Swing Stacks: Predicting political party affiliation using model stacking

Shisham Adhikari, Maggie Slein, Grayson White

Abstract

As American politics has become increasingly polarized over the last several decades, predicting the likelihood of presidential victories has become more difficult. With just a few states' electoral votes deciding the figurehead of the executive branch, two out of the four last elections have been decided based on the electoral college and not on the popular vote. Given this, many previous models used to predict election outcomes have failed to predict the true winner. We aim to address in two ways: First, by using the `survey` package to create the "classic" logistic regression models with weights, and then by using `tidymodels` and `stacks` to layer multiple models and model types to better predict party affiliation using several predictors from the General Social Survey (GSS) dataset from 2000-2016.

Introduction

In the last two decades, the polarization of American politics has yielded unexpected victories by minority political groups, both in the 2000 and 2016 election. The Republican party represents less Americans by volume than both the Democratic party and the Independent parties, yet has continued to secure political power as a result of the electoral college. The two unprecedented victories in 2000 and 2016 by candidates who lost the popular vote but won the presidency as a result of the electoral college have been attributed to highly variable party affiliation in what have been dubbed "swing states". In these states, the split of Democratic to Republican votes is fairly even, which makes winning their popular votes crucial for ultimately securing their electoral votes, in the electoral college's "winner takes all" framework. These states have been a source of contention in 2 of the 4 elections.

In 2000, the race between Governor George W. Bush vs. Vice President Al Gore for the presidency came down to one state: Florida. Gore had secured the popular vote, but he needed to win Florida to win the necessary electoral votes to secure the presidency. Early on during election night, Bush was called as the winner with margin of 100,000 votes. However, as votes from highly democratic districts poured in, the margin between Bush and Gore narrowed to just 2,000 votes. The vote counts in Florida were so close, the law demanded a recount. However, the legally mandated recount was to be performed by machine, not by hand. At the time, some counties used punch ballots and there was concern over the anomalies present in ballots cast with such a narrow margin between the two candidates. Gore pushed to have hand-counted recounts in particular counties via litigation. With the saga of who would ultimately become president-elect enduring for over a month after election day, the Supreme Court Case, Bush v. Gore ended the recount on December 12th, 2000. The verdict in that case was essentially that Gore did not have the grounds to request anything beyond a machine-automated recount of the votes. Thus, the electoral votes from the state of Florida ultimately decided that outcome of the 2000 election: President-elect Governor George W. Bush.

For the fifth time in US history, the 2016 race between businessman Donald J. Trump and former Secretary of State Hilary Clinton was also decided by the electoral college and not the popular vote. Less contentious than the 2000 election, the results of the 2016 election have been attributed to a lack of concern for winning electoral votes in key swing states. As these highly influential and dynamic "swing states" shift with each passing election, a better of understanding and prediction of their political party tendencies becomes increasing important. To better understand how external factors drive political party affiliation and ultimately predict political party affiliation, we are interested using techniques that combine many model types to provide more accurate conclusions.

Previous models have utilized one statistical learning method: Principal Component Analysis (Newman and Sheth 1985), LASSO (Kristensen et al. 2017), and multiple linear regression (Ben-Bassat and Dahan 2012) when attempting understand how social attributes influence party affiliation or election outcome. While these models have demonstrated that social behavior, like a single Facebook like can predict party affiliation (Kristensen et al. 2017) and identity, like Arab voter turnout and political affinity is Israeli elections (Ben-Bassat and Dahan 2012), they do not explain the trend towards polarization in last decade of American elections, domestically or globally. Thus, to better understand this phenomenon, we asked the question: How can we use a combination of models to predict political party affiliation using small sample sizes? Our group wishes to understand the external factors related to political party affiilation, and to understand how well we can predict political party affiliation by using a combination of the modeling techniques learned in class. Model stacking is an ensemble method that combines a variety of model types to optimize model predictions by training the model with a variety of model types.

Methods

The GSS dataset

The general social survey (GSS) is a massive survey conducted of people within the United States since 1972 by the National Opinion Research Center at the University of Chicago. The data collected survey includes both demographic information like age, race, gender etc. and respondents' opinions about various social and political issues like political affiliations, the state of race, government spending etc. This comprehensive survey is widely used by social scientists to see how the demographic factors interact with various beliefs.

