Predicting Gubernatorial Elections
PPHA 31720 – The Science of Elections and Politics (Fall, 2019)

Overview

The goal of this analysis is to predict the outcome of the 2019 gubernatorial elections in Kentucky, Louisiana, and Mississippi. To make these predictions, I use information on the incumbency status of the candidates, polling data, and historical presidential and gubernatorial election results. I then train a simple machine learning model on the data from the 2018 gubernatorial elections, and then use this model to predict the election results for the 2019 gubernatorial elections. My model predicts the following –

- Kentucky: Democratic nominee Andy Beshear will receive 48.81% of the two-party vote share
- Louisiana: Democratic nominee John Bel Edwards will receive 50.00% of the two-party vote share
- Mississippi: Democratic nominee Jim Hood will receive 46.95% of the two-party vote share

Methodology

Model Specification

My target – or the variable I am trying to predict – is the Democratic candidate's share of the two-party vote (`dem_vtsh_actual`). To do so, I include five features –

- `dem_incumbent` : Whether the Democratic candidate is the incumbent
- `rep_incumbent` : Whether the Republican candidate is the incumbent
- `dem_vtsh_poll` : The Democratic candidate's share of the two-party vote in the most recent poll
- `dem_vtsh_last_gov` : The Democratic candidate's share of the two-party vote in the last gubernatorial election
- `dem_vtsh_2016_pres` : Hillary Clinton's share of the two-party vote in the 2016 presidential election

Using data on the 36 gubernatorial elections in 2018, I build a random forest regression model, one of the most common – and often best-performing – machine learning models in predictive analytics. I chose this model over, for example, a linear regression because random forests are able to capture non-linear interactions between the features and the target, which are likely relevant in this context. Additionally, given the small size of the training set here, a bagging ensemble method like a random forest can prevent over-fitting.

Data Sources & Feature Generation

Using election results data from [Dave Leip's Atlas of U.S. Elections (https://uselectionatlas.org/RESULTS/)](https://uselectionatlas.org/RESULTS/), I compute the Democratic candidate's share of the two-party vote in gubernatorial elections in 2018 for the 36 training elections (`dem_vtsh_actual`) and in each state's most recent gubernatorial election – 2014 for the 36 training elections and 2015 for the three predicted states (`dem_vtsh_last_gov`). In both cases, this was simply the number of votes received by the Democratic candidate over the number of votes received by the Democratic and Republican candidates combined. I also compute Hillary Clinton's share of the two-party vote in the 2016 presidential election as the number of votes that she received over the total number of votes that she and Donald Trump received in each state (`dem_vtsh_2016_pres`).

I use [FiveThirtyEight's gubernatorial forecast data (https://github.com/fivethirtyeight/data/tree/master/governors-forecast-2018)](https://github.com/fivethirtyeight/data/tree/master/governors-forecast-2018) to pull information on whether the Democratic or Republican candidate running in the election is the incumbent. Note that I do NOT use the numerical precictions from the forecast – I simply use the party and incumbency status fields included in this dataset to create indicator variables for whether each election has an incumbent Democrat running (`dem_incumbent`) or incumbent Republican running (`rep_incumbent`).

Finally, I use [FiveThirtyEight's gubernatorial polls (https://projects.fivethirtyeight.com/polls/governor/)](https://projects.fivethirtyeight.com/polls/governor/) to compute the Democratic candidate's predicted share of the two-party vote in the single most recent poll available for each election (`dem_vtsh_poll`). For Mississippi's election, this was the Mason-Dixon poll released on October 23. Louisiana's most recent poll was the We Ask America poll released on October 17, and Kentucky's most recent poll was the Mason-Dixon poll released on October 16. For the 2018 races, the poll released closest to the actual elections were used.

The fully reproducible code used to generate these predictions is shown below. This also includes the full modelling dataset, with the full set of features and target for the 2018 and 2019 gubernatorial elections.

