Python Tutorial 3

February 3, 2022

This tutorial is for Prof. Xin Tong's DSO 530 class at the University of Southern California in spring 2022. It provides some supplementary code for *Lecture 2b: Multiple Linear Regression*.

1 Supplementary Part of Multiple Linear Regression

We still use the same *Housing* dataset.

[1]:
```
import pandas as pd
housing  = pd.read_csv("housing.csv"); display(housing.head())
```

	crim	zn	river	rm	ptratio	medv
0	0.00632	18.0	0	6.575	15.3	24.0
1	0.02731	0.0	0	6.421	17.8	21.6
2	0.02729	0.0	0	7.185	17.8	34.7
3	0.03237	0.0	0	6.998	18.7	33.4
4	0.06905	0.0	0	7.147	18.7	36.2

1.1 Simple Linear Regression

We will start by using the `smf.ols()` function to fit a simple linear regression model, with `medv` as the response and `crim` as the predictor. The basic syntax is `smf.ols('y ~ x', data)`,where `y` is the response, `x` is the predictor, and `data` is the data set in which these two variables are kept.

P.S. `smf.ols()` function can takes in data as *Pandas Dataframes*.

[2]:
```
import statsmodels.formula.api as smf

result1 = smf.ols('medv ~ crim', data=housing).fit()
```

We use `results.summary()` to output some detailed informaton about the model.

[3]:
```
result1.summary()
```

[3]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.151
Model:                            OLS   Adj. R-squared:                  0.149
```

```
Method:                 Least Squares   F-statistic:                     89.49
Date:                Thu, 03 Feb 2022   Prob (F-statistic):           1.17e-19
Time:                        16:31:38   Log-Likelihood:                -1798.9
No. Observations:                 506   AIC:                             3602.
Df Residuals:                     504   BIC:                             3610.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     24.0331      0.409     58.740      0.000      23.229      24.837
crim          -0.4152      0.044     -9.460      0.000      -0.501      -0.329
==============================================================================
Omnibus:                      139.832   Durbin-Watson:                   0.713
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              295.404
Skew:                           1.490   Prob(JB):                     7.14e-65
Kurtosis:                       5.264   Cond. No.                         10.1
==============================================================================

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

1.2 Multiple Regression

In order to fit a multiple linear regression model using least squares, we again use the `smf.ols()` function. The syntax `smf.ols('y ~ x1+x2+x3', data)` is used to fit a model with three predictors, `x1`, `x2`,and `x3`. The `summary()` function now outputs the regression coefficients for all the predictors.

[4]:
```
result2 = smf.ols('medv ~ crim+rm', data=housing).fit()
result2.summary()
```

[4]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.542
Model:                            OLS   Adj. R-squared:                  0.540
Method:                 Least Squares   F-statistic:                     297.6
Date:                Thu, 03 Feb 2022   Prob (F-statistic):           5.22e-86
Time:                        16:31:38   Log-Likelihood:                -1642.7
No. Observations:                 506   AIC:                             3291.
Df Residuals:                     503   BIC:                             3304.
Df Model:                           2
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -29.2447      2.588    -11.300      0.000     -34.330     -24.160
crim          -0.2649      0.033     -8.011      0.000      -0.330      -0.200
rm             8.3911      0.405     20.726      0.000       7.596       9.186
==============================================================================
Omnibus:                      172.412   Durbin-Watson:                   0.807
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1047.536
Skew:                           1.349   Prob(JB):                    3.39e-228
Kurtosis:                       9.512   Cond. No.                         92.3
==============================================================================

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

The housing data set contains 5 covariates so it would be cumbersome to have to type all of these in order to perform a regression using all of the predictors. Instead, we can use the following short-hand:

[5]:
```python
string_cols = ' + '.join(housing.columns[:-1])
result3 = smf.ols('medv ~ {}'.format(string_cols), data=housing).fit()
result3.summary()
```

[5]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.605
Model:                            OLS   Adj. R-squared:                  0.601
Method:                 Least Squares   F-statistic:                     153.3
Date:                Thu, 03 Feb 2022   Prob (F-statistic):           1.76e-98
Time:                        16:31:39   Log-Likelihood:                -1605.1
No. Observations:                 506   AIC:                             3222.
Df Residuals:                     500   BIC:                             3248.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.8294      4.014     -1.203      0.230     -12.716       3.057
crim          -0.1981      0.032     -6.237      0.000      -0.260      -0.136
zn             0.0277      0.012      2.230      0.026       0.003       0.052
river          3.3018      1.033      3.196      0.001       1.272       5.332
rm             7.1443      0.405     17.627      0.000       6.348       7.941
```

```
ptratio       -0.9409      0.138     -6.800      0.000      -1.213      -0.669
==============================================================================
Omnibus:                      220.975   Durbin-Watson:                   0.883
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1796.516
Skew:                           1.703   Prob(JB):                         0.00
Kurtosis:                      11.580   Cond. No.                         436.
==============================================================================

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

We could see that here we use *string_cols = ' + '.join(boston.columns[:-1])* to get all the variable with correct format which would be used in `smf.ols` except the target `medv`.

str.format() is one of the string formatting methods in Python3, which allows multiple substitutions and value formatting. This method lets us concatenate elements within a string through positional formatting. If you never use that before, you can see more details on the following webpage: https://www.geeksforgeeks.org/python-format-function/

[6]:
```python
print(string_cols)
```

```
crim + zn + river + rm + ptratio
```

What if we would like to perform a regression using all of the variables but one? For example, in the above regression output, `zn` has the highest p-value (still small though). The following syntax results in a regression using all predictors except `zn`.

[7]:
```python
string_cols = ' + '.join(housing.columns[:-1].difference(['zn']))
result4 = smf.ols('medv ~ {}'.format(string_cols), data=housing).fit()
result4.summary()
```

[7]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.601
Model:                            OLS   Adj. R-squared:                  0.598
Method:                 Least Squares   F-statistic:                     188.9
Date:                Thu, 03 Feb 2022   Prob (F-statistic):           1.42e-98
Time:                        16:31:39   Log-Likelihood:                -1607.6
No. Observations:                 506   AIC:                             3225.
Df Residuals:                     501   BIC:                             3246.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
Intercept     -3.8668      4.007     -0.965      0.335     -11.739       4.005
crim          -0.2035      0.032     -6.402      0.000      -0.266      -0.141
ptratio       -1.0345      0.132     -7.816      0.000      -1.295      -0.774
river          3.0411      1.031      2.951      0.003       1.016       5.066
rm             7.3223      0.399     18.354      0.000       6.538       8.106
==============================================================================
Omnibus:                      215.939   Durbin-Watson:                   0.887
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1751.602
Skew:                           1.655   Prob(JB):                         0.00
Kurtosis:                      11.493   Cond. No.                         311.
==============================================================================

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

Here, we use `difference` to exclude `zn`. The function *difference()* returns a set that is the difference between two sets. For example, if A = *{100, 60}* and B = *{60, 20}*. Then A.*difference(B)* = {100} and B.*difference(A)* = {20}.

[8]:
```
string_cols = ' + '.join(housing.columns[:-1].difference(['zn']))
print(string_cols)
```

```
crim + ptratio + river + rm
```

1.3 Interaction Terms

It is easy to include interaction terms in a linear model using the `smf.ols()` function. The syntax `x1:x2` tells Python to include an interaction term between `x1` and `x2`.The syntax `river*rm` simultaneously includes `river`, `rm`, and the interaction term `river×rm` as predictors; it is a shorthand for `river+rm+river:rm`.

[9]:
```
result4 = smf.ols('medv ~ river * rm', data=housing).fit() # is same as:␣
↪result4 = smf.ols('medv ~ river+rm+river:rm', data=housing).fit()
result4.summary()
```

[9]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.496
Model:                            OLS   Adj. R-squared:                  0.493
Method:                 Least Squares   F-statistic:                     164.9
Date:                Thu, 03 Feb 2022   Prob (F-statistic):           2.24e-74
Time:                        16:31:39   Log-Likelihood:                -1666.7
No. Observations:                 506   AIC:                             3341.
Df Residuals:                     502   BIC:                             3358.
```

```
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -34.4616      2.776    -12.415      0.000     -39.915     -29.008
river          7.5633      8.871      0.853      0.394      -9.865      24.992
rm             9.0241      0.440     20.496      0.000       8.159       9.889
river:rm      -0.5361      1.355     -0.396      0.692      -3.198       2.125
==============================================================================
Omnibus:                       93.496   Durbin-Watson:                   0.748
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              564.362
Skew:                           0.638   Prob(JB):                    2.82e-123
Kurtosis:                       8.014   Cond. No.                         199.
==============================================================================

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

1.4 Non-linear Transformations of the Predictors

The `smf.ols()` function can also accommodate non-linear transformations of the predictors. For instance, given a predictor X, we can create a predictor X^2 using `np.power(X, 2)`. We now perform a regression of `medv` onto `rm` and $\texttt{rm}^2$.

[10]:
```
import numpy as np
result5 = smf.ols('medv ~ rm + np.power(rm, 2)', data=housing).fit()
result5.summary()
```

[10]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.548
Model:                            OLS   Adj. R-squared:                  0.547
Method:                 Least Squares   F-statistic:                     305.4
Date:                Thu, 03 Feb 2022   Prob (F-statistic):           1.46e-87
Time:                        16:31:39   Log-Likelihood:                -1639.1
No. Observations:                 506   AIC:                             3284.
Df Residuals:                     503   BIC:                             3297.
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
===
                      coef    std err          t      P>|t|      [0.025
```

```
0.975]
--------------------------------------------------------------------------------
---
Intercept          66.0588     12.104      5.458      0.000      42.278
89.839
rm                -22.6433      3.754     -6.031      0.000     -30.019
-15.267
np.power(rm, 2)     2.4701      0.291      8.502      0.000       1.899
3.041
==============================================================================
Omnibus:                       82.173   Durbin-Watson:                   0.689
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              934.337
Skew:                           0.224   Prob(JB):                   1.29e-203
Kurtosis:                       9.642   Cond. No.                     1.91e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.91e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

1.5 Confidence Interval and Prediction Inverval

Here we'd like to talk about how to code to calculate **Confidence Interval** and **Prediction Interval** of the response. The point is that the confidence interval is about an average response and the prediction interval is about a particular response. Note that it is a slight abuse of language to name an interval prediction of the response CI here. But we will resolve the conflict at the end of this section.

We can read off the confidence intervals for the coefficient estimates in `summary()`:

[11]:
```
result1 = smf.ols('medv ~ rm', data=housing).fit()
result1.summary()
```

[11]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.484
Model:                            OLS   Adj. R-squared:                  0.483
Method:                 Least Squares   F-statistic:                     471.8
Date:                Thu, 03 Feb 2022   Prob (F-statistic):           2.49e-74
Time:                        16:31:39   Log-Likelihood:                -1673.1
No. Observations:                 506   AIC:                             3350.
Df Residuals:                     504   BIC:                             3359.
Df Model:                           1
```

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -34.6706      2.650    -13.084      0.000     -39.877     -29.465
rm             9.1021      0.419     21.722      0.000       8.279       9.925
==============================================================================
Omnibus:                      102.585   Durbin-Watson:                   0.684
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              612.449
Skew:                           0.726   Prob(JB):                    1.02e-133
Kurtosis:                       8.190   Cond. No.                         58.4
==============================================================================

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

If we want to produce confidence intervals and prediction intervals for the prediction of `medv` for a given value of `rm`, we can do as follows:

Assume that we want predict for the following given value of `rm`:

[12]:
```python
test_data = {'rm':[5,10,15,20]}
test_data_df = pd.DataFrame(test_data)
test_data_df
```

[12]:

	rm
0	5
1	10
2	15
3	20

We can use `get_prediction()` function to produce confidence intervals and prediction intervals for the prediction.

[13]:
```python
prediction1 = result1.get_prediction(test_data_df)
prediction1.summary_frame(alpha=0.05)
```

[13]:
```
         mean   mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  \
0   10.839924  0.613410       9.634769      12.045079     -2.214474
1   56.350469  1.584377      53.237672      59.463266     42.984303
2  101.861014  3.663795      94.662822     109.059205     87.002385
3  147.371559  5.754624     136.065553     158.677565    130.143945

   obs_ci_upper
0     23.894322
1     69.716635
```

```
2     116.719643
3     164.599173
```

For instance, the 95% confidence interval associated with a `rm` value of 10 is (53.237672, 59.463266), and the 95% prediction interval is (42.984303, 69.716635). As expected, the confidence and prediction intervals are centered around the same point (a predicted value of 56.350469 for `medv` when `rm` equals 10), but the latter are substantially wider.

P.S. The confidence interval here, in this case, is the confidence interval of $\beta_0 + 10\beta_1$.

References:

James, G. , Witten, D. , Hastie, T. , & Tibshirani, R. . (2013). An Introduction to Statistical Learning: With Applications in R.

Müller, Andreas C; Guido, Sarah. (2017). Introduction to Machine Learning with Python.

https://github.com/tdpetrou/Machine-Learning-Books-With-Python

https://scikit-learn.org/stable/index.html

https://www.statsmodels.org/dev/index.html

http://www.science.smith.edu/~jcrouser/SDS293/labs/lab4-py.html