17.835: Problem Set 4

Professor: In Song Kim
TA: Jesse Clark, Sean Shiyao Liu, and Nicole Wilson

Due: Wednesday October 21, 11:00 AM
(Late submissions will not be accepted)

Please post all questions to the Piazza discussion forum, which you can access through the link on the class website or at:

https://piazza.com/mit/fall2020/17835

For this and future problem sets, you will need to submit two files onto the Canvas (under `Assignment`): 1) your write-up with your answers to all questions (including codes and results for computational questions and any related graphics) as a PDF file, and 2) your code as an `R` file (so that we can verify it runs without errors). Both files should be identified with your last name (e.g., `clark.R` and `clark.pdf`). Please ensure that all of these are completed *before* class on the day of the due date – late submissions will be automatically flagged to us by Canvas. For problems that require calculations, please show all the steps of your work. For problems involving `R`, please ensure that your code is thoroughly commented.

This week's problems focus on applying supervised machine learning techniques. As we progress through the problems, you will learn about using OLS and SVM to predict US legislators' roll-call voting behaviors. We also apply LASSO model to predict the presidential election returns. For all the plots in this problem set, you may use the base R graphics or the `ggplot2()` library in R (more information on `ggplot` can be found here `http://ggplot2.tidyverse.org/`). Another package you may want to install and use for this problem set is `matrixStats`. You can do the whole problem set, however, using base `R`, that is without `ggplot2`. You may also use the machine learning packages such as `e1071` and `glmnet`.

Problem 1: Supervised Learning and Congressional Roll-Call

As we saw in lecture, the OLS model may suffer from over-fitting problem if we are dealing with a large number of variables with a small training set (See Slide 6 from the lecture note). Predicting legislators' roll-call voting behaviors represents a case in which the problem of over-fitting arises. The US Congress has a fixed number of House of Representatives (currently 435) while researchers often include a variety of variables such as gender, age, race/ethnicity, and partisanship of the legislators as well as the social and economic profiles of their districts to model the roll-call voting. In this problem, we will practice using supervised learning approaches to studying congressional roll call vote data (this is where members of Congress vote on bills to potentially become law). We will focus on the final passage vote of the Tax Cuts and Jobs Act of 2017 in the House of Representatives.

Load in the `house_votes.csv` dataset. Each row corresponds to a particular member of the House of Representatives in the 115th Congress. We won't list out all of the variables. The variables that begin with `vote_` followed by a number (x) are indicators for whether that member voted yay on the xth bill in that congress. `vote_` variable excludes the Tax Cuts and Jobs Act of 2017. Other variables summarize demographic characteristics of the member such as age, gender, race and the demographics

of their districts (You may consult the Appendix if you want to know more about the demographic variables in the data set)

Please note that in this data, several house members have been removed due to missing data. In this problem set, we're just going to focus on the members in the data.

1. First, calculate the number of members who voted for the tax bill from each party (we've renamed the tax bill vote `outcome` for convenience: 1 = "Yay", 0 = "Nay"). What do you observe?

2. Now, we are more interested in the variation within a party. We observed that a small portion of Republicans voted *against* the bill. Our goal is to apply supervised-learning models to detect those Republican "defectors."

   Before applying the model, we need to create a training set and a test set. Subset the dataset to include only Republican legislators. Then, randomly sample 160 observations from the 203 GOP members *without replacement* and treat the sampled observations as the training data. To make sure that we can replicate your analysis, make sure that you *set seed equal to* 02139. The remaining 43 observations will be the test data. Political science scholars often pay attention to three groups of variables that predict legislators roll-call behaviors: legislators' personal characteristics, the preferences of the congressional district they represented, and legislators' past voting records. To illustrate, an African American member who represents a largely Democratic district is likely to vote for a bill that promotes racial equality and legislator who voted against Obamacare is likely to vote for the tax-cut bill. In this problem, we are interested in all three types of variables. So we wish to include as many variables as we can. This is often called a "kitchen-sink" approach.

   To create a "kitchen-sink model," select colunm 499 to column 657 from your training set as the predictors. Those columns include all legislators' personal characteristics except for partisanship, the demographic characteristics of the congressional districts, and 140 roll-call actions prior to the tax bill votes. You will have 159 variables in your matrix. This will push the OLS model to its limit.

