wbtliapnf

July 28, 2023

[3]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```python
df=pd.read_csv("/content/1_fiat500_VehicleSelection_Dataset.csv")
df
```

[4]:
```
        ID   model  engine_power  age_in_days        km  previous_owners  \
0      1.0  lounge          51.0        882.0   25000.0              1.0
1      2.0     pop          51.0       1186.0   32500.0              1.0
2      3.0   sport          74.0       4658.0  142228.0              1.0
3      4.0  lounge          51.0       2739.0  160000.0              1.0
4      5.0     pop          73.0       3074.0  106880.0              1.0
…     …       …             …            …         …                …
1544   NaN     NaN           NaN          NaN       NaN              NaN
1545   NaN     NaN           NaN          NaN       NaN              NaN
1546   NaN     NaN           NaN          NaN       NaN              NaN
1547   NaN     NaN           NaN          NaN       NaN              NaN
1548   NaN     NaN           NaN          NaN       NaN              NaN

            lat          lon     price  Unnamed: 9 Unnamed: 10
0     44.907242  8.611559868      8900         NaN         NaN
1     45.666359  12.24188995      8800         NaN         NaN
2     45.503300     11.41784      4200         NaN         NaN
3     40.633171  17.63460922      6000         NaN         NaN
4     41.903221  12.49565029      5700         NaN         NaN
…          …           …       …         …           …
1544        NaN       length         5         NaN         NaN
1545        NaN       concat  lonprice         NaN         NaN
1546        NaN  Null values        NO         NaN         NaN
1547        NaN         find         1         NaN         NaN
1548        NaN       search         1         NaN         NaN

[1549 rows x 11 columns]
```

[5]:
```python
df.head()
```

[5]:
```
    ID   model  engine_power  age_in_days        km  previous_owners  \
0  1.0  lounge          51.0        882.0   25000.0              1.0
1  2.0     pop          51.0       1186.0   32500.0              1.0
2  3.0   sport          74.0       4658.0  142228.0              1.0
3  4.0  lounge          51.0       2739.0  160000.0              1.0
4  5.0     pop          73.0       3074.0  106880.0              1.0

         lat          lon price  Unnamed: 9 Unnamed: 10
0  44.907242  8.611559868  8900         NaN         NaN
1  45.666359  12.24188995  8800         NaN         NaN
2  45.503300     11.41784  4200         NaN         NaN
3  40.633171  17.63460922  6000         NaN         NaN
4  41.903221  12.49565029  5700         NaN         NaN
```

1 DATA CLEANING AND DATA PREPROCESSING

[6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1549 entries, 0 to 1548
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               1538 non-null   float64
 1   model            1538 non-null   object
 2   engine_power     1538 non-null   float64
 3   age_in_days      1538 non-null   float64
 4   km               1538 non-null   float64
 5   previous_owners  1538 non-null   float64
 6   lat              1538 non-null   float64
 7   lon              1549 non-null   object
 8   price            1549 non-null   object
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      1 non-null      object
dtypes: float64(7), object(4)
memory usage: 133.2+ KB
```

[7]:
```python
df.describe()
```

[7]:
```
                ID  engine_power  age_in_days            km  previous_owners  \
count  1538.000000   1538.000000  1538.000000   1538.000000      1538.000000
mean    769.500000     51.904421  1650.980494  53396.011704         1.123537
std     444.126671      3.988023  1289.522278  40046.830723         0.416423
min       1.000000     51.000000   366.000000   1232.000000         1.000000
25%     385.250000     51.000000   670.000000  20006.250000         1.000000
50%     769.500000     51.000000  1035.000000  39031.000000         1.000000
```

```
75%     1153.750000     51.000000  2616.000000   79667.750000           1.000000
max     1538.000000     77.000000  4658.000000  235000.000000           4.000000

               lat  Unnamed: 9
count  1538.000000         0.0
mean     43.541361         NaN
std       2.133518         NaN
min      36.855839         NaN
25%      41.802990         NaN
50%      44.394096         NaN
75%      45.467960         NaN
max      46.795612         NaN
```

[8]:
```
df.columns
```

[8]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10'],
      dtype='object')
```

[9]:
```
df1=df[0:1500]
```

[10]:
```
df1=df1.dropna(axis=1)
df1
```

[10]:
```
          ID   model  engine_power  age_in_days        km  previous_owners  \
0        1.0  lounge          51.0        882.0   25000.0              1.0
1        2.0     pop          51.0       1186.0   32500.0              1.0
2        3.0   sport          74.0       4658.0  142228.0              1.0
3        4.0  lounge          51.0       2739.0  160000.0              1.0
4        5.0     pop          73.0       3074.0  106880.0              1.0
…        …      …            …            …         …              …
1495  1496.0     pop          62.0       3347.0   80000.0              3.0
1496  1497.0     pop          51.0       1461.0   91055.0              3.0
1497  1498.0  lounge          51.0        397.0   15840.0              3.0
1498  1499.0   sport          51.0       1400.0   60000.0              1.0
1499  1500.0     pop          51.0       1066.0   53100.0              1.0

            lat          lon  price
0     44.907242  8.611559868   8900
1     45.666359  12.24188995   8800
2     45.503300     11.41784   4200
3     40.633171  17.63460922   6000
4     41.903221  12.49565029   5700
…           …           …     …
1495  44.283878  11.88813972   7900
1496  44.508839  11.46907997   7450
1497  38.122070  13.36112022  10700
```

```
1498  45.802021  9.187789917  10800
1499  38.122070  13.36112022   8900

[1500 rows x 9 columns]
```

[11]:
```
df1.columns
```

[11]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

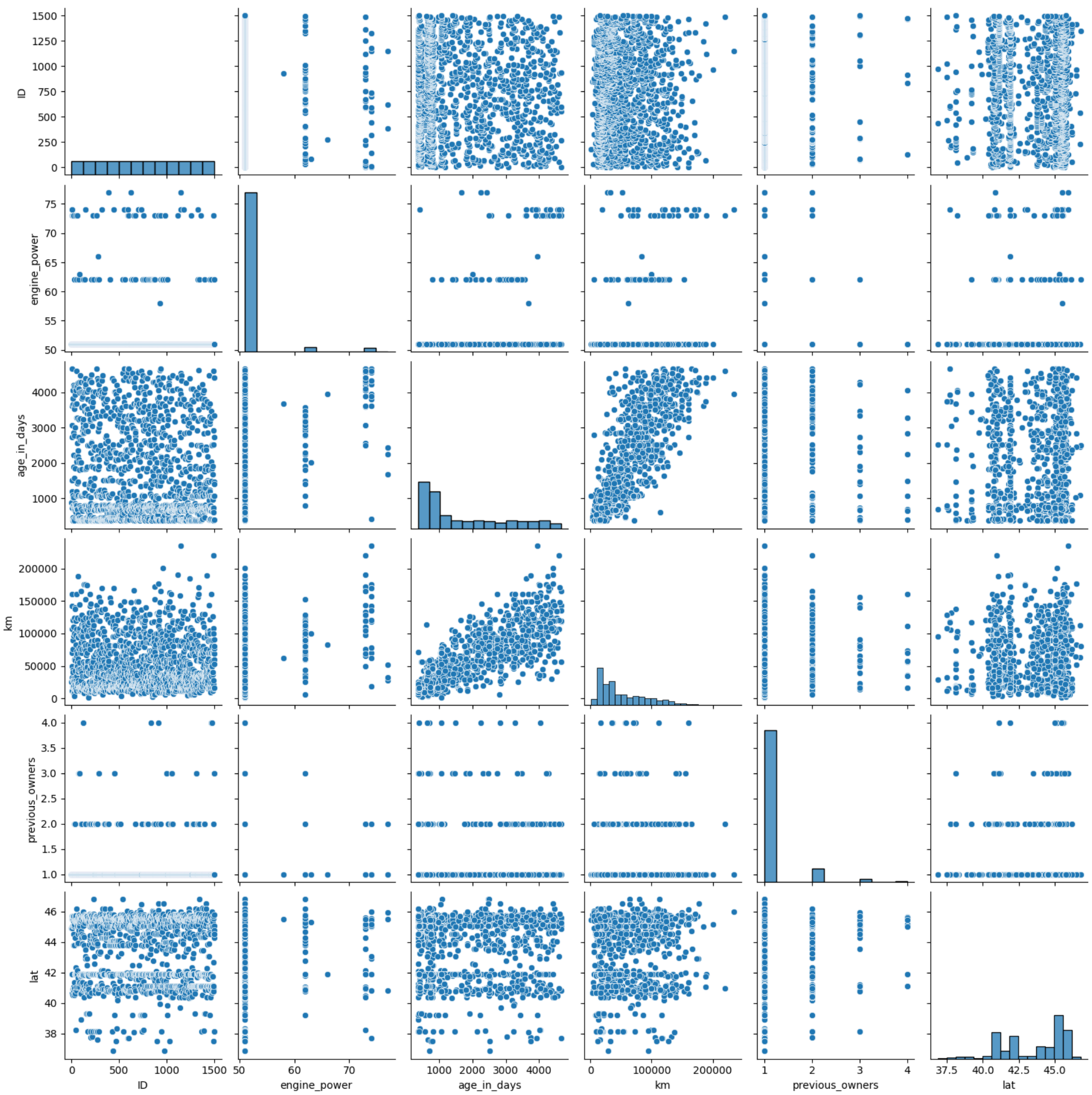
[11]:
```

```

[12]:
```
df1=df1[['ID',  'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat']]
```

2 EDA AND VISUALIZATION

[13]:
```
sns.pairplot(df1)
```

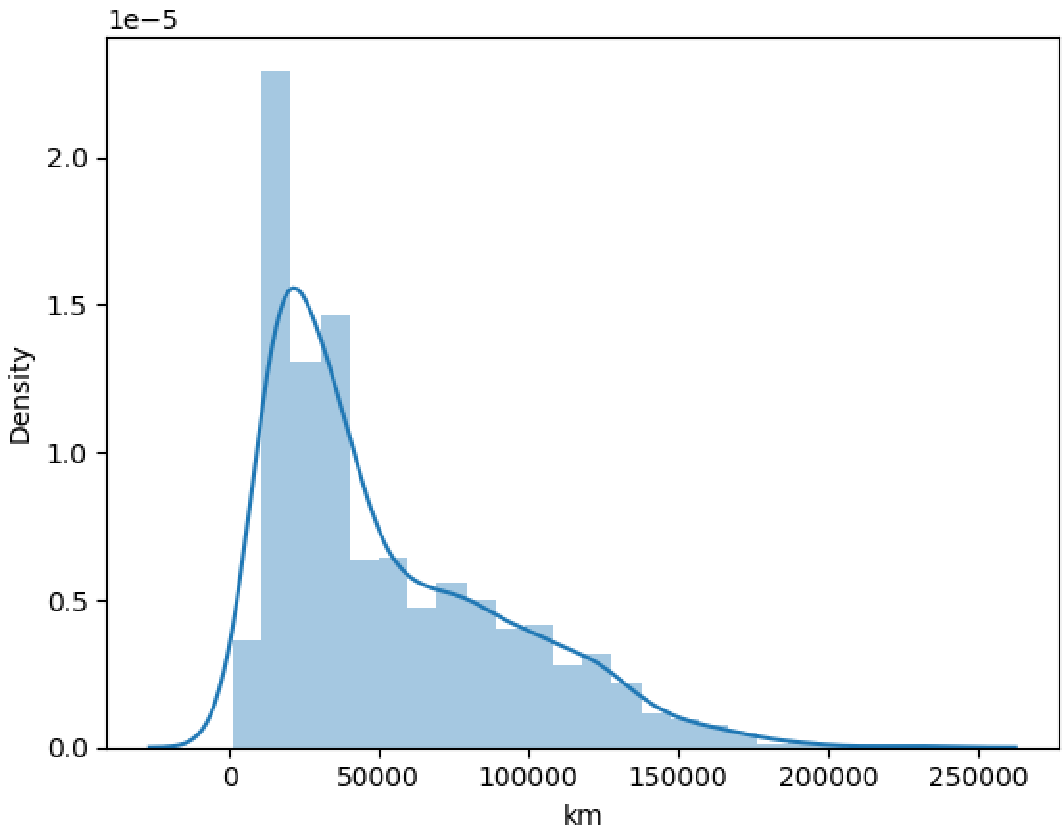
[13]:
```
<seaborn.axisgrid.PairGrid at 0x7e6dfe587430>
```



[14]:
```
sns.distplot(df1['km'])
```

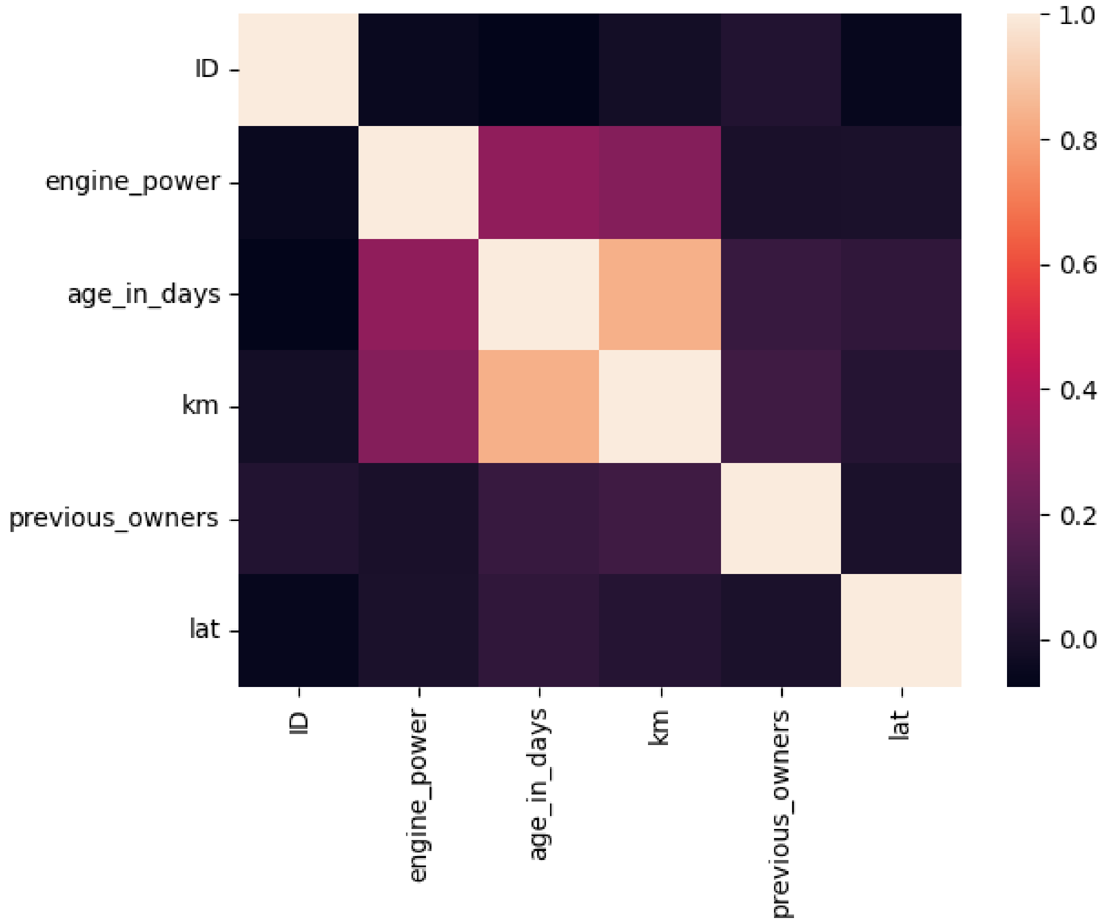
```
<ipython-input-14-ad27032804f7>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df1['km'])
```

[14]: <Axes: xlabel='km', ylabel='Density'>



[15]:
```
sns.heatmap(df1.corr())
```

[15]: <Axes: >



3 TO TRAIN THE MODEL AND MODEL BULDING

[16]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               1500 non-null   float64
 1   engine_power     1500 non-null   float64
 2   age_in_days      1500 non-null   float64
 3   km               1500 non-null   float64
 4   previous_owners  1500 non-null   float64
 5   lat              1500 non-null   float64
dtypes: float64(6)
memory usage: 70.4 KB
```