Import Modules

First, I import the Python libraries and packages used in this analysis. This includes `predicting_elections` , which contains the helper functions used below.

In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
import predicting_elections as pred
```

Generate Features

Next, I generate the target and five features discussed above (`dem_incumbent` , `rep_incumbent` , `dem_vtsh_poll` , `dem_vtsh_last_gov` , and `dem_vtsh_2016_pres`). This process relies upon the helper functions called from `predicting_elections` to wrangle the raw data. This creates five separate datasets.

In [2]:
```python
# wrangle dem_vtsh_actual
gov_results_18 = pred.wrangle_dem_vtshr('raw/2018_gov_election_results.csv',
                                        'dem_vtsh_actual',
                                        'Democr..1',
                                        'Republ..1',
                                        'State')
new_rows = [{'State': 'Kentucky', 'dem_vtsh_actual': np.NaN},
            {'State': 'Louisiana', 'dem_vtsh_actual': np.NaN},
            {'State': 'Mississippi', 'dem_vtsh_actual': np.NaN}]
gov_results_18 = pred.append_rows(gov_results_18, new_rows)

# wrangle dem_incumbent / rep_incumbent
incumbents = pred.wrangle_incumbents('raw/governor_state_forecast.csv')
new_rows = [{'state': 'KY', 'dem_incumbent': 0, 'rep_incumbent': 1},
            {'state': 'LA', 'dem_incumbent': 1, 'rep_incumbent': 0},
            {'state': 'MS', 'dem_incumbent': 0, 'rep_incumbent': 0}]
incumbents = pred.append_rows(incumbents, new_rows)

# wrangle dem_vtsh_poll
polls = pred.wrangle_dem_poll('raw/governor_polls.csv')

# wrangle dem_vtsh_2016_pres
pres_results = pred.wrangle_dem_vtshr('raw/2016_pres_election_results.csv',
                                      'dem_vtsh_2016_pres',
                                      'Clinton.1',
                                      'Trump.1',
                                      'State')

# wrangle dem_vtsh_last_gov
gov_results_last = pred.wrangle_dem_vtshr('raw/last_gov_election_results.csv',
                                          'dem_vtsh_last_gov',
                                          'Democr..1',
                                          'Republ..1',
                                          'State')
```

Combine Datasets

I next combine the five datasets created above to get a single dataset (`merged`) with the full set of five features and the target. I also print out this final modelling dataset.

In [3]:
```python
# combine dfs
gov_results_18['state'] = gov_results_18['State'].apply(lambda x: pred.get_abbr(x))
merged = gov_results_18.merge(
    incumbents, on = 'state', how = 'left').merge(
    polls, left_on = 'State', right_on = 'state', how = 'left').merge(
    pres_results, on = 'State', how = 'left').merge(
    gov_results_last, on = 'State').drop(['state_x', 'state_y'], axis=1)

# replace if neither candidate is an incumbent
merged['dem_incumbent'].fillna(0, inplace=True)
merged['rep_incumbent'].fillna(0, inplace=True)