3. Regress the outcome variable on all 159 variables. To avoid explicitly writing out 159 covariates by hand, you may want to combine the outcome vector with the matrix that contains 159 predictors and then use the `lm` function below:

   ```
   lm(outcome ~.   , data = your_combined_matrix)
   ```

   Report the *unadjusted* R^2 of the model. What does the R^2 say about the model?

4. Use the fitted OLS model to predict the test data. You may want to use the test set to generate the predicted probabilities in the first place. Then create a binary vector which indicates if the OLS prediction is greater than or equal to 0.5:

$$\hat{Y}\_vote = \begin{cases} 1, & \text{if } \hat{Y}\_ols \geq 0.5 \\ 0, & \text{if } \hat{Y}\_ols < 0.5 \end{cases}$$

   Note in this process, it is possible that your prediction goes beyond the range 0-1, because the linear probability model (OLS) does not prevent this from happening, and is another reason why OLS might not be ideal for this kind of prediction problem. Compute the number of correct predictions, precision, recall, and F_1 score of the fitted OLS model. To caculate the precision, recall, and F_1

score, you can consult Slide 25 of the lecture note. In this problem we define false positive as the case in which a legislator voted *against* the bill in reality while the predicted vote is 1, and false negative as the case in which the actual vote was 1 while the predicted vote was 0. Based on those statistics, what can you conclude about the performace of OLS? Also, did OLS model successfully detected all Republican "defectors?"

5. Now, we turn to the SVM model and see whether SVM outperforms the OLS. For the SVM model, we use the same training set and test set. Use the `svm()` function from the `e1071` package to estimate an SVM on the training data. Choose the default `radial` kernel. The SVM model should include the same 159 predictors specified in the previous question. Then, plug in the test data and obtain the predicted voted based on the estimated SVM model.

6. Compute the number of correct predictions, precision, recall, and F_1 score of the SVM model. Did the SVM model successfully identified the GOP defectors? Based on the statistics, did SVM outperform OLS in terms of prediction?

7. ***Extra Credit***: Now we want to see if the Random Forest algorithm can successfully predict GOP legislators' voting behaviors. Use `randomForest` function from the `randomForest` with the same training and test set. Then report number of correct predictions, precision, recall, and F_1 score of the Random Forest model. Does the Random Forest algorithm outperform OLS and SVM?

Problem 2: LASSO and Predicting Presidential Election Returns

In this problem, we will practice using LASSO to predict continuous outcomes. In this case, we'll develop a LASSO model to predict the proportion of voters in each county who voted for Trump in 2016.

The dataset[1] contains data on the vote share for President Trump in the 2016 election for every county, as well as many political and demographic covariates of that county. Aside from details in the Appendix, here are key covariates of the dataset:

- **FIPS**: Five digit unique identifier for each county
- **Geographic.Name**: Name of each county
- **Total.Vote**: Total counts of votes in each county
- **Hillary.Clinton**: Total counts of votes for Hilary Clinton in each county
- **Donald.J..Trump**: Total counts of votes for Donald J Trump in each county

In this problem, we will walk you through LASSO estimation and the cross-validation process. We will also learn that LASSO may also be subject to the problem of over-fitting if not implemented properly.

You may find `glmnet` package helpful for this problem.

1. Load the `pres_results_by_county_train.csv` dataset. The data contain vote share for presidential candidates Clinton and Trump, as well as a lot of demographic information. We will use this dataset as our training data. The dataset contains information from 2699 counties. After loading the data, create a new variable `trump.share` which equals the proportion of votes Trump

[1] The dataset comes from Kaufman, Aaron Russell, Peter Kraft, and Maya Sen. "Improving Supreme Court Forecasting Using Boosted Decision Trees." *Political Analysis* 27, no. 3 (2019): 381-387.

received in 2016 out of the total number of votes in each county. Exclude cases where Trump's voteshare is NA due to data missingness. Summarize the vote share for Trump in each county with function `summary`. Does this summary contradicts with the fact that Clinton won the popular votes? Explain.

2. Estimate a LASSO model for Trump's voteshare with the function `glmnet` without cross-validation. Use all of the 9^{th} column to 59^{th} column as predicting covariates in the model. For example,

```
#  mod_lasso <- glmnet(y = # your outcome vector,
#                      x = # your predictor matrix from training set,
#                      alpha = 1 # 1 stands for LASSO while 0 stands for Ridge)
```

3. Select the tuning parameter λ that produces the smallerst in-sample Mean Squared Errors.

   There are different ways of doing this. One way would be to loop over the pool of λ obtained by `glmnet.object$lambda`. Within each loop, we produce the MSE for one λ in the pool. Specfically, use `predict` function to produce a vector for predicted outcomes. Remember to specify the argument `s` in `predict`, so that `R` predict with the specified λ. Calculate the difference between the predicted outcome and the true outcome (as observed in the dataset) to produce the in-sample MSE for this specific λ. See Slide 36 for the definition of MSE.

4. Load the test data `pres_results_by_county_test.csv`. Predict vote share for Trump with model parameters and tuning parameters trained and determined in the previous parts of this question. Calculate the MSE for the test data. Remember to exclude NAs from the dataset. Produce a scatterplot where y-axis is the predicted vote share for Trump and x-axis is the true vote share for Trump.

5. Now we select the tuning parameter λ by cross-validation. We will use the canned `cv.glmnet` function, which automatically uses k-fold cross-validation (the default is 10 folds) to select optimal lambda values. Estimate the LASSO model with the training data, using the `cv.glmnet()` function. Note that to apply the canned function, you need to fit the entire training data set into the `cv.glmnet()` function. Be sure to set the seed to 02139. Use the `plot()` function to plot the cross-validated MSE against possible logged lambda values. What is the lambda selected by the 10-fold cross-validation process? What is the corresponding cross-validated MSE?

6. Predict the Trump vote share in the test data with lambda and model selected with cross validation. What is MSE for the test data? Which procedure is better in terms for prediction performance? Discuss your results.

7. In the model with the optimal lambda value selected by cross-validation, how many coefficients are non-zero? (hint: use the `fitted_object_name$nzero` object)

8. ***Extra Credit:*** Hand-code the cross-validation step by finishing the following loop. Loop through 100 evenly distributed (on logarithm scale) possible value of the tuning parameter λ from 10^{-5} to 0.1,

```
set.seed(02139)
```

```
# Randomly shuffle the data
data <- data[sample(nrow(data), replace = FALSE), ]
Create 10 equally size folds
folds <- cut(seq(1, nrow(data)), breaks = 10, labels=FALSE)

# create empty vectors to store the values
vec_mse <- c()
for(i in 1:101){
  # initiate an mse vector for each fold
  mse_vals <- c()
  for(j in 1:10){
  # Segement your data by fold using the which() function
    test_index <- which(folds == j, arr.ind = TRUE)
    test_set <- data[test_index, ]
    training_set <- data[-test_index, ]
  # fit the model
    mod_lasso <- glmnet(y = # your outcome vector,
                        x = # your predictor matrix from training set,
                        alpha = 1,
                        lambda = exp(log(.1)-(log(.1) - log(10^(-5)))/99 * (i-1)))
  # prediction
    yhat_lasso <- predict(mod_lasso,
                       newx = # your predictor matrix from test set
                         )
    my_mse <- # compute mse
    mse_vals <- c(mse_vals, my_mse)
}
 # compute the mean of mse for each 10-fold cross validation iteration
 mean_mse_vals <- mean(mse_vals)
 # fill the vector
 vec_mse[i] <- mean_mse_vals
}
```

Appendix 1: Roll Call Voting

- `full_district`: the congressional district code.
- `icpsr`: Unique ICPSR id for that member
- `vote_[x]`: A dichotomous indicator for whether that member voted for the xth roll call. 1 is "Yay," 0 is "Nay" or "Present."
- `born`: The year that member was born
- `lgbtq`: Whether that member is openly LGBTQ
- `house_dem_2016`: Proportion of votes for the Democratic candidate in that district in 2016.

- `house_rep_2016`: Proportion of votes for the Republican candidate in that district in 2016.
- `house_dem_2014`: Proportion of votes for the Democratic candidate in that district in 2014.
- `house_rep_2014`: Proportion of votes for the Republican candidate in that district in 2014.
- `pres_dem_2012`: Proportion of votes for Barack Obama in that district in 2012.
- `pres_rep_2012`: Proportion of votes for Mitt Romney in that district in 2012.
- `pres_dem_2008`: Proportion of votes for Barack Obama for President in that congressional district in 2008.
- `pres_rep_2008`: Proportion of votes for John McCain for President in that congressional district in 2008.
- `adult_pop_white_2010`: The proportion of adults living in that congressional district who are white, according to the 2010 census.
- `adult_pop_black_2010`: The proportion of adults living in that congressional district who are black, according to the 2010 census.
- `adult_pop_latino_2010`: The proportion of adults living in that congressional district who are Latino, according to the 2010 census.
- `adult_pop_asian_2010`: The proportion of adults living in that congressional district who are asian, according to the 2010 census.
- `adult_pop_native_2010`: The proportion of adults living in that congressional district who are native, according to the 2010 census.
- `adult_pop_other_2010`: The proportion of adults living in that congressional district who are categorized as "other race/ethnicity," according to the 2010 census.
- `college_degree_percent_2016`: The proportion of adults living in that congressional district who are native, according to the 2010 census.
- `median_income`: The Median Income in the district.
- `female`: Is the member female?
- `republican`: Is the member a Republican?
- `democrat`: Is the member a Democrat?
- `white`: Is the member white?
- `black`: Is the member black?
- `hispanic`: Is the member hispanic?
- `asian`: Is the member asian?
- `christian`: Is the member christian?
- `catholic`: Is the member catholic?
- `jewish`: Is the member jewish?
- `median_income_log`: The median income in that district, logged.
- `outcome`: Whether the member voted for passage of the Tax Cut and Jobs Acts.

Appendix 2: Presidential Elections in 2016

- `area_name`: Name of each county
- `state_abbreviation`: Abbreviation of the state each county belongs to
- `PST045214`: Population, 2014 estimate
- `PST040210`: Population, 2010 (April 1) estimates base
- `PST120214`: Population, percent change - April 1, 2010 to July 1, 2014
- `POP010210`: Population, 2010
- `AGE135214`: Persons under 5 years, percent, 2014
- `AGE295214`: Persons under 18 years, percent, 2014
- `AGE775214`: Persons 65 years and over, percent, 2014
- `SEX255214`: Female persons, percent, 2014
- `RHI125214`: White alone, percent, 2014
- `RHI225214`: Black or African American alone, percent, 2014
- `RHI325214`: American Indian and Alaska Native alone, percent, 2014
- `RHI425214`: Asian alone, percent, 2014
- `RHI525214`: Native Hawaiian and Other Pacific Islander alone, percent, 2014
- `RHI625214`: Two or More Races, percent, 2014
- `RHI725214`: Hispanic or Latino, percent, 2014
- `RHI825214`: White alone, not Hispanic or Latino, percent, 2014
- `POP715213`: Living in same house 1 year and over, percent, 2009-2013
- `POP645213`: Foreign born persons, percent, 2009-2013
- `POP815213`: Language other than English spoken at home, pct age 5+, 2009-2013
- `EDU635213`: High school graduate or higher, percent of persons age 25+, 2009-2013
- `EDU685213`: Bachelor's degree or higher, percent of persons age 25+, 2009-2013
- `VET605213`: Veterans, 2009-2013
- `LFE305213`: Mean travel time to work (minutes), workers age 16+, 2009-2013
- `HSG010214`: Housing units, 2014
- `HSG445213`: Homeownership rate, 2009-2013
- `HSG096213`: Housing units in multi-unit structures, percent, 2009-2013
- `HSG495213`: Median value of owner-occupied housing units, 2009-2013
- `HSD410213`: Households, 2009-2013
- `HSD310213`: Persons per household, 2009-2013
- `INC910213`: Per capita money income in past 12 months (2013 dollars), 2009-2013
- `INC110213`: Median household income, 2009-2013
- `PVY020213`: Persons below poverty level, percent, 2009-2013

- BZA010213: Private nonfarm establishments, 2013
- BZA110213: Private nonfarm employment, 2013
- BZA115213: Private nonfarm employment, percent change, 2012-2013
- NES010213: Nonemployer establishments, 2013
- SBO001207: Total number of firms, 2007
- SBO315207: Black-owned firms, percent, 2007
- SBO115207: American Indian- and Alaska Native-owned firms, percent, 2007
- SBO215207: Asian-owned firms, percent, 2007
- SBO515207: Native Hawaiian- and Other Pacific Islander-owned firms, percent, 2007
- SBO415207: Hispanic-owned firms, percent, 2007
- SBO015207: Women-owned firms, percent, 2007
- MAN450207: Manufacturers shipments, 2007 ($1,000)
- WTN220207: Merchant wholesaler sales, 2007 ($1,000)
- RTN130207: Retail sales, 2007 ($1,000)
- RTN131207: Retail sales per capita, 2007
- AFN120207: Accommodation and food services sales, 2007 ($1,000)
- BPS030214: Building permits, 2014
- LND110210: Land area in square miles, 2010
- POP060210: Population per square mile, 2010