Date : 21/03/2024

EXPERIMENT NO. 7

AIM: To study Multi Linear regression in data science.

SOFTWARE USED: Jupyter Notebook

THEORY:

Multiple linear regression (MLR) is a statistical method used to analyze the relationship between a dependent variable and two or more independent variables. It extends the concept of simple linear regression, where there's only one independent variable. In MLR, the relationship between the dependent variable and multiple independent variables is represented by a linear equation of the form:

$$y = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n + \epsilon$$

Where:

- y = the predicted value of the dependent variable
- B_0 = the y-intercept (value of y when all other parameters are set to 0)
- B_1X_1 = the regression coefficient (B_1) of the first independent variable (X_1) (a.k.a. the effect that increasing the value of the independent variable has on the predicted y value)
- … = do the same for however many independent variables you are testing
- B_nX_n = the regression coefficient of the last independent variable
- epsilon = model error (a.k.a. how much variation there is in our estimate of y)

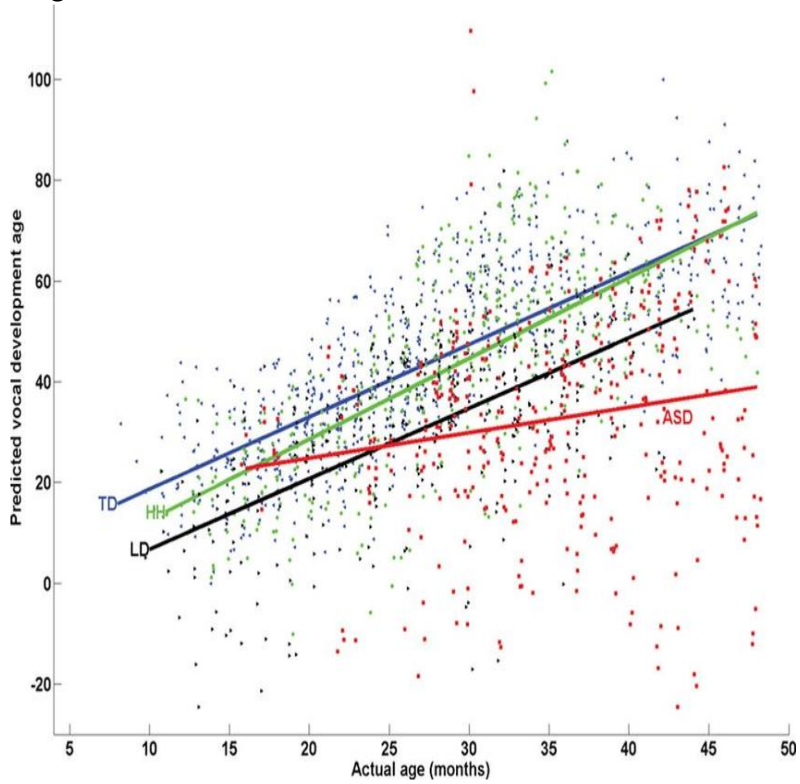
The goal of MLR is to estimate the coefficients that best fit the observed data points. This is typically done using the method of least squares, where the coefficients are chosen to minimize the sum of the squared differences between the observed and predicted values of the dependent variable.

Key concepts in MLR theory include:

- **Assumptions:** MLR relies on several assumptions, including linearity, independence of errors, constant variance of errors (homoscedasticity), and normality of errors.
- **Coefficient Estimation:** Coefficients in MLR are estimated using techniques such as ordinary least squares (OLS), which minimizes the sum of the squared differences between the observed and predicted values.
- **Model Evaluation:** Various metrics are used to evaluate the goodness of fit of the MLR model, including the coefficient of determination rsquared, adjusted rsquared, and significance tests for individual coefficients.
- **Variable Selection:** MLR allows for the inclusion of multiple independent variables, but careful selection is necessary to avoid multicollinearity (high correlation between independent variables) and overfitting (when the model fits the noise in the data).
- **Model Assumptions Checking:** Residual analysis is performed to assess whether the model assumptions are met. This involves examining the residuals (differences between

observed and predicted values) for patterns or trends that indicate violations of the model assumptions.

- **Interpretation of Coefficients:** Coefficients in MLR represent the change in the dependent variable associated with a one-unit change in the corresponding independent variable, holding all other variables constant.



OUTPUT CODE:

In [73]:
```
#multi linear regression
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.regressionplots import influence_plot
import statsmodels.formula.api as smf
```

In [6]:
```
cars = pd.read_csv('Cars.csv')
```

In [7]:
```
cars.head()
```

Out[7]:

	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 [8]:
```
cars.describe()
```

Out[8]:

	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 [9]:
```
cars.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 [10]:
```
cars.corr()
```

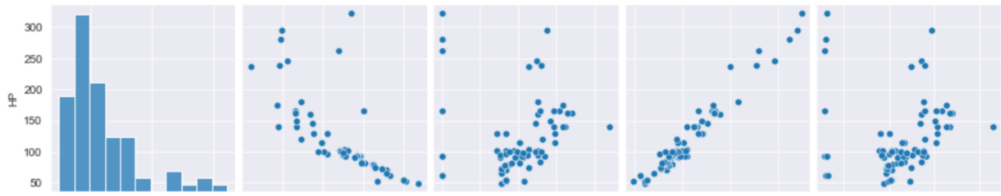
Out[10]:

	HP	MPG	VOL	SP	WT
HP	1.000000	-0.725038	0.077459	0.973848	0.076513
MPG	-0.725038	1.000000	-0.529057	-0.687125	-0.526759
VOL	0.077459	-0.529057	1.000000	0.102170	0.999203
SP	0.973848	-0.687125	0.102170	1.000000	0.102439
WT	0.076513	-0.526759	0.999203	0.102439	1.000000

In [11]:
```
sns.set_style(style='darkgrid')
sns.pairplot(cars)
```

```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\axisgrid.py:123: UserWarning: The figure layout has changed to t
ght
  self._figure.tight_layout(*args, **kwargs)
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x27be7eb1e10>



In [12]:
```python
import statsmodels.formula.api as smf
model = smf.ols('MPG~WT+VOL+SP+HP', data=cars).fit()
```

In [13]:
```python
#coefficients
model.params
```

Out[13]:
```
Intercept    30.677336
WT            0.400574
VOL          -0.336051
SP            0.395627
HP           -0.205444
dtype: float64
```

In [14]:
```
#t values and p values
print(model.tvalues, '\n', model.pvalues)
```

```
Intercept    2.058841
WT           0.236541
VOL         -0.590970
SP           2.499880
HP          -5.238735
dtype: float64
 Intercept    0.042936
WT           0.813649
VOL          0.556294
SP           0.014579
HP           0.000001
dtype: float64
```

In [15]:
```
(model.rsquared, model.rsquared_adj)
```

Out[15]:
```
(0.7705372737359844, 0.7584602881431415)
```

In [16]:
```
# WT and VOL are not significant variables but HP, SP are significant but in reality WT and VOL are also significant
```

In [17]:
```
ml_v = smf.ols('MPG~VOL', data=cars).fit()
#t and p values
print(ml_v.tvalues, '\n', ml_v.pvalues)
```

```
Intercept    14.106056
VOL          -5.541400
dtype: float64
 Intercept    2.753815e-23
VOL          3.822819e-07
dtype: float64
```

In [18]:
```
ml_w = smf.ols('MPG~WT', data=cars).fit()
#t and p values
print(ml_w.tvalues, '\n', ml_w.pvalues)
```

```
Intercept    14.248923
WT           -5.508067
dtype: float64
 Intercept    1.550788e-23
WT           4.383467e-07
dtype: float64
```

In [19]:
```
ml_wv = smf.ols('MPG~VOL+WT', data=cars).fit()
#t and p values
print(ml_wv.tvalues, '\n', ml_wv.pvalues)
```

```
Intercept    12.545736
VOL          -0.709604
WT            0.489876
dtype: float64
 Intercept    2.141975e-20
VOL          4.800657e-01
WT           6.255966e-01
dtype: float64
```

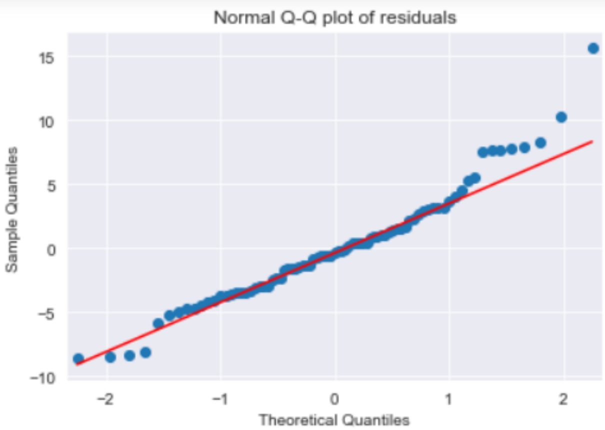
In [26]:
```python
#calculating VIF
rsp_hp = smf.ols('HP~WT+VOL+SP', data = cars).fit().rsquared
vif_hp = 1/(1-rsp_hp) #16.33
rsp_wt = smf.ols('WT~HP+VOL+SP', data = cars).fit().rsquared
vif_wt = 1/(1-rsp_wt) #564.98
rsp_vol = smf.ols('VOL~WT+HP+SP', data = cars).fit().rsquared
vif_vol = 1/(1-rsp_vol) #564.84
rsp_sp = smf.ols('SP~WT+VOL+HP', data = cars).fit().rsquared
vif_sp = 1/(1-rsp_sp) #16.35
d1 = {'Variables' : ['Hp','WT','VOL','SP'],'VIF ' : [vif_hp,vif_wt,vif_vol,vif_sp]}
Vif_frame = pd.DataFrame(d1)
Vif_frame
```

Out[26]:

	Variables	VIF
0	Hp	19.926589
1	WT	639.533818
2	VOL	638.806084
3	SP	20.007639

In [27]:
```python
# vif should be less than 20 to be significant variables
```

In [28]:
```python
import statsmodels.api as sm
qqplot = sm.qqplot(model.resid, line='q')
plt.title('Normal Q-Q plot of residuals')
plt.show()
```



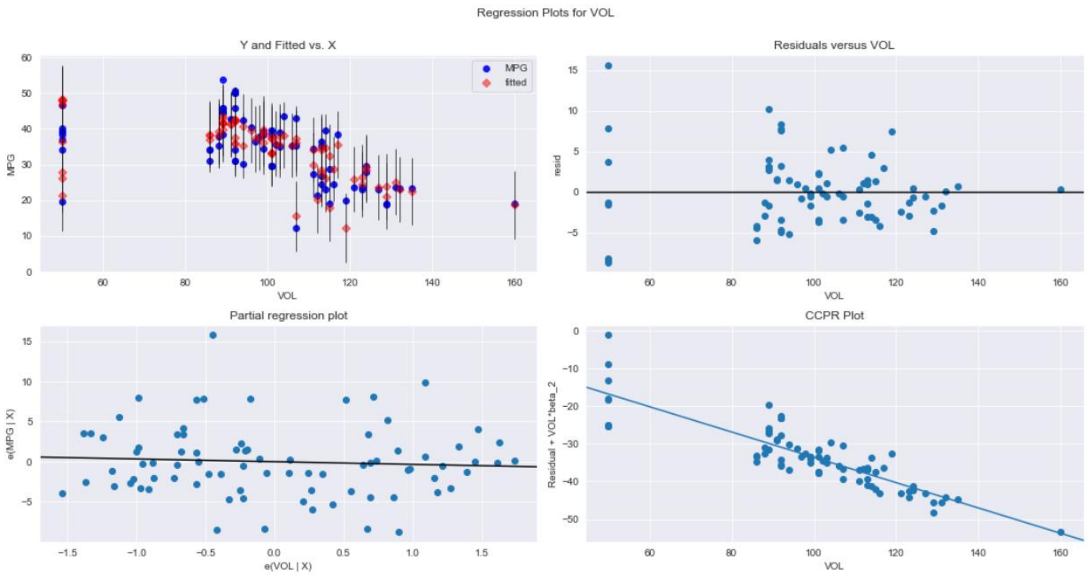
In [29]:
```python
list(np.where(model.resid>10))
```

