22862152 50.86652319 50.73912561
 50.77211268 52.78186156 51.69863804 52.31187698 53.23448286 51.19036804
 50.82669171 51.19282793 52.03224591 54.45384868 50.89318211 51.27491268
 51.22195129 51.7122503  50.75538217 52.86215481 50.79957935 50.60438537
 50.92635741 51.59356971 52.1635857  53.17767134 50.86863626 53.99323821
 52.97565332 50.86868436 51.10367041 51.4343301  51.32732073 50.84088637
 51.2135418  51.61894152 50.79926616 53.88681486 50.72618618 52.03766434
 54.51289902 54.90740901 51.3215635  53.50185289 52.65721426 51.06636907
 50.83345166 54.17828788 53.07620284 50.88697493 50.57301299 51.06755484
 52.52627595 52.7411545  51.16531256 51.29880782 50.68643455 54.69997234
 51.19090532 54.88056037 50.76639174 51.9312796  51.42129557 50.922232
 51.06234546 50.93693519 50.82549132 50.75511555 54.17097323 53.59012663
 51.78377245 50.77714477 53.25581535 50.49900692 52.99124419 51.590806
 51.63673108 51.01311218 50.72891065 52.45427574 52.08290115 50.81919006
 50.65631568 51.23028566 53.37727151 53.09437195 51.3140726  54.13397539
 50.96478609 52.69223046 52.97682015 50.57535852 53.03072069 52.8422771
 51.74848197 50.42491698 51.12705662 51.65099887 52.72155782 53.46088406
 50.65741177 50.60895754 50.93654202 50.88276416 50.74395193 52.80661111
 53.08248074 50.72514329 50.65578611 50.96648133 50.72196997 51.98991205
 51.07941165 52.08450627 52.0590918  51.12475377 50.99253182 50.84664709
 52.96854764 51.28360718 50.59425526 50.75070234 51.39784086 51.12427367
 51.22411279 51.00684657 50.57416256 52.61660247 50.5797545  51.78253059
 50.41761852 53.80991286 52.9472605  51.57024322 51.40474964 50.86569267
 50.70275935 50.69391533 51.41714809 51.03310242 53.32655651 50.87963971
 51.01657459 50.99619111 50.42733018 51.29885807 51.47031274 54.04787372
 50.83473996 50.86080505 51.05315334 50.79840699 53.90722553 50.73850333
 50.86714657 51.35027325 51.72523453 51.74952812 52.9311397  51.0235495
 50.78556564 51.75679743 51.26424095 51.79919913 52.92780754 54.36255657
 55.05353345 50.79156626 50.96838633 50.5958236  51.64311174 50.77829274
 51.50094763 50.75259238 51.08337842 52.43498653 51.39245575 51.10566811
 50.80786201 53.39266129 53.89139863 54.79946308 51.96591801 50.67452383
 50.45279764 50.47049959 51.0329638  50.7860209  51.52707234 51.31880492
 50.79467036 52.66618842 52.04836187 53.12463309 50.61247442 52.50020008
 50.71284523 52.26624653 51.50692268 52.43874544 52.13037025 50.80208857
 50.90239419 53.81024336 50.9473464  53.32754036 53.43754454 53.09565516
 54.09339023 50.97001832 50.80122252 52.6933505  51.53982984 50.98023401
 53.62770316 51.39365529 50.99711825 51.05844791 50.91906272 51.90079281
 50.62533136 50.56817182 50.87344232 50.8900982  50.47554222 50.81427878
 53.53057527 51.61255924 52.72917833 53.40869339 51.03160868 50.50915064
```

```
 51.159632   50.70644394 50.90054032 50.86628543 51.74144922 50.55848206
 53.19762748 54.16587792 53.15549115 51.14746219 50.65572419 50.48147042
 50.8187041  51.05552549 50.86683026 51.54868543 53.61152887 50.88405851
 51.5907741  53.13193688 51.01601228 50.85506101 51.8047449  54.42542265
 50.5088516  50.8456449  51.44466875 51.81692783 53.5442642  51.72891617
 52.99522149 51.03230049 53.09387672 50.74682689 51.09518727 52.72623711
 50.61925293 50.44323596 52.8818917  50.98371329 53.67165257 51.21182653
 53.17653499 51.00840314 52.15212113 53.44577886 52.27102642 52.64984538
 50.8920842  50.40475125 53.0172232  50.92306419 53.11940929 51.02932034
 50.93390384 50.92458261 51.79787546 52.86134589 50.8543491  50.72885718
 53.62200823 50.78857365 50.7012506  50.46284987 51.09949456 51.08133138
 53.74108463 54.75063743 50.44082431 50.73139247 53.03716513 50.63761143
 52.69622978 50.8851304  50.73491945 50.42885837 51.23575689 50.45361704
 53.61482608 53.68005819 51.29164251 50.9913395  53.68003436 52.36138327]
```

In [22]:
```
print(en.score(x_test,y_test))
```

```
0.09148080520644752
```

In [25]:
```
from sklearn import metrics
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Mean Absolytre Error: 1.6071776422470854
Mean Squared Error: 15.025416295329613
Root Mean Squared Error: 3.8762631870565256
```

In [ ]:
```

```