Apply linear regression on “Mileage prediction” after implementing PCA

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

In [2]:
```
df = pd.read_csv('Cars_mileage.csv')
```

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

Out[3]:

	HP	MPG	VOL	SP	WT
count	81.000000	81.000000	81.000000	81.000000	81.000000
mean	117.469136	34.422076	98.765432	121.540272	32.412577
std	57.113502	9.131445	22.301497	14.181432	7.492813
min	49.000000	12.101263	50.000000	99.564907	15.712859
25%	84.000000	27.856252	89.000000	113.829145	29.591768
50%	100.000000	35.152727	101.000000	118.208698	32.734518
75%	140.000000	39.531633	113.000000	126.404312	37.392524
max	322.000000	53.700681	160.000000	169.598513	52.997752

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

Out[4]:

	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

In [6]:
```
df.tail()
```

Out[6]:

	HP	MPG	VOL	SP	WT
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

In [7]:
```
df.shape
```

Out[7]:
```
(81, 5)
```

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

Out[8]:
```
Index(['HP', 'MPG', 'VOL', 'SP', 'WT'], dtype='object')
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   HP      81 non-null     int64  
 1   MPG     81 non-null     float64
 2   VOL     81 non-null     int64  
 3   SP      81 non-null     float64
 4   WT      81 non-null     float64
dtypes: float64(3), int64(2)
memory usage: 3.3 KB
```

In [11]:
```
#missing_data =
df.isnull()
```

Out[11]:

	HP	MPG	VOL	SP	WT
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
76	False	False	False	False	False
77	False	False	False	False	False
78	False	False	False	False	False
79	False	False	False	False	False
80	False	False	False	False	False

81 rows × 5 columns

In [13]:
```
from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test= train_test_split(A,B, test_size = 0.3)
```

In [14]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
A_train = sc.fit_transform(A_train)
B_test = sc.transform(A_test)
```

In [15]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
A_train = pca.fit_transform(A_train)
A_test = pca.transform(A_test)
```

In [16]:
```
#explained_variance = pca.explained_variance_ratio_
print(A_train,A_test)
```

```
[[-5.05664316e-01 -6.44295420e-01]
 [-1.09690877e+00 -5.95856640e-01]
 [ 2.44851209e+00 -3.00805324e-01]
 [-2.37528992e+00 -1.01144034e+00]
 [-3.59124001e+00  8.32074840e-01]
 [ 3.24581706e+00 -2.29265880e+00]
 [-1.66287118e+00 -5.79936969e-01]
 [-2.47865384e-01 -7.88031849e-01]
 [-7.10174162e-01  1.78884326e-01]
 [ 2.65417189e+00 -6.72439400e-01]
 [-9.02820109e-01  9.15534818e-02]
 [ 1.84170772e-01 -8.47653664e-01]
 [ 1.80349802e-01 -2.16052415e-01]
 [ 2.03487506e-01 -2.47317630e-01]
 [ 2.56334209e+00 -5.73817581e-01]
 [ 2.01949558e+00  1.74418668e-01]
 [ 3.90063948e-03 -1.07197012e+00]
 [ 6.94637786e-02 -5.53981520e-01]
 [-2.29764491e+00  1.67109374e+00]
 [ 1.01312990e+00 -8.20168756e-01]
 [ 1.47493047e+00 -7.88997711e-01]
 [-1.75080119e+00 -4.93468682e-01]
 [-7.77999649e-01 -5.01545225e-01]
 [-2.35361836e+00  1.58045824e+00]
 [ 7.58951977e-02 -8.54826786e-01]
 [-4.78980694e-01 -3.84712712e-01]
 [-3.56654399e+00  7.98703934e-01]
 [-2.00279427e+00 -6.55929041e-01]
 [ 3.21035928e-01  6.44220992e-01]
 [-1.44362700e+00 -3.60633135e-01]
 [ 2.07601458e+00 -1.01516206e+00]
 [-7.46397699e-01  2.27831995e-01]
 [-2.31890775e+00  1.69982552e+00]
 [-5.00673520e-01 -5.59799671e-02]
 [-6.38318936e-01 -3.17264998e-01]
 [ 2.87419646e+00 -3.30146965e-01]
 [ 2.62150124e-02 -1.09679708e+00]
 [ 1.91565553e+00 -4.19178125e-01]
 [-3.47104574e-01 -3.79695184e-01]
 [ 1.78997585e+00 -6.92950042e-01]
 [ 1.38739330e+00  1.49737914e-02]
 [-6.04574461e-01 -3.89113124e-01]
 [-5.79941464e-01 -4.22398877e-01]
 [ 1.04396964e-01  8.88731064e-01]
 [-1.38703450e+00  2.70412533e+00]
 [-1.12695902e+00 -1.10386297e-01]
 [-2.36764774e+00 -1.02176696e+00]
 [ 4.63503287e-01 -1.07498461e+00]
 [ 1.79102447e+00  2.75813259e-01]
 [-5.53521488e-01 -5.98415389e-01]
 [ 7.78805201e-01  4.54821464e+00]
 [ 2.31669617e+00 -1.54838159e-01]
 [-1.77205581e+00 -4.56512877e-01]
 [-4.06176888e-01 -2.69940373e-01]
 [ 4.82190967e+00  2.47368690e+00]
 [ 2.31066855e+00  5.25746008e+00]] [[132.67842595  21.59705198]
 [128.49989619   5.8745015 ]
```

```
 [131.92353854  29.52688789]
 [142.91149031  31.97539864]
 [218.67004825  94.99798286]
 [121.18933443  14.40891189]
 [ 87.83798023   2.57919077]
 [119.13897056  12.84667731]
 [106.66544732  17.04675613]
 [128.22492288  23.79483018]
 [117.39606235   4.77236897]
 [115.65764594  18.88893433]
 [ 94.80409049   3.75505242]
 [192.33460313  65.02275233]
 [ 93.08477766   2.51652361]
 [229.04485777 102.99284254]
 [155.93545193  16.75459055]
 [176.16512157  25.76287056]
 [106.6593963   17.05493267]
 [112.32842865  19.51563694]
 [121.93643291  30.2286324 ]
 [142.92021329  31.96361156]
 [226.71521364 187.95240807]
 [225.07465764  94.37858592]
 [100.82295536  -8.6898909 ]]
```

In [17]:

```python
from sklearn.linear_model import LinearRegression
cls = LinearRegression()
cls.fit( A_train,B_train)
cls = LinearRegression().fit(A_train,B_train)
```

In [18]:

```python
R_square = cls.score(A_train,B_train)
print('Coefficient of determination :',R_square)
print('intersept:', cls.intercept_)
print('slope:', cls.coef_ )
```

```
Coefficient of determination : 0.9572617781216625
intersept: [117.60714286  33.69825299  98.17857143 121.74379404  32.22296451]
slope: [[ 22.13601717  23.28923679]
 [ -4.36406452  -0.43938038]
 [  9.97176026 -13.30310102]
 [  5.58904252   5.48870748]
 [  3.32972518  -4.49934753]]
```

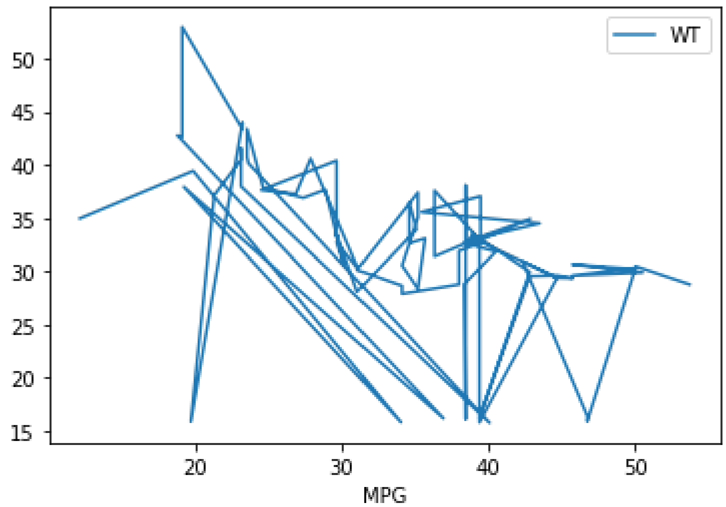
In [19]:
```
new_cls = LinearRegression().fit(A_train, B_train)
print('intercept:', new_cls.intercept_)
print('slope:', new_cls.coef_ )
y_pred = cls.predict(A_train)
print('predicted response:', y_pred, sep= '\n')
```

```
intercept: [117.60714286  33.69825299  98.17857143 121.74379404  32.22296451]
slope: [[ 22.13601717  23.28923679]
 [ -4.36406452  -0.43938038]
 [  9.97176026 -13.30310102]
 [  5.58904252   5.48870748]
 [  3.32972518  -4.49934753]]
predicted response:
[[ 91.40860028  36.18809545 101.70733516 115.38126559  33.43815032]
 [ 79.44890502  38.74704137  95.16720119 112.34264147  31.25152585]
 [164.80192219  23.14495618 126.59619063 133.77759981  41.72926457]
 [ 41.47201087  44.50857851  87.94804273 102.91669754  28.86472343]
 [ 57.48978034  49.00505874  51.29841138 106.23921635  16.52132836]
 [136.06233138  20.5406472  161.04455257 127.3010701   53.34611199]
 [ 67.29150852  41.20994302  89.31177879 109.26683195  29.29539846]
 [ 93.76773015  35.12619925 106.19018453 116.03318758  34.94327005]
 [106.05279484  36.71890059  88.71716868 118.7564442   29.05341697]
 [160.69933709  22.41073226 133.59086652 132.88725045  44.08616605]
 [ 99.75451213  37.59799139  87.95792052 117.20040435  28.80489073]
 [101.94274333  33.26696224 111.29150053 118.12060931  36.65009099]
 [116.5676733   33.00612401 102.85114352 121.56592825  33.79557469]
 [116.35170693  32.91888689 103.49779147 121.52364024  34.01328995]
 [160.98555383  22.7637869  131.37315746 132.92090512  43.33999392]
 [166.37280945  24.80840781 115.99618808 133.98817378  38.1625596 ]
 [ 92.7281216   34.15223298 112.47799442 115.88186449  37.05911867]
 [106.24298746  33.63851719 106.24091969 119.09138755  34.94681519]
 [105.66493354  42.99107781  53.03627844 118.07430368  17.05360693]
 [120.93269945  29.63725477 119.19204776 122.90455377  39.28663293]
 [131.88107458  27.60823136 123.38234074 125.65666552  40.68405254]
 [ 67.35886875  41.55568279  87.28466547 109.24998653  28.61356482]
 [ 88.70472376  37.31386279  97.09265224 114.6426859   31.88906576]
 [102.31507248  43.27517305  53.68385775 117.26399389  17.27503131]
 [ 99.37889682  33.74263556 110.30722723 117.47608136  36.32183745]
 [ 98.04475256  35.95759086  98.52016285 116.95517505  32.35904662]
 [ 57.25926878  48.91194627  51.98861064 106.19408027  16.7537066 ]
 [ 57.99716785  42.72677874  86.93307746 106.94988911  28.50546273]
 [139.7170149   32.01417342  92.8097278  127.07401807  30.39335179]
 [ 77.25212028  40.15678949  88.58060809 111.69589157  29.03869714]
 [139.91948763  25.0844337  132.38489457 127.7748002   43.70308944]
 [106.39090384  36.8554758   87.70480047 118.82264874  28.71256998]
 [105.86340005  43.07124605  52.44202869 118.11316506  16.85353324]
 [105.22049452  35.90782103  93.93068228 118.63825079  30.80773261]
 [ 96.08844427  36.62331802  96.03401635 116.43482761  31.52502336]
 [173.54153412  21.30013429 131.23134787 135.99572012  43.27869475]
 [ 92.64387189  34.0657601  113.03078361 115.87031252  37.24512454]
 [150.24978806  25.52238727 122.8573981  130.14972817  40.48759904]
 [101.080819    35.37987036  99.76845123 117.71977603  32.77558226]
 [141.09180133  26.19115155 125.24616586 127.94464508  41.30091522]
 [148.66723301  27.6369999  111.81412697 129.58018096  36.77523063]
 [ 95.16212454  36.50762361  97.32631104 116.22907357  31.96065288]
 [ 94.93220117  36.41474874  98.01474912 116.18405267  32.19243816]
 [140.61594403  32.85216691  87.39671381 127.20525795  28.57186779]
 [149.88073833  38.56322143  48.37414353 128.83375214  15.4377212 ]
 [ 90.08994594  38.66487646  88.40928627 114.83929405  28.96716698]
 [ 41.40067908  44.47976486  88.16162485 102.90273019  28.93663286]
 [102.83168855  32.14782188 117.10114388 118.43405758  38.6030324 ]
 [163.67677159  25.7609197  112.36906644 133.26776426  36.94560408]
 [ 91.41774399  36.37678846 100.61976822 115.36561189  33.07236888]
```

```
 [240.77123584  28.30110056  45.43927141 151.06038912  14.3521735 ]
 [165.28350655  23.65607426 123.33993793 133.84204609  40.63359677]
 [ 67.74904848  41.63220169  86.58109262 109.33403313  28.37651574]
 [102.32929904  35.58944164  97.71931693 117.99203041  32.08506265]
 [281.95529825  11.56823855 113.35379194 162.27099603  37.14862149]
 [291.19837442  21.30432152  51.27948169 163.5148793   16.26171572]]
```

In [20]:

```
#missing_data.head(5)
df.plot('MPG','WT')
A = df.iloc[: , 0:80].values
B = df.iloc[: , :].values
```



In [ ]: