Workshop - Regularization

In this workshop, we are going to:

1. Tune an elastic-net regression
2. Compare the following models:
    A. The null model
    B. The tuned elastic-net model
    C. The trimmed non-regularized model with standardized features
    D. The trimmed non-regularized model with non-standardized features

Preliminaries

- Load any necessary packages and/or functions
- Load in and prepare the class data
- Create x and y with a label of `pct_d_rgdp`
- Create `x_train`, `x_test`, `y_train`, `y_test` with
    - training size of two-thirds
    - random state of 490
- Standardize the features
- Add constants

In [24]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn import linear_model as lm
```

In [17]:
```
df = pd.read_pickle('/Users/liaohaitao/Desktop/ECON 490/class data/class_data.pkl')
df.columns
```

Out[17]:
```
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 [25]:
```
df = pd.read_csv('/Users/liaohaitao/Desktop/ECON 490/class data/class_data.csv')
df.set_index(['fips', 'year', 'GeoName'], inplace = True)
df
df['year'] = df.index.get_level_values('year')
```

In [26]:
```
df_prepped = df.drop(columns = ['urate_bin', 'year']).join([
    pd.get_dummies(df['urate_bin'], drop_first = True),
    pd.get_dummies(df.year, drop_first = True)
])
```

In [30]:
```
y = df_prepped['pct_d_rgdp']
x = df_prepped.drop(columns = 'pct_d_rgdp')

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 2/3, random_state = 490)

x_train_std = x_train.apply(lambda x: (x - np.mean(x))/np.std(x), axis = 0)
x_test_std  = x_test.apply(lambda x: (x - np.mean(x))/np.std(x), axis = 0)

x_train_std = sm.add_constant(x_train_std)
x_test_std  = sm.add_constant(x_test_std)
x_train     = sm.add_constant(x_train)
x_test      = sm.add_constant(x_test)
```

In [47]:
```
regr = lm.ElasticNet(random_state=0)
regr.fit(x_train, y_train)
lm.ElasticNet(random_state=0)
print(regr.coef_)
print(regr.intercept_)
```

```
[ 0.00000000e+00  0.00000000e+00 -0.00000000e+00  3.00009972e-01
 -1.22301753e-01 -3.91512002e-07 -2.17803760e-03  2.22947887e-02
  6.64307477e-02 -9.70663666e-06  0.00000000e+00 -0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00]
-4.927378834478454
```

In [43]:
```
fit_ridge = sm.OLS(y_train, x_train).fit_regularized(alpha = 10, L1_wt = 0)
fit_ridge.params
```

Out[43]:
```
array([-5.14962948e-03,  3.50074274e-02, -4.23729499e-02,  1.41694710e-01,
       -9.01252418e-02, -8.95294801e-08, -1.76898570e-02,  2.08023912e-02,
        2.69659720e-02,  8.02492341e-06,  1.65741497e-02, -3.95625994e-03,
        4.94056214e-03,  3.35189075e-03,  3.03008429e-03,  1.37542255e-02,
       -3.79052979e-03, -6.20923071e-03, -1.45032855e-02,  5.13035704e-03,
        5.43325760e-04, -5.75092238e-03,  3.98155190e-03, -3.53246120e-03,
       -1.21612077e-03, -1.01214271e-02, -2.19542860e-03,  2.33046282e-03])
```

In [45]:
```
fit_lasso = sm.OLS(y_train, x_train).fit_regularized(alpha = 1, L1_wt = 1)
fit_lasso.params
```

Out[45]:
```
const                 0.000000e+00
pos_net_jobs          0.000000e+00
emp_estabs            0.000000e+00
estabs_entry_rate     1.913649e-01
estabs_exit_rate     -9.842778e-02
pop                  -2.365903e-07
pop_pct_black        -2.046507e-02
pop_pct_hisp          1.434198e-02
lfpr                  1.614383e-02
density               0.000000e+00
lower                 0.000000e+00
similar               0.000000e+00
2003                  0.000000e+00
2004                  0.000000e+00
2005                  0.000000e+00
2006                  0.000000e+00
2007                  0.000000e+00
2008                  0.000000e+00
2009                  0.000000e+00
2010                  0.000000e+00
2011                  0.000000e+00
2012                  0.000000e+00
2013                  0.000000e+00
2014                  0.000000e+00
2015                  0.000000e+00
2016                  0.000000e+00
2017                  0.000000e+00
2018                  0.000000e+00
dtype: float64
```

Take a look at `lm.ElasticNet?` and

```
fit = sm.OLS(y_train, x_train)
fit.fit_regularized?
```

Determine which coefficients are the same, but named differently. Specifically, α and the weight on the different constraints (i.e. $||\beta||_2$ and $||\beta||_1$).

In [61]:
```
fit = sm.OLS(y_train, x_train)
fit.fit_regularized(alpha=1.0).params
```

Out[61]:
```
const                 0.000000e+00
pos_net_jobs          0.000000e+00
emp_estabs            0.000000e+00
estabs_entry_rate     1.913649e-01
estabs_exit_rate     -9.842778e-02
pop                  -2.365903e-07
pop_pct_black        -2.046507e-02
pop_pct_hisp          1.434198e-02
lfpr                  1.614383e-02
density               0.000000e+00
lower                 0.000000e+00
similar               0.000000e+00
2003                  0.000000e+00
2004                  0.000000e+00
2005                  0.000000e+00
2006                  0.000000e+00
2007                  0.000000e+00
2008                  0.000000e+00
2009                  0.000000e+00
2010                  0.000000e+00
2011                  0.000000e+00
2012                  0.000000e+00
2013                  0.000000e+00
2014                  0.000000e+00
2015                  0.000000e+00
2016                  0.000000e+00
2017                  0.000000e+00
2018                  0.000000e+00
dtype: float64
```

In [ ]:

Perform a 5-fold cross-validation grid search with a random state of 490. Identify the optimally tuned hyperparameters. Use this grid:

```
param_grid = {'alpha': 10.**np.arange(-5, -1, 1),
              'l1_ratio': np.arange(0, 1, 0.1)}
```

You will get a warning message about convergence. We will discuss it after the workshop. Think about why it occuring.

In [63]:

```
param_grid = {'alpha': 10.**np.arange(-5, -1, 1),
              'l1_ratio': np.arange(0, 1, 0.1)}
cv = lm.ElasticNet(fit_intercept = False, normalize = False,
                    random_state = 490)
grid_search = GridSearchCV(cv, param_grid, cv = 5,
                           scoring = 'neg_root_mean_squared_error')
grid_search.fit(x_train_std, y_train)
print(grid_search.best_params_)
best = grid_search.best_params_['alpha']
best
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1162767.3929822594, toleranc
e: 253.23442781744671
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1130609.4852849508, toleranc
e: 246.96229539243055
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1140144.2131046676, toleranc
e: 248.68990538784342
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1196542.6978201736, toleranc
e: 260.29355400963703
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1161936.264629494, toleranc
e: 252.92220066402044
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1162777.3637906245, toleranc
e: 253.23442781744671
  model = cd_fast.enet_coordinate_descent(
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1130619.7805362656, toleranc
e: 246.96229539243055
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1140154.4984219028, toleranc
e: 248.68990538784342
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1196552.982331442, toleranc
e: 260.29355400963703
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1161945.7997337482, toleranc
e: 252.92220066402044
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1162876.791867304, toleranc
e: 253.23442781744671
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1130722.4397446283, toleranc
e: 246.96229539243055
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1140257.0040295958, toleranc
e: 248.68990538784342
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1196655.5038959691, toleranc
e: 260.29355400963703
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1162040.9328701212, toleranc
e: 252.92220066402044
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1163845.8859214336, toleranc
e: 253.23442781744671
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1131722.599566463, toleranc
e: 246.96229539243055
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
```

```
tive did not converge. You might want to increase the number of iterations. Duality gap: 1141251.1692901086, toleranc
e: 248.68990538784342
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1197651.8882325457, toleranc
e: 260.29355400963703
  model = cd_fast.enet_coordinate_descent(
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1162972.3791399472, toleranc
e: 252.92220066402044
  model = cd_fast.enet_coordinate_descent(
{'alpha': 0.01, 'l1_ratio': 0.0}
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:529: ConvergenceWarning: Objec
tive did not converge. You might want to increase the number of iterations. Duality gap: 1449757.3812360957, toleranc
e: 315.5255958178446
  model = cd_fast.enet_coordinate_descent(
```

Out[63]:

```
0.01
```

Question

How many models did we just fit?

In [65]:

```
x_train_std.shape
y_train.shape
```

Out[65]:

```
(33889,)
```

Using the tuned hyperparameters, fit your elastic net model with `statsmodels`

In [68]:

```
fit_lasso_tuned = sm.OLS(y_train, x_train_std).fit_regularized(alpha = best)
fit_lasso_tuned.params
```

Out[68]:

```
const                 1.973286
pos_net_jobs          0.546067
emp_estabs           -0.164027
estabs_entry_rate     0.894423
```

```
estabs_exit_rate    -0.508984
pop                 -0.097182
pop_pct_black        0.000000
pop_pct_hisp         0.294657
lfpr                 0.487393
density             -0.000401
lower                0.585185
similar              0.237165
2003                 0.031446
2004                 0.000000
2005                 0.096228
2006                 0.469200
2007                -0.273997
2008                -0.306477
2009                -0.498656
2010                 0.204014
2011                 0.000000
2012                -0.333238
2013                 0.065622
2014                -0.258129
2015                -0.180687
2016                -0.529981
2017                -0.161577
2018                 0.000000
dtype: float64
```

Using the selected features refit

- the non-regularized model with standardized features
- the non-regularized model with non-standardized features

In [74]:
```
beta = fit_lasso_tuned.params
beta.index[beta == 0]
```

Out[74]:
```
Index(['pop_pct_black', 2004, 2011, 2018], dtype='object')
```

In [75]:
```
x_train_std_trim = x_train_std.loc[:, ~x_train_std.columns.isin(beta.index[beta == 0])]
x_test_std_trim = x_test_std.loc[:, ~x_test_std.columns.isin(beta.index[beta == 0])]
```

In [76]:
```
fit_std_final = sm.OLS(y_train, x_train_std_trim).fit()
fit_std_final.summary2()
```

Out[76]:

Model:	OLS	Adj. R-squared:	0.041
Dependent Variable:	pct_d_rgdp	AIC:	246977.1044
Date:	2021-02-26 17:02	BIC:	247179.4447
No. Observations:	33889	Log-Likelihood:	-1.2346e+05
Df Model:	23	F-statistic:	63.22
Df Residuals:	33865	Prob (F-statistic):	4.53e-287
R-squared:	0.041	Scale:	85.562

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	1.9833	0.0502	39.4707	0.0000	1.8848	2.0818
pos_net_jobs	0.5514	0.0541	10.1890	0.0000	0.4453	0.6575
emp_estabs	-0.1734	0.0537	-3.2316	0.0012	-0.2786	-0.0682
estabs_entry_rate	0.9019	0.0594	15.1881	0.0000	0.7855	1.0183
estabs_exit_rate	-0.5252	0.0578	-9.0859	0.0000	-0.6385	-0.4119
pop	-0.1052	0.0567	-1.8574	0.0633	-0.2163	0.0058
pop_pct_hisp	0.3075	0.0517	5.9454	0.0000	0.2061	0.4088
lfpr	0.4852	0.0586	8.2835	0.0000	0.3704	0.6000
density	-0.0071	0.0538	-0.1328	0.8944	-0.1126	0.0983
lower	0.6102	0.0682	8.9510	0.0000	0.4766	0.7439
similar	0.2625	0.0588	4.4641	0.0000	0.1473	0.3778
2003	0.0335	0.0548	0.6121	0.5405	-0.0739	0.1410
2005	0.1067	0.0585	1.8256	0.0679	-0.0079	0.2213
2006	0.4801	0.0596	8.0545	0.0000	0.3633	0.5970
2007	-0.2909	0.0549	-5.2972	0.0000	-0.3985	-0.1832
2008	-0.3212	0.0547	-5.8678	0.0000	-0.4285	-0.2139
2009	-0.5106	0.0562	-9.0929	0.0000	-0.6206	-0.4005
2010	0.2088	0.0552	3.7855	0.0002	0.1007	0.3169

2012	-0.3495	0.0548	-6.3744	0.0000	-0.4570	-0.2420
2013	0.0681	0.0548	1.2422	0.2142	-0.0393	0.1754
2014	-0.2758	0.0551	-5.0056	0.0000	-0.3838	-0.1678
2015	-0.1986	0.0549	-3.6176	0.0003	-0.3061	-0.0910
2016	-0.5485	0.0550	-9.9755	0.0000	-0.6562	-0.4407
2017	-0.1792	0.0550	-3.2615	0.0011	-0.2869	-0.0715

Omnibus:	34597.339	Durbin-Watson:	1.994
Prob(Omnibus):	0.000	Jarque-Bera (JB):	10663135.814
Skew:	4.482	Prob(JB):	0.000
Kurtosis:	89.436	Condition No.:	3

Compare the percent improvement from the null model RMSE to the elastic-net and OLS model.

