Penalized regression, logistic regression, and classification

Tom Ben-Shahar

February 22, 2025

```r
# Load Packages
pacman::p_load(tidyverse, tidymodels, magrittr, skimr, glmnet)

# Load Data
election <- read.csv("data/election.csv")

# Clean Data
election %<>% mutate(
  i_republican_2016_f = factor(i_republican_2016),
  i_republican_2012 = factor(i_republican_2012)
)

# Set Constants
set.seed(93284298)
nfold = 5
lambdas = 10^seq(from = 5, to = -2, length = 1e3)
alphas = seq(from = 0, to = 1, by = 0.05)
```

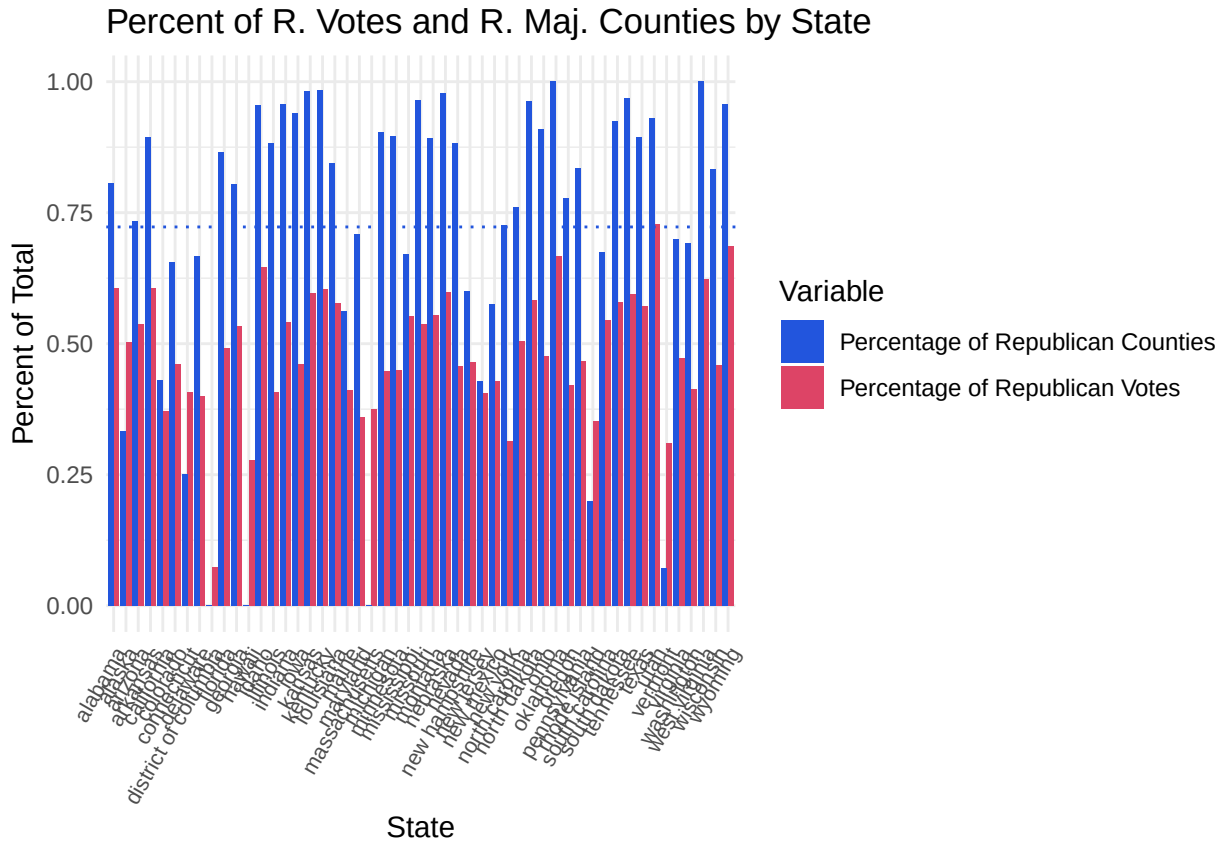
Data Overview

Create some nice plots of the raw data and brief discussion

```r
# Percent of Republican Votes by State
fig_1_df <- election %>%
  group_by(state) %>%
  summarise(
    pct_r_counties = mean(i_republican_2016),
    tot_votes_r = sum(n_votes_republican_2012),
    tot_votes = sum(n_votes_total_2012)
  ) %>%
  mutate(
    pct_r_votes = tot_votes_r / tot_votes
  )

long_df <- fig_1_df %>%
  pivot_longer(cols = c(pct_r_votes, pct_r_counties),
               names_to = "Variable",
               values_to = "Value")
```

```r
# Create the bar chart
ggplot(long_df, aes(x = state, y = Value, fill = Variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Percent of R. Votes and R. Maj. Counties by State",
       x = "State",
       y = "Percent of Total",
       fill = "Variable") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1, size = 8)) +
  scale_fill_manual(values = c("pct_r_counties" = "#25d", "pct_r_votes" = "#d46"),
                    labels = c("Percentage of Republican Counties", "Percentage of Republican Votes")) +
  geom_hline(yintercept = mean(fig_1_df$pct_r_counties), linetype = "dotted", color = "#25d", )
```



