

DATA COLLECTION

```
In [1]: # import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: # To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\\VehicleSelection.csv")
sd
```

Out[2]:


	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.6115598
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241889
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.417
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634609
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495650
...
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	leng
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	conc
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null valu
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	fi
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	sear

1549 rows × 11 columns

```
In [3]: # to display top 10 rows
sd.head(10)
```

```
Out[3]:
```

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



DATA CLEANING AND PRE_PROCESSING

```
In [4]: sd.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
```

```
In [5]: # to display summary of statistics
sd.describe()
```

```
Out[5]:
```

	ID	engine_power	age_in_days	km	previous_owners	lat	lon
count	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	10.834265
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.133518
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.852071
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	10.834265
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	10.834265
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	10.834265
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	10.834265



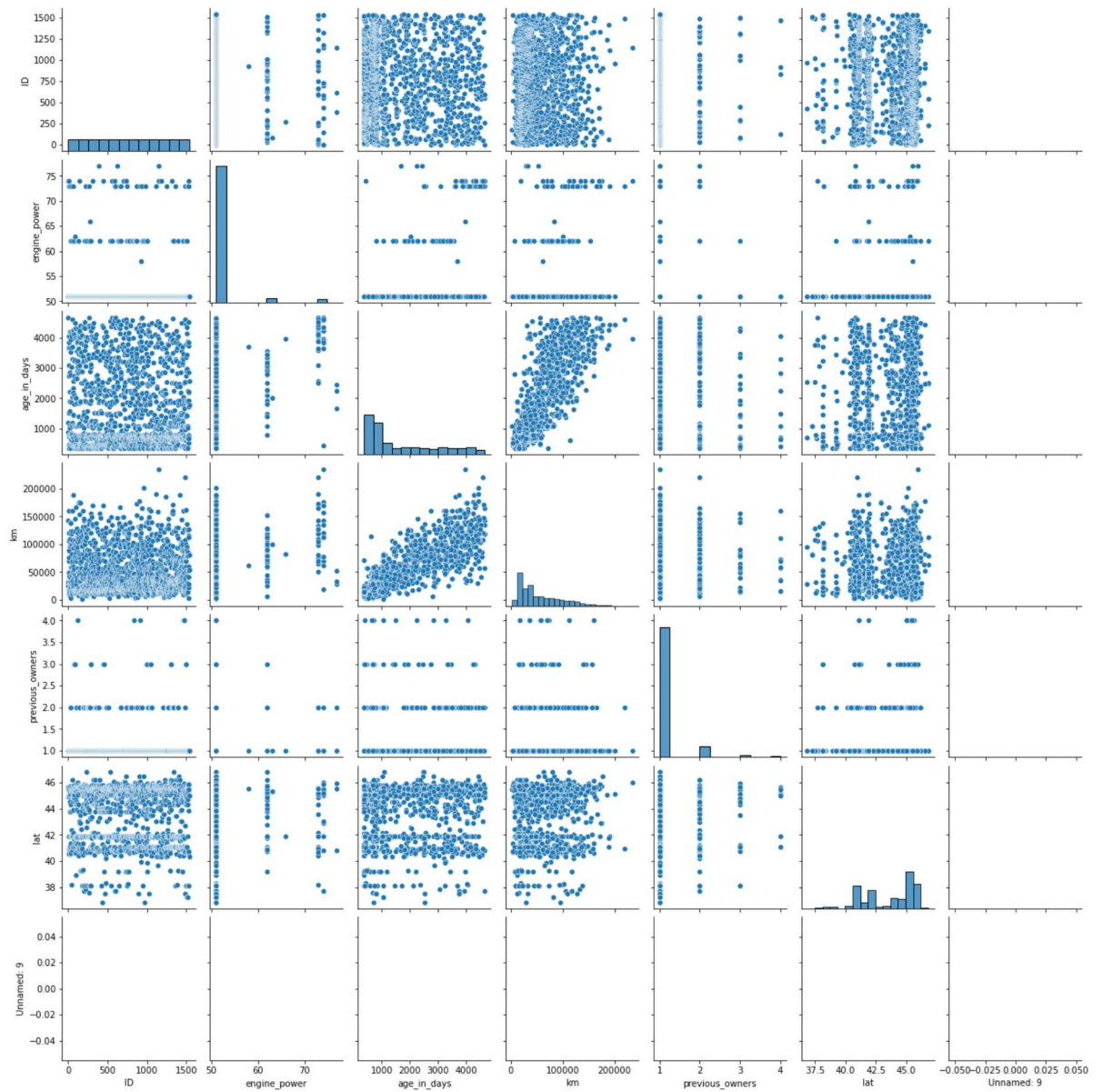
```
In [6]: #to display columns heading
sd.columns
```

```
Out[6]: Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
               'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10'],
              dtype='object')
```