The survey targets adults (18+) living in households in the United States. The GSS sample is drawn using an area probability design that randomly selects respondents in households across the US. Participation in the study is strictly voluntary but every respondent selected is crucial to the results. The survey is conducted face-to-face with an in-person interview by NORC at the University of Chicago. The survey was conducted every year from 1972 to 1994 (except in 1979, 1981, and 1992). Since 1994, it has been conducted every other year. It takes about 90 minutes to complete the main survey. After the US Census, the GSS is the most frequently used dataset in the social sciences. The survey result is used by scientists, researchers, government officials, and students to better understand different aspects of the residents of the United States.

Since the GSS gives an extensive information about a representative sample of people in the United States, we chose this dataset for our research. We have chosen some key variables collected from this survey, along with participants from 2000 or more recent, in order for us to attempt to classify political affiliation of participants. We also removed all NAs which was total 9577 observations. Our final subset of the GSS dataset contains 5,800 rows, 16 columns, and 0 NA's.

Filtering

Most of this filtering was done for the `infer` package `gss` dataset and can be attributed to authors of that package. We have included more rows and columns than that package, however, much initial tidying and subsetting can be attributed to them (Bray et al. 2020). The Code Appendix includes the code adapted from the `infer` package to attain our dataset, `gss_subset`.

Given our goal to understand which factors influence party affiliation in the US, we selected `year` (year of the election), `age` (age of time of survey), `college` (degree or no degree), `partyid` (democrat or republican), `hompop` (number of people in the respondent's household), `hours` (number of hours worked in the last week), `income` (total family income, categorical), `class` (socioeconomic class as described by respondent), `finrela` (respondent's opinion on family's income level), `wrkgovt` (whether or not the respondent works for the government), `marital` (respondent's martial status), `educ` (highest year of school completed), `race` (race of respondent), `income16` (respondent's family income at the age of 16), and `weight` (survey weight).

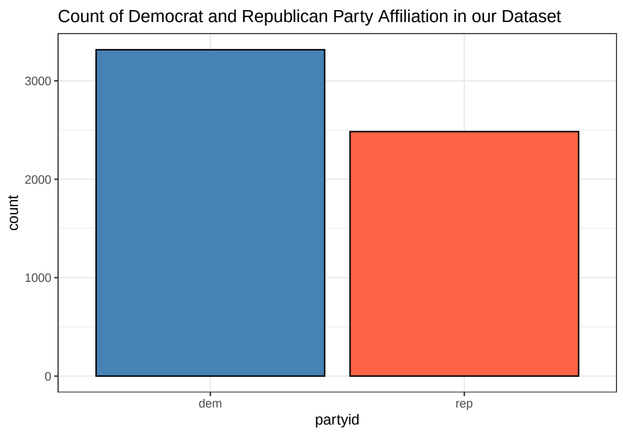
We made some choices while filtering the dataset which will effect the final results of our models. First of all, we have filtered all observations which do not state that their political affiliation was either democrat or republican. We are most interested in answering the question of whether or not we can classify between these parties rather than considering much smaller third parties. Also, we have filtered all observations with any NA's. We chose to do this for ease of analysis and because many of the models we use will not consider a row that includes NA's in any of the columns being used for the model.

Exploratory Data Analysis

Before we dig too deeply in to the dataset, it is important to understand its structure. We have 16 columns and 5800 rows (3316 democrats, 2484 republicans). Below is a glimpse of our dataset:

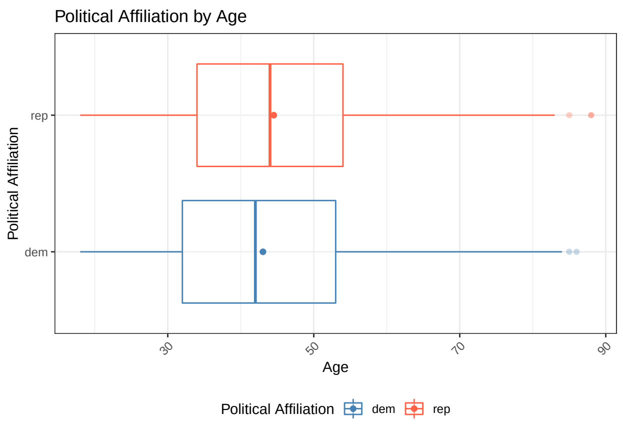
year	age	sex	college	party	home	hours	income	class	finrela	wrkgovt	marital	educ	race	incom16
2002	25	female	no degree	rep	1	40	$25000 or more	middle class	average	private	divorced	14	white	average
2002	43	male	degree	rep	1	72	$25000 or more	middle class	above average	private	married	16	white	above average
2002	46	male	no degree	rep	2	40	$25000 or more	middle class	above average	private	divorced	14	white	above average
2002	71	female	no degree	rep	1	24	$20000 - 24999	working class	average	private	divorced	12	white	average
2002	37	male	no degree	rep	1	50	$25000 or more	middle class	average	private	never married	15	white	below average
2002	23	male	no degree	dem	3	60	$25000 or more	working class	average	private	separated	12	black	average

As we first explore the dataset, we can look at the distribution of democrats and republications in our dataset in counts:

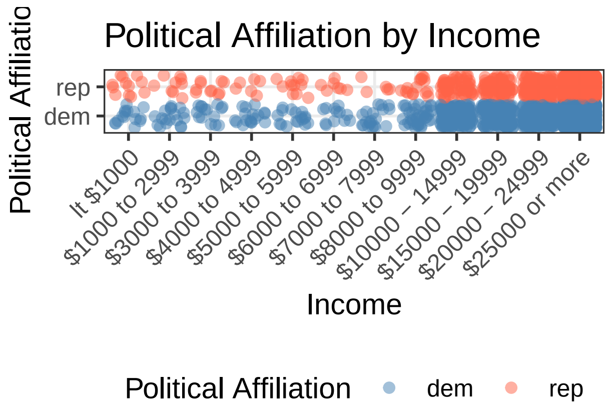


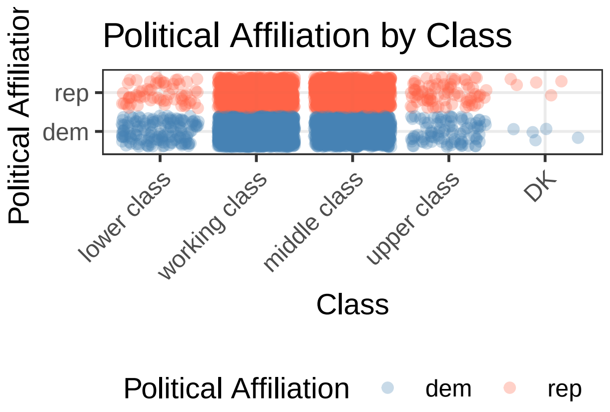
There appears to me more democrats than republicans represented in this dataset, which could be because democrats are more likely to participate in this survey, or it could be that the way we selected our data systemically oversampled democrats. Notably from this, it is the case that our the weights associated with our sample of the GSS dataset would not be the same as the weights that the GSS uses for the dataset, so the `weight` variable should be ignored entirely.

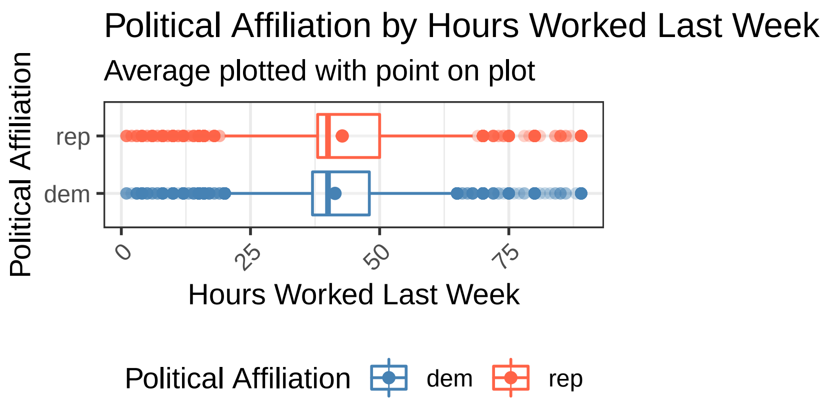
Now, we can examine some of our predictor variables with our response, `partyid`, to see the relationships there are between variables. First, we see in this side-by-side boxplot with the means plotted on top that republicans tend to be older on average:



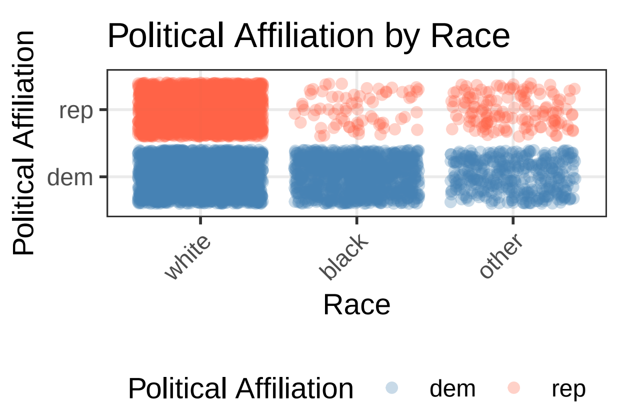
Next, it is interesting to consider economic status across political affiliations. By comparing political affiliation to income, class, and hours worked in the last week we can see small relationships between political affiliation and economic status:

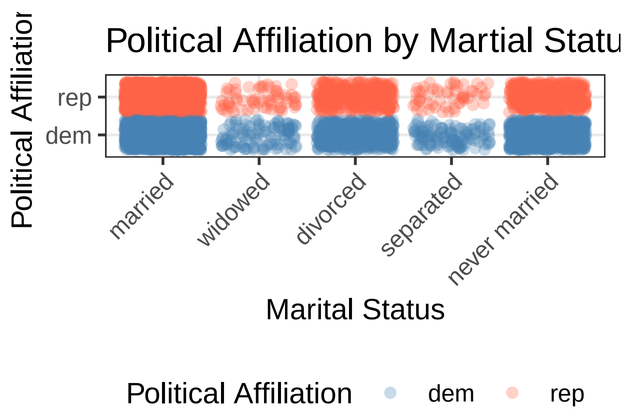


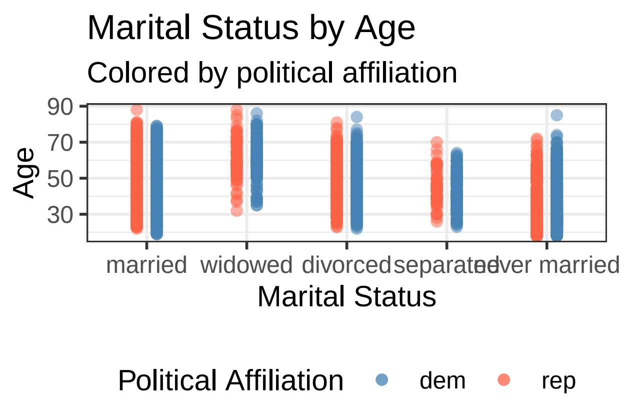


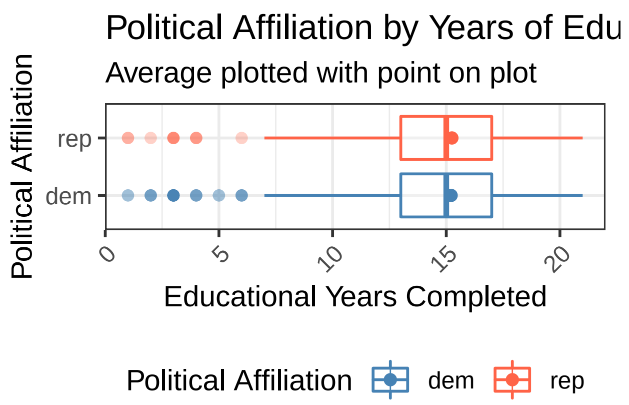


It is also relevant to look at other variables such as race, marital status, and education as factors related to political party affiliation. Most notably, there is a much larger proportion of white republicans than democrats. We can see this in the first plot in the following plots:









After completing these exploratory analyses, it is clear that while there are some weak relationships within many variables, we will likely need all of these variables to make models which have good predictive power. None of the predictors appear to have an extremely strong relationship with political party affiliation, and so we will need to use many of them for our models to perform well.