[17]:
```
x=df1[['ID', 'age_in_days', 'km','previous_owners',
       'lat']]
y=df1['engine_power']
```

[18]:
```
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)
```

[19]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[19]:
```
LinearRegression()
```

[20]:
```
lr.intercept_
```

[20]:
```
50.613595484526456
```

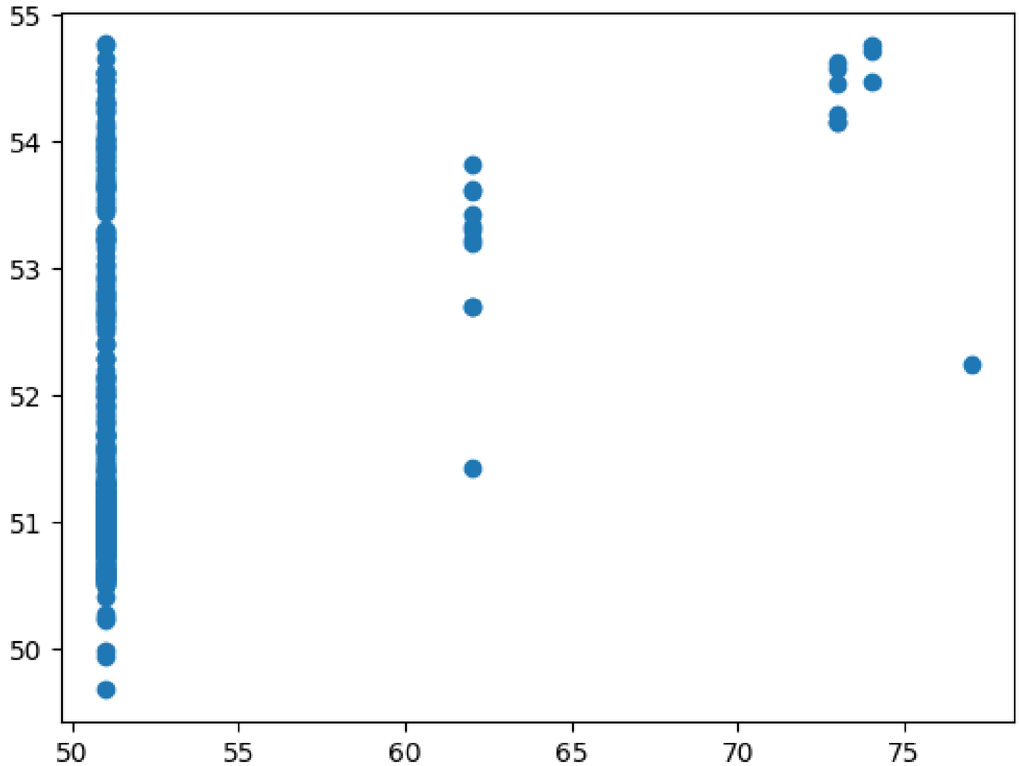
[21]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

[21]:

	Co-efficient
ID	-0.000267
age_in_days	0.000831
km	0.000004
previous_owners	-0.292565
lat	0.004989

[22]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

[22]:
```
<matplotlib.collections.PathCollection at 0x7e6dfaecd8d0>
```



4 ACCURACY

[23]:
```
lr.score(x_test,y_test)
```

[23]: 0.1174629752442442

[24]:
```
lr.score(x_train,y_train)
```

[24]: 0.09327242421014503

[25]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

[25]: Ridge(alpha=10)

[26]:
```
rr.score(x_train,y_train)
```

[26]: 0.09326942619342626

[27]:
```
rr.score(x_test,y_test)
```

[27]: 0.11747171879864649

[28]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

[28]: Lasso(alpha=10)

[29]:
```
la.score(x_train,y_train)
```

[29]: 0.09245387333030353

[30]:
```
la.score(x_test,y_test)
```

[30]: 0.11694127378125141