Consider only the below columns and prepare a prediction model for predicting Price.

Corolla<-Corolla[c("Price","Age_08_04","KM","HP","cc","Doors","Gears","Quarterly_Tax","Weight")]

Model -- model of the car
Price -- Offer Price in EUROs
Age_08_04 -- Age in months as in August 2004
Mfg_Month -- Manufacturing month (1-12)
Mfg_Year -- Manufacturing Year
KM -- Accumulated Kilometers on odometer
Fuel_Type -- Fuel Type (Petrol, Diesel, CNG)
HP -- Horsepower
Met_Color -- Metallic Color? (Yes=1, No=0)
Color -- Color (Blue, Red, Grey, Silver, Black, etc.)
Automatic -- Automatic ((Yes=1, No=0)
cc -- Cylinder Volume in cubic centimeters
Doors -- Number of doors
Cylinders -- Number of cylinders
Gears -- Number of gear positions
Quarterly_Tax -- Quarterly road tax in EUROs
Weight -- Weight in Kilograms
Mfr_Guarantee -- Within Manufacturer's Guarantee period (Yes=1, No=0)
BOVAG_Guarantee -- BOVAG (Dutch dealer network) Guarantee (Yes=1, No=0)
Guarantee_Period -- Guarantee period in months
ABS -- Anti-Lock Brake System (Yes=1, No=0)
Airbag_1 -- Driver_Airbag (Yes=1, No=0)
Airbag_2 -- Passenger Airbag (Yes=1, No=0)
Airco -- Airconditioning (Yes=1, No=0)
Automatic_airco -- Automatic Airconditioning (Yes=1, No=0)
Boardcomputer -- Boardcomputer (Yes=1, No=0)
CD_Player -- CD Player (Yes=1, No=0)
Central_Lock -- Central Lock (Yes=1, No=0)
Powered_Windows -- Powered Windows (Yes=1, No=0)
Power_Steering -- Power Steering (Yes=1, No=0)
Radio -- Radio (Yes=1, No=0)
Mistlamps -- Mistlamps (Yes=1, No=0)
Sport_Model -- Sport Model (Yes=1, No=0)
Backseat_Divider -- Backseat Divider (Yes=1, No=0)
Metallic_Rim --Metallic Rim (Yes=1, No=0)
Radio_cassette -- Radio Cassette (Yes=1, No=0)
Tow_Bar -- Tow Bar (Yes=1, No=0)

—----------------Import Important Libraries—-----------------------------

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import statsmodels.api as sm
from statsmodels.graphics.regressionplots import influence_plot
```

—-------------Read the Dataset—--------------------------------

```
data = pd.read_csv('Downloads/ToyotaCorolla.csv')
data
```

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_Color	...	Central_Lock	Powered_Windows	Power_Steering	Rad
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002	46986	Diesel	90	1	...	1	1	1	
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002	72937	Diesel	90	1	...	1	0	1	
2	3	�TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002	41711	Diesel	90	1	...	0	0	1	
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	7	2002	48000	Diesel	90	0	...	0	0	1	
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002	38500	Diesel	90	0	...	1	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1431	1438	TOYOTA Corolla 1.3 16V HATCHB	7500	69	12	1998	20544	Petrol	86	1	...	1	1	1	

—---------------------Get information—--------------------------------

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 38 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   Mfg_Month         1436 non-null   int64
 5   Mfg_Year          1436 non-null   int64
 6   KM                1436 non-null   int64
 7   Fuel_Type         1436 non-null   object
 8   HP                1436 non-null   int64
 9   Met_Color         1436 non-null   int64
 10  Color             1436 non-null   object
 11  Automatic         1436 non-null   int64
 12  cc                1436 non-null   int64
 13  Doors             1436 non-null   int64
 14  Cylinders         1436 non-null   int64
 15  Gears             1436 non-null   int64
 16  Quarterly_Tax     1436 non-null   int64
 17  Weight            1436 non-null   int64
 18  Mfr_Guarantee     1436 non-null   int64
 19  BOVAG_Guarantee   1436 non-null   int64
 20  Guarantee_Period  1436 non-null   int64
 21  ABS               1436 non-null   int64
 22  Airbag_1          1436 non-null   int64
 23  Airbag_2          1436 non-null   int64
 24  Airco             1436 non-null   int64
```

—----------Choose the specific columns from dataset—---------------------

```
data1 = pd.concat
([data.iloc[:,2:4],data.iloc[:,6:7],data.iloc[:,8:9],data.iloc[:,12:14],data.iloc[:,15:18]],axis=1)
data1
```

	Price	Age_08_04	KM	HP	cc	Doors	Gears	Quarterly_Tax	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	1300	3	5	69	1025
1432	10845	72	19000	86	1300	3	5	69	1015
1433	8500	71	17016	86	1300	3	5	69	1015
1434	7250	70	16916	86	1300	3	5	69	1015
1435	6950	76	1	110	1600	5	5	19	1114

1436 rows × 9 columns

—--------------Rename Columns—------------------------------

```
data2 = data1.rename
({'Price':'Price','Age_08_04':'Age','KM':'km','HP':'hp','cc':'cc','Doors':'Doors','Gears':'gears','Quarte
rly_Tax':'QT','Weight':'WT'},axis=1)
data2
```

	Price	Age	km	hp	cc	Doors	gears	QT	WT
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	1300	3	5	69	1025
1432	10845	72	19000	86	1300	3	5	69	1015
1433	8500	71	17016	86	1300	3	5	69	1015
1434	7250	70	16916	86	1300	3	5	69	1015
1435	6950	76	1	110	1600	5	5	19	1114

1436 rows × 9 columns

—-------------Find duplicate Values—-----------------------
data2[data2.duplicated()]

