```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Drop NA Version
# dt = pd.read_excel('/content/dropNa_USE_THIS_FINAL.xlsx',index_col = 'OBJECTID')
dt = pd.read_excel('/content/dropNa_USE_THIS_FINAL.xlsx')

# ORG Ver.
# dt = pd.read_excel('/content/USE_THIS_for_Regression_New.xlsx')
#
```

```
dt.head()
```

	Unnamed: 0	Unnamed: 0.1	OBJECTID	STATEFP	COUNTYFP	TRACTCE	BLKGRPCE	GEOID
0	3	3	4	36	5	7500	2	360050075002
1	5	5	6	36	5	12701	2	360050127012
2	6	6	7	36	5	5400	3	360050054003
3	8	8	9	36	5	22401	1	360050224011
4	9	9	10	36	5	7100	2	360050071002

```
list(dt.columns)
```

```
['Unnamed: 0',
 'Unnamed: 0.1',
 'OBJECTID',
 'STATEFP',
 'COUNTYFP',
 'TRACTCE',
 'BLKGRPCE',
 'GEOID',
 'NAMELSAD',
 'MTFCC',
 'FUNCSTAT',
 'ALAND',
 'AWATER',
 'INTPTLAT',
 'INTPTLON',
 'Area',
```

```
      'Number_of_trees',
      'Tree_density',
      'Mean PM 2.5',
      'Temp_Mean',
      'Mean O3',
      'Mean NO',
      'Mean NO2',
      'Shape_Length',
      'Shape_Area',
      'Mean greenview',
      'Count of Points',
      'Geo_FIPS',
      'Geo_NAME',
      'Geo_QName',
      'Geo_STUSAB',
      'Geo_STATE',
      'Geo_COUNTY',
      'Geo_TRACT',
      'Geo_BLKGRP',
      'Geo_GEOID',
      'Geo_AREALAND',
      'Geo_AREAWATR',
      'Total Population',
      'White Alone',
      'Black Alone',
      'Native American Alone',
      'Asian Alone',
      'Pacific Islander Alone',
      'Other Race Alone',
      'Two or More Races',
      'Pct White',
      'Pct Black',
      'Pct Native American',
      'Pct Asian',
      'Pct Pacific Islander',
      'Pct Other',
      'Pct Two or More Races',
      'Population 25 Over',
      'Less than HS',
      'HS or More',
      'Some C or More',
```

```
y_index = ['Temp_Mean','Mean greenview', 'Tree_density',
 'Mean PM 2.5',
 'Temp_Mean',
 'Mean O3',
 'Mean NO',
 'Mean NO2']

#impute missing data in y columns?
from sklearn.impute import SimpleImputer
#imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean')
```

```
x_index = ['Mean greenview','Pct White','Pct Bachelors or More','Pct Unemployed','Ln Incom
x = dt[x_index].values
x
```

```
array([[2.07878907e+01, 2.21600000e+01, 2.29800000e+01, ...,
        1.26847882e+01, 3.57900000e+01, 3.19127523e-02],
       [2.10217262e+01, 2.79100000e+01, 2.41100000e+01, ...,
        1.27390511e+01, 2.88700000e+01, 1.22113874e-02],
       [8.98164933e+00, 9.93000000e+00, 1.51000000e+00, ...,
        1.30619762e+01, 5.27400000e+01, 2.01799612e-02],
       ...,
       [1.31078934e+01, 8.93000000e+01, 2.19900000e+01, ...,
        1.32380108e+01, 7.04000000e+01, 2.58147571e-02],
       [1.27443491e+01, 6.32900000e+01, 2.39500000e+01, ...,
        1.31062340e+01, 1.03300000e+01, 2.65082094e-02],
       [1.96521962e+01, 7.97400000e+01, 1.96300000e+01, ...,
        1.32346201e+01, 1.50700000e+01, 2.45165092e-02]])
```