EDA and visualization

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

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x24c385f4c40>
```

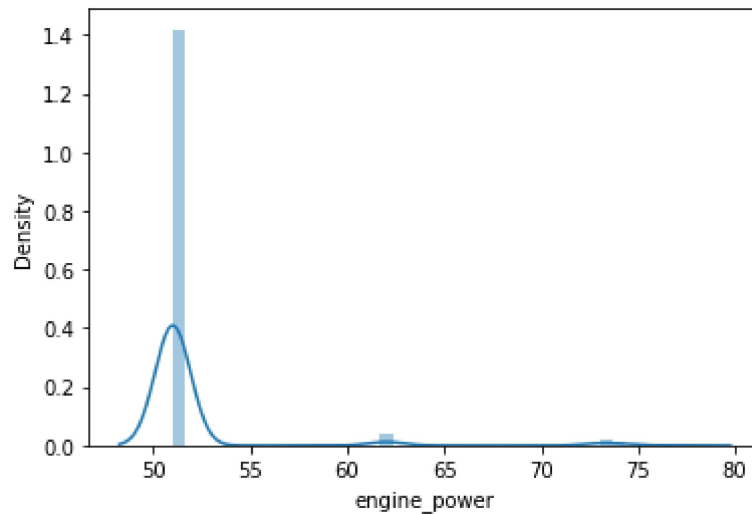


```
In [8]: sns.distplot(sd['engine_power'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `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 flexibility) or `histplot` (an axes-level function for histograms).

```
warnings.warn(msg, FutureWarning)
```

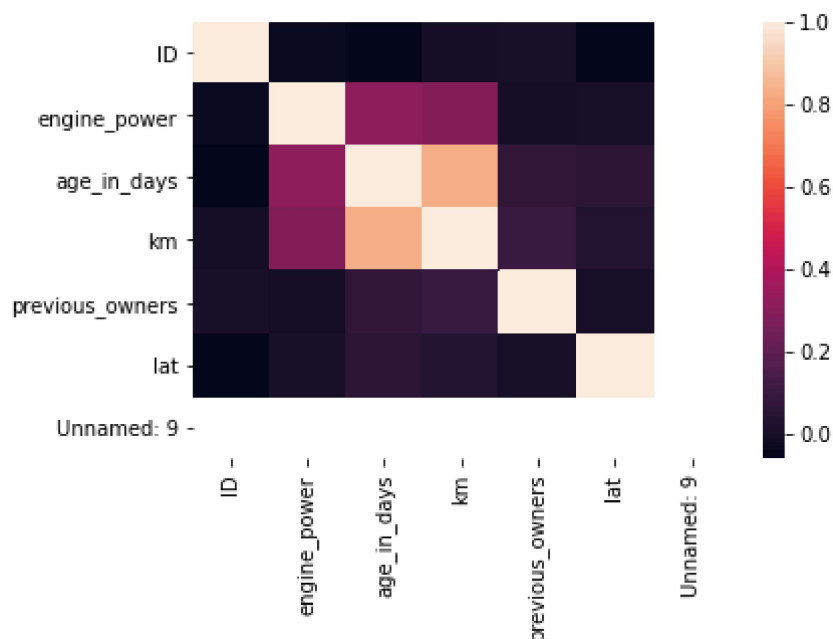
```
Out[8]: <AxesSubplot:xlabel='engine_power', ylabel='Density'>
```



```
In [9]: sd1=sd[['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',  
              'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10']]
```

```
In [10]: sns.heatmap(sd1.corr())
```

```
Out[10]: <AxesSubplot:>
```



TO TRAIN THE MODEL _MODEL BUILDING

we are going to train Linear Regression model; we need to split out the data into two variables x and y where x is independent on x (output) and y is dependent on x(output) address column as it is not required our model

```
In [11]: dss=sd.head(200)
dss
```

```
Out[11]:
```

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.6115598
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241889
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.417
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634609
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495650
...
195	196.0	lounge	51.0	517.0	9150.0	1.0	44.411758	12.20405
196	197.0	pop	51.0	1552.0	52026.0	1.0	45.069679	7.7049198
197	198.0	lounge	51.0	2282.0	145150.0	2.0	45.386841	11.790889
198	199.0	lounge	51.0	397.0	19783.0	2.0	38.122070	13.361120
199	200.0	lounge	51.0	3743.0	105610.0	2.0	37.727879	12.887470

200 rows × 11 columns



```
In [12]: x= dss[['age_in_days', 'km', 'previous_owners',
               'lat']]
y=dss[ 'engine_power']
```

```
In [13]: # To split my dataset into training data and test data
from sklearn .model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4)
```

```
In [14]: from sklearn.linear_model import LinearRegression

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

```
Out[14]: LinearRegression()
```

```
In [15]: print(lr.intercept_)
```

57.287116485755654

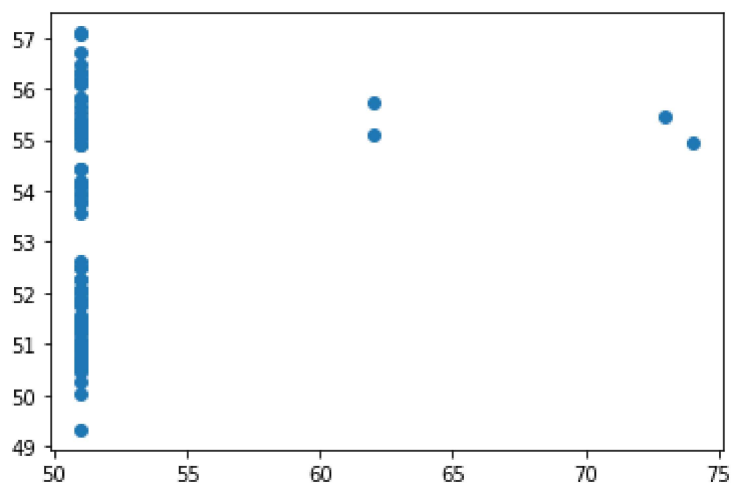
```
In [16]: coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

```
Out[16]:
```

	Co-efficient
age_in_days	0.001218
km	0.000010
previous_owners	-1.237513
lat	-0.137091

```
In [17]: prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

```
Out[17]: <matplotlib.collections.PathCollection at 0x24c3c455c40>
```



```
In [18]: print(lr.score(x_test,y_test))
-0.19302825676216084
```

```
In [19]: lr.score(x_train,y_train)
```

```
Out[19]: 0.16182642739380004
```

```
In [20]: from sklearn.linear_model import Ridge,Lasso
```

```
In [21]: dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

```
Out[21]: Ridge(alpha=10)
```

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

```
Out[22]: -0.2064739917770242
```

In [23]: `dr.score(x_train,y_train)`

Out[23]: 0.1607460504381495

In [24]: `la=Lasso(alpha=10)
la.fit(x_train,y_train)`

Out[24]: Lasso(alpha=10)

In [25]: `la.score(x_test,y_test)`

Out[25]: -0.22672079753185526

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

Out[26]: 0.1508021612247673

In []: