50_STARTUPS

In [74]:

```
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 snf
import numpy as np
import statsmodels.api as sm
```

In [53]:
```
st=pd.read_csv("C:\\Users\\nishi\\Downloads\\50_Startups.csv")
st
```

Out[53]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92

	R&D Spend	Administration	Marketing Spend	State	Profit
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

In [54]:
```
st.columns = st.columns.str.replace(' ','_')
```

In [55]:
```
st.head()
```

Out[55]:

	R&D_Spend	Administration	Marketing_Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

In [56]:
```
st['R&D_Spend'].mean()
```

Out[56]: 73721.61559999999

In [57]:
```
st['Administration'].mean()
```

Out[57]: 121344.63959999995

In [58]:
```
st['Marketing_Spend'].mean()
```
Out[58]:
```
211025.09780000005
```
In [59]:
```
st['Profit'].mean()
```
Out[59]:
```
112012.63920000002
```
In [60]:
```
st.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D_Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing_Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

In [61]:
```
st1=st.rename({'R&D_Spend':'rdSpend','Administration':'Adm','Marketing_Spend':'Ma
st1
```

Out[61]:

	rdSpend	Adm	Mark	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92

	rdSpend	Adm	Mark	State	Profit
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

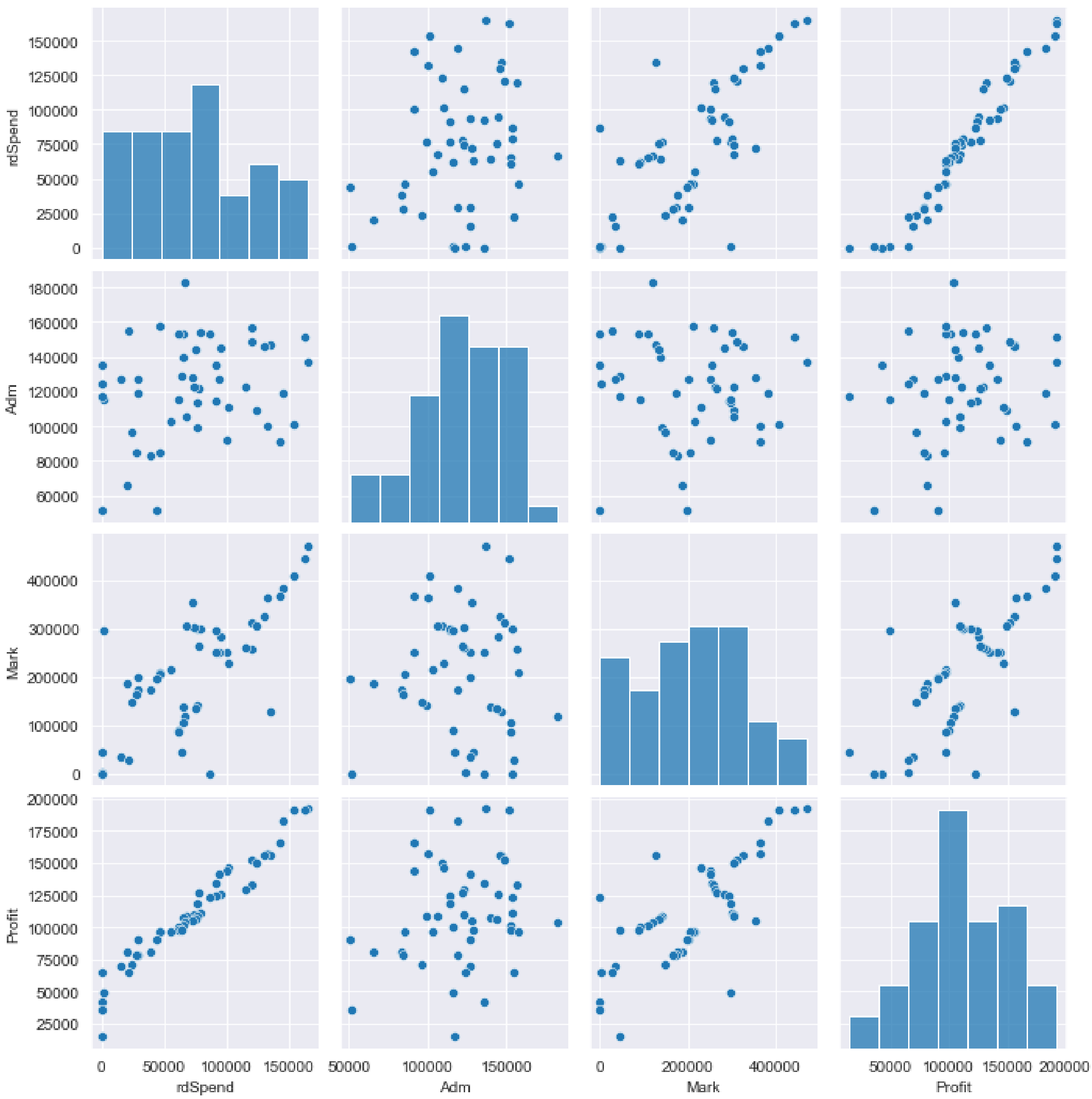
In [62]:
```
# finding the correlaation
st1.corr()
```

Out[62]:

	rdSpend	Adm	Mark	Profit
rdSpend	1.000000	0.241955	0.724248	0.972900
Adm	0.241955	1.000000	-0.032154	0.200717
Mark	0.724248	-0.032154	1.000000	0.747766
Profit	0.972900	0.200717	0.747766	1.000000

In [63]:
```
sns.set_style(style='darkgrid')
sns.pairplot(st1)
```

Out[63]: <seaborn.axisgrid.PairGrid at 0x1d0699d99a0>



In [64]:
```
# check for null values
st1.isna().sum()
```

Out[64]:
```
rdSpend    0
Adm        0
Mark       0
State      0
Profit     0
dtype: int64
```

In [65]:
```
#Build the model
import statsmodels.formula.api as snf
model=snf.ols('Profit~Adm+Mark+rdSpend',data=st1).fit()
```

In [13]:
```
model.fittedvalues
```

Out[13]:
```
0     192521.252890
1     189156.768232
2     182147.279096
3     173696.700026
4     172139.514183
5     163580.780571
6     158114.096669
7     160021.363048
8     151741.699699
9     154884.684110
10    135509.016367
11    135573.712961
12    129138.054182
13    127487.991663
14    149548.646335
15    146235.159985
16    116915.405401
17    130192.447208
18    129014.226806
19    115635.216367
20    116639.669231
21    117319.451640
22    114706.981717
23    109996.615221
24    113362.966113
25    102237.725065
26    110600.575350
27    114408.071457
28    101660.026005
29    101794.983452
30     99452.372936
31     97687.856276
32     99001.328985
33     97915.007805
34     89039.273741
35     90511.599568
36     75286.174585
37     89619.537708
38     69697.430648
39     83729.011977
40     74815.953991
41     74802.556239
42     70620.411821
43     60167.039963
44     64611.354916
45     47650.649687
46     56166.206853
47     46490.588983
48     49171.388158
49     48215.134111
dtype: float64
```

In [66]:
```
# beta coefficients
model.params
```

Out[66]:
```
Intercept    50122.192990
Adm             -0.026816
Mark             0.027228
rdSpend          0.805715
dtype: float64
```

In [67]:
```
# t and p values
(model.pvalues,"\n",model.tvalues)
```

Out[67]:
```
(Intercept    1.057379e-09
 Adm          6.017551e-01
 Mark         1.047168e-01
 rdSpend      2.634968e-22
 dtype: float64,
 '\n',
 Intercept     7.626218
 Adm          -0.525507
 Mark          1.655077
 rdSpend      17.846374
 dtype: float64)
```

In [68]:
```
# r squared values
(model.rsquared,model.rsquared_adj)
```

Out[68]:
```
(0.9507459940683246, 0.9475337762901719)
```

In [ ]:
```
#NOW FOR THE MULTI COLLINEARITY PROBLEM WE NEED TO FIND OUT THE Simple REGRESSION
#IT IS A GOOD PRACTICE
```

In [69]:
```
m1_v=snf.ols('Profit~Adm',data=st1).fit()
#p and the t values
print(m1_v.pvalues,"\n",m1_v.tvalues)
```

```
Intercept    0.003824
Adm          0.162217
dtype: float64
 Intercept    3.040044
Adm          1.419493
dtype: float64
```

In [70]:
```python
m1_v=snf.ols('Profit~Mark',data=st1).fit()
#p and the t values
print(m1_v.pvalues,"\n",m1_v.tvalues)
```

```
Intercept    4.294735e-10
Mark         4.381073e-10
dtype: float64
 Intercept    7.808356
Mark         7.802657
dtype: float64
```

In [71]:
```python
m1_v=snf.ols('rdSpend~Adm+Mark',data=st1).fit()
#p and the t values
print(m1_v.pvalues,"\n",m1_v.tvalues)
```

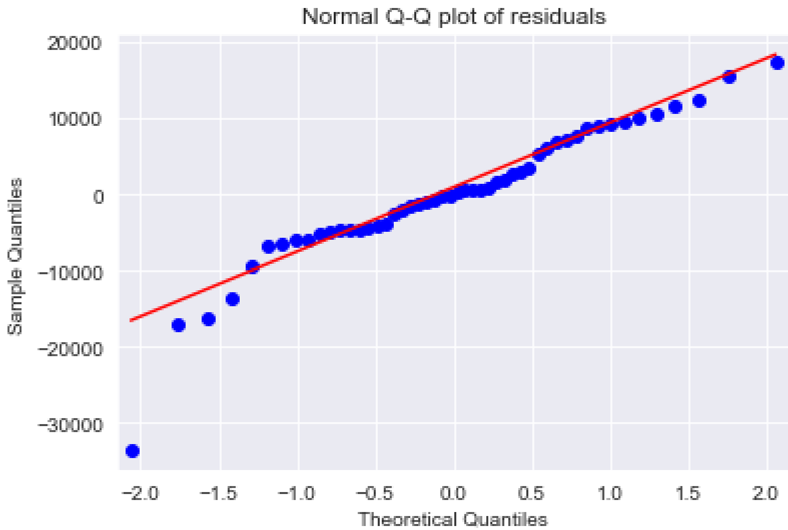
```
Intercept    7.713519e-02
Adm          6.322635e-03
Mark         3.724804e-10
dtype: float64
 Intercept   -1.807194
Adm          2.858702
Mark         7.889568
dtype: float64
```

In [73]:
```python
rsq_rd=snf.ols("rdSpend~Adm+Mark",data=st1).fit().rsquared
vif_rd=1/(1-rsq_rd)

rsq_adm=snf.ols("Adm~rdSpend+Mark",data=st1).fit().rsquared
vif_adm=1/(1-rsq_adm)

rsq_mark=snf.ols("Mark~Adm+rdSpend",data=st1).fit().rsquared
vif_mark=1/(1-rsq_mark)

# Putting the values in Dataframe format
d1={'Variables':['rdSpend','Administration','Marketing_Spend'],'Vif':[vif_rd,vif_
Vif_df=pd.DataFrame(d1)
Vif_df
```

Out[73]:

	Variables	Vif
0	rdSpend	2.468903
1	Administration	1.175091
2	Marketing_Spend	2.326773

In [75]:
```
sm.qqplot(model.resid,line='q')
plt.title("Normal Q-Q plot of residuals")
plt.show()
```



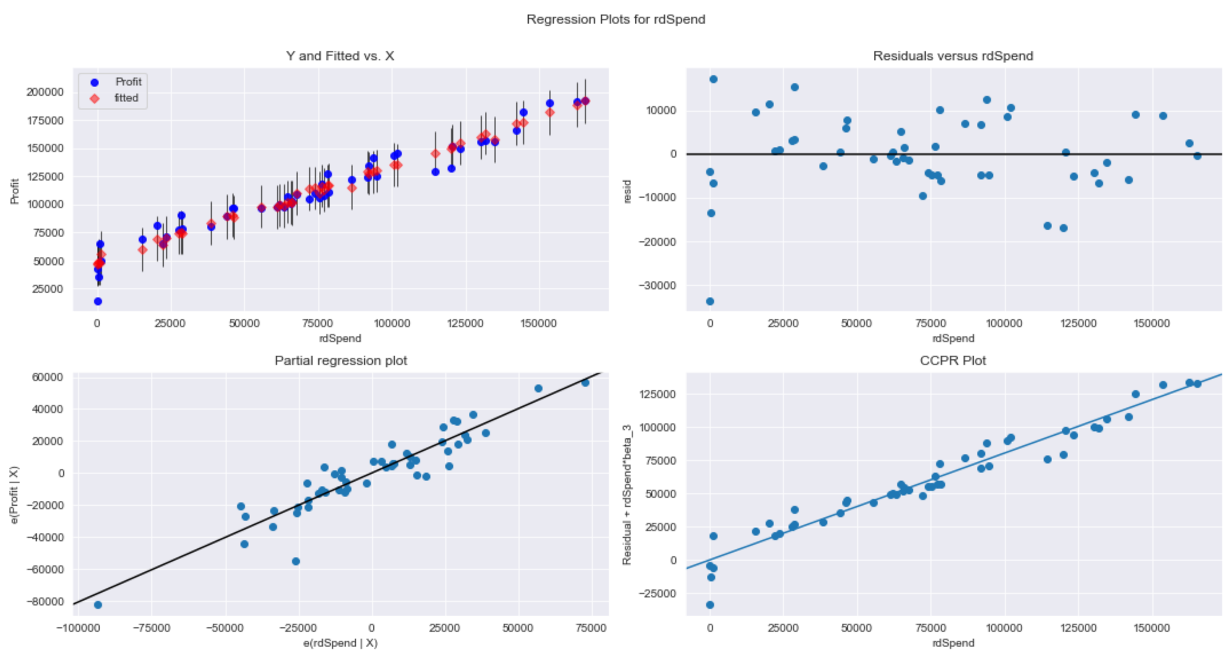
In [76]:
```
list(np.where(model.resid<-20000))
```

Out[76]:
```
[array([49], dtype=int64)]
```

In [77]:
```
def standard_values(vals) : return (vals-vals.mean())/vals.std()
```

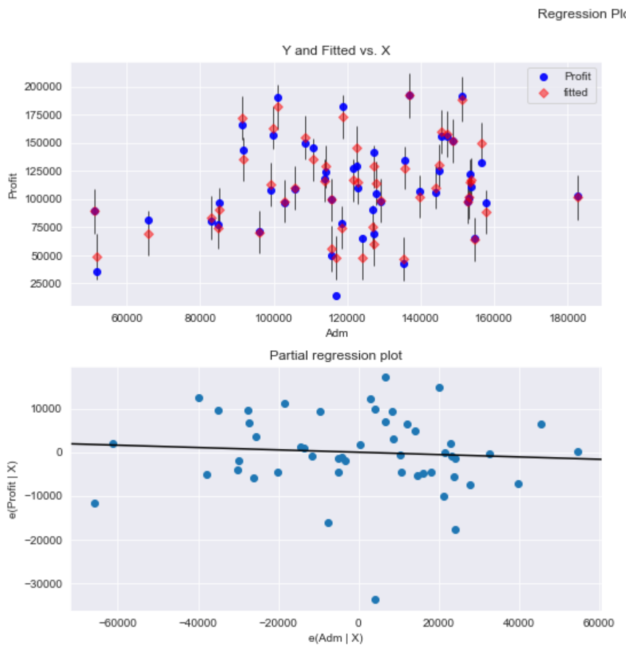
In [78]:
```python
plt.scatter(standard_values(model.fittedvalues),standard_values(model.resid))
plt.title('Residual Plot')
plt.xlabel('standardized fitted values')
plt.ylabel('standardized residual values')
plt.show()
```

In [79]:
```python
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'rdSpend',fig=fig)
plt.show()
```

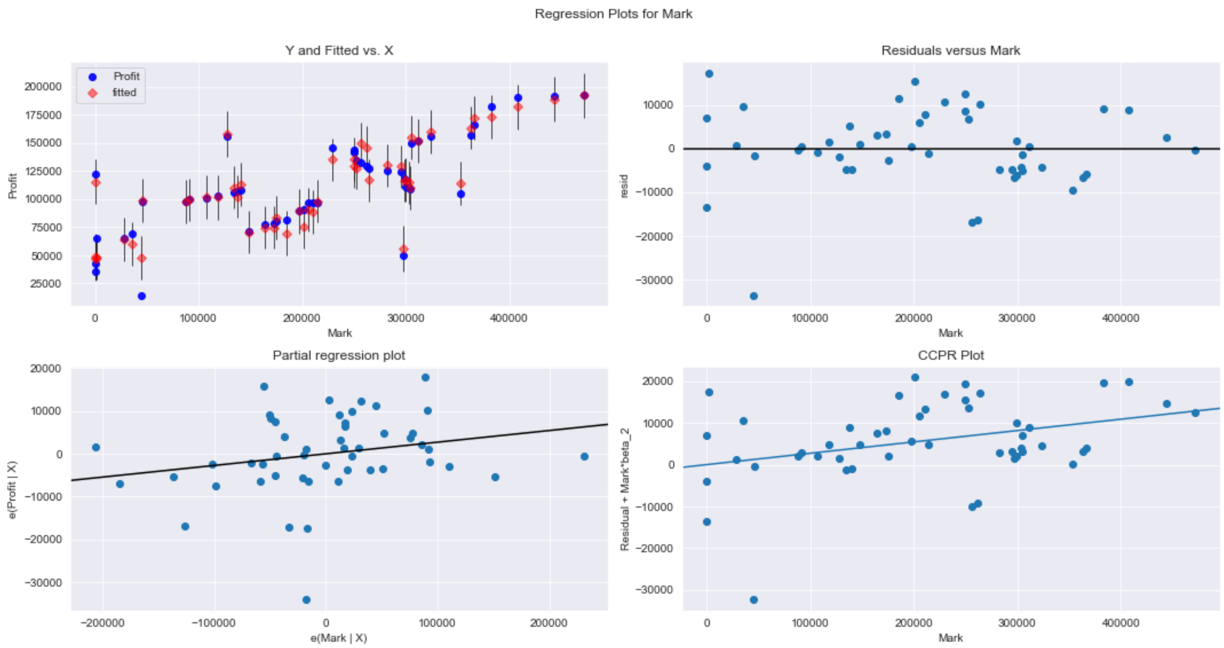


In [80]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Adm',fig=fig)
plt.show()
```



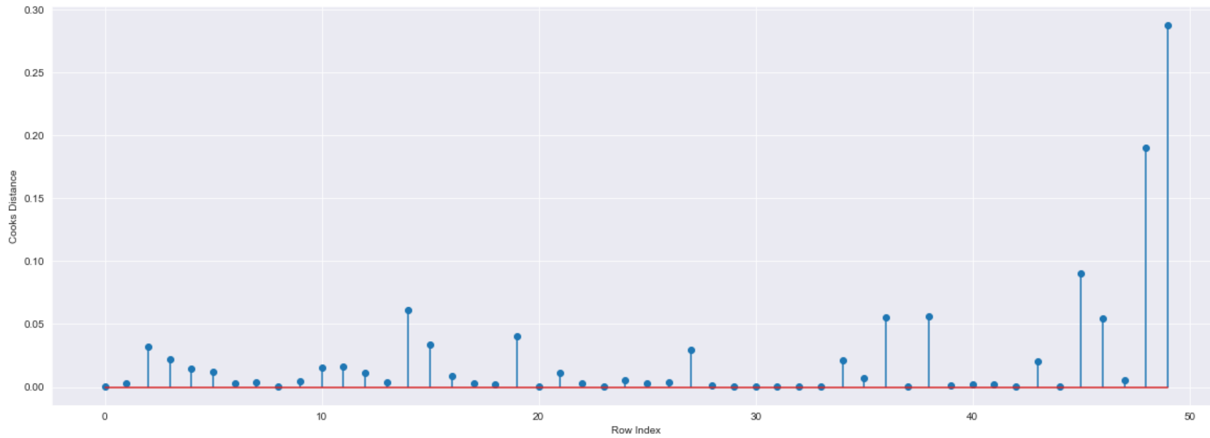
In [81]:
```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Mark',fig=fig)
plt.show()
```



In [82]:
```
(c,_)=model.get_influence().cooks_distance
c
```

Out[82]:
```
array([3.21825244e-05, 3.27591036e-03, 3.23842699e-02, 2.17206555e-02,
       1.44833032e-02, 1.17158463e-02, 2.91766303e-03, 3.56513444e-03,
       4.04303948e-05, 4.86758017e-03, 1.51064757e-02, 1.63564959e-02,
       1.15516625e-02, 4.01422811e-03, 6.12934253e-02, 3.40013448e-02,
       8.33556413e-03, 3.30534399e-03, 2.16819303e-03, 4.07440577e-02,
       4.25137222e-04, 1.09844352e-02, 2.91768000e-03, 2.76030254e-04,
       5.04643588e-03, 3.00074623e-03, 3.41957068e-03, 2.98396413e-02,
       1.31590664e-03, 1.25992620e-04, 4.18505125e-05, 9.27434786e-06,
       7.08656521e-04, 1.28122674e-04, 2.09815032e-02, 6.69508674e-03,
       5.55314705e-02, 6.55050578e-05, 5.61547311e-02, 1.54279607e-03,
       1.84850929e-03, 1.97578066e-03, 1.36089280e-04, 2.05553171e-02,
       1.23156041e-04, 9.03234206e-02, 5.45303387e-02, 5.33885616e-03,
       1.90527441e-01, 2.88082293e-01])
```

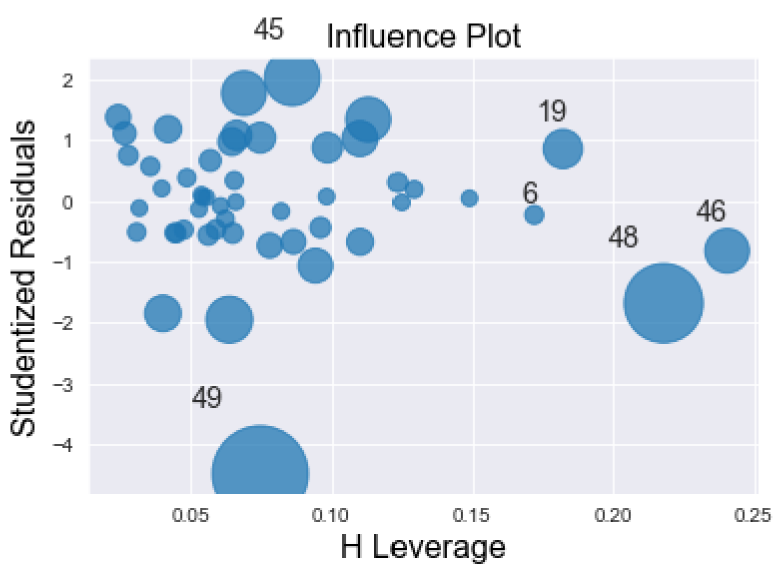
In [83]:
```
fig=plt.figure(figsize=(20,7))
plt.stem(np.arange(len(st1)),np.round(c,5))
plt.xlabel('Row Index')
plt.ylabel('Cooks Distance')
plt.show()
```



In [84]:
```
np.argmax(c) , np.max(c)
```

Out[84]: (49, 0.28808229275432634)

In [85]:
```
influence_plot(model)
plt.show()
```



In [87]:
```
k=st1.shape[1]
n=st1.shape[0]
leverage_cutoff = (3*(k+1))/n
leverage_cutoff
```

Out[87]: 0.36

In [88]:
```
st1[st1.index.isin([49])]
```

Out[88]:

	rdSpend	Adm	Mark	State	Profit
49	0.0	116983.8	45173.06	California	14681.4

In [89]:
```
st2=st1.drop(st1.index[[49]],axis=0).reset_index(drop=True)
st2
```

Out[89]:

	rdSpend	Adm	Mark	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92

	rdSpend	Adm	Mark	State	Profit
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41

In [90]:

```
while np.max(c)>0.5 :
    model=snf.ols("Profit~rdSpend+Adm+Mark",data=st2).fit()
    (c,_)=model.get_influence().cooks_distance
    c
    np.argmax(c) , np.max(c)
    st2=st2.drop(st2.index[[np.argmax(c)]],axis=0).reset_index(drop=True)
    data2
else:
    final_model=snf.ols("Profit~rdSpend+Adm+Mark",data=st2).fit()
    final_model.rsquared , final_model.aic
    print("Thus model accuracy is improved to",final_model.rsquared)
```

```
Thus model accuracy is improved to 0.9613162435129847
```

In [91]:

```
final_model.rsquared
```

Out[91]:

```
0.9613162435129847
```

In [92]:
```
st2
```

Out[92]:

	rdSpend	Adm	Mark	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92

	rdSpend	Adm	Mark	State	Profit
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41

In [93]:

```
new_data=pd.DataFrame({'rdSpend':60000,"Adm":70000,"Mark":130000},index=[0])
new_data
```

Out[93]:

	rdSpend	Adm	Mark
0	60000	70000	130000

In [94]:

```
final_model.predict(new_data)
```

Out[94]:

```
0    101088.827113
dtype: float64
```

In [95]:
```
pred_y=final_model.predict(st2)
pred_y
```

Out[95]:
```
0     190716.676999
1     187537.122227
2     180575.526396
3     172461.144642
4     170863.486721
5     162582.583177
6     157741.338633
7     159347.735318
8     151328.826941
9     154236.846778
10    135507.792682
11    135472.855621
12    129355.599449
13    127780.129139
14    149295.404796
15    145937.941975
16    117437.627921
17    130408.626295
18    129129.234457
19    116641.003121
20    117097.731866
21    117911.019038
22    115248.217796
23    110603.139045
24    114051.073877
25    103398.054385
26    111547.638935
27    114916.165026
28    103027.229434
29    103057.621761
30    100656.410227
31     99088.213693
32    100325.741335
33     98962.303136
34     90552.307809
35     91709.288672
36     77080.554255
37     90722.503244
38     71433.021956
39     85147.375646
40     76625.510303
41     76492.145175
42     72492.394974
43     62592.049718
44     67025.731107
45     50457.297206
46     58338.443625
47     49375.776655
48     51658.096812
dtype: float64
```

In [96]:
```
d2={'Prep_Models':['Model','Final_Model'],'Rsquared':[model.rsquared,final_model.
table=pd.DataFrame(d2)
table
```

Out[96]:

	Prep_Models	Rsquared
0	Model	0.950746
1	Final_Model	0.961316

ToyotaCorolla

In [1]:
```
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 snf
import numpy as np
import statsmodels.api as sm
```

In [2]:
```
cars=pd.read_csv("C:\\Users\\nishi\\Desktop\\Assignments\\Multi_Linear_Regressior
```

In [16]:
```
cars
```

Out[16]:

	Id	Model	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	46986	90	3	4	5	1165
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	72937	90	3	4	5	1165
2	3	ÊTOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	41711	90	3	4	5	1165
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	48000	90	3	4	5	1165
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...	...	...
1431	1438	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	20544	86	3	4	5	1025
1432	1439	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	19000	86	3	4	5	1015
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	17016	86	3	4	5	1015
1434	1441	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	16916	86	3	4	5	1015
1435	1442	TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors	6950	76	1	110	5	4	5	1114

1436 rows × 10 columns

In [19]:
```
cars1=cars.drop("Cylinders",axis=1) # We drop the Cylinders column as it has no v
```

In [20]:
```
cars1
```

Out[20]:

	Id	Model	Price	Age_08_04	KM	HP	Doors	Gears	Weight
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	46986	90	3	5	1165
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	72937	90	3	5	1165
2	3	ÊTOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	41711	90	3	5	1165
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	48000	90	3	5	1165
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	38500	90	3	5	1170
...	...	...	...	...	...	...	...	...	...
1431	1438	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	20544	86	3	5	1025
1432	1439	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	19000	86	3	5	1015
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	17016	86	3	5	1015
1434	1441	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	16916	86	3	5	1015
1435	1442	TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors	6950	76	1	110	5	5	1114

1436 rows × 9 columns

In [21]:
```
cars1["Price"].mean()
```

Out[21]: 10730.824512534818

In [22]:
```
cars1["HP"].mean()
```

Out[22]: 101.50208913649026

In [23]:
```
cars1["Doors"].mean()
```

Out[23]: 4.0334261838440115

In [24]:
```
cars1["Gears"].mean()
```

Out[24]: 5.0264623955431755

In [25]:
```
cars1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Id         1436 non-null   int64 
 1   Model      1436 non-null   object
 2   Price      1436 non-null   int64 
 3   Age_08_04  1436 non-null   int64 
 4   KM         1436 non-null   int64 
 5   HP         1436 non-null   int64 
 6   Doors      1436 non-null   int64 
 7   Gears      1436 non-null   int64 
 8   Weight     1436 non-null   int64 
dtypes: int64(8), object(1)
memory usage: 101.1+ KB
```

In [32]:
```
cars2=pd.concat([cars1.iloc[:,0:]],axis=1)
cars2
```

Out[32]:

	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
0	13500	23	46986	90	3	4	5	1165
1	13750	23	72937	90	3	4	5	1165
2	13950	24	41711	90	3	4	5	1165
3	14950	26	48000	90	3	4	5	1165
4	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	3	4	5	1025
1432	10845	72	19000	86	3	4	5	1015
1433	8500	71	17016	86	3	4	5	1015
1434	7250	70	16916	86	3	4	5	1015
1435	6950	76	1	110	5	4	5	1114

1436 rows × 8 columns

In [39]:
```
cars3=cars2.rename({'Age_08_04':'Age'},axis=1)
cars3
```

Out[39]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
0	13500	23	46986	90	3	4	5	1165
1	13750	23	72937	90	3	4	5	1165
2	13950	24	41711	90	3	4	5	1165
3	14950	26	48000	90	3	4	5	1165
4	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	3	4	5	1025
1432	10845	72	19000	86	3	4	5	1015
1433	8500	71	17016	86	3	4	5	1015
1434	7250	70	16916	86	3	4	5	1015
1435	6950	76	1	110	5	4	5	1114

1436 rows × 8 columns

In [40]:
```
cars3[cars3.duplicated()]
```

Out[40]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
113	24950	8	13253	116	5	4	5	1320

In [41]:
```
cars3=cars3.drop_duplicates().reset_index(drop=True)
cars3
```

Out[41]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
0	13500	23	46986	90	3	4	5	1165
1	13750	23	72937	90	3	4	5	1165
2	13950	24	41711	90	3	4	5	1165
3	14950	26	48000	90	3	4	5	1165
4	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...
1430	7500	69	20544	86	3	4	5	1025
1431	10845	72	19000	86	3	4	5	1015
1432	8500	71	17016	86	3	4	5	1015
1433	7250	70	16916	86	3	4	5	1015
1434	6950	76	1	110	5	4	5	1114

1435 rows × 8 columns

In [42]:
```
cars3
```

Out[42]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
0	13500	23	46986	90	3	4	5	1165
1	13750	23	72937	90	3	4	5	1165
2	13950	24	41711	90	3	4	5	1165
3	14950	26	48000	90	3	4	5	1165
4	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...
1430	7500	69	20544	86	3	4	5	1025
1431	10845	72	19000	86	3	4	5	1015
1432	8500	71	17016	86	3	4	5	1015
1433	7250	70	16916	86	3	4	5	1015
1434	6950	76	1	110	5	4	5	1114

1435 rows × 8 columns

In [43]:
```
cars3.describe()
```

Out[43]:

	Price	Age	KM	HP	Doors	Cylinders	Gears
count	1435.000000	1435.000000	1435.000000	1435.000000	1435.000000	1435.0	1435.00000(
mean	10720.915679	55.980488	68571.782578	101.491986	4.032753	4.0	5.02648′
std	3608.732978	18.563312	37491.094553	14.981408	0.952667	0.0	0.18857ŧ
min	4350.000000	1.000000	1.000000	69.000000	2.000000	4.0	3.00000(
25%	8450.000000	44.000000	43000.000000	90.000000	3.000000	4.0	5.00000(
50%	9900.000000	61.000000	63451.000000	110.000000	4.000000	4.0	5.00000(
75%	11950.000000	70.000000	87041.500000	110.000000	5.000000	4.0	5.00000(
max	32500.000000	80.000000	243000.000000	192.000000	5.000000	4.0	6.00000(

