Workshop - OLS Python

In this workshop, we are going to:

1. perform backward selection on the class data set
   A. fit the full model with $\%\Delta rGDP$ as the label
   B. remove the feature with the highest p-value
   C. refit the model
   D. repeat steps B. and C. until all features have p-values below 0.05
2. evaluatate the model performance

Do not use interactions or polynomial terms in this workshop.

Preliminaries

- Load any necessary packages and/or functions
  - For backward select, I recommend using `statsmodels.api` instead of `statsmodels.formula.api`. Your choice.
- Load in the class data
- Define `x` and `y`
- Create a train-test split with
  - training size of two-thirds
  - random state of 490

In [35]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
```

In [37]:
```
df = pd.read_csv('/Users/liaohaitao/Desktop/ECON 490/Lecture 2.2/class_data.csv')
df.set_index(['fips', 'year', 'GeoName'], inplace = True)
```

```
df
df['year'] = df.index.get_level_values('year')
```

In [36]:
```
df = pd.read_pickle('/Users/liaohaitao/Desktop/ECON 490/Lecture 2.2/class_data.pkl')
df.columns
```

Out[36]:
```
Index(['GeoName', 'pct_d_rgdp', 'urate_bin', 'pos_net_jobs', 'emp_estabs',
       'estabs_entry_rate', 'estabs_exit_rate', 'pop', 'pop_pct_black',
       'pop_pct_hisp', 'lfpr', 'density', 'year'],
      dtype='object')
```

In [38]:
```
y = df['pct_d_rgdp']
x = df.drop(columns = 'pct_d_rgdp')

# Creating dummies
x = x.join([pd.get_dummies(x['year'], prefix = 'year', drop_first = True),
          pd.get_dummies(x['urate_bin'], prefix = 'urate', drop_first = True)]).drop(columns = ['year', 'urate_bin'])
x = sm.add_constant(x)

# Sorting the columns for output
x.sort_index(axis = 'columns', inplace = True)

# Dropping un
x.columns
```

Out[38]:
```
Index(['const', 'density', 'emp_estabs', 'estabs_entry_rate',
       'estabs_exit_rate', 'lfpr', 'pop', 'pop_pct_black', 'pop_pct_hisp',
       'pos_net_jobs', 'urate_lower', 'urate_similar', 'year_2003',
       'year_2004', 'year_2005', 'year_2006', 'year_2007', 'year_2008',
       'year_2009', 'year_2010', 'year_2011', 'year_2012', 'year_2013',
       'year_2014', 'year_2015', 'year_2016', 'year_2017', 'year_2018'],
      dtype='object')
```

Backward Selection

In [39]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 2/3, random_state = 490)
```

In [40]:
```
fit = sm.OLS(y_train, x_train.drop(columns = ['density',
```

```
                                                    'year_2005',
                                                    'pop_pct_black',
                                                    'year_2013',
                                                    'year_2003',
                                                    'pop'])).fit()
print(fit.summary2())
```

```
                  Results: Ordinary least squares
====================================================================
Model:              OLS              Adj. R-squared:     0.041
Dependent Variable: pct_d_rgdp       AIC:                246970.3167
Date:               2021-02-25 18:08 BIC:                247155.7953
No. Observations:   33889            Log-Likelihood:     -1.2346e+05
Df Model:           21               F-statistic:        69.38
Df Residuals:       33867            Prob (F-statistic): 1.65e-289
R-squared:          0.041            Scale:              85.550
--------------------------------------------------------------------
                     Coef.  Std.Err.    t     P>|t|   [0.025  0.975]
--------------------------------------------------------------------
const               -2.1414   0.5048  -4.2422 0.0000 -3.1308 -1.1520
emp_estabs          -0.0434   0.0108  -4.0255 0.0001 -0.0645 -0.0223
estabs_entry_rate    0.2878   0.0197  14.6260 0.0000  0.2492  0.3263
estabs_exit_rate    -0.2135   0.0226  -9.4646 0.0000 -0.2577 -0.1693
lfpr                 0.0433   0.0052   8.3381 0.0000  0.0331  0.0534
pop_pct_hisp         0.0225   0.0039   5.7863 0.0000  0.0149  0.0301
pos_net_jobs         1.1354   0.1091  10.4029 0.0000  0.9215  1.3494
urate_lower          1.2342   0.1334   9.2536 0.0000  0.9728  1.4956
urate_similar        0.6491   0.1475   4.4017 0.0000  0.3601  0.9382
year_2004           -0.5085   0.2339  -2.1739 0.0297 -0.9670 -0.0500
year_2006            1.6836   0.2409   6.9874 0.0000  1.2113  2.1558
year_2007           -1.5648   0.2314  -6.7612 0.0000 -2.0185 -1.1112
year_2008           -1.7170   0.2344  -7.3262 0.0000 -2.1763 -1.2576
year_2009           -2.5129   0.2387 -10.5262 0.0000 -2.9808 -2.0450
year_2010            0.5159   0.2344   2.2010 0.0277  0.0565  0.9754
year_2011           -0.5958   0.2327  -2.5604 0.0105 -1.0518 -0.1397
year_2012           -1.8647   0.2357  -7.9128 0.0000 -2.3266 -1.4028
year_2014           -1.5355   0.2337  -6.5694 0.0000 -1.9936 -1.0774
year_2015           -1.2351   0.2396  -5.1554 0.0000 -1.7047 -0.7656
year_2016           -2.7089   0.2369 -11.4366 0.0000 -3.1731 -2.2446
year_2017           -1.1367   0.2371  -4.7941 0.0000 -1.6015 -0.6720
year_2018           -0.4968   0.2424  -2.0493 0.0404 -0.9719 -0.0216
--------------------------------------------------------------------
Omnibus:              34634.651     Durbin-Watson:        1.994
Prob(Omnibus):        0.000         Jarque-Bera (JB):     10695929.309
Skew:                 4.490         Prob(JB):             0.000
```

```
Kurtosis:                  89.569   Condition No.:            872
==============================================================================
```

Testing

Evaluate two RMSEs:

1. null model
2. backward-selected model

Then, determine the percent improvement of the backward-selected model from the null model.

In [41]:
```
rmse_null = np.sqrt(  np.mean((y_test - np.mean(y_train))**2)  )
rmse_null
```

Out[41]: 9.40322930944683

In [42]:
```
yhat_fit = fit.predict(x_test.drop(columns = ['density',
                                               'year_2005',
                                               'pop_pct_black',
                                               'year_2013',
                                               'year_2003',
                                               'pop']))
rmse_fit = np.sqrt(  np.mean((y_test - yhat_fit)**2)  )
rmse_fit
```

Out[42]: 9.216942512937898

In [43]:
```
print(round((rmse_null - rmse_fit)/rmse_null*100, 3), '%')
```

```
1.981 %
```

In [ ]: