Presidents2016

Isaac Wilfong

2025-02-22

Introduction

For this post, I wanted to predict each county in the presidential election of 2016. In this document, I will mainly be using the tidymodels framework. I will talk about setbacks to models. The idea isn't getting the absolute best model. Each model will have set backs but rather providing a base of knowledge in how I would tackle this problem.

```
library(pacman)
p_load(tidyverse, modeldata, skimr, janitor, tidymodels, magrittr)
```

Step One

Before doing any machine learning method, it is important to know what the data you are working with is. As well to load that data into R.

```
AGAM <- read.csv("election-2016.csv")
RED <- AGAM %>% select(-state, -county)
columns <- colnames(RED)
columns
head(RED)
#skim(RED)
sum(AGAM$i_republican_2016)
length(AGAM$fips)
```

Lucky for us in this scenario is the data is as perfect as it gets. There are no missing variables. The data set includes 33 variables to work with. In a perfect scenario, I would want more in depth variables but this is what we have with this data set.

Five Fold Cross Validation

For all my regression based models, I will be using five fold cross validation. I will be separating my training data into five sub samples and I will test my model on each one while training on the other four.

```
#setting up the 5 fold
set.seed(8008)
RED_cv = RED %>% vfold_cv(v = 5)
```

Elasticnet Regression

For the first method, I am going to use Lasso regression. I want to tune the hyperparameter for penalizing my different variables. I am going to pick 5 I think are good predictors for a county to go Republican during an election. Elasticnet combines both Lasso and Ridge so not only can I tune the penalty but also the mixture.

```
#defining our lambdas
lambdas = 10^seq(from = 5, to = -2, length = 1e2)
alphas = seq(from = 0, to = 1, by = 0.1)
#setting up a recipe
penalized_recipe = recipe(i_republican_2016 ~ pop_pct_hispanic  + pop_pct_veteran +
                            pop_pct_homeowner + land_area_mi2, data=RED) %>%
  step_normalize(all_numeric_predictors())
# There is no missing data so there is no reason to imputate the data.
#setting up our model
Penalized_model =
  linear_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet")
```

```
#workflow

fitred = workflow() %>%
  add_model(Penalized_model) %>%
  add_recipe(penalized_recipe)

#tuning grid
RED_Grid =
  fitred %>%
  tune_grid(
    RED_cv,
    grid = expand_grid(mixture = alphas, penalty = lambdas),
    metrics = metric_set(rmse)
  )
```

```
#Finding my best metric for elasticnet
RED_Grid %>% collect_metrics()
```

```
## # A tibble: 1,100 x 8
##    penalty mixture .metric .estimator  mean     n std_err .config
##      <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1  0.01         0 rmse    standard   0.322     5 0.00789 Preprocessor1_Model00~
##  2  0.0118       0 rmse    standard   0.322     5 0.00789 Preprocessor1_Model00~
##  3  0.0138       0 rmse    standard   0.322     5 0.00789 Preprocessor1_Model00~
##  4  0.0163       0 rmse    standard   0.322     5 0.00789 Preprocessor1_Model00~
##  5  0.0192       0 rmse    standard   0.322     5 0.00785 Preprocessor1_Model00~
##  6  0.0226       0 rmse    standard   0.322     5 0.00782 Preprocessor1_Model00~
##  7  0.0266       0 rmse    standard   0.322     5 0.00778 Preprocessor1_Model00~
##  8  0.0313       0 rmse    standard   0.322     5 0.00774 Preprocessor1_Model00~
##  9  0.0368       0 rmse    standard   0.322     5 0.00769 Preprocessor1_Model00~
## 10  0.0433       0 rmse    standard   0.322     5 0.00764 Preprocessor1_Model00~
## # i 1,090 more rows
```

```
best = RED_Grid %>% select_best()
best
```

```
## # A tibble: 1 x 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1    0.01       0 Preprocessor1_Model0001
```