We also examined two classification model methods for accuracy in predicting `partyid` based on some of our 16 predictors. Linear discriminant analysis (LDA) appear to perform better job correctly classifying Democrats than Republicans based on these 6 predictors, as there was an equal amount of Republicans incorrectly predicted to those correctly predicted. Our logistic regression model with all 16 predictors also appears to better classify Democrats than Republicans, but not by much, with an overall training error rate of about 32%. These results suggest that the current classification models we have used throughout this course may not be successful in predicting `partyid` with high accuracy on their own. We hope to leverage these methods through model stacking in our Methods and Results section.

Results

Classical Approach for Survey Data

First, we construct a complex sample survey design.

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

Now, we fit the logistic-regression model with weights using the `svyglm()` function from the `survey` package. A slight wrinkle is that we must use the quasibinomial rather than the binomial family to avoid a warning about noninteger counts produced by the use of differential sampling weights.

After performing the appropiate calculations, we see that the training error rate is 0.3275862 for the logistic regression with weights.

Training and Testing

To compare our result to the tidymodels approach, we must perform the same analysis with a training and testing set. We do so with the same initial split used in the tidymodels approach:

```
# Set-up to calculate the test error rate
set.seed(1)
split <- initial_split(data = gss_survey, prop = 3/4)
gss_train <- training(split)
gss_test <- testing(split)

# A complex survey design based on the training data
gss_design_train <-
    svydesign( ~ vpsu ,
        strata = ~ vstrat ,
        data = gss_train ,
        weights = ~ weight ,
        nest = TRUE)
```

Now, we fit the logistic-regression model with weights using the svyglm() function to our training set and see how our model performs on the test set by calculating the test error rate. The testing error rate is 0.3627586 for the logistic regression with weights and the amount correctly predicted is 0.6372414.

Model Stacking with `tidymodels`

Our second approach to this classification problem was by using packages from the `tidymodels`, and aggregating their results with the `stacks` package (also part of the `tidymodels`.) This approach allowed us to combine the power of many of the models learned in our course and implement them with convenient syntax. The models we specified included: logistic regression, penalized logistic regression, linear discriminant analysis, random forests, and K nearest neighbors. Model stacking, unlike many other ensembling methods, relies on the fact that the models used in the stack have heterogeneous types, so we specified many different types of models.

In ordered to create our ensemble we created many models for each model type, by using 5-fold cross validation and tuning methods. We then used `stacks` to see how well each model performs on our test set and it automatically chooses which models to include in the final stack. `stacks` then assigns weights to each model and aggregates the output.

We specified our models for the stack like this:

```
# Specify random forest
rf_spec <- rand_forest(mtry = tune(),
                       min_n = tune(),
                       trees = 1000) %>%
  set_mode("classification") %>%
  set_engine("ranger")

# Workflow
rf_workflow <-
  gss_workflow %>%
  add_model(rf_spec)

# Tuning
set.seed(13)
rf_res <-
  tune_grid(
    rf_workflow,
    resamples = folds,
    grid = 3,
    control = control_stack_grid()
  )
```

This is an example of a random forest that we specified for our model stack, and since we used `tidymodels`, all of our other models used very similar syntax. This allowed us to specify many models very quickly and efficiently.

We initialize our model stack with 88 models:

```
## # A data stack with 5 model definitions and 88 candidate members:
## #   basic_logreg_resamples: 1 model configuration
## #   logreg_resamples: 65 model configurations
## #   lda_resamples: 10 model configurations
## #   rf_res: 3 model configurations
## #   knn_res: 9 model configurations
## # Outcome: partyid (factor)
```

