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

In [2]:
```
cars=pd.read_csv("C:/Users/USER/Desktop/Datasets/Cars.csv")
cars
```

Out[2]:

	HP	MPG	VOL	SP	WT
0	49	53.700681	89	104.185353	28.762059
1	55	50.013401	92	105.461264	30.466833
2	55	50.013401	92	105.461264	30.193597
3	70	45.696322	92	113.461264	30.632114
4	53	50.504232	92	104.461264	29.889149
...	...	...	...	...	...
76	322	36.900000	50	169.598513	16.132947
77	238	19.197888	115	150.576579	37.923113
78	263	34.000000	50	151.598513	15.769625
79	295	19.833733	119	167.944460	39.423099
80	236	12.101263	107	139.840817	34.948615

81 rows × 5 columns

In [3]:
```
cols =['HP','VOL','SP','WT']
x=cars[cols]
y=cars.MPG
```

In [4]:
```
x
```

Out[4]:

	HP	VOL	SP	WT
0	49	89	104.185353	28.762059
1	55	92	105.461264	30.466833
2	55	92	105.461264	30.193597
3	70	92	113.461264	30.632114
4	53	92	104.461264	29.889149
...	...	...	...	...
76	322	50	169.598513	16.132947
77	238	115	150.576579	37.923113

	HP	VOL	SP	WT
78	263	50	151.598513	15.769625
79	295	119	167.944460	39.423099
80	236	107	139.840817	34.948615

81 rows × 4 columns

In [5]:
```
y
```

Out[5]:
```
0     53.700681
1     50.013401
2     50.013401
3     45.696322
4     50.504232
        ...
76    36.900000
77    19.197888
78    34.000000
79    19.833733
80    12.101263
Name: MPG, Length: 81, dtype: float64
```

In [6]:
```
x.describe()
```

Out[6]:

	HP	VOL	SP	WT
count	81.000000	81.000000	81.000000	81.000000
mean	117.469136	98.765432	121.540272	32.412577
std	57.113502	22.301497	14.181432	7.492813
min	49.000000	50.000000	99.564907	15.712859
25%	84.000000	89.000000	113.829145	29.591768
50%	100.000000	101.000000	118.208698	32.734518
75%	140.000000	113.000000	126.404312	37.392524
max	322.000000	160.000000	169.598513	52.997752

In [7]:
```
x.median()
```

Out[7]:
```
HP     100.000000
VOL    101.000000
SP     118.208698
WT      32.734518
dtype: float64
```

In [8]:
```
x.mode()
```

Out[8]:

	HP	VOL	SP	WT

	HP	VOL	SP	WT
0	92.0	50.0	118.288996	15.712859
1	NaN	NaN	NaN	15.753535
2	NaN	NaN	NaN	15.769625
3	NaN	NaN	NaN	15.823060
4	NaN	NaN	NaN	15.847758
...	...	...	...	...
76	NaN	NaN	NaN	42.778219
77	NaN	NaN	NaN	43.353123
78	NaN	NaN	NaN	43.390988
79	NaN	NaN	NaN	44.013139
80	NaN	NaN	NaN	52.997752

81 rows × 4 columns

In [9]:
```
x.var()
```

Out[9]:
```
HP     3261.952160
VOL     497.356790
SP      201.113002
WT       56.142247
dtype: float64
```

In [10]:
```
x.std()
```

Out[10]:
```
HP     57.113502
VOL    22.301497
SP     14.181432
WT      7.492813
dtype: float64
```

In [11]:
```
x.skew()
```

Out[11]:
```
HP     1.716216
VOL   -0.590197
SP     1.611450
WT    -0.614753
dtype: float64
```