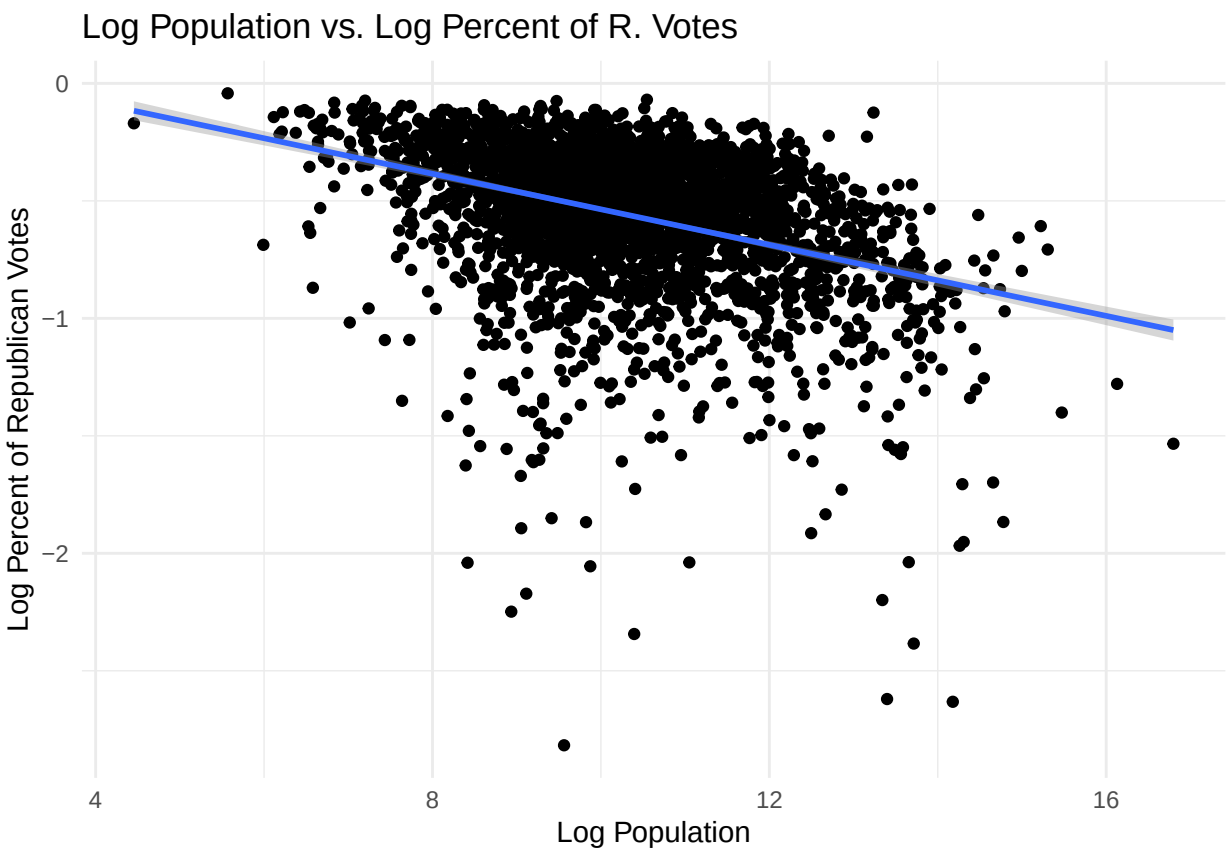
What I find interesting about this visualization is that the proportion of republican-majority counties is consistently larger than the proportion of republican votes, indicating that republican votes may be disproportionately represented. In other words, republicans seem to be winning more counties than they "should" given their voting population. Given the American democratic system, this is expected.

```r
# Plot population vs. pct of r votes
election_plot_df <- election %>%
  mutate(
  pct_r_votes = n_votes_republican_2012 / n_votes_total_2012
  )

ggplot(data = election_plot_df, aes(x = log(pop), y = log(pct_r_votes))) +
  geom_point()+
```

```
  geom_smooth(method='lm') +
  labs(
    title = "Log Population vs. Log Percent of R. Votes",
    x = "Log Population",
    y = "Log Percent of Republican Votes"
  ) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



This shows a weakly negative relationship between population and republican votes, suggesting more populous areas are more likely democratic-aligned. This is expected.

Penalized Regression

Using 5-fold cross validation, tune a Lasso model

```
# Recipe
rec_1 <- recipe(data = election, i_republican_2016 ~ .) %>%
```

```
  update_role(fips, new_role = "id") %>%
  step_rm(i_republican_2016_f) %>%
  step_novel(county, state) %>%
  step_dummy(i_republican_2012, county, state) %>%
  step_normalize(all_numeric_predictors())

# Model : Lasso
mod_lasso <- linear_reg(penalty = tune(),mixture = 1) %>%
  set_engine("glmnet")

# Workflow
wkfl_lasso <- workflow() %>%
  add_model(mod_lasso) %>%
  add_recipe(rec_1)

# Cross Validate
elec_cv <- election %>% vfold_cv(v = nfold)

cv_lasso <- wkfl_lasso %>%
  tune_grid(
    elec_cv,
    grid = data.frame(penalty = lambdas),
    metrics = metric_set(rmse)
  )
```

```
## > A | warning: !  The following columns have zero variance so scaling cannot be used:
##                  county_new and state_new.
##                i Consider using ?step_zv (`?recipes::step_zv()`) to remove those columns
##                  before normalizing.
```

```
## There were issues with some computations   A: x1There were issues with some computations   A: x2There
```

```
# Find optimal penalty

cv_lasso %>% show_best()
```

```
## # A tibble: 5 x 7
##   penalty .metric .estimator  mean     n std_err .config
##     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1  0.0103 rmse    standard   0.204     5 0.00645 Preprocessor1_Model0003
## 2  0.0102 rmse    standard   0.204     5 0.00647 Preprocessor1_Model0002
## 3  0.0105 rmse    standard   0.204     5 0.00643 Preprocessor1_Model0004
## 4  0.01   rmse    standard   0.204     5 0.00650 Preprocessor1_Model0001
## 5  0.0107 rmse    standard   0.204     5 0.00641 Preprocessor1_Model0005
```

Highest Performing Lambda:

$$\lambda = 0.0107$$

Which metric was used? RMSE was the metric used. The error looks fairly high, at 0.205. This isn't great, as it would seem that even a null classifier would have a similar RMSE.

Tune an Elasticnet model

```
# Elasticnet Model
mod_ent <- linear_reg(
  penalty = tune(), mixture = tune()
) %>%
  set_engine(
    "glmnet"
  )

# Workflow
wkfl_ent <- workflow() %>%
  add_model(mod_ent) %>%
  add_recipe(rec_1)

# Cross validating alphas and lambdas
cv_ent <- wkfl_ent %>%
  tune_grid(
    elec_cv,
    grid = expand_grid(mixture = alphas, penalty = lambdas),
    metrics = metric_set(rmse)
  )
```

```
## > A | warning: !  The following columns have zero variance so scaling cannot be used:
##                  county_new and state_new.
##                i Consider using ?step_zv (`?recipes::step_zv()`) to remove those columns
##                  before normalizing.
```

```
## There were issues with some computations   A: x1There were issues with some computations   A: x2There
```

```
cv_ent %>% show_best()
```

```
## # A tibble: 5 x 8
##   penalty mixture .metric .estimator  mean     n std_err .config
##     <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1  0.0168    0.6  rmse    standard   0.204     5 0.00623 Preprocessor1_Model120~
## 2  0.0155    0.65 rmse    standard   0.204     5 0.00628 Preprocessor1_Model130~
## 3  0.0182    0.55 rmse    standard   0.204     5 0.00619 Preprocessor1_Model110~
## 4  0.0165    0.6  rmse    standard   0.204     5 0.00626 Preprocessor1_Model120~
## 5  0.0179    0.55 rmse    standard   0.204     5 0.00622 Preprocessor1_Model110~
```

Highest Performing Lambda and Alpha:

$$\lambda = 0.0125,\ \alpha = 0.85$$

The algorithm tuned α to 0.85, indicating that a Lasso regression with a weak Ridge regression effect performs well with this data.

Logistic Regression

Use 5-fold cross validation to tune a logistic regression model

