Steps for Regression Model with Statsmodels

Basic steps of Regression Models in Statsmdels Library

Import Library

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

Import CSV as DataFrame

Use URL of file directly

```
df =
pd.read_csv(r'https://github.com/YBI-Foundation/Dataset/raw/main/Fish.
csv')
```

or use localfile path in Jupyter Notebook

```
# df = pd.read_csv(r'C:\Users\YBI-Foundation\Desktop\Fish.csv')
```

or use file path after uploading in Google Colab Notebook

```
#df = pd.read_csv(r'/content/Fish.csv')
```

Get the First Rows of Dataframe

```
df.head()
```

```
   Category Species  Weight   Height   Width  Length1  Length2
Length3
0         1   Bream   242.0  11.5200  4.0200     23.2     25.4
30.0
1         1   Bream   290.0  12.4800  4.3056     24.0     26.3
31.2
2         1   Bream   340.0  12.3778  4.6961     23.9     26.5
31.1
3         1   Bream   363.0  12.7300  4.4555     26.3     29.0
33.5
4         1   Bream   430.0  12.4440  5.1340     26.5     29.0
34.0
```

Get Information of DataFrame

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Category  159 non-null    int64
 1   Species   159 non-null    object
 2   Weight    159 non-null    float64
 3   Height    159 non-null    float64
 4   Width     159 non-null    float64
 5   Length1   159 non-null    float64
 6   Length2   159 non-null    float64
 7   Length3   159 non-null    float64
dtypes: float64(6), int64(1), object(1)
memory usage: 10.1+ KB
```

Get the Summary Statistics

```
df.describe()
```

```
         Category       Weight      Height       Width     Length1  \
count  159.000000   159.000000  159.000000  159.000000  159.000000
mean     3.264151   398.326415    8.970994    4.417486   26.247170
std      1.704249   357.978317    4.286208    1.685804    9.996441
min      1.000000     0.000000    1.728400    1.047600    7.500000
25%      2.000000   120.000000    5.944800    3.385650   19.050000
50%      3.000000   273.000000    7.786000    4.248500   25.200000
75%      4.500000   650.000000   12.365900    5.584500   32.700000
max      7.000000  1650.000000   18.957000    8.142000   59.000000

          Length2     Length3
count  159.000000  159.000000
mean    28.415723   31.227044
std     10.716328   11.610246
min      8.400000    8.800000
25%     21.000000   23.150000
50%     27.300000   29.400000
75%     35.500000   39.650000
max     63.400000   68.000000
```

Get Shape of DataFrame

```
df.shape
```

```
(159, 8)
```

Get Column Names

```
df.columns
```

```
Index(['Category', 'Species', 'Weight', 'Height', 'Width', 'Length1',
       'Length2', 'Length3'],
      dtype='object')
```

Define y(dependent or label or target variable) and X)independent or features or attribute Variable)

```
y = df['Weight']
```

```
y.shape
```

```
(159,)
```

```
y
```

```
0      242.0
1      290.0
2      340.0
3      363.0
4      430.0
       ...
154     12.2
155     13.4
156     12.2
157     19.7
158     19.9
Name: Weight, Length: 159, dtype: float64
```

```
x = df[['Height','Width','Length1','Length2','Length3']]
```

or use .drop function to define x

```
x = df.drop(['Category','Species','Weight'],axis = 1)
```

```
x.shape
```

```
(159, 5)
```

```
x
```

```
      Height   Width  Length1  Length2  Length3
0    11.5200  4.0200     23.2     25.4     30.0
1    12.4800  4.3056     24.0     26.3     31.2
2    12.3778  4.6961     23.9     26.5     31.1
3    12.7300  4.4555     26.3     29.0     33.5
4    12.4440  5.1340     26.5     29.0     34.0
..       ...     ...      ...      ...      ...
154   2.0904  1.3936     11.5     12.2     13.4
155   2.4300  1.2690     11.7     12.4     13.5
156   2.2770  1.2558     12.1     13.0     13.8
157   2.8728  2.0672     13.2     14.3     15.2
158   2.9322  1.8792     13.8     15.0     16.2
```

```
[159 rows x 5 columns]
```

Add Constant to Features (X) for Intercept Estimation

```
import statsmodels.api as sm
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/
_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use
the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

```
x = sm.add_constant(x)
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/
tsatools.py:117: FutureWarning: In a future version of pandas all
arguments of concat except for the argument 'objs' will be keyword-
only
  x = pd.concat(x[::order], 1)
```

```
x.head()
```

	const	Height	Width	Length1	Length2	Length3
0	1.0	11.5200	4.0200	23.2	25.4	30.0
1	1.0	12.4800	4.3056	24.0	26.3	31.2
2	1.0	12.3778	4.6961	23.9	26.5	31.1
3	1.0	12.7300	4.4555	26.3	29.0	33.5
4	1.0	12.4440	5.1340	26.5	29.0	34.0

Get Train Test Split

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size =
0.3,random_state = 2529)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((111, 6), (48, 6), (111,), (48,))
```

Get Model Train

You can choose machine learning models as per requirement

Linear models with independently and identically distributed errors,and for errrors with heteroscedically or autocorrealation.This module allows estimation by ordinary least squares(WLS),generated leastsquares(GLS),and feasible generalized least squares with autocorrelated AR(P=p) errors.