Next, we blended our predictions and the model stack retained 11 models, here are the top ten weighted models. Interestingly, the top three models were (penalized) logistic regression models, indicating that overall logistic regression did a very good job at predicting party affiliation.

```
## # A tibble: 10 x 3
##    member                          type            weight
##    <chr>                           <chr>            <dbl>
##  1 .pred_rep_logreg_resamples_1_100 logistic_reg   1.57
##  2 .pred_rep_logreg_resamples_1_097 logistic_reg   1.36
##  3 .pred_rep_logreg_resamples_1_073 logistic_reg   0.956
##  4 .pred_rep_rf_res_1_2             rand_forest    0.911
##  5 .pred_rep_rf_res_1_3             rand_forest    0.701
##  6 .pred_rep_logreg_resamples_1_112 logistic_reg   0.417
##  7 .pred_rep_lda_resamples_1_07     discrim_linear 0.272
##  8 .pred_rep_logreg_resamples_1_118 logistic_reg   0.203
##  9 .pred_rep_logreg_resamples_1_015 logistic_reg   0.0441
## 10 .pred_rep_lda_resamples_1_04     discrim_linear 0.00112
```

Finally, we can see our results. Notably, one member, a random forest, performed slightly better than the overall stack. This is not to say that the random forest is actually better than the model stack though, as since the model stack is aggregating models, it has quite low variance.

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

Overall, the model stack did quite well, performing over 1% better than the logistic regression model fit in our first section.

Discussion

Our results demonstrate that social predictors vary highly in predicting party affiliation. The relationship between these cultural and social factors of citizens identities and major political party affiliation remains complicated, as many previous studies have shown. While our ability to predict party affiliation remains limited as a result of study, our understanding of how model stacking can improve these predictions has improved. From applying simple classification models like logistic regression and QDA to utilizing the `survey` package to predict party affiliation, `stacks` performed competitively, though always supreme. Leveraging and aggregating all of these approaches through model stacking has clearly been demonstrated its ability to improve performance under variable conditions.

Limitations of our project

Further Directions

Code Appendix

Data loading and filtering

```r
#Load data
load("gss/gss_orig.rda")
#Appropriate filtering
gss_subset <- gss_orig %>%
  filter(!stringr::str_detect(sample, "blk oversamp")) %>% # this is for weighting
  dplyr::select(year, age, sex, college = degree, partyid, hompop, hours = hrs1, income,
         class, finrela, wrkgovt, marital, educ, race, incom16, weight = wtssall, vpsu,
         vstrat) %>%
  mutate_if(is.factor, ~ fct_collapse(., NULL = c("IAP", "NA", "iap", "na"))) %>%
  mutate(
    age = age %>%
      fct_recode("89" = "89 or older",
                 NULL = "DK") %>%
      as.character() %>%
      as.numeric(),
    hompop = hompop %>%
      fct_collapse(NULL = c("DK")) %>%
      as.character() %>%
      as.numeric(),
    hours = hours %>%
      fct_recode("89" = "89+ hrs",
                 NULL = "DK") %>%
      as.character() %>%
      as.numeric(),
    weight = weight %>%
      as.character() %>%
      as.numeric(),
    partyid = fct_collapse(
      partyid,
      dem = c("strong democrat", "not str democrat"),
      rep = c("strong republican", "not str republican"),
      ind = c("ind,near dem", "independent", "ind,near rep"),
      other = "other party"
    ),
    income = factor(income, ordered = TRUE),
    college = fct_collapse(
      college,
      degree = c("junior college", "bachelor", "graduate"),
      "no degree"  = c("lt high school", "high school"),
      NULL = "dk"
    )
  ) %>%
  filter(year >= 2000) %>%
  filter(partyid %in% c("dem", "rep")) %>%
  drop_na()
```

```r
# Number of rows
nrow(gss_subset)

# Number of columns
ncol(gss_subset)

# Response variable summary
summary(gss_subset$partyid)

# Data Structure
str(gss_subset)
```