In [12]:
```
x.kurtosis()
```

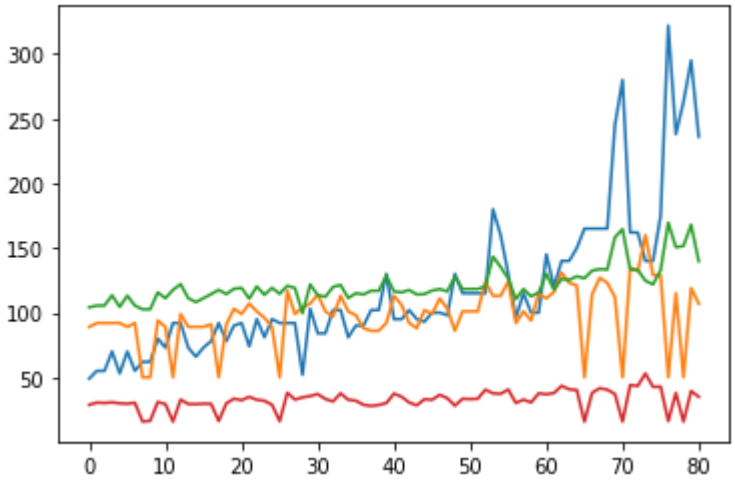
Out[12]:
```
HP     2.960025
VOL    0.920229
SP     2.977329
WT     0.950291
dtype: float64
```

In [13]:
```
x.isnull().sum()
```

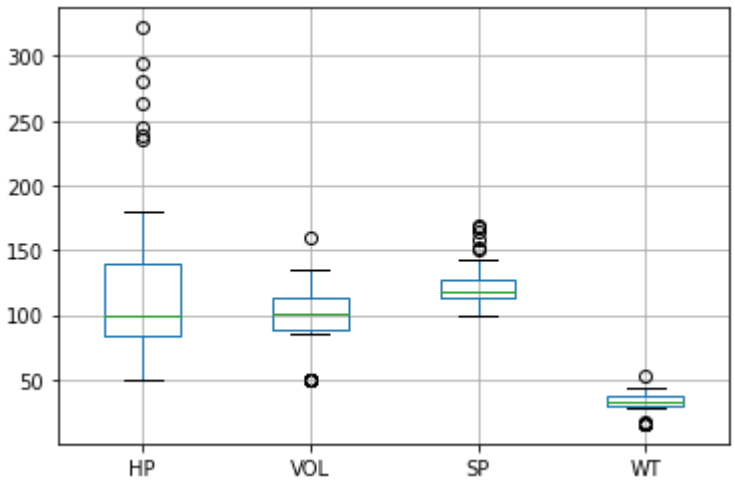
Out[13]:
```
HP     0
VOL    0
SP     0
WT     0
dtype: int64
```

In [14]:
```
plt.plot(x)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x206d8e07190>,
 <matplotlib.lines.Line2D at 0x206d8e071c0>,
 <matplotlib.lines.Line2D at 0x206d8e072e0>,
 <matplotlib.lines.Line2D at 0x206d8e07400>]
```

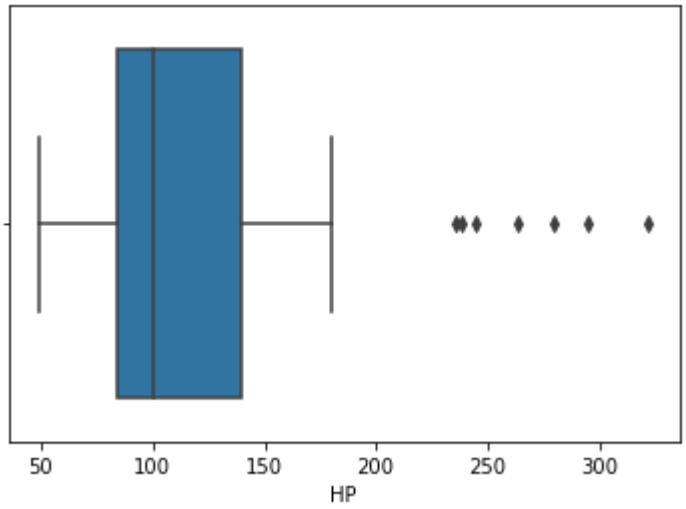


In [15]:
```
boxplot=x.boxplot()
```



In [16]:
```
sns.boxplot(x["HP"])
```

Out[16]:
```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
<AxesSubplot:xlabel='HP'>
```



In [17]:
```
from feature_engine.outliers import Winsorizer
win=Winsorizer(capping_method="iqr", tail="both", fold=1.5, variables=["HP"])
New_HP=win.fit_transform(x[["HP"]])
```

In [18]:
```
x.insert(loc=1,column='New_HP', value=New_HP)
```

In [19]:
```
del x["HP"]
x
```

Out[19]:

	New_HP	VOL	SP	WT
0	49.0	89	104.185353	28.762059
1	55.0	92	105.461264	30.466833
2	55.0	92	105.461264	30.193597
3	70.0	92	113.461264	30.632114
4	53.0	92	104.461264	29.889149
...	...	...	...	...
76	224.0	50	169.598513	16.132947
77	224.0	115	150.576579	37.923113
78	224.0	50	151.598513	15.769625
79	224.0	119	167.944460	39.423099
80	224.0	107	139.840817	34.948615

81 rows × 4 columns

boxplot=x.boxplot()

In [20]:
```
x.corr()
```

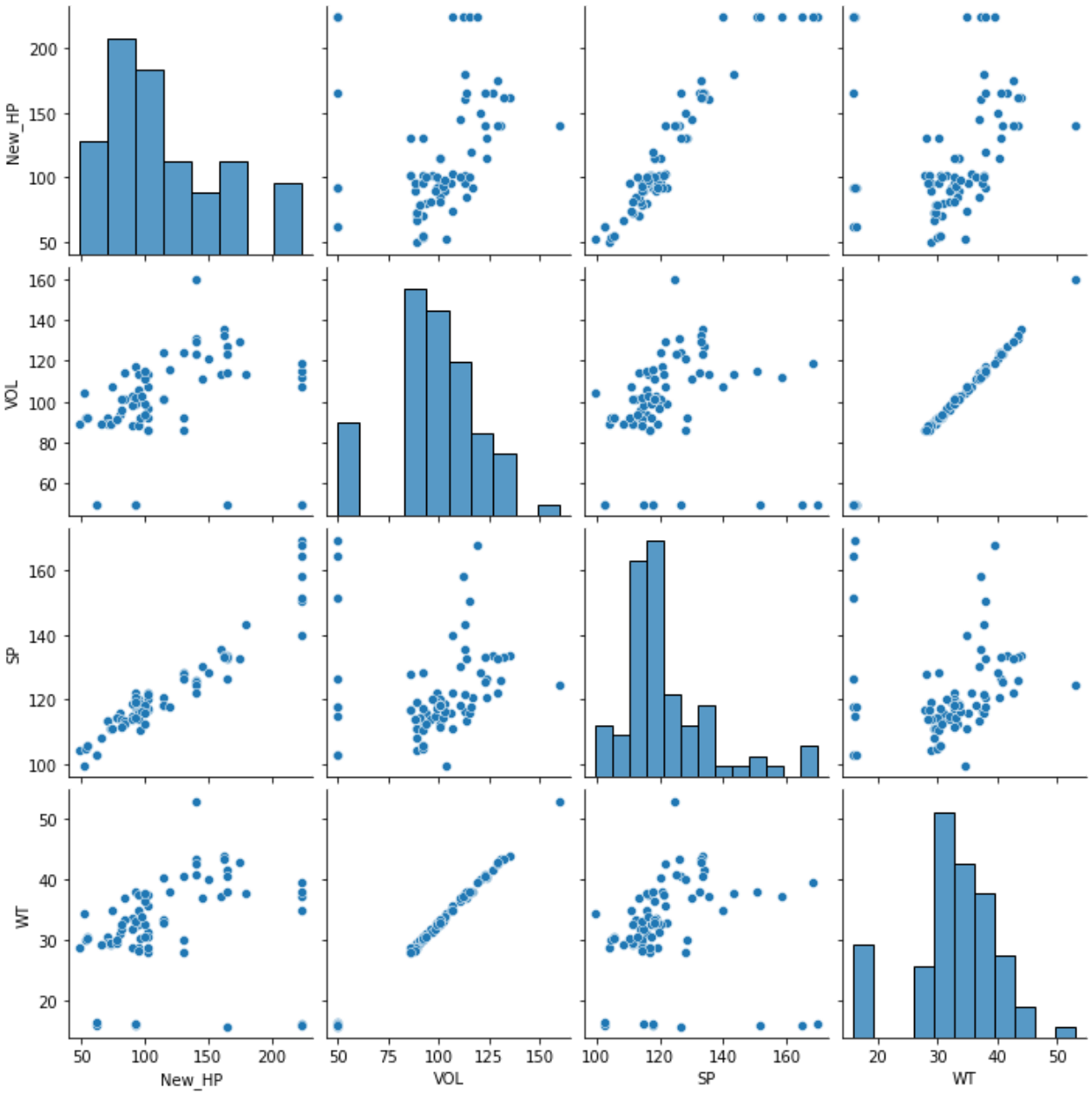
Out[20]:

	New_HP	VOL	SP	WT

	New_HP	VOL	SP	WT
New_HP	1.000000	0.183756	0.948771	0.182321
VOL	0.183756	1.000000	0.102170	0.999203
SP	0.948771	0.102170	1.000000	0.102439
WT	0.182321	0.999203	0.102439	1.000000

In [21]:
```
sns.pairplot(x)
```

Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x206d902b2b0>
```



In [22]:
```
names=x.columns
names
```

Out[22]:
```
Index(['New_HP', 'VOL', 'SP', 'WT'], dtype='object')
```

In [25]:
```
from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
S_x=scale.fit(x)
S_x=scale.transform(x)

S_x=pd.DataFrame(S_x, columns=names)
S_x
```

Out[25]:

	New_HP	VOL	SP	WT
0	0.000000	0.354545	0.065975	0.349986
1	0.034286	0.381818	0.084193	0.395709
2	0.034286	0.381818	0.084193	0.388381
3	0.120000	0.381818	0.198424	0.400142
4	0.022857	0.381818	0.069914	0.380215
...	...	...	...	...
76	1.000000	0.000000	1.000000	0.011267
77	1.000000	0.590909	0.728388	0.595690
78	1.000000	0.000000	0.742981	0.001523
79	1.000000	0.627273	0.976382	0.635921
80	1.000000	0.518182	0.575094	0.515913

81 rows × 4 columns

In [26]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(S_x,y, test_size=0.2)
```

In [27]:
```
x_train
```

Out[27]:

	New_HP	VOL	SP	WT
28	0.017143	0.490909	0.000000	0.503430
67	0.662857	0.700000	0.487128	0.693608
71	0.645714	0.772727	0.483355	0.759028
54	0.634286	0.572727	0.511579	0.577755
4	0.022857	0.381818	0.069914	0.380215
...	...	...	...	...
78	1.000000	0.000000	0.742981	0.001523
55	0.462857	0.672727	0.383236	0.667193
47	0.280000	0.481818	0.241427	0.488058

	New_HP	VOL	SP	WT
25	0.245714	0.000000	0.214663	0.008859
22	0.262857	0.463636	0.295916	0.454956

64 rows × 4 columns

In [28]:
```
from sklearn.linear_model import LinearRegression
rgsr=LinearRegression()
rgsr.fit(x_train, y_train)
y_pred=rgsr.predict(x_test)
y_pred
```

Out[28]:
```
array([22.87126169, 39.44267825, 18.15228001, 14.20081617, 38.48251104,
       38.9758946 , 36.45555826, 43.91231198, 21.74910438, 27.66116864,
       17.77210199, 41.7611509 , 45.63690802, 37.84592873, 39.11881947,
       35.79710839, 32.99787181])
```

In [29]:
```
from sklearn.metrics import mean_squared_error
```

In [30]:
```
RMSE=mean_squared_error(y_test, y_pred)
```

In [31]:
```
RMSE
```

Out[31]:
```
8.91223610053179
```