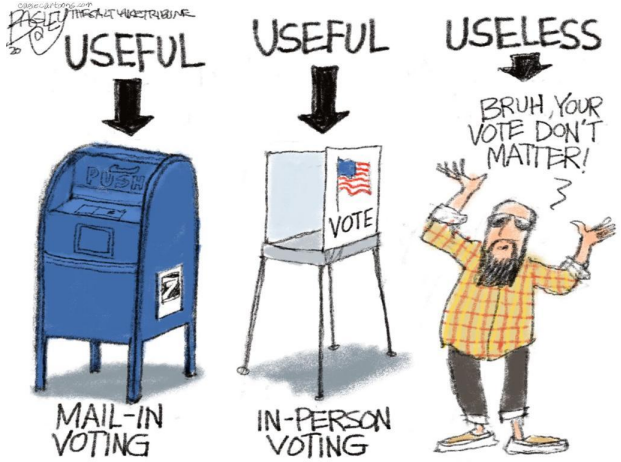
Swing Stacks:

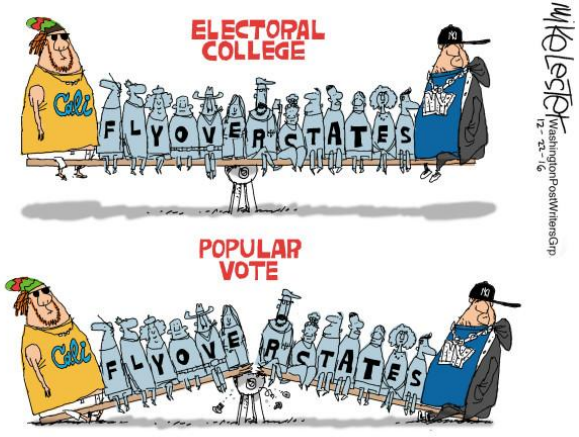
Predicting political party affiliation using model stacking

Shisham Adhikari[1], Maggie Slein[1], Grayson White[1], Nate Wells[1]
[1]Department of Mathematics, Reed College, Portland, OR

Introduction

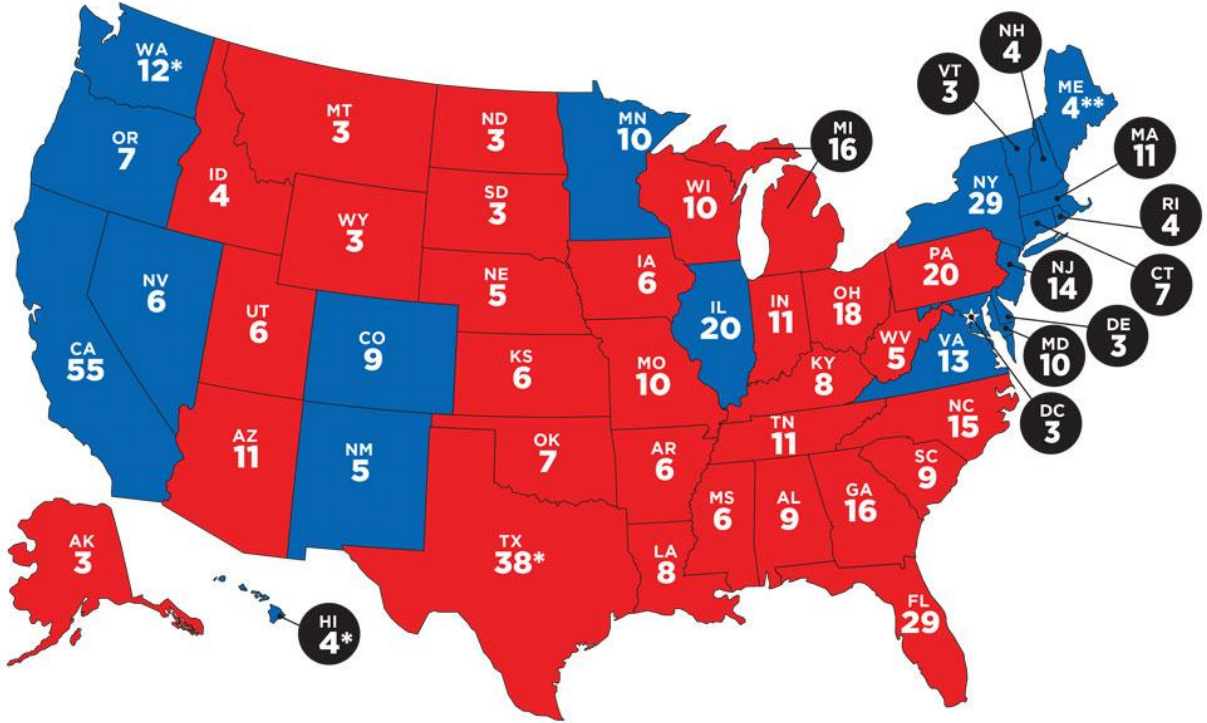
- ★ In the United States, citizens do not directly vote for the candidates on the ballot
- ★ Instead, citizens vote for the electors of the Electoral College, essentially other people who will vote for them





Introduction

And that process looks like this...



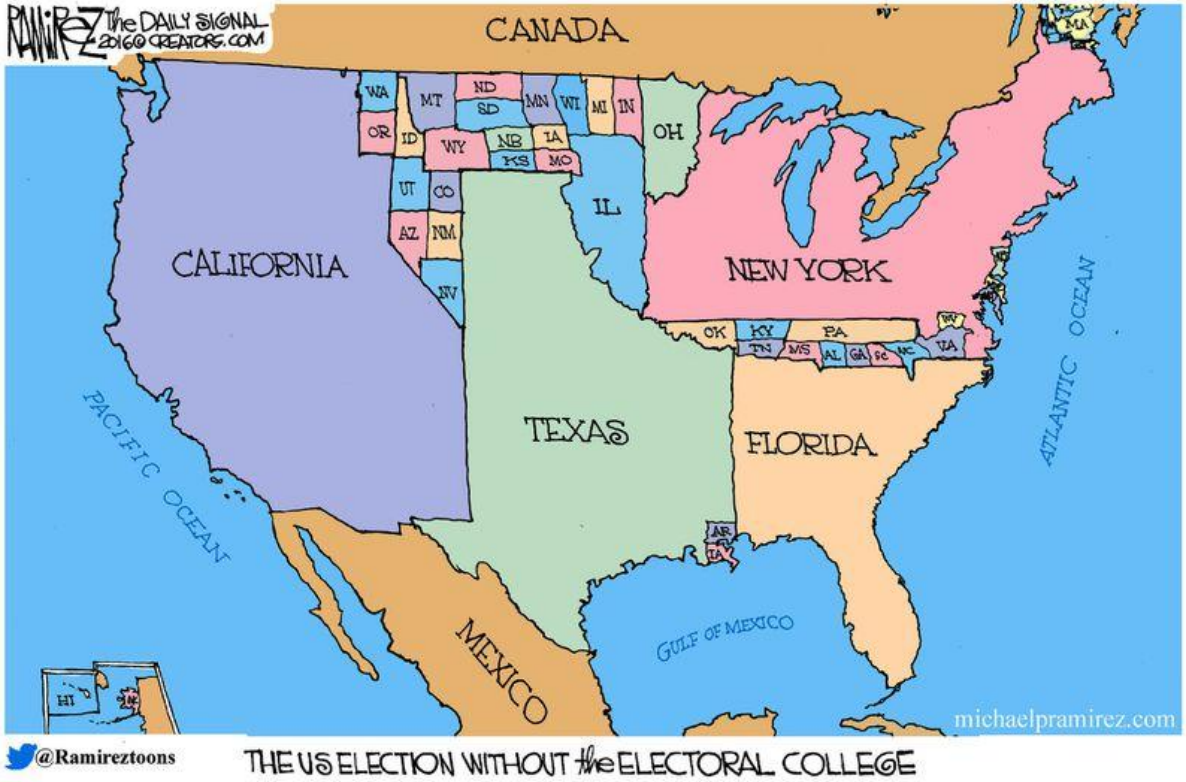
Introduction

And that process looks like this...

1. Each state receives a number of electoral votes based on:
   - ★ the number of representatives in the House of Representatives (based on population size)
   - ★ the representatives each state has in the Senate (every state has two)
2. Based on which candidate wins the popular vote in the state, all of those state's electoral votes are awarded to that candidate
3. On December 14th, each state sends electors to vote for the candidate that won their state's electoral votes and the winner with the most electoral votes becomes President-elect

Introduction

Why do we even have an Electoral College?



Introduction

Why do we even have an Electoral College?

- ★ The Great Compromise, as known as the Connecticut Compromise, was established so that smaller states would not be overpowered by the larger states
- ★ The Electoral College is an active example of this compromise at work, such that, the smaller states have essentially decided elections in modern history

Introduction

What has the Electoral College meant in modern history?

- ★ With increasingly polarized politics, the last several elections have been decided by a small number of “swing states” or states in which voters lean towards either party
- ★ 2 out of the last 4 presidents have won the electoral college but not the popular vote
- ★ There have only been 5 presidents in history to have done so, perhaps most notably, Rutherford B. Hayes, whose 1877 presidential win was orchestrated by the Democrats acceptance of Hayes as president, in exchange for Republicans’ removal of federal troops from the South, ending Reconstruction and ushering in the era of Jim Crow in the South

Introduction

With the increasing polarization of American elections in the 21st century and the increasing importance of understanding "swing state" dynamics, we were interested in improving how we predict party affiliation based on a number of socio-economic predictors using data from the GSS (General Social Survey)

For our project, we decided to implement model stacking using the {*stacks*} package from {*tidymodels*} and compare it to the classical approach of logistic regression with {*survey*} weights.

Methods

Dataset: General Social Survey (GSS)

- ★ Since 1972 by the National Opinion Research Center, the University of Chicago; 64,184 obs and 6,110 vars
- ★ Demographic information and opinions on various topics
- ★ Adults (18+) living in households in the United States
- ★ Face-to-face, 90-minutes, in-person interviews
- ★ After the US Census, the most frequently used dataset in the social sciences

Our Subset

- ★ 2000+, -NAs, 18 vars, 14 predictors, ‘partyid’ as response
- ★ Final subset: 5,800 rows, 16 columns, and 0 NA's
- ★ 3:1 split into train and test data to compare test MSEs

Methods

Weighted Logistic regression with {*survey*}

Overview

★ We specified a complex survey design with weights:
- Survey Weights
  - A value assigned to each case in the data
  - Make our results more representative of the population
  - Bias-variance tradeoff: increases variance, hope of lowering bias
- Logistic regression using svyglm()

★ Code Example:

```
#Creating survey design
gss_design <-
    svydesign(~ vpsu ,
        strata = ~ vstrat ,
        data = gss_survey ,
        weights = ~ weight ,
        nest = TRUE)
```

Methods

Modeling stacking with {*stacks*}

Overview

- We specified five model definitions to stack:
  - Logistic regression
  - Penalized logistic regression
  - Linear discriminant analysis
  - Random forest
  - K nearest neighbors

- Each model definition has submodels which are added to the stack (88 total submodels)

- The {stacks} engine performs model selection and retains a subset of these models

Methods

Modeling stacking with {*stacks*}

Specifying a model definition

★ KNN Example:

```
knn_spec <- nearest_neighbor(neighbors = tune()) %>%
  set_mode("classification") %>%
  set_engine("kknn")

knn_wf <- gss_workflow %>%
  add_model(knn_spec)

# tuning
set.seed(13)
knn_res <-
  tune_grid(
    knn_wf,
    resamples = folds,
    control = control_stack_grid()
  )
```

Methods

Modeling stacking with {*stacks*}

{*stacks*} workflow

★ After specifying all of our model definitions, we can easily use {*stacks*} to *add_candidates()* and *blend_predictions()*

★ These functions put all of our models together, compare performance, and select (and weight) the best models

★ Finally, the model stack uses these weighted models to make predictions on the test set, and gives us our final model accuracy

Results

Weighted logistic regression with {*survey*}

Model accuracy

- ★ Assessing model accuracy using svyglm() is similar to using a simple glm() method
  - Model on train -> predict on test -> test MSE
- ★ Training error rate is 0.3275862 for the logistic regression with weights
- ★ To compare our result to the {tidymodels} approach, use 3:1 split into a training and testing set
- ★ Test error rate is 0.3627586 for the logistic regression with weights

Results

Modeling stacking with {*stacks*}

★ We retained 11 models, below are the weights for the top 10:

member	type	weight
.pred_rep_logreg_resamples_1_100	logistic_reg	1.57000
.pred_rep_logreg_resamples_1_097	logistic_reg	1.36000
.pred_rep_logreg_resamples_1_073	logistic_reg	0.95600
.pred_rep_rf_res_1_2	rand_forest	0.91100
.pred_rep_rf_res_1_3	rand_forest	0.70100
.pred_rep_logreg_resamples_1_112	logistic_reg	0.41700
.pred_rep_lda_resamples_1_07	discrim_linear	0.27200
.pred_rep_logreg_resamples_1_118	logistic_reg	0.20300
.pred_rep_logreg_resamples_1_015	logistic_reg	0.04410
.pred_rep_lda_resamples_1_04	discrim_linear	0.00112

Results

Modeling stacking with {*stacks*}

★ Accuracy of each model member and the stack:

metric	estimate	member
accuracy	0.6475862	Model Stack
accuracy	0.6386207	Penalized Logistic Regression
accuracy	0.6406897	Penalized Logistic Regression
accuracy	0.6386207	Penalized Logistic Regression
accuracy	0.6393103	Penalized Logistic Regression
accuracy	0.6434483	Penalized Logistic Regression
accuracy	0.6358621	Penalized Logistic Regression
accuracy	0.5703448	Penalized Logistic Regression
accuracy	0.6372414	LDA
accuracy	0.6372414	LDA
accuracy	0.6400000	Random Forest
accuracy	0.6517241	Random Forest

Conclusions

Overall, the model stack outperformed the survey regression:

★ {stacks} classification rate: 0.6476

★ {survey} classification rate: 0.6372

★ Difference: 0.0104 (Just over 1%)

Do the benefits of model stacking outweigh the cost in this application?

Conclusions

Costs and benefits of {*survey*}

Benefits:

★ Interpretability

★ Simple to execute

★ Handles complex survey design well

Costs:

★ Handling {*survey*} package

★ Slightly worse predictive power than {*stacks*}

★ Little nuances/wrinkles

Conclusions

Costs and benefits of {*stacks*}

Benefits:

★ Slightly improved model predictive power

★ Lower variance

★ Requires less thinking about model form

Costs:

★ Computationally intense

★ Loss of interpretability

★ Requires less thinking about model form

Future Directions

- ★ More predictors (we used 14 predictors)
- ★ More model definition types in model stack
- ★ Different time frame
- ★ An alternate method to handle NAs (we simply deleted them)
  - Suggestion: imputation
- ★ Alternate dataset to answer the research question
  - Compare with our results

Questions?