```
# I did logistic lasso first by mistake, this code is the simplified and repurposed code from the next

mod_log0 <- logistic_reg(
  penalty = 0.2
  # I think that just defining penalty arbitrarily is what makes it not logistic lasso..?
) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

# Define data
election_log_df0 <- election %>%
  select(-i_republican_2016)

# Recipe
rec_20 <- recipe(data = election_log_df0, i_republican_2016_f ~ .) %>%
  #step_select(-i_republican_2016) %>%
  update_role(fips, new_role = "id") %>%
  step_novel(county, state) %>%
  step_dummy(i_republican_2012, county, state) %>%
  step_normalize(all_numeric_predictors())

wkfl_log0 <- workflow() %>%
  add_model(mod_log0) %>%
  add_recipe(rec_20)

# # Cross validating alphas and lambdas
# cv_log0 <- wkfl_log0 %>%
#   tune_grid(
#     elec_cv,
#     grid = expand_grid(mixture = alphas, penalty = lambdas),
#     metrics = metric_set(accuracy)
#   )

# cv_log0 %>% show_best()
# cv_log %>% collect_metrics()

# Obtain Predictions
log_fit0 <- wkfl_log0 %>%
#  finalize_workflow(select_best(cv_log0, metric = 'accuracy')) %>%
  fit(data = election_log_df0)
```

```
## Warning: !  The following columns have zero variance so scaling cannot be used:
##   county_new and state_new.
## i Consider using ?step_zv (`?recipes::step_zv()`) to remove those columns
##   before normalizing.
```

```
log_pred0 = predict(log_fit0, new_data = election)

election_log_df0 %<>%
  mutate(
    pred = log_pred0$.pred_class
  )

# Find accuracy, precision, specificity, sensitivity, ROC AUC

accuracy(election_log_df0, i_republican_2016_f, pred)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.842
```

```
sensitivity(election_log_df0, i_republican_2016_f, pred)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 sensitivity binary             0
```

```
specificity(election_log_df0, i_republican_2016_f, pred)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 specificity binary             1
```

```
election_log_df0 %<>% mutate(
  i_republican_2016_f = factor(i_republican_2016_f),
  pred = as.numeric(pred)
)
roc_auc(election_log_df0, i_republican_2016_f, pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary           0.5
```

Outcome Statistics

$$\text{Accuracy} = 0.8421 \text{Sensitivity} = 0 \text{Specificity} = 1 \text{ROC AUC} = 0.5$$

Discussion

This accuracy is not great. As we saw in the first graph, about 73% of counties vote Republican. So even the null classifier would be right 73% of the time. This only improves upon that by about 10%, which is not much. The sensitivity was found to equal 0, which implies that the model is not getting any true positives. This implies that the model may just be guessing negative 100% of the time (though this is not likely actually the case, because accuracy is relatively better).

Logistic Lasso Regression

Use 5-fold cross validation to tune a logistic lasso regression model

```
mod_log <- logistic_reg(
  penalty = tune(),
  mixture = tune()
) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

# Define data
election_log_df <- election %>%
  select(-i_republican_2016)

# Recipe
rec_2 <- recipe(data = election_log_df, i_republican_2016_f ~ .) %>%
  #step_select(-i_republican_2016) %>%
  update_role(fips, new_role = "id") %>%
  step_novel(county, state) %>%
  step_dummy(i_republican_2012, county, state) %>%
  step_normalize(all_numeric_predictors())

wkfl_log <- workflow() %>%
  add_model(mod_log) %>%
  add_recipe(rec_2)

# Cross validating alphas and lambdas
cv_log <- wkfl_log %>%
  tune_grid(
    elec_cv,
    grid = expand_grid(mixture = alphas, penalty = lambdas),
    metrics = metric_set(accuracy)
  )
```

```
## > A | warning: !  The following columns have zero variance so scaling cannot be used:
##                  county_new and state_new.
##                i Consider using ?step_zv (‘?recipes::step_zv()‘) to remove those columns
##                  before normalizing.

## There were issues with some computations   A: x1There were issues with some computations   A: x2There
```

```
cv_log %>% show_best()
```

```
## Warning in show_best(.): No value of ‘metric‘ was given; "accuracy" will be
## used.