```
import statsmodels.api as sm
```

```
model = sm.OLS(y_train,x_train).fit()
```

Get Model Prediction

```
y_pred = model.predict(x_test)
```

```
y_pred
```

```
6      485.768263
54     502.247209
80      94.723820
138    876.571171
91     184.078918
48     219.301305
52     322.325322
103    376.223260
57     372.357305
149   -182.675371
153   -160.604868
108    454.335862
90     159.597558
118    843.485252
131    587.216806
100    299.535214
15     597.729508
46     197.146054
132    639.890467
79      91.200679
64     150.954248
35    -103.083206
133    627.197128
116    795.691769
31     814.687330
146   -204.149651
53     329.987469
28     715.892880
1      359.756344
117    792.324392
9      532.703671
12     552.008323
129    433.484727
111    687.617503
147   -204.763625
125    932.536683
120    810.742342
158    -80.062172
51     284.362879
34     907.080360
23     642.582834
127    959.338482
```

```
21     675.287923
113    718.863055
109    623.898492
101    376.483470
10     530.838281
157    -86.235707
dtype: float64
```

Get Model Evaluation

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
mean_absolute_percentage_error, r2_score

mean_squared_error(y_test,y_pred)

16397.34452441141

mean_absolute_error(y_test,y_pred)

103.02952922678567

mean_absolute_percentage_error(y_test,y_pred)

2.508285347160016

r2_score(y_test,y_pred)

0.8349141424416875
```

Get Model Summary

```
print(model.summary())
```

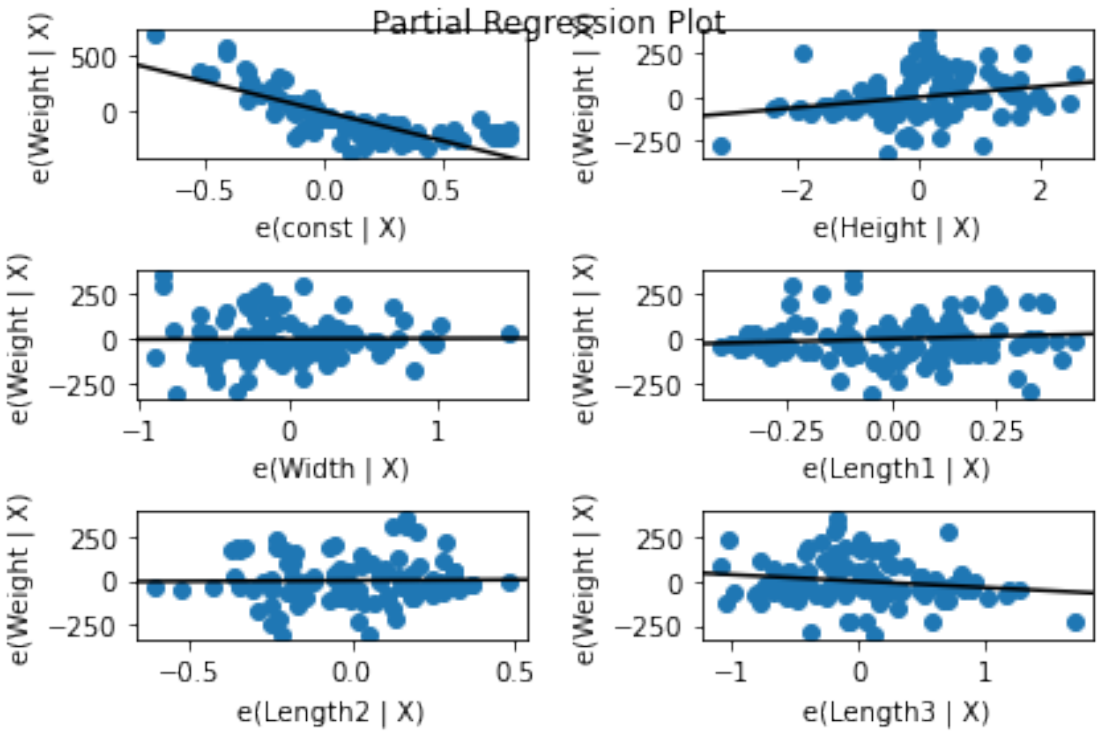
```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                 Weight   R-squared:
0.896
Model:                            OLS   Adj. R-squared:
0.891
Method:                 Least Squares   F-statistic:
181.2
Date:                Sat, 23 Apr 2022   Prob (F-statistic):
5.84e-50
Time:                        13:03:30   Log-Likelihood:
-689.20
No. Observations:                 111   AIC:
1390.
Df Residuals:                     105   BIC:
1407.
```

```
Df Model:                           5

Covariance Type:            nonrobust

==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -519.2834     34.659    -14.983      0.000    -588.005    -450.562
Height        29.8643     10.826      2.759      0.007       8.398      51.330
Width          2.2594     26.105      0.087      0.931     -49.502      54.020
Length1       58.3379     52.151      1.119      0.266     -45.068     161.743
Length2        8.5339     51.806      0.165      0.869     -94.189     111.256
Length3      -36.1521     21.444     -1.686      0.095     -78.671       6.367
==============================================================================
Omnibus:                        5.384   Durbin-Watson:                   2.008
Prob(Omnibus):                  0.068   Jarque-Bera (JB):                4.993
Skew:                           0.391   Prob(JB):                       0.0824
Kurtosis:                       3.684   Cond. No.                         331.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

***Note : Model has insignificant variables and hence need remodeling**

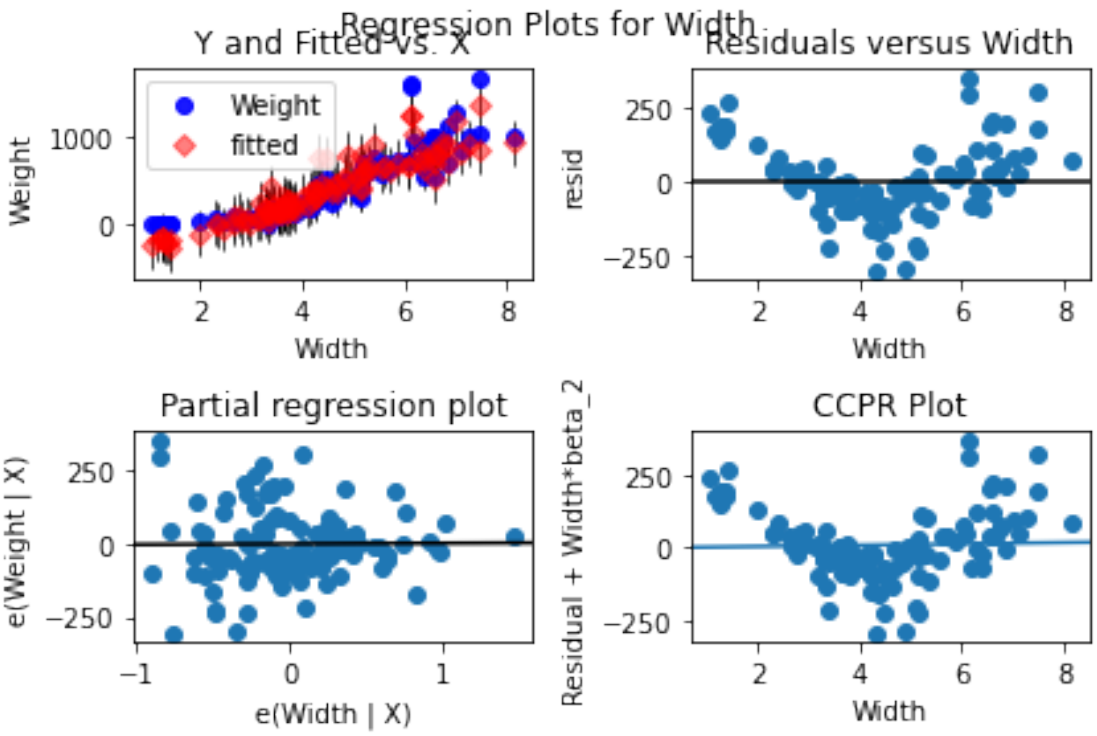
For a quick check of all the regressors,you can use plot_partregress_grid.These plots will not label the points,but you can use them to identify problems and then use plot_partregress to get more information.

```
fig =sm.graphics.plot_partregress_grid(model)
```



The plot_regrress_exog function is a convenience function that gives a 2x2 plot containing the dependent variable and fitted values with confidence intervals vs. theindependent variable chosen,the residuals of the model vs.the chosen independent variable,a partial regression pot,and a CCPR plot. This function can be used for quickly checking modeling assumptions with respect to a single regressor.

```
fig = sm.graphics.plot_regress_exog(model,"Width")
```



Get Future Predictions

Lets select a random sample from existing dataset as new value

Steps to follow

1.Extract a random row using sample function

2.Separate x and y

3.Predict

```
df_new = df.sample(1)
```

```
df_new
```

```
     Category Species  Weight   Height   Width  Length1  Length2
Length3
141         4    Pike  1250.0  10.6863  6.9849     52.0     56.0
59.7
```

```
x_new = df_new[['Height','Width','Length1','Length2','Length3']]
```

```
x_new = sm.add_constant(x_new,has_constant= 'add')
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/
tsatools.py:117: FutureWarning: In a future version of pandas all
arguments of concat except for the argument 'objs' will be keyword-
```

```
only
  x = pd.concat(x[::order], 1)

x_new

     const   Height   Width  Length1  Length2  Length3
141    1.0  10.6863  6.9849     52.0     56.0     59.7

x_new.shape

(1, 6)

y_pred_new = model.predict(x_new)

y_pred_new

141    1168.825036
dtype: float64
```