In [44]:
```
cars3.corr()
```

Out[44]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
Price	1.000000	-0.876273	-0.569420	0.314134	0.183604	NaN	0.063831	0.575869
Age	-0.876273	1.000000	0.504575	-0.155293	-0.146929	NaN	-0.005629	-0.466484
KM	-0.569420	0.504575	1.000000	-0.332904	-0.035193	NaN	0.014890	-0.023969
HP	0.314134	-0.155293	-0.332904	1.000000	0.091803	NaN	0.209642	0.087143
Doors	0.183604	-0.146929	-0.035193	0.091803	1.000000	NaN	-0.160101	0.301734
Cylinders	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gears	0.063831	-0.005629	0.014890	0.209642	-0.160101	NaN	1.000000	0.021238
Weight	0.575869	-0.466484	-0.023969	0.087143	0.301734	NaN	0.021238	1.000000

In [45]:
```
sns.set_style(style='darkgrid')
sns.pairplot(cars3)
```

Out[45]: <seaborn.axisgrid.PairGrid at 0x2c7b2c7c970>

In [46]:
```
model=snf.ols('Price~Age+KM+HP+Doors+Gears+Weight',data=cars3).fit()
```

In [47]:
```
model.params
```

Out[47]:
```
Intercept   -6838.987234
Age          -122.288250
KM             -0.019928
HP             28.327782
Doors          -8.715826
Gears         625.297840
Weight         18.455133
dtype: float64
```

In [48]:
```
model.tvalues , np.round(model.pvalues,5)
```

Out[48]:
```
(Intercept    -5.204801
 Age         -46.774676
 KM          -16.490150
 HP           10.840831
 Doors        -0.218061
 Gears         3.169455
 Weight       22.141591
 dtype: float64,
 Intercept    0.00000
 Age          0.00000
 KM           0.00000
 HP           0.00000
 Doors        0.82741
 Gears        0.00156
 Weight       0.00000
 dtype: float64)
```

In [49]:
```
model.rsquared , model.rsquared_adj
```

Out[49]:
```
(0.8615946984866649, 0.8610131636063568)
```

In [50]:
```
slr_c=snf.ols('Price~Doors',data=cars3).fit()
slr_c.tvalues , slr_c.pvalues
```

Out[50]:
```
(Intercept    19.421546
 Doors         7.070520
 dtype: float64,
 Intercept    8.976407e-75
 Doors        2.404166e-12
 dtype: float64)
```

In [51]:

```python
rsq_age=snf.ols('Age~KM+HP+Doors+Gears+Weight',data=cars3).fit().rsquared
vif_age=1/(1-rsq_age)

rsq_KM=snf.ols('KM~Age+HP+Doors+Gears+Weight',data=cars3).fit().rsquared
vif_KM=1/(1-rsq_KM)

rsq_HP=snf.ols('HP~Age+KM+Doors+Gears+Weight',data=cars3).fit().rsquared
vif_HP=1/(1-rsq_HP)

rsq_DR=snf.ols('Doors~Age+KM+HP+Gears+Weight',data=cars3).fit().rsquared
vif_DR=1/(1-rsq_DR)

rsq_GR=snf.ols('Gears~Age+KM+HP+Doors+Weight',data=cars3).fit().rsquared
vif_GR=1/(1-rsq_GR)

rsq_WT=snf.ols('Weight~Age+KM+HP+Doors+Gears',data=cars3).fit().rsquared
vif_WT=1/(1-rsq_WT)

# Putting the values in Dataframe format
d1={'Variables':['Age','KM','HP','Doors','Gears','Weight'],
    'Vif':[vif_age,vif_KM,vif_HP,vif_DR,vif_GR,vif_WT]}
Vif_df=pd.DataFrame(d1)
Vif_df
```

Out[51]:

	Variables	Vif
0	Age	1.866057
1	KM	1.626264
2	HP	1.214147
3	Doors	1.148708
4	Gears	1.096575
5	Weight	1.502749

In [58]:
```
sm.qqplot(model.resid,line='q') # 'q' - A line is fit through the quartiles # lin
plt.title("Normal Q-Q plot of residuals")
plt.show()
```

In [71]:
```
list(np.where(model.resid<-6000))
```

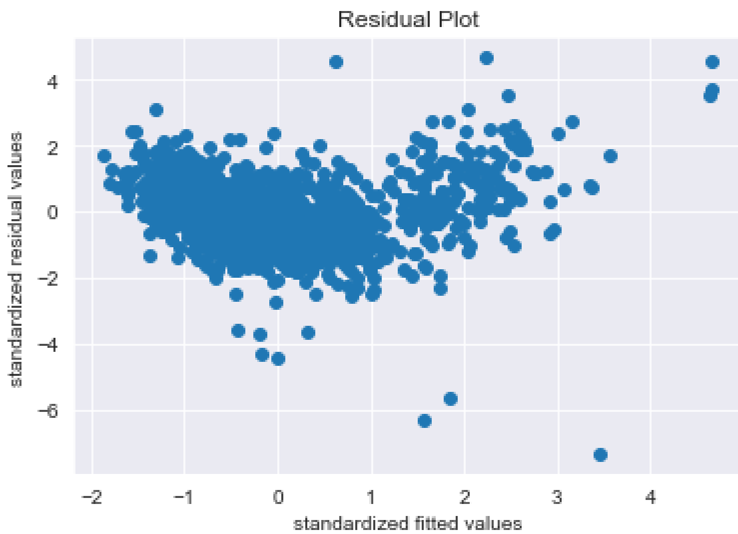
Out[71]: [array([220, 600, 959], dtype=int64)]

In [72]:
```
list(np.where(model.resid>6000))
```

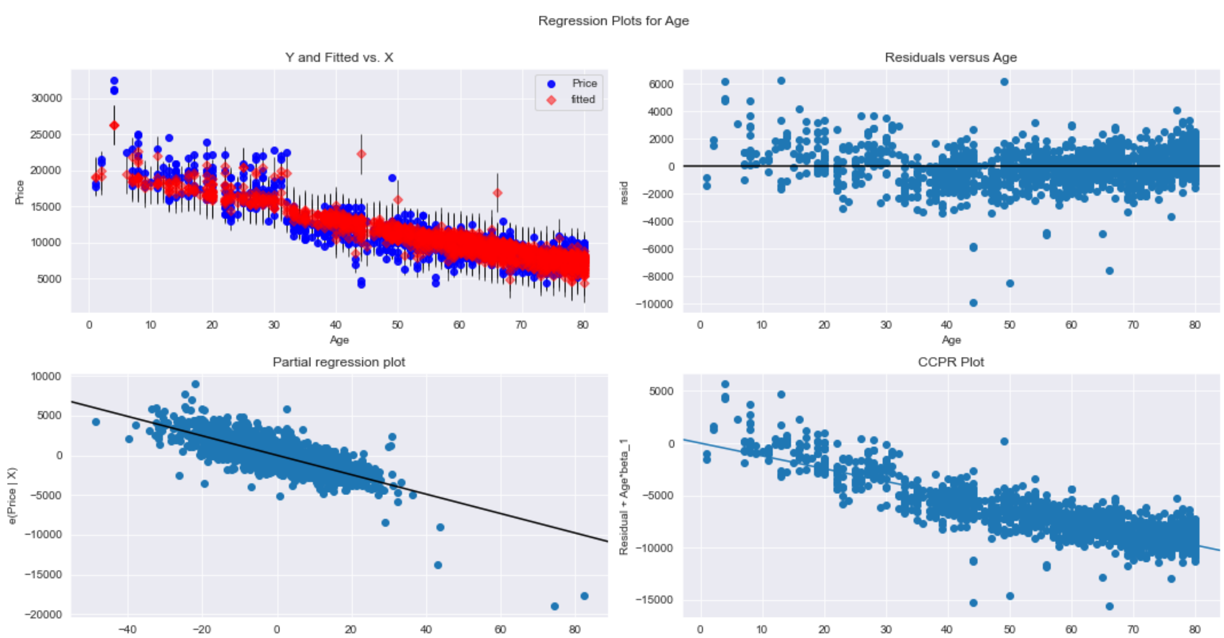
Out[72]: [array([109, 146, 522], dtype=int64)]

In [73]:
```
def standard_values(vals) : return (vals-vals.mean())/vals.std()
```

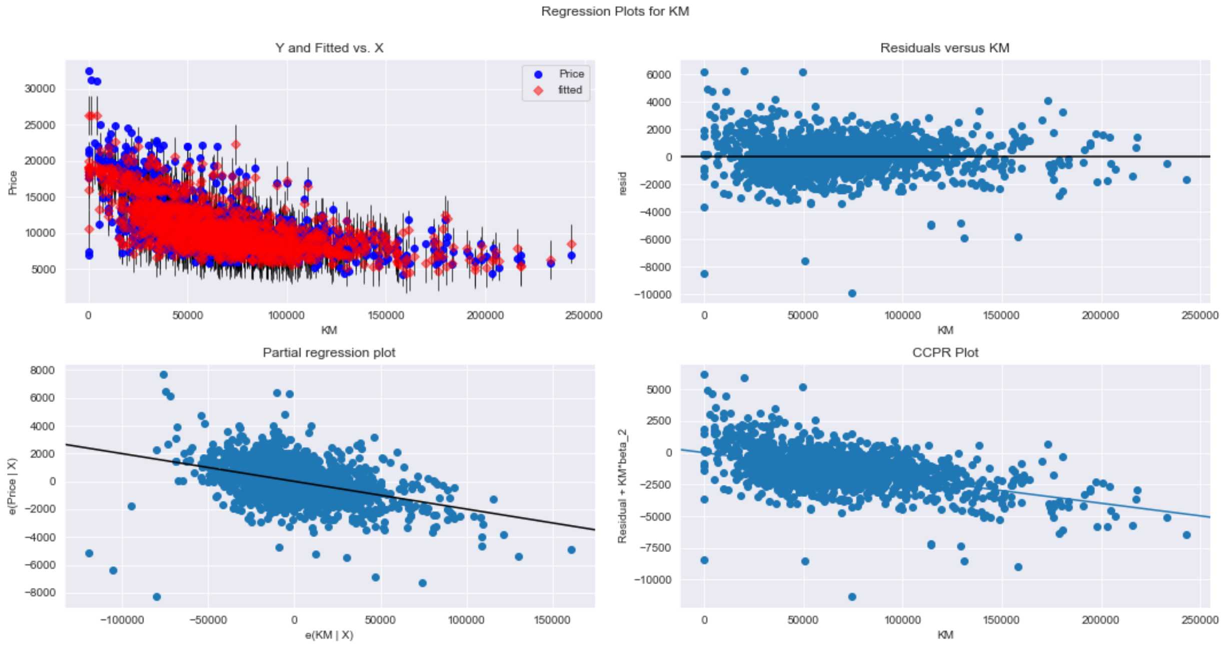
In [74]:
```
plt.scatter(standard_values(model.fittedvalues),standard_values(model.resid))
plt.title('Residual Plot')
plt.xlabel('standardized fitted values')
plt.ylabel('standardized residual values')
plt.show()
```



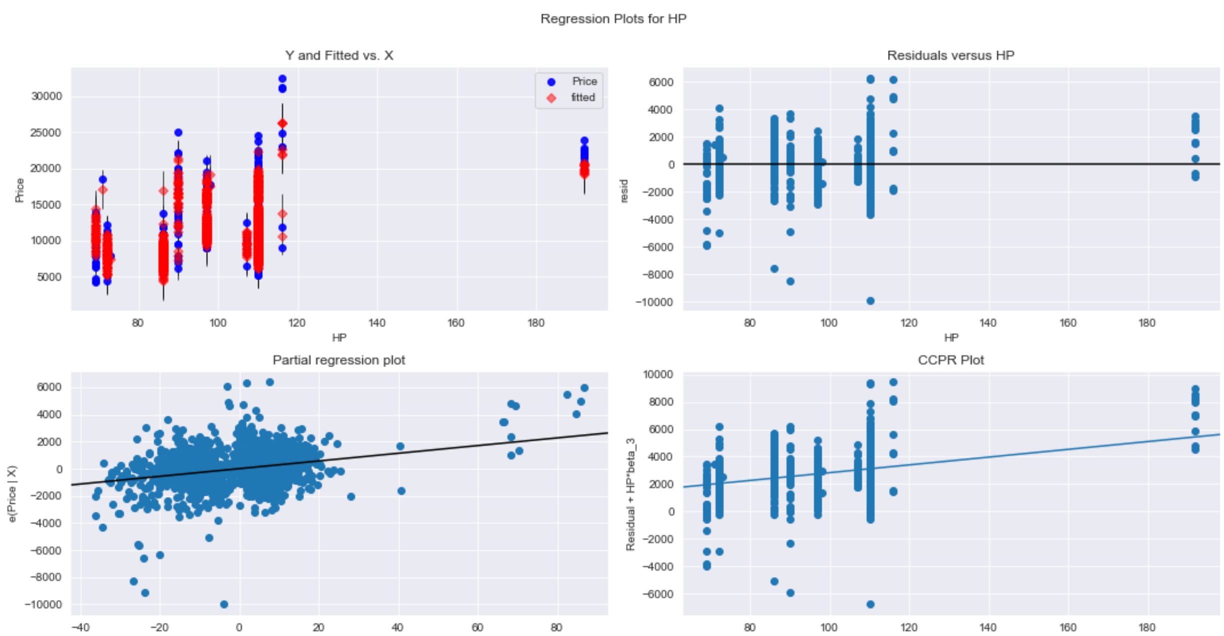
In [76]:
```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Age',fig=fig)
plt.show()
```



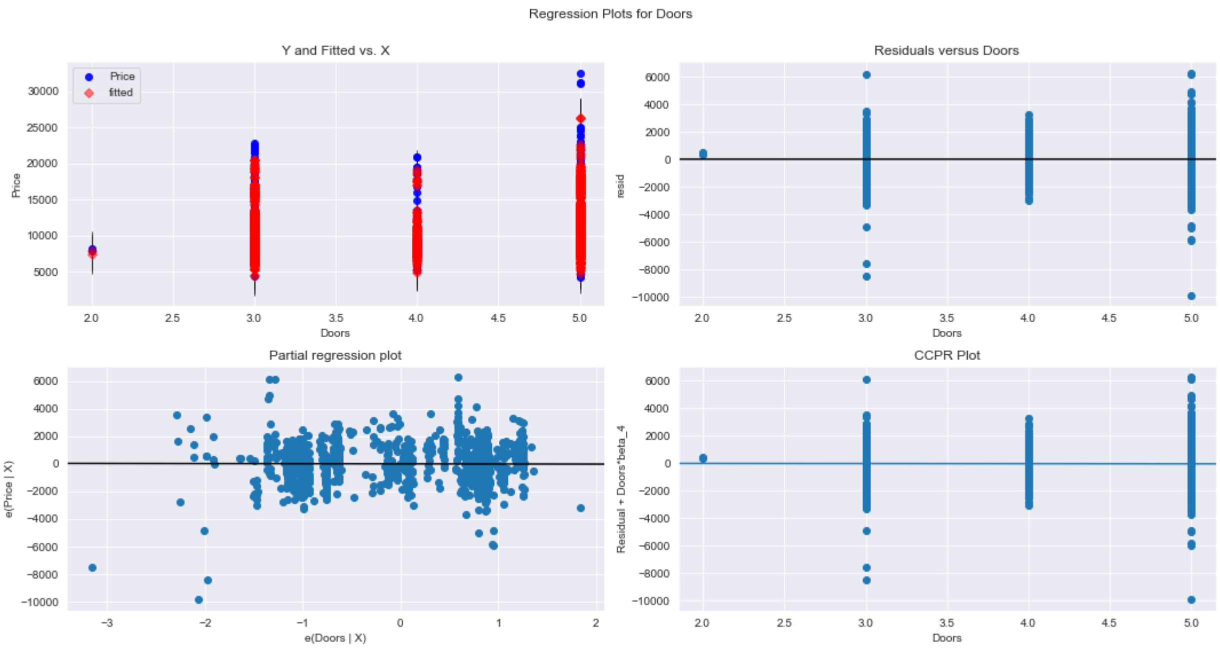
In [77]:
```python
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'KM',fig=fig)
plt.show()
```



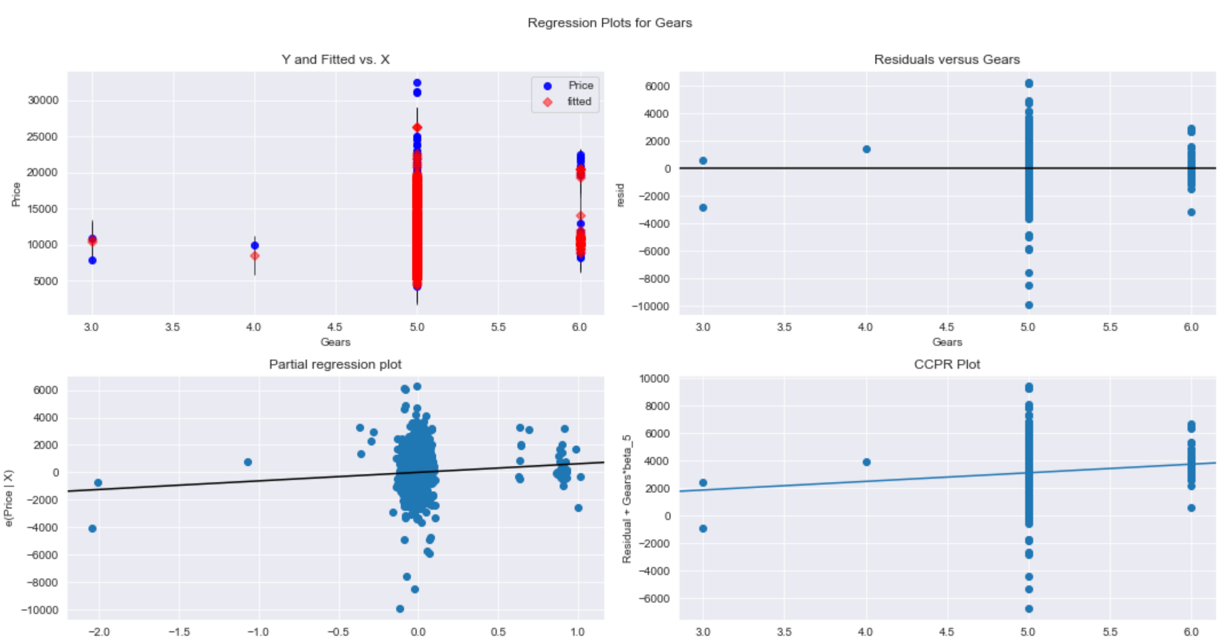
In [56]:
```python
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'HP',fig=fig)
plt.show()
```



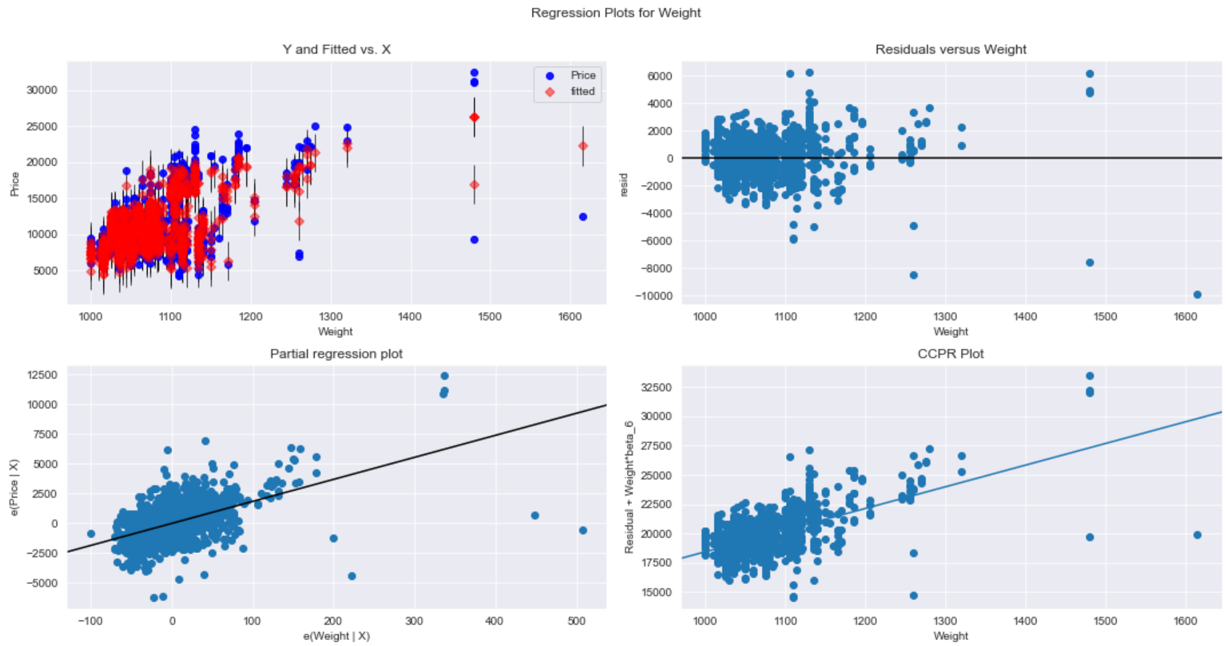
In [78]:
```python
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Doors',fig=fig)
plt.show()
```



In [64]:
```python
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Gears',fig=fig)
plt.show()
```



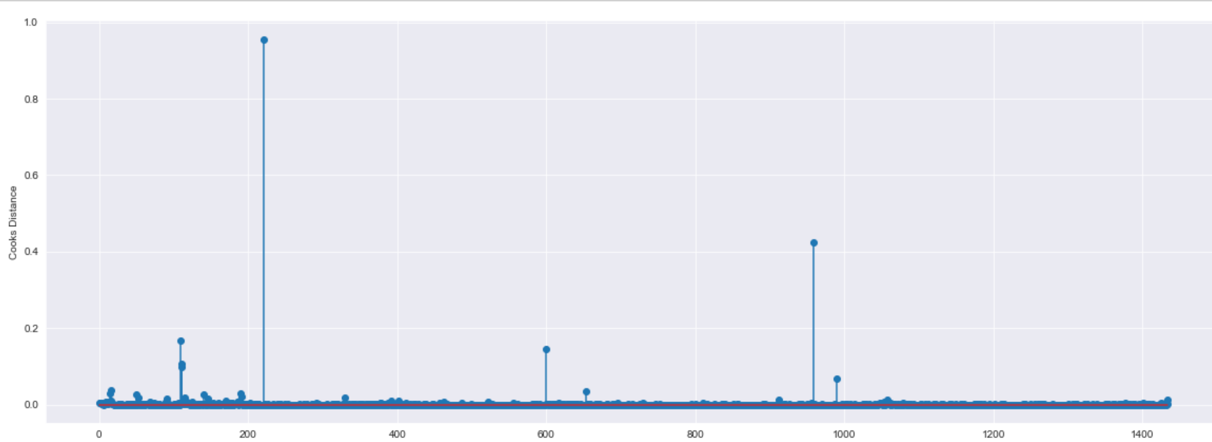
In [79]:
```python
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Weight',fig=fig)
plt.show()
```



In [80]:
```python
(c,_)=model.get_influence().cooks_distance
c
```

Out[80]:
```
array([4.84834865e-03, 2.81504747e-03, 3.49062601e-03, ...,
       4.29681961e-06, 8.15626746e-04, 1.20038984e-02])
```

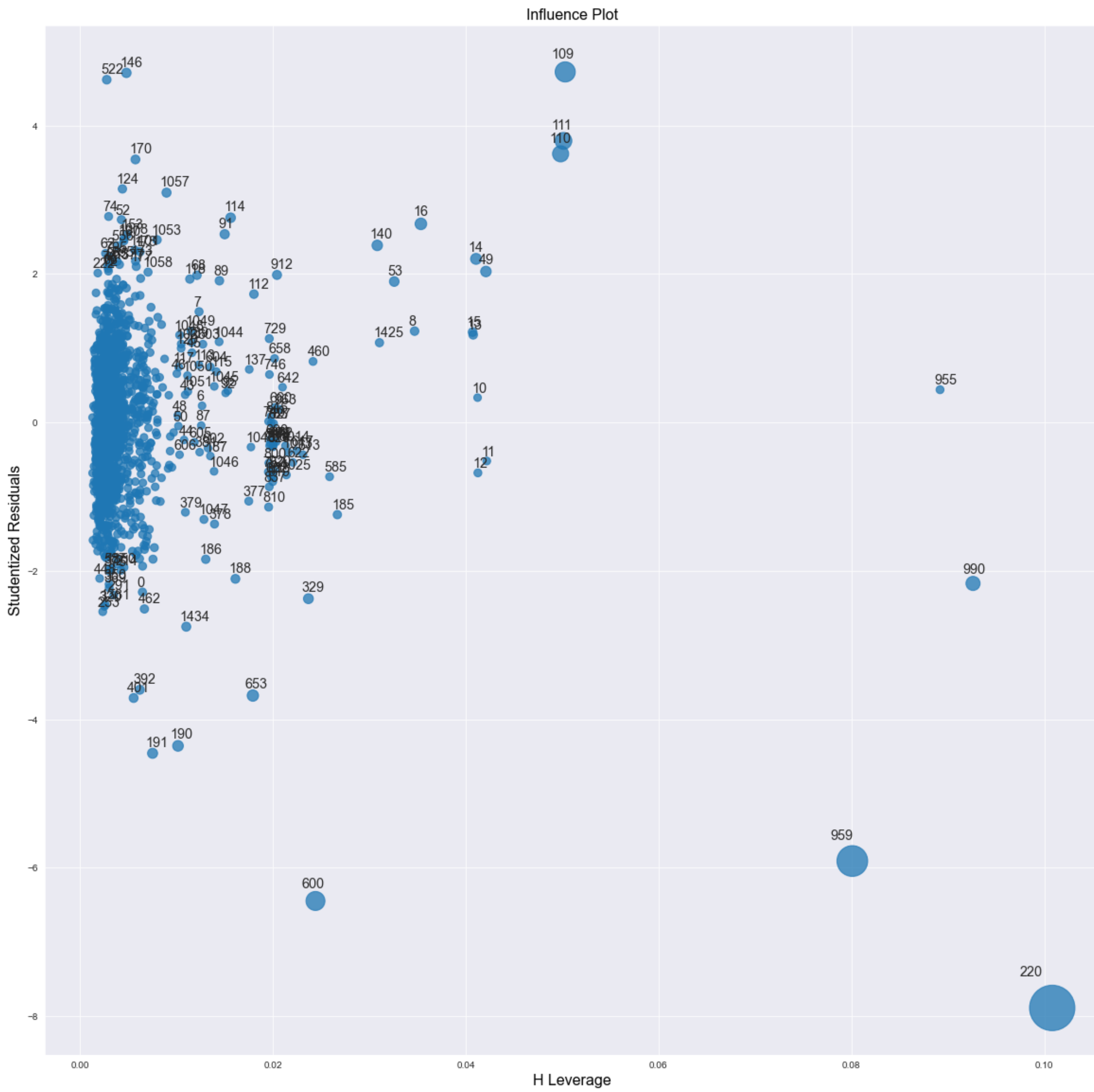
In [66]:
```python
fig=plt.figure(figsize=(20,7))
plt.stem(np.arange(len(cars3)),np.round(c,3))
plt.xlabel('Row Index')
plt.ylabel('Cooks Distance')
plt.show()
```



In [81]:
```python
np.argmax(c) , np.max(c)
```

Out[81]:
```
(220, 0.9561392473392505)
```

In [82]:
```python
fig,ax=plt.subplots(figsize=(20,20))
fig=influence_plot(model,ax = ax)
```



In [84]:
```python
k=cars3.shape[1]
n=cars3.shape[0]
leverage_cutoff = (3*(k+1))/n
leverage_cutoff
```

Out[84]:
```
0.018815331010452963
```

In [93]:
```
cars3[cars3.index.isin([220])]
```

Out[93]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
220	12450	44	74172	110	5	4	5	1615

In [86]:
```
cars4=cars3.copy()
cars4
```

Out[86]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
0	13500	23	46986	90	3	4	5	1165
1	13750	23	72937	90	3	4	5	1165
2	13950	24	41711	90	3	4	5	1165
3	14950	26	48000	90	3	4	5	1165
4	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...
1430	7500	69	20544	86	3	4	5	1025
1431	10845	72	19000	86	3	4	5	1015
1432	8500	71	17016	86	3	4	5	1015
1433	7250	70	16916	86	3	4	5	1015
1434	6950	76	1	110	5	4	5	1114

1435 rows × 8 columns

In [94]:
```
cars5=cars4.drop(cars4.index[[220]],axis=0).reset_index(drop=True)
cars5
```

Out[94]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
0	13500	23	46986	90	3	4	5	1165
1	13750	23	72937	90	3	4	5	1165
2	13950	24	41711	90	3	4	5	1165
3	14950	26	48000	90	3	4	5	1165
4	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...
1429	7500	69	20544	86	3	4	5	1025
1430	10845	72	19000	86	3	4	5	1015
1431	8500	71	17016	86	3	4	5	1015
1432	7250	70	16916	86	3	4	5	1015
1433	6950	76	1	110	5	4	5	1114

1434 rows × 8 columns

In [95]:
```
while np.max(c)>0.5 :
    model=snf.ols('Price~Age+KM+HP+Doors+Gears+Weight',data=cars5).fit()
    (c,_)=model.get_influence().cooks_distance
    c
    np.argmax(c) , np.max(c)
    cars5=cars5.drop(cars5.index[[np.argmax(c)]],axis=0).reset_index(drop=True)
    cars5
else:
    final_model=snf.ols('Price~Age+KM+HP+Doors+Gears+Weight',data=cars5).fit()
    final_model.rsquared , final_model.aic
    print("Thus model accuracy is improved to",final_model.rsquared)
```

```
Thus model accuracy is improved to 0.8673586781804876
```

In [96]:
```
final_model.rsquared
```

Out[96]:
```
0.8673586781804876
```

In [97]:
```
cars5
```

Out[97]:

	Price	Age	KM	HP	Doors	Cylinders	Gears	Weight
0	13500	23	46986	90	3	4	5	1165
1	13750	23	72937	90	3	4	5	1165
2	13950	24	41711	90	3	4	5	1165
3	14950	26	48000	90	3	4	5	1165
4	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...
1429	7500	69	20544	86	3	4	5	1025
1430	10845	72	19000	86	3	4	5	1015
1431	8500	71	17016	86	3	4	5	1015
1432	7250	70	16916	86	3	4	5	1015
1433	6950	76	1	110	5	4	5	1114

1434 rows × 8 columns

In [98]:
```
new_data=pd.DataFrame({'Age':12,"KM":40000,"HP":80,"Doors":4,"Gears":5,"QT":69,"W
new_data
```

Out[98]:

	Age	KM	HP	Doors	Gears	QT	Weight
0	12	40000	80	4	5	69	1012

In [99]:
```
final_model.predict(new_data)
```

Out[99]:
```
0    14699.298039
dtype: float64
```

In [100]:
```
pred_y=final_model.predict(cars5)
pred_y
```

Out[100]:
```
0       16704.446380
1       16168.971437
2       16694.086877
3       16325.910639
4       16148.065524
            ...
1429     8771.451048
1430     8239.800877
1431     8399.943149
1432     8521.210819
1433    10811.940431
Length: 1434, dtype: float64
```