Some basic models

```r
#taking a look at how LDA could perform on our dataset with just a couple of variables
set.seed(2020)
mlda <- lda(partyid ~ race + age + year + hompop + income + wrkgovt, data = gss_subset)
mlda_pred <- predict(mlda)
conf_mlda <- table(mlda_pred$class,gss_subset$partyid)
conf_mlda


#taking a look at how logistic regression could perform on our dataset
# with just a couple of variables and then full model
simple_logreg <- glm(partyid ~ race + age + year + hompop +
                     income + wrkgovt, data = gss_subset, family= "binomial")
summary(simple_logreg)

full_logreg <- glm(partyid ~ ., data = gss_subset, family= "binomial")
summary(full_logreg)

probs<-predict(full_logreg, gss_subset, type = "response")
preds<-ifelse(probs >=.5, 1, 0)
conf_log <- table(preds, gss_subset$partyid)
conf_log

n <- length(gss_subset$partyid)
false_pos <- conf_log[1,2]
false_neg <- conf_log[2,1]
error <- 1/n *(false_pos + false_neg)
error
```

The survey regression

```r
# An alternative to specifying fpc, useful to run the regression without error
 options(survey.lonely.psu="certainty")

# Fit a logistic-regression model based on the complex survey design
glm_result <-
    svyglm(
        partyid ~ age + sex + college + hompop + hours +
          income + class + finrela + wrkgovt + marital +
        educ + race + incom16 + weight, design=gss_design, family=quasibinomial)
```

Note: The global option, options(survey.lonely.psu="fail"), makes it an error to have a stratum with a single, non-certainty PSU. Changing it to options(survey.lonely.psu="certainty"), single-PSU stratum makes no contribution to the variance (for multistage sampling it makes no contribution at that level of sampling). This is an alternative to specifying fpc, and is useful to run the regression without error.

```r
# An alternative to specifying fpc, useful to run the regression without error
 options(survey.lonely.psu="certainty")

# Fit a logistic-regression model based on the complex survey design
```

```r
glm_result <-
    svyglm(
        partyid ~ age + sex + college + hompop + hours +
          income + class + finrela + wrkgovt + marital +
        educ + race + incom16 + weight, design=gss_design, family=quasibinomial)
```

```r
# Calculation of the training error rate
probs_survey<-predict(glm_result, gss_survey, type = "response")
preds_survey<-ifelse(probs_survey >=.5, 1, 0)
conf_log_survey <- table(preds_survey, gss_survey$partyid)
conf_log_survey
n <- length(gss_survey$partyid)
false_pos_survey <- conf_log_survey[1,2]
false_neg_survey <- conf_log_survey[2,1]
error_survey <- 1/n *(false_pos_survey + false_neg_survey)
error_survey
1 - error_survey
```

```r
 options(survey.lonely.psu="certainty")

# Fit a logistic-regression model on the training data
glm_result_train <-
    svyglm( partyid ~ age + sex + college + hompop + hours +
          income + class + finrela + wrkgovt + marital +
        educ + race + incom16 + weight, design=gss_design_train, family=quasibinomial)
```

```r
#Calculation of the test error rate
probs_survey_test <- predict(glm_result_train, gss_test, type = "response")
preds_survey_test <- ifelse(probs_survey_test >=.5, 1, 0)
conf_log_survey_test <- table(preds_survey_test, gss_test$partyid)
conf_log_survey_test
n_test <- length(gss_test$partyid)
false_pos_survey_test <- conf_log_survey_test[1,2]
false_neg_survey_test <- conf_log_survey_test[2,1]
error_survey_test <- 1/n_test *(false_pos_survey_test + false_neg_survey_test)
error_survey_test
1 - error_survey_test
```

The model stacking

Let's first split our data into training and testing datasets:

```r
set.seed(1)
split <- initial_split(data = gss_subset, prop = 3/4)
gss_train <- training(split)
gss_test <- testing(split)
```

Next, let's use 5-fold cross validation:

```
folds <- rsample::vfold_cv(gss_train, v = 5)
```

Now, let's make our `recipe()` and `workflow()` that will be used for each of our models:

```
# Create the recipe
gss_recipe <- recipe(partyid ~ ., data = gss_train) %>%
  step_rm(year) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_zv(all_predictors())

# Create the workflow
gss_workflow <- workflow() %>%
  add_recipe(gss_recipe)

# View the workflow
gss_workflow
```