	Price	Age	km	hp	cc	Doors	gears	QT	WT
113	24950	8	13253	116	2000	5	5	234	1320

—-----------------Drop duplicates from dataset—-------------------------
drop = data2.drop_duplicates().reset_index(drop=True)
drop

	Price	Age	km	hp	cc	Doors	gears	QT	WT
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1430	7500	69	20544	86	1300	3	5	69	1025
1431	10845	72	19000	86	1300	3	5	69	1015
1432	8500	71	17016	86	1300	3	5	69	1015
1433	7250	70	16916	86	1300	3	5	69	1015
1434	6950	76	1	110	1600	5	5	19	1114

1435 rows × 9 columns

—-----------------Get information—-------------------------------
drop.describe()

	Price	Age	km	hp	cc	Doors	gears	QT	WT
count	1435.000000	1435.000000	1435.000000	1435.000000	1435.000000	1435.000000	1435.000000	1435.000000	1435.000000
mean	10720.915679	55.980488	68571.782578	101.491986	1576.560976	4.032753	5.026481	87.020209	1072.287108
std	3608.732978	18.563312	37491.094553	14.981408	424.387533	0.952667	0.188575	40.959588	52.251882
min	4350.000000	1.000000	1.000000	69.000000	1300.000000	2.000000	3.000000	19.000000	1000.000000
25%	8450.000000	44.000000	43000.000000	90.000000	1400.000000	3.000000	5.000000	69.000000	1040.000000
50%	9900.000000	61.000000	63451.000000	110.000000	1600.000000	4.000000	5.000000	85.000000	1070.000000
75%	11950.000000	70.000000	87041.500000	110.000000	1600.000000	5.000000	5.000000	85.000000	1085.000000
max	32500.000000	80.000000	243000.000000	192.000000	16000.000000	5.000000	6.000000	283.000000	1615.000000

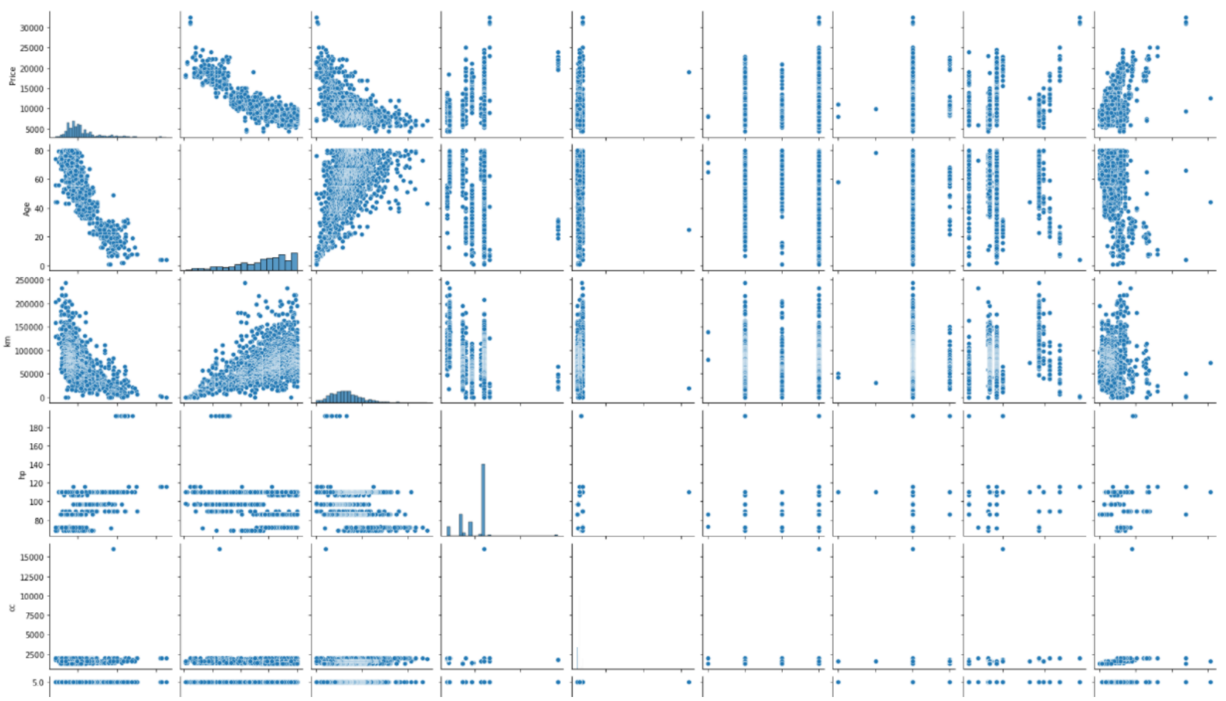
—------------------Correlation Analysis—-----------------------------

drop.corr()

	Price	Age	km	hp	cc	Doors	gears	QT	WT
Price	1.000000	-0.876273	-0.569420	0.314134	0.124375	0.183604	0.063831	0.211508	0.575869
Age	-0.876273	1.000000	0.504575	-0.155293	-0.096549	-0.146929	-0.005629	-0.193319	-0.466484
km	-0.569420	0.504575	1.000000	-0.332904	0.103822	-0.035193	0.014890	0.283312	-0.023969
hp	0.314134	-0.155293	-0.332904	1.000000	0.035207	0.091803	0.209642	-0.302287	0.087143
cc	0.124375	-0.096549	0.103822	0.035207	1.000000	0.079254	0.014732	0.305982	0.335077
Doors	0.183604	-0.146929	-0.035193	0.091803	0.079254	1.000000	-0.160101	0.107353	0.301734
gears	0.063831	-0.005629	0.014890	0.209642	0.014732	-0.160101	1.000000	-0.005125	0.021238
QT	0.211508	-0.193319	0.283312	-0.302287	0.305982	0.107353	-0.005125	1.000000	0.621988
WT	0.575869	-0.466484	-0.023969	0.087143	0.335077	0.301734	0.021238	0.621988	1.000000

—-------------------Pairplot visualization—---------------------------

sns.pairplot(drop)



—-------------------Model Building—-------------------------

model = smf.ols ('Price~Age+km+hp+cc+Doors+gears+QT+WT', data=drop).fit()

—-------------------Find Parameters—--------------------

model.params

```
Intercept     -5472.540368
Age            -121.713891
km               -0.020737
hp               31.584612
cc               -0.118558
Doors            -0.920189
gears           597.715894
QT                3.858805
WT               16.855470
dtype: float64
```

—-----------Find T Values and P Values—---------------
print (model.tvalues, '\n', model.pvalues)

```
Intercept     -3.875273
Age          -46.551876
km           -16.552424
hp            11.209719
cc            -1.316436
Doors         -0.023012
gears          3.034563
QT             2.944198
WT            15.760663
dtype: float64
 Intercept      1.113392e-04
Age          1.879217e-288
km            1.994713e-56
hp            5.211155e-28
cc            1.882393e-01
Doors         9.816443e-01
gears         2.452430e-03
QT            3.290363e-03
WT            1.031118e-51
dtype: float64
```

—----------------Find R^2 Values—----------------------
print (model.rsquared, model.rsquared_adj)

0.8625200256947 0.8617487495415146

—-----------Find T values and P values for [Price,cc]--------------
mlv = smf.ols ('Price~cc',data=drop).fit()
print (mlv.tvalues, '\n', mlv.pvalues)

```
Intercept     24.879592
cc             4.745039
dtype: float64
 Intercept     7.236022e-114
cc             2.292856e-06
dtype: float64
```

—-----------Find T values and P values for [Price,Doors]--------------
nlv = smf.ols ('Price~Doors',data=drop).fit()
print (nlv.tvalues, '\n', nlv.pvalues)

```
Intercept     19.421546
Doors          7.070520
dtype: float64
 Intercept     8.976407e-75
Doors         2.404166e-12
dtype: float64
```

—-----------Find T values and P values for [Price,cc,Doors]---------------
hlv = smf.ols ('Price~cc+Doors',data=drop).fit()
print (hlv.tvalues, '\n', hlv.pvalues)

```
Intercept    12.786341
cc            4.268006
Doors         6.752236
dtype: float64
 Intercept    1.580945e-35
cc           2.101878e-05
Doors        2.109558e-11
dtype: float64
```

—--------------------Find R^2 values—-------------------------------------
a = smf.ols ('Age~km+hp+cc+Doors+gears+QT+WT', data=drop).fit().rsquared
a1 = 1/(1-a)
a1
1.8762358497682887

In [ ]:

b = smf.ols ('km~Age+hp+cc+Doors+gears+QT+WT',data=drop).fit().rsquared
b1=1/(1-b)
b1
1.75717802398104

In [ ]:

c = smf.ols ('hp~Age+km+cc+Doors+gears+QT+WT',data=drop).fit().rsquared
c1=1/(1-c)
c1
1.4191801087182137

In [ ]:

d = smf.ols ('cc~Age+km+hp+Doors+gears+QT+WT', data = drop).fit().rsquared
d1=1/(1-d)
d1
1.163470364594085

e = smf.ols ('Doors~Age+km+hp+cc+gears+QT+WT',data=drop).fit().rsquared
e1=1/(1-e)
e1
1.1558898658142074

In [ ]:

f = smf.ols ('gears~Age+km+hp+cc+Doors+QT+WT',data=drop).fit().rsquared
f1=1/(1-f)
f1
1.098842908163115

g = smf.ols('QT~Age+km+hp+cc+Doors+gears+WT',data=drop).fit().rsquared
g1=1/(1-g)
g1
2.2953745089857147

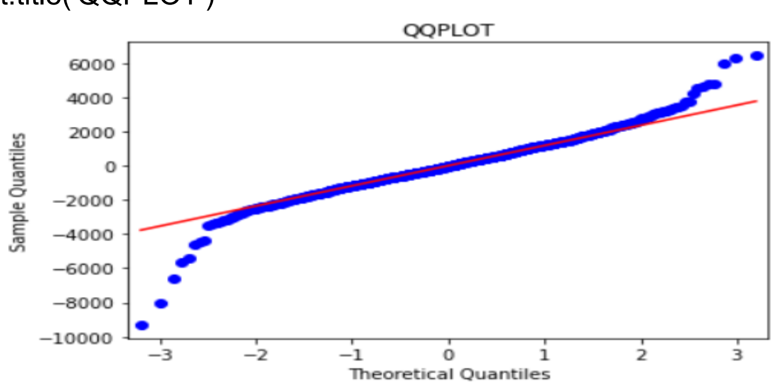
In [ ]:

h = smf.ols('WT~Age+km+hp+cc+Doors+gears+QT',data=drop).fit().rsquared
h1=1/(1-h)
h1
2.4871800071791856

—----------------------Put all R^2 values in dataframe—----------------
list = {'Variables':['Age','km','hp','cc','Doors','gears','QT','WT'],'Values':[a,b,c,d,e,f,g,h]}
simple=pd.DataFrame (list)
simple

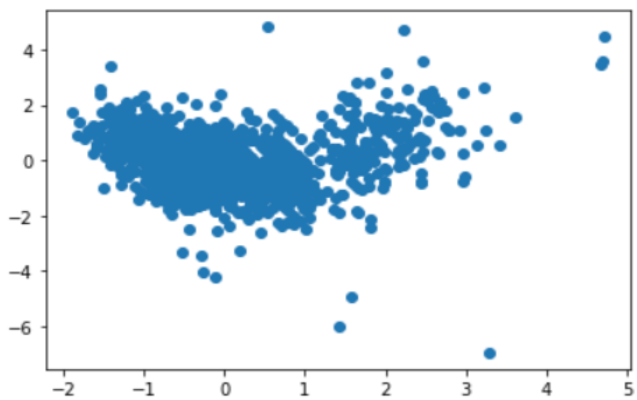
	Variables	Values
0	Age	1.876236
1	km	1.757178
2	hp	1.419180
3	cc	1.163470
4	Doors	1.155890
5	gears	1.098843
6	QT	2.295375
7	WT	2.487180

—-----------------------QQPLOT—-------------------------------------
sm.qqplot(model.resid, line='q')
plt.title('QQPLOT')



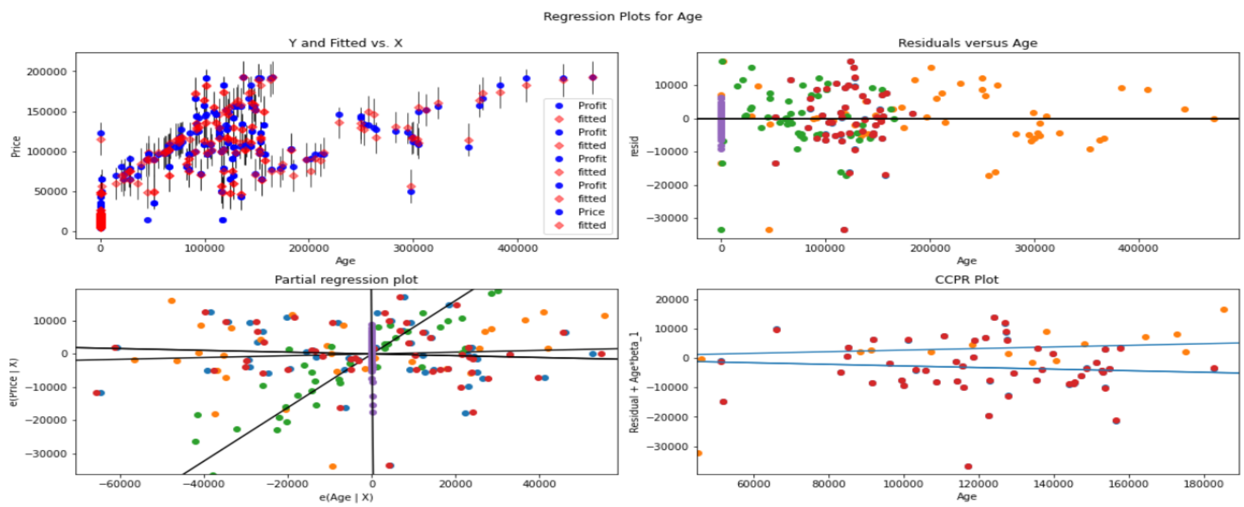
—-----------------------Residual Plot—-------------------------------------

```
def standard_values(vals) : return (vals-vals.mean())/vals.std()
plt.scatter(standard_values(model.fittedvalues),standard_values(model.resid))
```



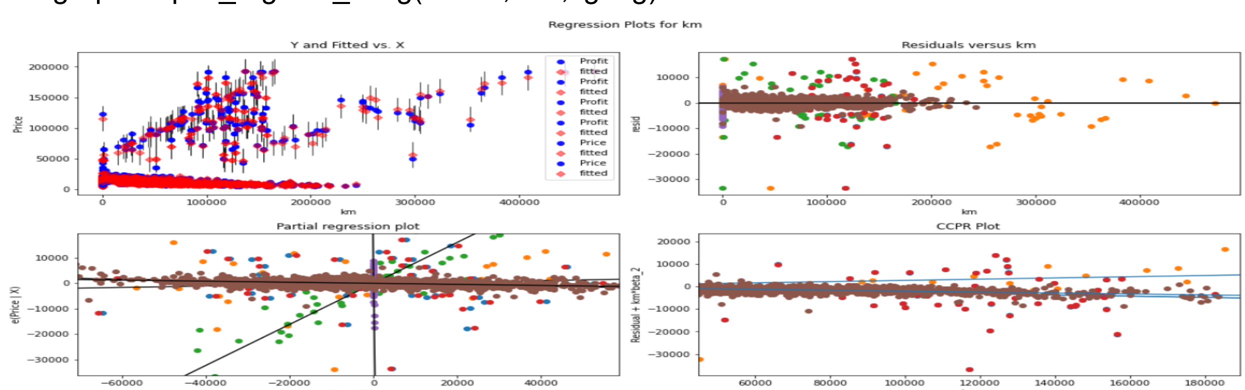
—-------------Regression Plot for Age—------------------------------

```
sm.graphics.plot_regress_exog(model,'Age',fig=fig)
```



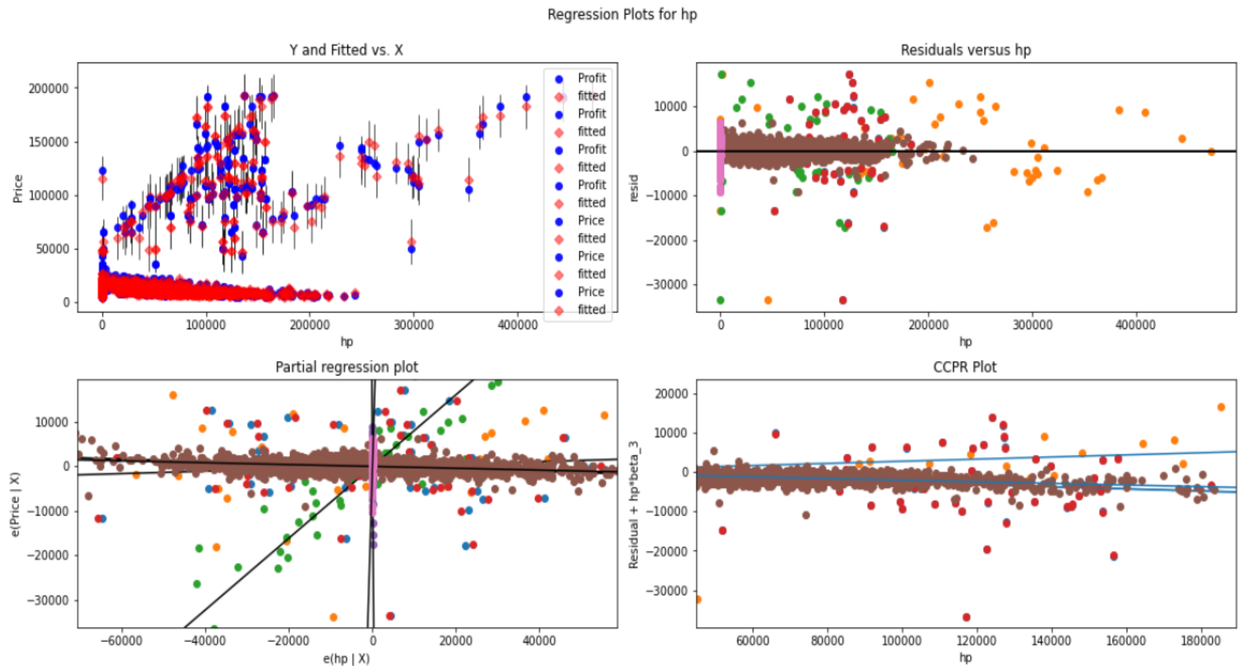
—-----------------------Regression Plot for km—-------------------------------

```
sm.graphics.plot_regress_exog(model,'km',fig=fig)
```



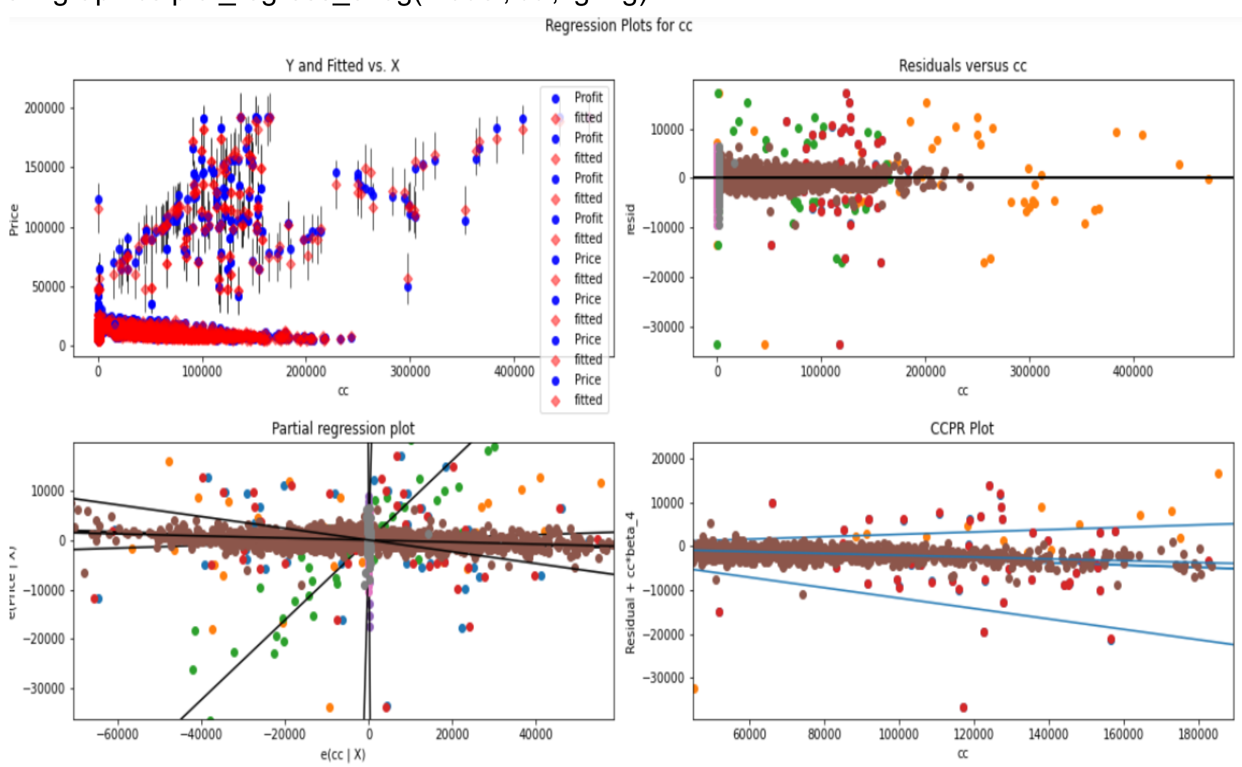
—--------------------------Regression Plot for HP—--------------------------------

sm.graphics.plot_regress_exog(model,'hp',fig=fig)



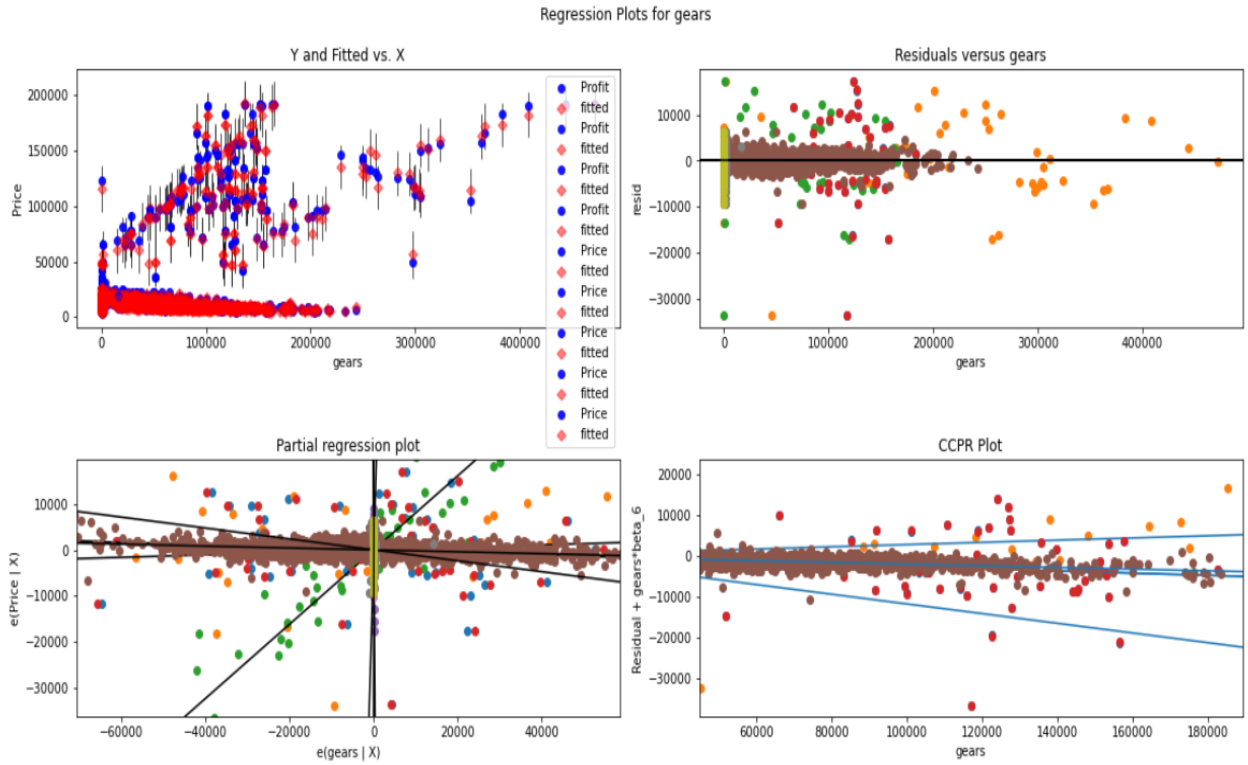
—-------------------------------Regression Plot for cc—---------------------------------------

sm.graphics.plot_regress_exog(model,'cc',fig=fig)



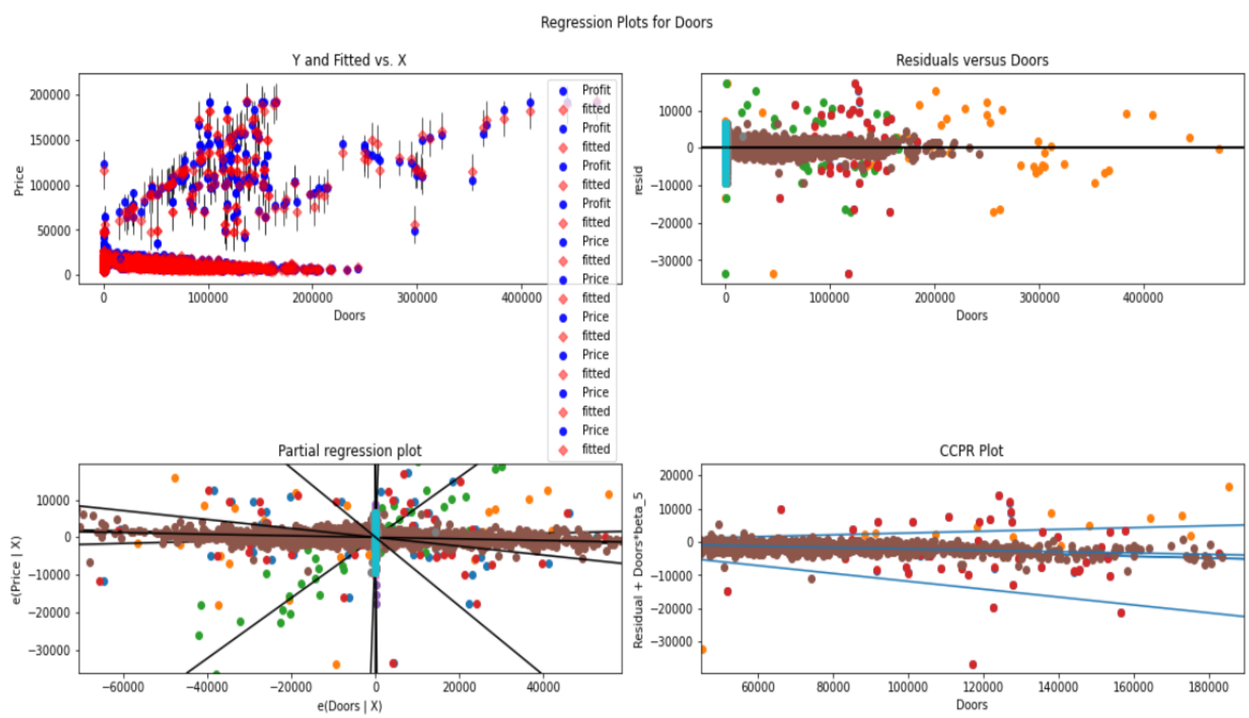
—-----------------------Regression plot for gears—---------------------------------------

sm.graphics.plot_regress_exog(model,'gears',fig=fig)



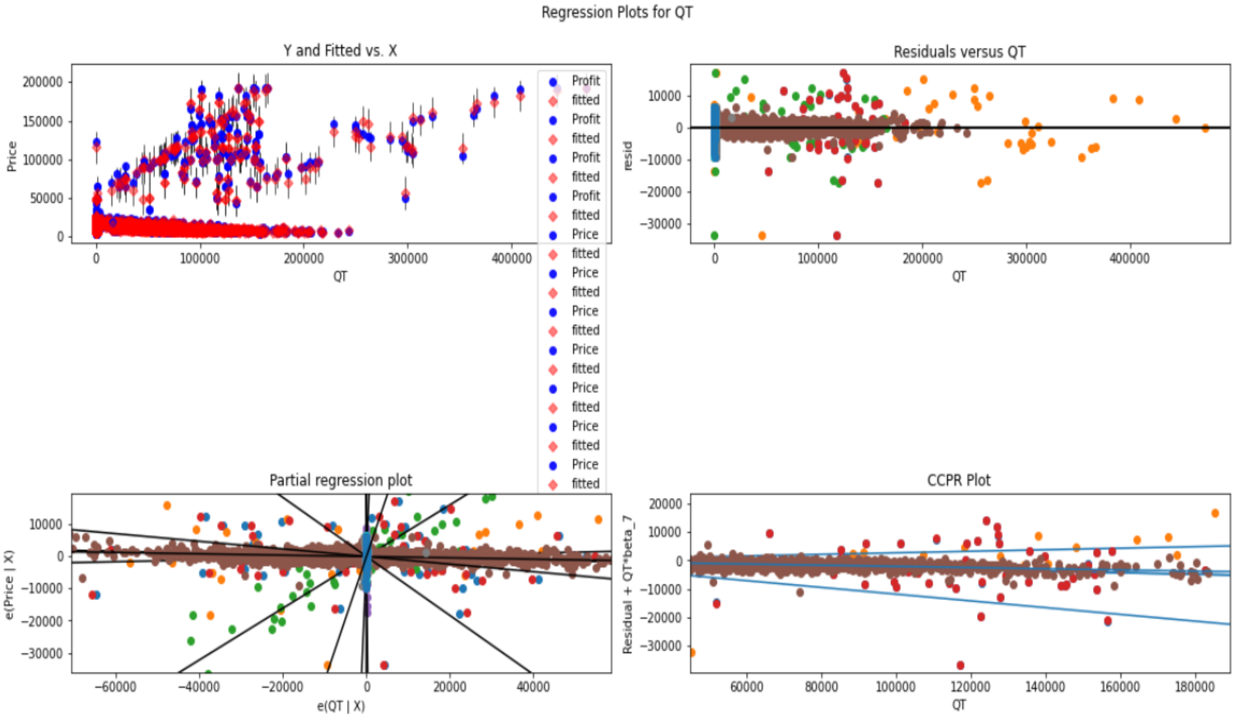
—--------------------------Regression Plot for doors—------------------------

sm.graphics.plot_regress_exog(model,'Doors',fig=fig)



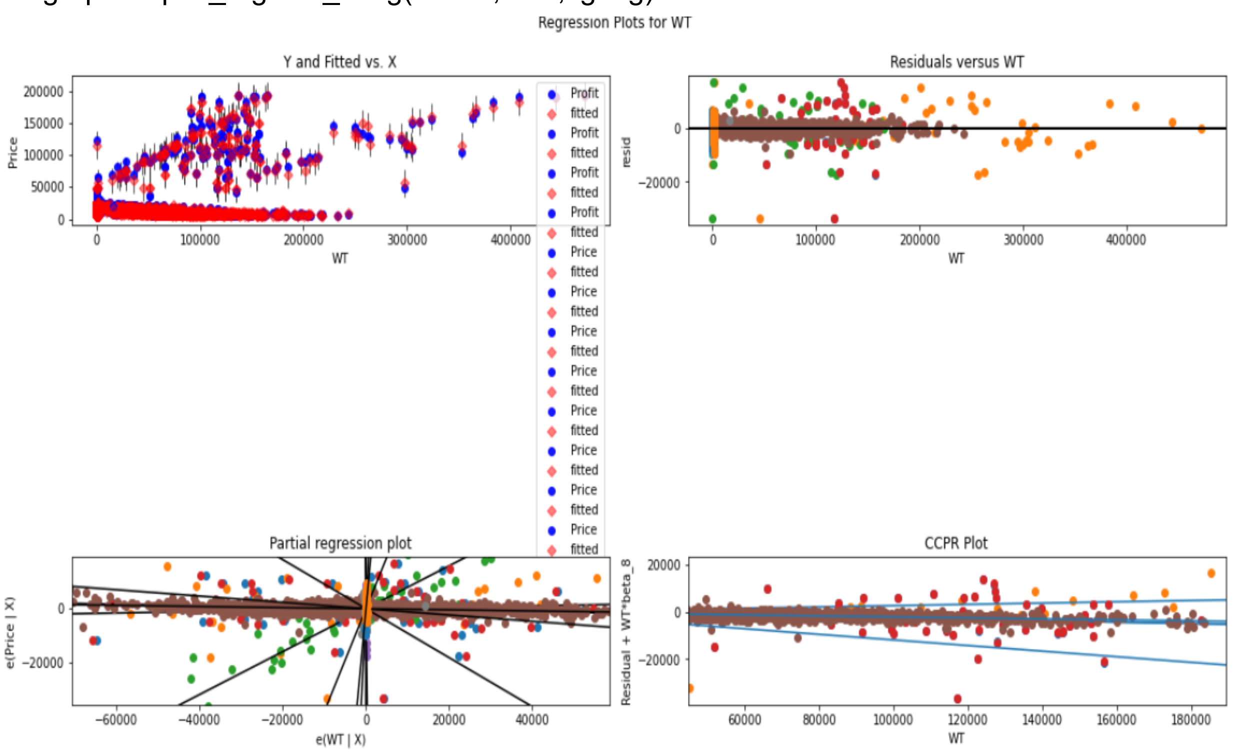
—-------------------Regression plot for QT—---------------------------------

sm.graphics.plot_regress_exog(model,'QT',fig=fig)



—--------------------------Regression plot for WT—--------------------------------------

sm.graphics.plot_regress_exog(model,'WT',fig=fig)

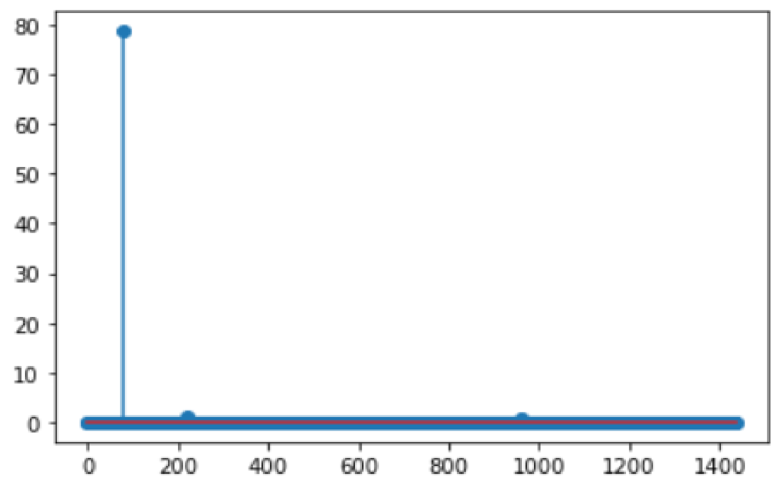


—------------------Cooks Distance—---------------------

model_influence = model.get_influence()
(c,_) = model_influence.cooks_distance
C
array([7.22221054e-03, 3.94547973e-03, 5.44224039e-03, ...,
      8.04110550e-07, 6.99854767e-04, 1.08408002e-02])

—------------------Plotting using stem function—---------------------

plt.stem(np.arange(len(drop)),np.round(c,5))
fig = plt.figure (figsize=(16,8))



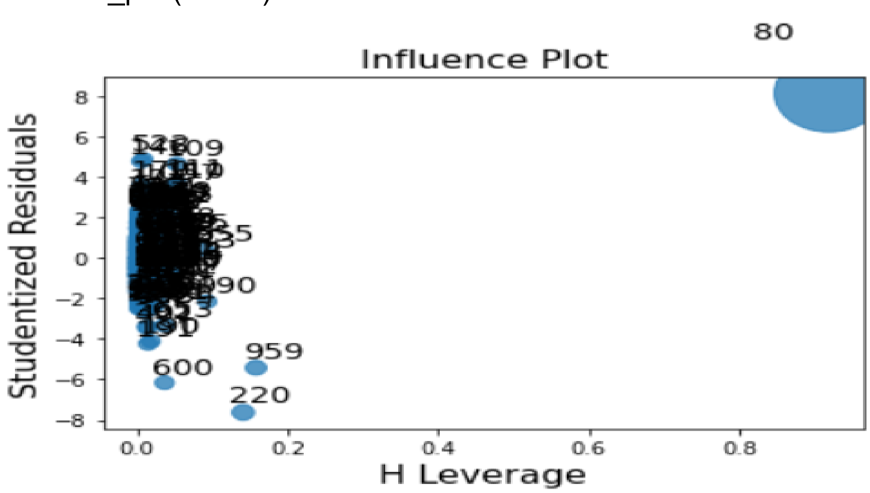
<Figure size 1152x576 with 0 Axes>

np.argmax(c), np.max(c)

(80, 78.7295058224916)

—-----------Influence Plot—-------------------------------

influence_plot(model)



—---------------Leverage value—-------------------

```
k=drop.shape[1]
n=drop.shape[0]
leverage = 3*(k+1)/n
leverage
```

—-------------------See 80th row—--------------------------

```
drop[drop.index.isin([80])]
```

	Price	Age	km	hp	cc	Doors	gears	QT	WT
80	18950	25	20019	110	16000	5	5	100	1180

—---------------Make a copy—----------------------------

```
drop.copy()
```

	Price	Age	km	hp	cc	Doors	gears	QT	WT
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1430	7500	69	20544	86	1300	3	5	69	1025
1431	10845	72	19000	86	1300	3	5	69	1015
1432	8500	71	17016	86	1300	3	5	69	1015
1433	7250	70	16916	86	1300	3	5	69	1015
1434	6950	76	1	110	1600	5	5	19	1114

1435 rows × 9 columns

—---------------Drop Column—-----------------------

```
drrop = drop.drop(drop.index[[80]],axis=0).reset_index(drop=True)
drrop
```

	Price	Age	km	hp	cc	Doors	gears	QT	WT
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1429	7500	69	20544	86	1300	3	5	69	1025
1430	10845	72	19000	86	1300	3	5	69	1015
1431	8500	71	17016	86	1300	3	5	69	1015
1432	7250	70	16916	86	1300	3	5	69	1015
1433	6950	76	1	110	1600	5	5	19	1114

1434 rows × 9 columns

—-------------Model Diagnosis—------------------------------------

```
while np.max(c)>0.5:
    model=smf.ols('p~a+km+hp+cc+doors+gears+qt+wt', data=drop).fit()
    (c,_) = model.get_influence().cooks_distance
    c
    np.argmax(c),np.max(c)
    drop=drop.drop(drop.index[[np.argmax(c)]],axis=0).reset_index(drop=True)
    drop
else:
        final_model=smf.ols('p~a+km+hp+cc+doors+gears+qt+wt',data=drop).fit()
        final_model.rsquared,final_model.aic
        print('Final result is', final_model.rsquared)
```

```
Final result is 0.8882395145171204
```

—----------------Model Prediction—----------------------------

```
model.predict(drop)
```

```
0       16326.634426
1       15886.220972
2       16304.093367
3       15973.237208
4       15839.043084
            ...
1426     9114.821644
1427     8499.169594
1428     8644.902871
1429     8758.662855
1430    10638.570082
Length: 1431, dtype: float64
```