In [77]:

```
x_train_trim = x_train.loc[:, ~x_train.columns.isin(beta.index[beta == 0])]
x_test_trim = x_test.loc[:, ~x_test.columns.isin(beta.index[beta == 0])]
```

In [78]:

```
fit_final = sm.OLS(y_train, x_train_trim).fit()
fit_final.summary2()
```

Out[78]:

Model:	OLS	Adj. R-squared:	0.041
Dependent Variable:	pct_d_rgdp	AIC:	246977.1044
Date:	2021-02-26 17:03	BIC:	247179.4447
No. Observations:	33889	Log-Likelihood:	-1.2346e+05
Df Model:	23	F-statistic:	63.22
Df Residuals:	33865	Prob (F-statistic):	4.53e-287
R-squared:	0.041	Scale:	85.562

	Coef.	**Std.Err.**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-2.7552	0.5018	-5.4904	0.0000	-3.7388	-1.7716

pos_net_jobs	1.1110	0.1090	10.1890	0.0000	0.8973	1.3247
emp_estabs	-0.0363	0.0112	-3.2316	0.0012	-0.0584	-0.0143
estabs_entry_rate	0.2979	0.0196	15.1881	0.0000	0.2595	0.3364
estabs_exit_rate	-0.2053	0.0226	-9.0859	0.0000	-0.2495	-0.1610
pop	-0.0000	0.0000	-1.8574	0.0633	-0.0000	0.0000
pop_pct_hisp	0.0234	0.0039	5.9454	0.0000	0.0157	0.0312
lfpr	0.0435	0.0053	8.2835	0.0000	0.0332	0.0538
density	-0.0000	0.0000	-0.1328	0.8944	-0.0001	0.0001
lower	1.2576	0.1405	8.9510	0.0000	0.9823	1.5330
similar	0.6784	0.1520	4.4641	0.0000	0.3806	0.9763
2003	0.1420	0.2320	0.6121	0.5405	-0.3128	0.5968
2005	0.4516	0.2473	1.8256	0.0679	-0.0332	0.9363
2006	2.0237	0.2513	8.0545	0.0000	1.5313	2.5162
2007	-1.2268	0.2316	-5.2972	0.0000	-1.6808	-0.7729
2008	-1.3693	0.2334	-5.8678	0.0000	-1.8267	-0.9119
2009	-2.1596	0.2375	-9.0929	0.0000	-2.6251	-1.6941
2010	0.8825	0.2331	3.7855	0.0002	0.4256	1.3394
2012	-1.4919	0.2340	-6.3744	0.0000	-1.9506	-1.0332
2013	0.2899	0.2333	1.2422	0.2142	-0.1675	0.7472
2014	-1.1585	0.2314	-5.0056	0.0000	-1.6121	-0.7049
2015	-0.8580	0.2372	-3.6176	0.0003	-1.3229	-0.3931
2016	-2.3362	0.2342	-9.9755	0.0000	-2.7952	-1.8772
2017	-0.7632	0.2340	-3.2615	0.0011	-1.2219	-0.3046

Omnibus:	34597.339	Durbin-Watson:	1.994
Prob(Omnibus):	0.000	Jarque-Bera (JB):	10663135.814
Skew:	4.482	Prob(JB):	0.000

Kurtosis:	89.436	Condition No.:	3468030

In [79]:
```
rmse_null = np.sqrt(np.mean(  (y_test - np.mean(y_train))**2  ))
rmse_null
```

Out[79]: 9.40322930944683

In [80]:
```
rmse_lasso = np.sqrt(np.mean(  (y_test - fit_lasso_tuned.predict(x_test_std))**2  ))
print(rmse_lasso)
round((rmse_lasso - rmse_null)/rmse_null*100, 2)
```

9.217124732339427

Out[80]: -1.98

In [81]:
```
rmse_std_final = np.sqrt(np.mean(  (y_test - fit_std_final.predict(x_test_std_trim))**2  ))
print(rmse_std_final)
round((rmse_std_final - rmse_null)/rmse_null*100, 2)
```

9.216854907402741

Out[81]: -1.98

In [82]:
```
rmse_final = np.sqrt(np.mean(  (y_test - fit_final.predict(x_test_trim))**2  ))
print(rmse_final)
round((rmse_final - rmse_null)/rmse_null*100, 2)
```

9.216827559489085

Out[82]: -1.98

In [ ]: