DAY 10:

Vehicle Dataset

In [1]:

```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\ve.csv")[0:500]
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.6115
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495
...	...	...	...	...	...	...	...	
495	496.0	lounge	51.0	397.0	15003.0	1.0	45.512569	10.329
496	497.0	pop	51.0	790.0	38718.0	1.0	43.782372	11.254
497	498.0	lounge	51.0	397.0	17488.0	1.0	40.967571	14.207
498	499.0	lounge	51.0	425.0	24281.0	1.0	45.438110	12.318
499	500.0	lounge	51.0	701.0	25076.0	1.0	45.512569	10.329

500 rows × 11 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               500 non-null    float64
 1   model            500 non-null    object 
 2   engine_power     500 non-null    float64
 3   age_in_days      500 non-null    float64
 4   km               500 non-null    float64
 5   previous_owners  500 non-null    float64
 6   lat              500 non-null    float64
 7   lon              500 non-null    object 
 8   price            500 non-null    object 
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      0 non-null      object 
dtypes: float64(7), object(4)
memory usage: 43.1+ KB
```

In [4]:

```
df.columns
```

Out[4]:

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

Linear Regression

In [9]:

```
x=df[['ID', 'engine_power', 'age_in_days']]
y=df[ 'km']
```

In [10]:

```
# to split my dataset into test and train data
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 [11]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[11]:

```
LinearRegression()
```

In [12]:

```
print(lr.score(x_test,y_test))
```

```
0.7009371105896656
```

In [13]:

```
lr.score(x_train,y_train)
```

Out[13]:

```
0.7416159320000355
```

Ridge Regression

In [14]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [15]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[15]:

```
0.7009385740603824
```

Lasso Regression

In [16]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[16]:

```
Lasso(alpha=10)
```

In [17]:

```
la.score(x_test,y_test)
```

Out[17]:

```
0.7009402223028935
```

Elastic regression

In [18]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[18]:

```
ElasticNet()
```

In [19]:

```
print(en.intercept_)
```

```
-1512.7563567000325
```

In [20]:

```
predict=(en.predict(x_test))
```

In [21]:

```
print(en.score(x_test,y_test))
```

```
0.7009619748331201
```

Evalution matrics

In [22]:

```
from sklearn import metrics
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute Error: 16871.50071182913
```

In [23]:

```
print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Square Error: 556364078.5051523
```

In [24]:

```
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root Mean Square Error: 23587.371165629127
```

In [ ]:

```

```