Out[29]: [array([0, 76], dtype=int64)]

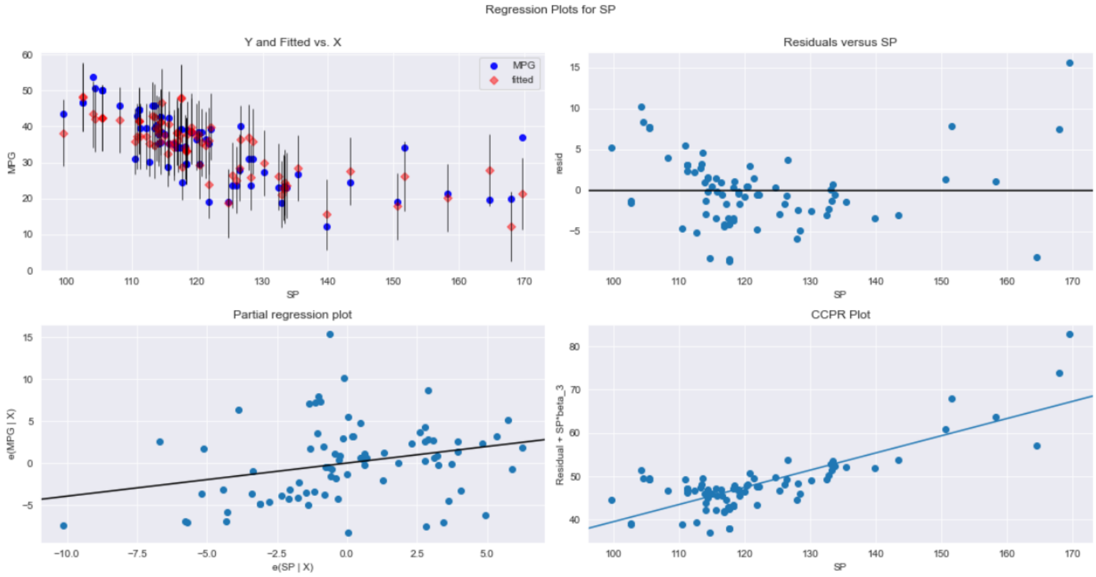
In [30]:
```python
def get_standardized_values(vals):
    return (vals- vals.mean())/vals.std()
```

In [32]:
```python
plt.scatter(get_standardized_values(model.fittedvalues),
            get_standardized_values(model.resid))
plt.title('Residual plot')
plt.xlabel('Standardized Fitted values')
plt.ylabel('Standardized Residual values')
plt.show()
```

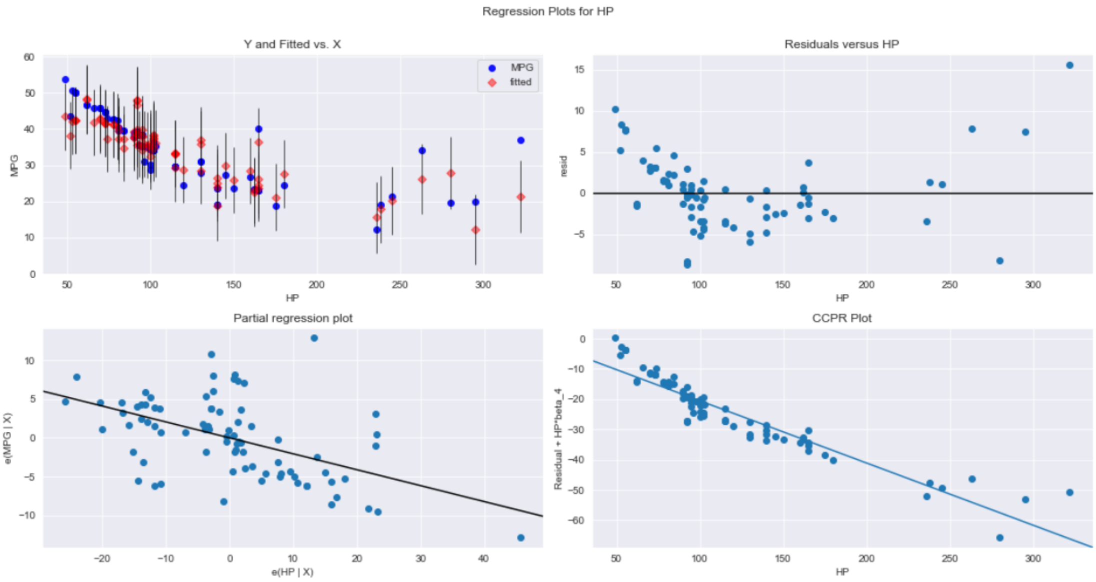
In [34]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "VOL", fig=fig)
plt.show()
```



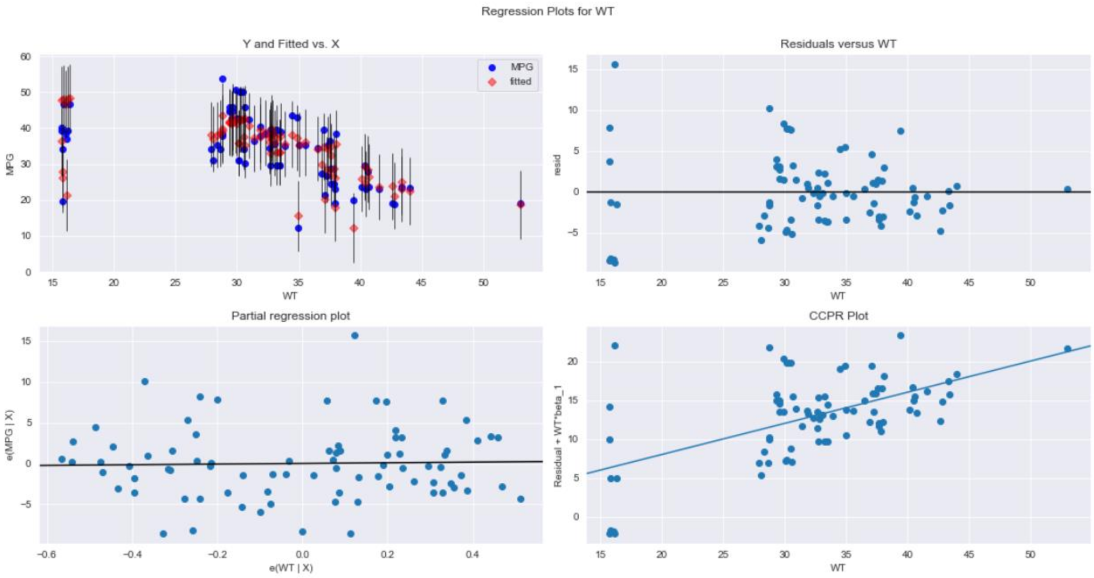
In [35]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "SP", fig=fig)
plt.show()
```



In [36]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "HP", fig=fig)
plt.show()
```

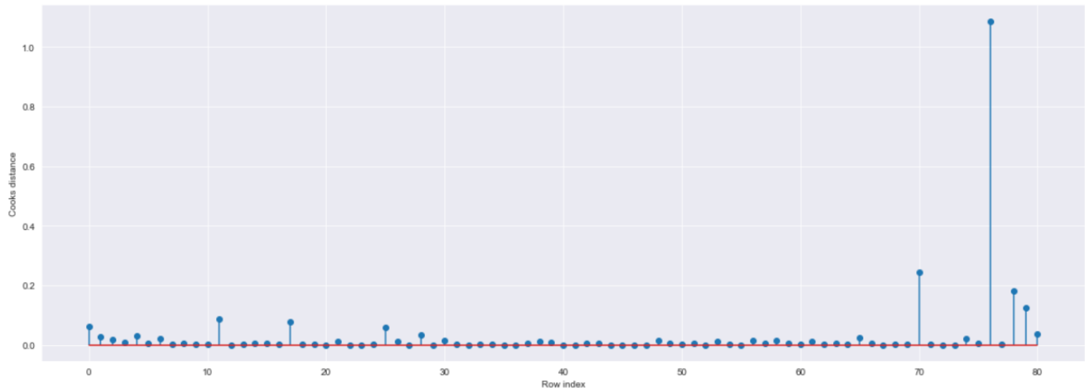


In [37]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "WT", fig=fig)
plt.show()
```



In [38]:
```
model_influence = model.get_influence()
(c,_)= model_influence.cooks_distance
```

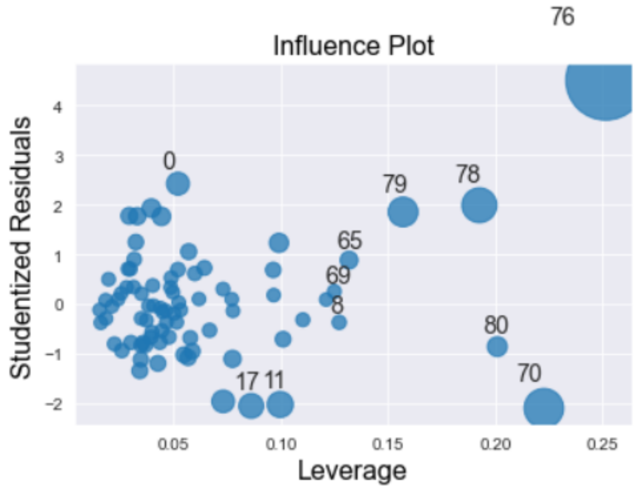
In [39]:
```
fig = plt.subplots(figsize=(20,7))
plt.stem(np.arange(len(cars)), np.round(c,3))
plt.xlabel('Row index')
plt.ylabel('Cooks distance')
plt.show()
```



In [41]:
```
(np.argmax(c),np.max(c))
```

Out[41]:
```
(76, 1.0865193998179907)
```

In [40]:
```
from statsmodels.graphics.regressionplots import influence_plot
influence_plot(model)
plt.show()
```



In [42]:
```
k = cars.shape[1]
n = cars.shape[0]
leverage_cutoff = 3*((k+1)/n)
```

In [43]:
```
cars[cars.index.isin([70,76])]
```

Out[43]:

	HP	MPG	VOL	SP	WT
70	280	19.678507	50	164.598513	15.823060
76	322	36.900000	50	169.598513	16.132947

In [44]:
```
cars.head()
```

Out[44]:

	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 [45]:
```
cars_new = pd.read_csv('Cars.csv')
```

In [46]:
```
car1 = cars_new.drop(cars_new.index[[70,76]],axis=0).reset_index()
```

In [47]:
```
car1=car1.drop(['index'],axis=1)
```

In [48]:
```
car1
```

In [48]:
```
car1
```

Out[48]:

	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
...	...	...	...	...	...
74	175	18.762837	129	132.864163	42.778219
75	238	19.197888	115	150.576579	37.923113
76	263	34.000000	50	151.598513	15.769625
77	295	19.833733	119	167.944460	39.423099
78	236	12.101263	107	139.840817	34.948615

79 rows × 5 columns

In [49]:
```
final_ml_V = smf.ols('MPG~VOL+SP+HP', data=car1).fit()
```

In [50]:
```
(final_ml_V.rsquared, final_ml_V.aic)
```

Out[50]: (0.8161692010376008, 446.1172263944772)

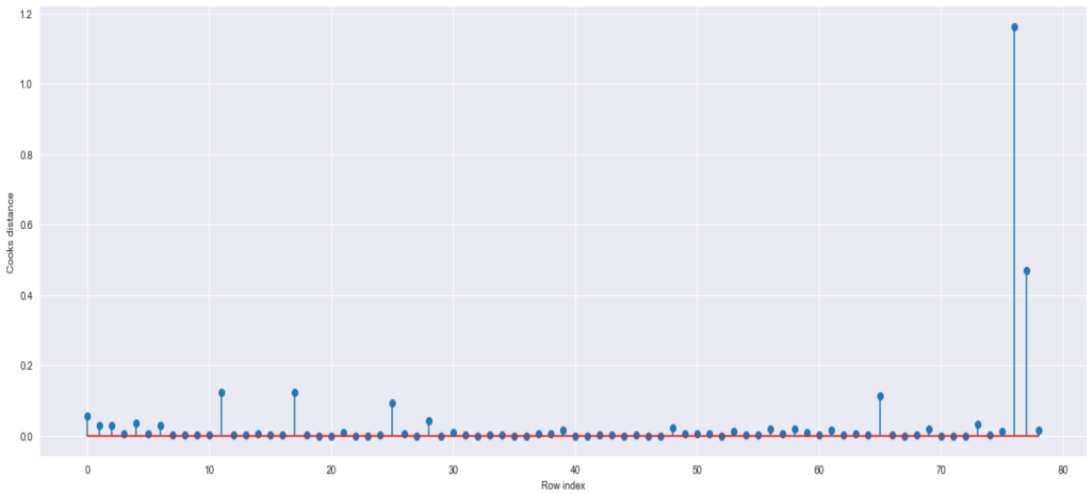
In [51]:
```
final_ml_W = smf.ols('MPG~WT+SP+HP', data=car1).fit()
```

In [52]:
```
(final_ml_W.rsquared, final_ml_W.aic)
```

Out[52]: (0.8160034320495304, 446.18843235750313)

In [53]:
```
model_influence_V = final_ml_V.get_influence()
(c_V,_)= model_influence_V.cooks_distance
```

In [54]:
```
fig = plt.subplots(figsize=(20,7))
plt.stem(np.arange(len(car1)), np.round(c_V,3))
plt.xlabel('Row index')
plt.ylabel('Cooks distance')
plt.show()
```



In [55]:
```
(np.argmax(c_V),np.max(c_V))
```
Out[55]:
```
(76, 1.1629387469135182)
```

In [56]:
```
car2 = car1.drop(car1.index[[76,77]],axis=0)
```

In [57]:
```
car2
```
Out[57]:

	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
...	...	...	...	...	...
72	140	19.086341	160	124.715241	52.997752
73	140	19.086341	129	121.864163	42.618698
74	175	18.762837	129	132.864163	42.778219
75	238	19.197888	115	150.576579	37.923113
78	236	12.101263	107	139.840817	34.948615

77 rows × 5 columns

In [58]:
```
car3 =car2.reset_index()
```

In [60]:
```
car4 = car3.drop(['index'],axis=1)
```

In [61]:
```
car4
```
Out[61]:

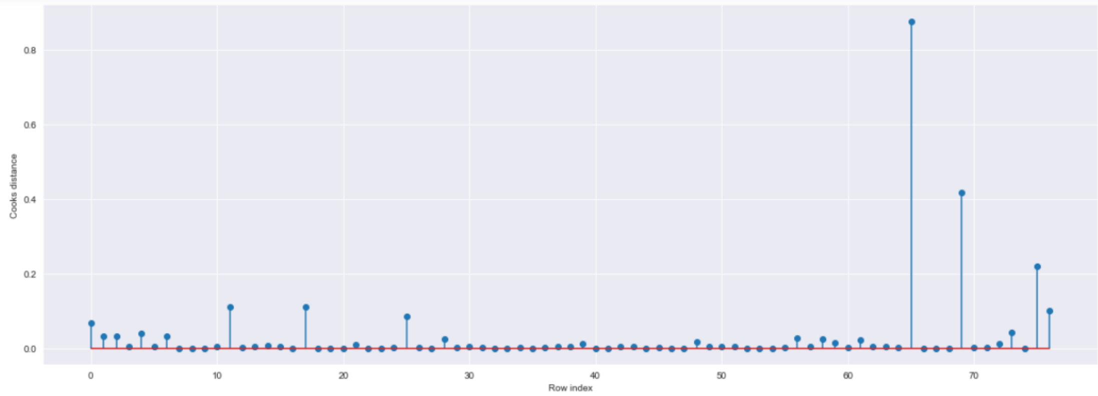
	HP	MPG	VOL	SP	WT
0	49	53.700681	89	104.185353	28.762059
1	55	50.013401	92	105.461264	30.466833
2	55	50.013401	92	105.461264	30.193597
3	70	45.696322	92	113.461264	30.632114
4	53	50.504232	92	104.461264	29.889149
...	...	...	...	...	...
72	140	19.086341	160	124.715241	52.997752
73	140	19.086341	129	121.864163	42.618698
74	175	18.762837	129	132.864163	42.778219
75	238	19.197888	115	150.576579	37.923113
76	236	12.101263	107	139.840817	34.948615

77 rows × 5 columns

In [62]:
```
final_ml_V = smf.ols('MPG~VOL+SP+HP', data=car4).fit()
```

In [63]:
```
model_influence_V = final_ml_V.get_influence()
(c_V,_)= model_influence_V.cooks_distance
```

In [64]:
```
fig = plt.subplots(figsize=(20,7))
plt.stem(np.arange(len(car4)), np.round(c_V,3))
plt.xlabel('Row index')
plt.ylabel('Cooks distance')
plt.show()
```




```
In [65]: (np.argmax(c_V),np.max(c_V))
Out[65]: (65, 0.8774556986296674)

In [66]: final_ml_V = smf.ols('MPG~VOL+SP+HP', data=car4).fit()

In [67]: (final_ml_V.rsquared, final_ml_V.aic)
Out[67]: (0.8669636111859063, 409.4153062719508)
```

```
In [68]: new_data = pd.DataFrame({'HP':40,'VOL':95,'SP':102,'WT':35}, index=[1])

In [69]: final_ml_V.predict(new_data)
Out[69]: 1    46.035594
         dtype: float64

In [70]: final_ml_V.predict(cars_new.iloc[0:5,])
Out[70]: 0    45.428872
         1    43.992392
         2    43.992392
         3    43.508150
         4    44.085858
         dtype: float64

In [71]: pred_y = final_ml_V.predict(cars_new)

In [72]: pred_y
Out[72]: 0     45.428872
         1     43.992392
         2     43.992392
         3     43.508150
         4     44.085858
                 ...
         76     7.165876
         77    12.198598
         78    14.908588
         79     4.163958
         80     9.161202
         Length: 81, dtype: float64
```

CONCLUSION:

In summary, the multiple linear regression (MLR) experiment revealed insights into the relationship between a dependent variable and multiple independent variables. Through the method of least squares, we estimated coefficients and evaluated model fit using metrics like . We also checked assumptions and interpreted coefficients to understand each variable's impact on the outcome. This experiment underscores the importance of robust variable selection and provides valuable tools for analyzing complex data relationships across various fields.