## # A tibble: 5 x 8
```

```
##   penalty mixture .metric  .estimator  mean     n std_err .config
##     <dbl>   <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1  0.0427    0.25 accuracy binary     0.966     5 0.00410 Preprocessor1_Model05~
## 2  0.0434    0.25 accuracy binary     0.966     5 0.00410 Preprocessor1_Model05~
## 3  0.0441    0.25 accuracy binary     0.966     5 0.00410 Preprocessor1_Model05~
## 4  0.0448    0.25 accuracy binary     0.966     5 0.00410 Preprocessor1_Model05~
## 5  0.0456    0.25 accuracy binary     0.966     5 0.00410 Preprocessor1_Model05~
```

```
# cv_log %>% collect_metrics()

# Obtain Predictions
log_fit <- wkfl_log %>%
  finalize_workflow(select_best(cv_log, metric = 'accuracy')) %>%
  fit(data = election_log_df)
```

```
## Warning: !  The following columns have zero variance so scaling cannot be used:
##   county_new and state_new.
## i Consider using ?step_zv (`?recipes::step_zv()`) to remove those columns
##   before normalizing.
```

```
log_pred = predict(log_fit, new_data = election)

election_log_df %<>%
  mutate(
    pred = log_pred$.pred_class
  )

# Find accuracy, precision, specificity, sensitivity, ROC AUC

accuracy(election_log_df, i_republican_2016_f, pred)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.968
```

```
sensitivity(election_log_df, i_republican_2016_f, pred)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 sensitivity binary         0.850
```

```
specificity(election_log_df, i_republican_2016_f, pred)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 specificity binary         0.990
```

```r
election_log_df %<>% mutate(
  i_republican_2016_f = factor(i_republican_2016_f),
  pred = as.numeric(pred)
)
roc_auc(election_log_df, i_republican_2016_f, pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary        0.0802
```

Outcome Statistics

Confusion Matrix Statistics:

$$\text{Accuracy} = 0.9688 \text{Sensitivity} = 0.8801 \text{Specificity} = 0.9855 \text{ROC AUC} = 0.0672$$

Highest Performing Lambda and Alpha:

$$\lambda = 0.0247,\ \alpha = 0.45$$

Discussion

I think this accuracy is pretty good! Much closer to the ideal 100%. Sensitivity is now positive, and neither sensitivity nor specificity have taken integer values, implying that the model is actually guessing positive or negative with some consideration. I would be interested to see how this model translates to predicting ultimate electoral college outcomes. Logistic elasticnet would seem to provide the best results, as parameters can all be tuned optimally.

Reflection

Why might we prefer Lasso to elasticnet (or vice versa)? It seems that elasticnet likely dominates lasso in most settings, as it simply allows for an additional dimension of optimization through tuning more parameters.

What are the the differences between logistic regression and linear regression? What are the similarities? Logistic regressions fit a logistic function rather than a line. This may be preferable in classification settings, as the logistic function can be manipulated to more closely approximate the step function. The step function which our logistic regression seeks to approximate gives a binary output, allowing for a model more suited to classification.

Imagine you worked for a specific political party. In this setting do you care about accuracy, precision, sensitivity, or something else? Explain. Even working for a specific party, there is not much difference between a false positive and false negative (at least not in a scenario I can think of). Therefore accuracy is the optimal metric to measure the model's effectiveness at predicting election outcomes, regardless of party.

Do you think that your estimated test accuracy will apply to the 2020 or 2024 (or future) presidential elections? Why or why not? Probably not. If the model could be trained on several election cycles over a larger number of years, then it would probably do a much better job predicting future elections. However, the provided training data only covers two cycles of election outcomes. The political landscape changes over 4 years, and without a larger history to inform a prediction about that change, the model is unlikely to apply to the future or past elections. Even a model trained on *every* election would likely struggle, as political attitudes have morphed rapidly in recent years.

Does the standard "random sampling" approach to cross validation make sense with these data? If not: What approach could make more sense? You would probably want to choose your sample in such a way that each county's proportional representation is unchanged throughout the validation folds. So if "County A" has 100 people, 20 of those people will be assigned to each fold. This would ensure that counties are not randomly left out of some folds.

Using *math* (not code): Derive what *R-squared* tells us when the outcome is binary and the prediction is also binary.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

If the data is unbalanced (meaning more T than F, or vice versa), then the denominator will be smaller, increasing R^2 . So for an application like this, where most counties are republican, R^2 will be higher than it would be with an equivalent model trained on balanced data.

Similarly, derive what *R-squared* tells us when the outcome is binary and the prediction is a probability. If the prediction is a probability then the numerator will be larger as erroneous guesses will be closer to the correct value. This will increase R^2 compared to when the prediction is binary.

What is the most interesting concept you've learned in this class? Why do you find it interesting/important? I am really enjoying learning about these models and using them for prediction. I feel (perhaps overzealously) that I can predict anything, given the right data. I am already using skills from this class in other research endeavors as I am learning them, which is really igniting my passion as I continue my academic career. **10/10 class**.