Using `tidymodels`

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

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

Next, let's use 10-fold cross validation:

```
folds <- rsample::vfold_cv(gss_train, v = 10)
```

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
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: None
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_rm()
## * step_dummy()
## * step_zv()
```

To asses our models, we will use the proportion of correct predictions made by each model. We can set this metric to be used easily with `yardstick`:

```
# metric <- metric_set(precision)
```

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

```
# Logistic regression specification
logreg_spec <- logistic_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet")
# add grid
lr_reg_grid <- tibble(penalty = 10^seq(-4, -1, length.out = 30))
```

```r
# Add logistic regression to workflow
logreg_workflow <- gss_workflow %>%
  add_model(logreg_spec)
# Fit with our cross validation
set.seed(13)
logreg_resamples <- tune_grid(
  logreg_workflow,
  resamples = folds,
  grid = lr_reg_grid,
  # metrics = metric,
  control = control_stack_grid()
)
```

```
##
## Attaching package: 'rlang'

## The following objects are masked from 'package:purrr':
##
##     %@%, as_function, flatten, flatten_chr, flatten_dbl, flatten_int,
##     flatten_lgl, flatten_raw, invoke, list_along, modify, prepend,
##     splice

##
## Attaching package: 'vctrs'

## The following object is masked from 'package:dplyr':
##
##     data_frame

## The following object is masked from 'package:tibble':
##
##     data_frame

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.0-2

## The workflow being saved contains a recipe, which is 0.4 Mb in memory. If this was not intentional,
```

```r
# LDA specification
lda_spec <- discrim_linear(penalty = tune()) %>%
  set_engine("MASS")
# Add LDA to workflow
```

```
lda_workflow <- gss_workflow %>%
  add_model(lda_spec)
# Fit with our cross validation
set.seed(13)
lda_resamples <- fit_resamples(
  lda_workflow,
  resamples = folds,
  # metrics = metric,
  control = control_stack_grid()
)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## The workflow being saved contains a recipe, which is 0.4 Mb in memory. If this was not intentional, p
```

Now, we can stack these models:

```
gss_stack <- stacks() %>%
  add_candidates(logreg_resamples) %>%
  add_candidates(lda_resamples) %>%
  blend_predictions() %>%
  fit_members()
```

```
## Warning: Predictions from the candidates c(".pred_dem_logreg_resamples_1_02",
## ".pred_dem_logreg_resamples_1_03", ".pred_dem_logreg_resamples_1_04",
## ".pred_rep_logreg_resamples_1_02", ".pred_rep_logreg_resamples_1_03",
## ".pred_rep_logreg_resamples_1_04") were identical to those from existing
## candidates and were removed from the data stack.
```

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

colnames(gss_preds) %>%
  map_dfr(
    .f = accuracy,
    truth = partyid,
    data = gss_preds
  ) %>%
  mutate(member = colnames(gss_preds))
```

```
## # A tibble: 4 x 4
##   .metric  .estimator .estimate member
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         1     partyid
## 2 accuracy binary         0.634 .pred_class
## 3 accuracy binary         0.642 .pred_class_logreg_resamples_1_17
## 4 accuracy binary         0.639 .pred_class_logreg_resamples_1_18
```

For a base-line, let's fit a simple logistic regression:

```
lr_baseline <- logistic_reg(mode = "classification") %>%
  set_engine("glm") %>%
  fit(partyid ~ ., data = gss_train)

lr_preds <- lr_baseline %>%
  predict(new_data = gss_test) %>%
  bind_cols(gss_test %>% select(partyid))

logreg_preds %>%
  precision(partyid, .pred_class)
```