Now, we can begin to specify our models for our model stack:

```
# Basic Logistic regression specification
basic_logreg_spec <- logistic_reg() %>%
  set_engine("glm")

# Add logistic regression to workflow
basic_logreg_workflow <- gss_workflow %>%
  add_model(basic_logreg_spec)

# Cross validation
set.seed(13)
basic_logreg_resamples <- fit_resamples(
  basic_logreg_workflow,
  resamples = folds,
  control = control_stack_resamples()
)
```

```
# Penalized Logistic regression specification
logreg_spec <- logistic_reg(penalty = tune(),
                            mixture = tune()) %>%
  set_engine("glmnet")

# add grid
lr_reg_grid <- tidyr::crossing(
  penalty = 10 ^ seq(-6, -1, length.out = 20),
  mixture = c(0.05, 0.2, 0.4, 0.6, 0.8, 1)
)

# Add logistic regression to workflow
logreg_workflow <- gss_workflow %>%
  add_model(logreg_spec)

# Tuning hyperparameters
set.seed(13)
logreg_resamples <- tune_grid(
```

```r
  logreg_workflow,
  resamples = folds,
  grid = lr_reg_grid,
  control = control_stack_grid()
)
```

```r
# LDA specification
lda_spec <- discrim_linear(penalty = tune()) %>%
  set_engine("mda")

# Add LDA to workflow
lda_workflow <- gss_workflow %>%
  add_model(lda_spec)

# Fit with our cross validation
set.seed(13)
lda_resamples <- tune_grid(
  lda_workflow,
  resamples = folds,
  control = control_stack_grid()
)
```

```r
# Specify random forest
rf_spec <- rand_forest(mtry = tune(),
                       min_n = tune(),
                       trees = 1000) %>%
  set_mode("classification") %>%
  set_engine("ranger")

# Workflow
rf_workflow <-
  gss_workflow %>%
  add_model(rf_spec)

# tuning
set.seed(13)
rf_res <-
  tune_grid(
    rf_workflow,
    resamples = folds,
    grid = 3,
    control = control_stack_grid()
  )
```

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

knn_wf <- gss_workflow %>%
  add_model(knn_spec)

# tuning
set.seed(13)
```

```r
knn_res <-
  tune_grid(
    knn_wf,
    resamples = folds,
    control = control_stack_grid()
  )
```

Now, we can stack these models:

```r
gss_stack1 <- stacks() %>%
  add_candidates(basic_logreg_resamples) %>%
  add_candidates(logreg_resamples) %>%
  add_candidates(lda_resamples) %>%
  add_candidates(rf_res) %>%
  add_candidates(knn_res)

gss_stack1
```

```r
gss_stack2 <- gss_stack1 %>%
  blend_predictions()
gss_stack2
```

```r
gss_stack3 <- gss_stack2 %>%
  fit_members()
gss_stack3
```

```r
gss_preds <-
  gss_test %>%
  dplyr::select(partyid) %>%
  bind_cols(
    predict(
      gss_stack3,
      gss_test,
      members = TRUE
    )
  )

colnames(gss_preds) %>%
  map_dfr(
    .f = accuracy,
    truth = partyid,
    data = gss_preds
  ) %>%
  mutate(member = colnames(gss_preds)) %>%
  select(-.estimator) %>%
  rename(metric = .metric,
         estimate = .estimate) %>%
  filter(estimate != 1)
```

References

Ben-Bassat, Avi, and Momi Dahan. 2012. *Social Identity and Voting Behavior*. *Public Choice*. Vol. 151. https://doi.org/10.1007/s11127-010-9742-2.

Bray, Andrew, Chester Ismay, Evgeni Chasnovski, Ben Baumer, and Mine Cetinkaya-Rundel. 2020. *Infer: Tidy Statistical Inference*. https://CRAN.R-project.org/package=infer.

Kristensen, Jakob Baek, Thomas Albrechtsen, Emil Dahl-Nielsen, Michael Jensen, Magnus Skovrind, and Tobias Bornakke. 2017. *Parsimonious Data: How a Single Facebook Like Predicts Voting Behavior in Multiparty Systems*. *PLoS ONE*. Vol. 12. https://doi.org/10.1371/journal.pone.0184562.

Newman, Bruce I., and Jagdish N. Sheth. 1985. *A Model of Primary Voter Behavior*. *Journal of Consumer Research*. Vol. 12. https://doi.org/10.1086/208506.