```
#============================================#
# Simple Multiple Linear Regression [Feature Scaling?]
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
for ele in y_index:

    #process x and y, as there are NAN values in y, and we dont want to drop them
    x_for_y = dt.copy()
    x_for_y = x_for_y.dropna(subset=[ele,'Mean greenview','Pct White','Pct Bachelors or More
    y = x_for_y[ele].values
    x = x_for_y[x_index].values
    #x = x[~np.isnan(x).any(axis=1)]

    #print(np.argwhere(np.isnan(x)))
    #print(np.argwhere(np.isnan(y)))
    #print(x.shape)
    print(f'Obs = {y.shape}')


    regressor.fit(x,y)
    print(f'Current Y: {ele}')
    cof_count = 0
    for cofs in regressor.coef_:
        print(f'Coef {x_index[cof_count]} = {cofs}')
        cof_count += 1
    print(f'Constant = {regressor.intercept_}')
    print(f'R_score = {regressor.score(x,y)}')
    print('\n')
```

```
Coef Pct White = -0.0021912467644557036
Coef Pct Bachelors or More = 0.01309983317855168
Coef Pct Unemployed = 0.00020891592507072104
Coef Ln Income = -0.10557092508305839
Coef Ln Housing Value = 0.21353061769685494
Coef Pct Rent 50 or More = -0.001054861853072057
Coef Population Density = 10.116947910953616
Constant = 4.248898526181845
R_score = 0.35434611490733214


Obs = (4081,)
Current Y: Temp_Mean
Coef Mean greenview = -0.05840433312057996
```

```
Coef Pct White = -0.0057237769922766826
Coef Pct Bachelors or More = -0.01825250097648058
Coef Pct Unemployed = 0.0005077481604984907
Coef Ln Income = 0.19379991800962185
Coef Ln Housing Value = -0.17794003441675385
Coef Pct Rent 50 or More = 0.0037545223542598772
Coef Population Density = -12.341718905565513
Constant = 40.392852411799325
R_score = 0.186356821977948


Obs = (2920,)
Current Y: Mean O3
Coef Mean greenview = -0.018681535866619206
Coef Pct White = 0.00224761388206378
Coef Pct Bachelors or More = -0.015491075001107715
Coef Pct Unemployed = -0.001014777220969748
Coef Ln Income = -0.15998691909417187
Coef Ln Housing Value = -0.11759065107687483
Coef Pct Rent 50 or More = -0.0018705430124355835
Coef Population Density = -8.420645653679571
Constant = 34.140959839614325
R_score = 0.10787204787286908


Obs = (2920,)
Current Y: Mean NO
Coef Mean greenview = -0.10182341432361303
Coef Pct White = -0.010458692410916261
Coef Pct Bachelors or More = 0.040662977050988904
Coef Pct Unemployed = 0.012886505711410137
Coef Ln Income = -0.4638438291704686
Coef Ln Housing Value = 0.8854369926074198
Coef Pct Rent 50 or More = -0.006712531817653218
Coef Population Density = 32.52393364245999
Constant = 4.4480251433834805
R_score = 0.22844240404575478


Obs = (2920,)
Current Y: Mean NO2
Coef Mean greenview = -0.10635225791267663
Coef Pct White = -0.009366456030613951
Coef Pct Bachelors or More = 0.0510578316945225
```

```
#=============================================#
# Simple Multiple Linear Regression [Feature Scaling?]
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
for ele in y_index:

    #process x and y, as there are NAN values in y, and we dont want to drop them
    x_for_y = dt.copy()
    x_for_y = x_for_y.dropna(subset=[ele,'Mean greenview','Pct White','Pct Bachelors or More
    y = x_for_y[ele].values
    x = x_for_y[x_index].values
    #x = x[~np.isnan(x).any(axis=1)]
```

```
    #print(np.argwhere(np.isnan(x)))
    #print(np.argwhere(np.isnan(y)))
    #print(x.shape)
    print(f'Obs = {y.shape}')


    regressor.fit(x,y)
    print(f'Current Y: {ele}')
    cof_count = 0
    for cofs in regressor.coef_:
        print(f'Coef {x_index[cof_count]} = {cofs}')
        cof_count += 1
    print(f'Constant = {regressor.intercept_}')
    print(f'R_score = {regressor.score(x,y)}')
    print('\n')
    N = int(y.shape[0])
    p = 9  # plus one because LinearRegression adds an intercept term

    x_with_intercept = np.empty(shape=(N, p), dtype=np.float)
    x_with_intercept[:, 0] = 1
    x_with_intercept[:, 1:p] = x

    ols = sm.OLS(y, x_with_intercept)
    ols_result = ols.fit()
    result = ols_result.summary()
    print(result)
    print('\n')
    print('+====================================+')
```

```
Coef Pct Bachelors or More = -0.01825250097648058
Coef Pct Unemployed = 0.0005077481604984907
Coef Ln Income = 0.19379991800962185
Coef Ln Housing Value = -0.17794003441675385
Coef Pct Rent 50 or More = 0.0037545223542598772
Coef Population Density = -12.341718905565513
Constant = 40.392852411799325
R_score = 0.186356821977948


                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.186
Model:                            OLS   Adj. R-squared:                  0.185
Method:                 Least Squares   F-statistic:                     116.6
Date:                Sat, 14 May 2022   Prob (F-statistic):          4.04e-176
Time:                        03:32:57   Log-Likelihood:                -7450.6
No. Observations:                4081   AIC:                         1.492e+04
Df Residuals:                    4072   BIC:                         1.498e+04
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------

const         40.3929      0.982     41.122      0.000      38.467      42.319
x1            -0.0584      0.004    -14.239      0.000      -0.066      -0.050
x2            -0.0057      0.001     -6.087      0.000      -0.008      -0.004
x3            -0.0183      0.002    -10.868      0.000      -0.022      -0.015
```

```
x4             0.0005      0.005      0.098      0.922      -0.010       0.011
x5             0.1938      0.076      2.545      0.011       0.045       0.343
x6            -0.1779      0.049     -3.651      0.000      -0.274      -0.082
x7             0.0038      0.001      2.536      0.011       0.001       0.007
x8           -12.3417      1.603     -7.700      0.000     -15.484      -9.199
==============================================================================
Omnibus:                      561.097   Durbin-Watson:                   1.667
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1082.882
Skew:                          -0.862   Prob(JB):                    7.16e-236
Kurtosis:                       4.842   Cond. No.                     5.35e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.35e+03. This might indicate that there are
strong multicollinearity or other numerical problems.


+=====================================+
Obs = (4081,)
Current Y: Mean greenview
Coef Mean greenview = 0.9999999999999998
Coef Pct White = -9.316116465476137e-17
Coef Pct Bachelors or More = 3.758378274528841e-16
Coef Pct Unemployed = -5.607015999705018e-17
Coef Ln Income = 6.796094498376125e-18
Coef Ln Housing Value = -8.640026131948859e-17
Coef Pct Rent 50 or More = -1.7506698079008746e-16
Coef Population Density = 6.925903045554331e-16
```

✓ 0 秒 完成时间：23:32