Logistic Regression

```
#For logistic we need to overwrite the data set
RED$PoliticalParty <- ifelse(RED$i_republican_2016 == 1, "Republican", "Democrat")
#RE do our 5 fold
RED_cv = RED %>% vfold_cv(v = 5) # I am aware I overwrote this variable
#Logistic Model
Logistic_model =
  logistic_reg("classification") %>%
  set_engine("glm")
#Logistic Recipe
Logistic_recipe = recipe(PoliticalParty ~ pop_pct_hispanic  + pop_pct_veteran + pop_pct_homeowner + lan
#Logistic WorkFlow
Logistic_WF = workflow() %>%
  add_model(Logistic_model) %>%
  add_recipe(Logistic_recipe) %>%
  fit_resamples(
    resamples = RED_cv,
    metrics = metric_set(accuracy,precision, specificity, sensitivity, roc_auc)) #add the other metrics
```

```
## > A | warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## There were issues with some computations   A: x1There were issues with some computations   A: x3Ther
```

```
#Collecting Metrics

Logistic_WF %>% collect_metrics(summarize = TRUE)
```

```
## # A tibble: 5 x 6
##   .metric     .estimator  mean     n std_err .config
##   <chr>       <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy    binary     0.876     5 0.00579 Preprocessor1_Model1
## 2 precision   binary     0.741     5 0.0233  Preprocessor1_Model1
## 3 roc_auc     binary     0.824     5 0.00686 Preprocessor1_Model1
## 4 sensitivity binary     0.338     5 0.0213  Preprocessor1_Model1
## 5 specificity binary     0.978     5 0.00314 Preprocessor1_Model1
```

Logistic Lasso

I can combine the Logistic regression with a Lasso regression. This way I can add a penalty to my regression and get the benefit of using logistic regression on our binary dependent variable.

```r
Lasso_Logistic_Model =
  logistic_reg(penalty = tune()) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

Lasso_Logistic_WF = workflow() %>%
  add_model(Lasso_Logistic_Model) %>%
  add_recipe(Logistic_recipe)

Lasso_Log_Grid =
  Lasso_Logistic_WF %>%
  tune_grid(
    RED_cv,
    grid = expand_grid(penalty = lambdas),
    metrics = metric_set(accuracy,precision, specificity, sensitivity, roc_auc)
  )

Lasso_Log_Grid %>% collect_metrics(summarize = TRUE)
```

```
## # A tibble: 500 x 7
##    penalty .metric     .estimator  mean     n std_err .config
##      <dbl> <chr>       <chr>      <dbl> <int>   <dbl> <chr>
##  1  0.01   accuracy    binary     0.874     5 0.00596 Preprocessor1_Model001
##  2  0.01   precision   binary     0.767     5 0.0352  Preprocessor1_Model001
##  3  0.01   roc_auc     binary     0.823     5 0.00713 Preprocessor1_Model001
##  4  0.01   sensitivity binary     0.292     5 0.0149  Preprocessor1_Model001
##  5  0.01   specificity binary     0.983     5 0.00313 Preprocessor1_Model001
##  6  0.0118 accuracy    binary     0.873     5 0.00619 Preprocessor1_Model002
##  7  0.0118 precision   binary     0.772     5 0.0328  Preprocessor1_Model002
##  8  0.0118 roc_auc     binary     0.823     5 0.00716 Preprocessor1_Model002
##  9  0.0118 sensitivity binary     0.282     5 0.0149  Preprocessor1_Model002
## 10  0.0118 specificity binary     0.984     5 0.00281 Preprocessor1_Model002
## # i 490 more rows
```

```r
best_Log_Lasso = Lasso_Log_Grid %>% select_best()
best_Log_Lasso
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <chr>
## 1    0.01 Preprocessor1_Model001
```

Random Forests

I can not only use regression style approaches to solve this problem but I can also use random forests. Random forests is an assortment of decision trees that can help classify our predictors up to solve our prediction problem.

```r
AGAM <- AGAM %>% select(-state, -county)
AGAM$PoliticalParty <- ifelse(RED$i_republican_2016 == 1, "Republican", "Democrat")
```

```r
RF_Recipe = penalized_recipe = recipe(PoliticalParty ~ ., data=AGAM) %>%
  step_normalize(all_predictors())

AGAM_cv = AGAM %>% vfold_cv(v = 5)

RF_Model =
  rand_forest(mtry = 10, trees = 500) %>%
  set_engine("randomForest") %>%
  set_mode("classification")

RF_WF = workflow() %>%
  add_model(RF_Model) %>%
  add_recipe(RF_Recipe)


RF_Results <- fit_resamples(
  RF_WF,
  resamples = AGAM_cv,
  metrics = metric_set(accuracy),
  control = control_resamples(save_pred = TRUE)
)

 RF_Results %>% collect_metrics()
```

```
## # A tibble: 1 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary         1     5       0 Preprocessor1_Model1
```

```r
 show_notes(RF_Results)
```

```
## Great job! No notes to show.
```

That is how you could use a random forest, Ridge regression, Lasso regression, and a Logistic regression to solve a prediction problem as it relates to county elections. I understand the models aren't refined. That wasn't the point. The point was to show how you would use these different methods in the tidymodel framework.

If you are a future employer, I really appreciate you reading this far in. If you ask any questions about my work I would love to talk more in depth with you about this.