# print full modeling dataset
merged
```

Out[3]:

	State	dem_vtsh_actual	dem_incumbent	rep_incumbent	dem_vtsh_poll	dem_vtsh_2016_pres	dem_vtsh_last_gov
0	Alabama	0.404493	0.0	1.0	0.391304	0.356259	0.636859
1	Alaska	0.463349	0.0	0.0	0.498821	0.416143	1.000000
2	Arizona	0.427636	0.0	1.0	0.421687	0.481100	0.562201
3	Arkansas	0.327184	0.0	1.0	0.285714	0.357149	0.571920
4	California	0.619485	0.0	0.0	0.563830	0.661282	0.400298
5	Colorado	0.555169	0.0	0.0	0.529412	0.526833	0.482455
6	Connecticut	0.516499	0.0	0.0	0.554217	0.571415	0.487036
7	Florida	0.498001	0.0	0.0	0.482402	0.493812	0.505660
8	Georgia	0.492988	0.0	0.0	0.520833	0.473388	0.540249
9	Hawaii	0.650291	1.0	0.0	0.626506	0.674413	0.428473
10	Idaho	0.389851	0.0	0.0	0.414894	0.316898	0.581303
11	Illinois	0.584089	0.0	1.0	0.597561	0.590201	0.520297
12	Iowa	0.486024	0.0	1.0	0.515789	0.449365	0.612815
13	Kansas	0.527867	0.0	0.0	0.492554	0.388885	0.519235
14	Maine	0.541002	0.0	0.0	0.591398	0.515968	0.526273
15	Maryland	0.440085	0.0	1.0	0.391304	0.640162	0.519230
16	Massachusetts	0.332108	0.0	1.0	0.268817	0.646513	0.509801
17	Michigan	0.549283	0.0	0.0	0.543820	0.498823	0.520791
18	Minnesota	0.559296	0.0	0.0	0.563830	0.508285	0.470601
19	Nebraska	0.409995	0.0	1.0	NaN	0.364523	0.593004
20	Nevada	0.521561	0.0	0.0	0.501126	0.512937	0.747183
21	New Hampshire	0.464251	0.0	1.0	0.500000	0.501970	0.475214
22	New Mexico	0.571991	0.0	0.0	0.563830	0.546508	0.572231
23	New York	0.622176	1.0	0.0	0.593407	0.617723	0.426158
24	Ohio	0.480921	0.0	0.0	0.527473	0.457324	0.658332
25	Oklahoma	0.437320	0.0	0.0	0.484083	0.306953	0.576427
26	Oregon	0.534156	1.0	0.0	0.517241	0.561558	0.469355
27	Pennsylvania	0.586695	1.0	0.0	0.557895	0.496245	0.450678
28	Rhode Island	0.586066	1.0	0.0	0.569054	0.583107	0.470980
29	South Carolina	0.459789	0.0	1.0	0.412338	0.425397	0.574405
30	South Dakota	0.482866	0.0	0.0	0.531250	0.340281	0.734804
31	Tennessee	0.392946	0.0	0.0	0.453608	0.363757	0.754775
32	Texas	0.432366	0.0	1.0	0.455804	0.452868	0.603726
33	Vermont	0.421776	0.0	1.0	0.443182	0.651864	0.493108
34	Wisconsin	0.505579	0.0	1.0	0.505618	0.495920	0.528706
35	Wyoming	0.290913	0.0	0.0	0.306818	0.242947	0.685449
36	Kentucky	NaN	0.0	1.0	0.500000	0.343294	0.545178
37	Louisiana	NaN	1.0	0.0	0.500000	0.398283	0.438855
38	Mississippi	NaN	0.0	0.0	0.483146	0.409102	0.672978

Build Model

Next I actually build the machine learning model by fitting to the training data – or on the 2018 elections.

In [4]:
```python
# specify model parameters
features = ['dem_incumbent',
            'rep_incumbent',
            'dem_vtsh_poll',
            'dem_vtsh_2016_pres',
            'dem_vtsh_last_gov',
            'dem_vtsh_last_gov']
target = 'dem_vtsh_actual'

# split training and testing data
X_train, y_train, X_test, test = pred.split(merged, target, features)

# fit model
rfr = RandomForestRegressor(n_estimators = 10)
rfr.fit(X_train, y_train.values.ravel())
```

Out[4]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

Make Predictions

Finally, I use the model built above to predict elections results for the 2019 elections, and print out these predictions.

In [5]:
```python
# make predictions
test['dem_vtsh_predicted'] = rfr.predict(X_test)
test[['State', 'dem_vtsh_predicted']]
```

Out[5]:

	State	dem_vtsh_predicted
36	Kentucky	0.488137
37	Louisiana	0.500352
38	Mississippi	0.469504