Final Project (PSTAT 131/231, FALL 2022)

Due December 8, 2022, 7:00 PM PT

Instructions and Expectations

- You are allowed and encouraged to work with one partner on this project. Include your names, perm numbers, and whether you are taking the class for 131 or 231 credit.
- Format the assignment like a long homework by addressing each question in parts.
- All of your results should be formatted in a professional and visually appealing manner. That means, either as a polished visualization or for tabular data, a nicely formatted table.
- All R code should be available from your Rmarkdown file, but does not need to be shown in the body of the report! Use the chunk option `echo=FALSE` to exclude code from appearing in your write-up when necessary. In addition to your Rmarkdown, you should turn in the write-up as either a pdf document or an html file (both are acceptable).
- All files should be submitted electronically via GauchoSpace. See course syllabus for submission instructions.

Predicting voter behavior is complicated for many reasons despite the tremendous effort in collecting, cleaning, analyzing, and understanding many available datasets. As the midterm elections in the United States is currently being held near the midpoint of the current president's four-year term of office, it is a good time for us to review the 2020 United States presidential election data! Despite that the 2016 presidential election came as a big surprise (https://fivethirtyeight.com/features/the-polls-missed-trump-we-asked-pollsters-why/) to many, Biden's victory in the 2020 presidential election has been widely predicted (e.g., see the well-known Nate Silver (https://en.wikipedia.org/wiki/Nate_Silver) in FiveThirtyEight (https://projects.fivethirtyeight.com/2020-election-forecast/)).

For our final project, we will analyze and visualize the 2020 presidential election dataset. We will primarily work towards building state/county-level red/blue map plots that are commonly shown on media coverage (https://www.nytimes.com/interactive/2020/11/03/us/elections/results-president.html).

In addition, we will combine the Untied States county-level census data with the election data. Our target would then be building and selecting classification models (among many predictive models that we've covered in this quarter) to predict the election winner.

Data

We will start the analysis with two data sets. The first one is the election data, which is drawn from here (https://www.kaggle.com/unanimad/us-election-2020?select=president_county.csv). The data contains county-level election results.

The second dataset is the 2017 United States county-level census data, which is available here (https://www.kaggle.com/muonneutrino/us-census-demographic-data).

The following code load in these two data sets: `election.raw` and `census`.

```
## read data and convert candidate names and party names from string to factor
## we manually remove the variable "won", the indicator of county level winner
## In Problem 5 we will reproduce this variable!
election.raw <- read_csv("candidates_county.csv", col_names = TRUE) %>%
  mutate(candidate = as.factor(candidate), party = as.factor(party), won = NULL)

## remove the word "County" from the county names
words.to.remove = c("County")
remove.words <- function(str, words.to.remove){
  sapply(str, function(str){
    x <- unlist(strsplit(str, " "))
    x <- x[!x %in% words.to.remove]
    return(paste(x, collapse = " "))
  }, simplify = "array", USE.NAMES = FALSE)
}
election.raw$county <- remove.words(election.raw$county, words.to.remove)

## read census data
census <- read_csv("census_county.csv")
```

Election data

1. (1 pts) Report the dimension of `election.raw`. (1 pts) Are there missing values in the data set? (1 pts) Compute the total number of distinct values in `state` in `election.raw` to verify that the data contains all states and a federal district.

Census data

Following is the first few rows of the `census` data. The column names are all very self-explanatory:

CountyId	State	County	TotalPop	Men	Women	Hispanic	White	Black	Native	Asian	Pacific	V
1001	Alabama	Autauga County	55036	26899	28137	2.7	75.4	18.9	0.3	0.9	0	
1003	Alabama	Baldwin County	203360	99527	103833	4.4	83.1	9.5	0.8	0.7	0	
1005	Alabama	Barbour County	26201	13976	12225	4.2	45.7	47.8	0.2	0.6	0	
1007	Alabama	Bibb County	22580	12251	10329	2.4	74.6	22.0	0.4	0.0	0	
1009	Alabama	Blount County	57667	28490	29177	9.0	87.4	1.5	0.3	0.1	0	
1011	Alabama	Bullock County	10478	5616	4862	0.3	21.6	75.6	1.0	0.7	0	

2. (1 pts) Report the dimension of `census` . (1 pts) Are there missing values in the data set? (1 pts) Compute the total number of distinct values in `county` in `census` . (1 pts) Compare the values of total number of distinct county in `census` with that in `election.raw` . (1 pts) Comment on your findings.

Data wrangling

3. (4 pts) Construct aggregated data sets from `election.raw` data: i.e.,

- Keep the county-level data as it is in `election.raw` .
- Create a state-level summary into a `election.state` .
- Create a federal-level summary into a `election.total` .

4. (1 pts) How many named presidential candidates were there in the 2020 election? (2 pts) Draw a bar chart of all votes received by each candidate. You can split this into multiple plots or may prefer to plot the results on a log scale. Either way, the results should be clear and legible! (For fun: spot Kanye West among the presidential candidates!)

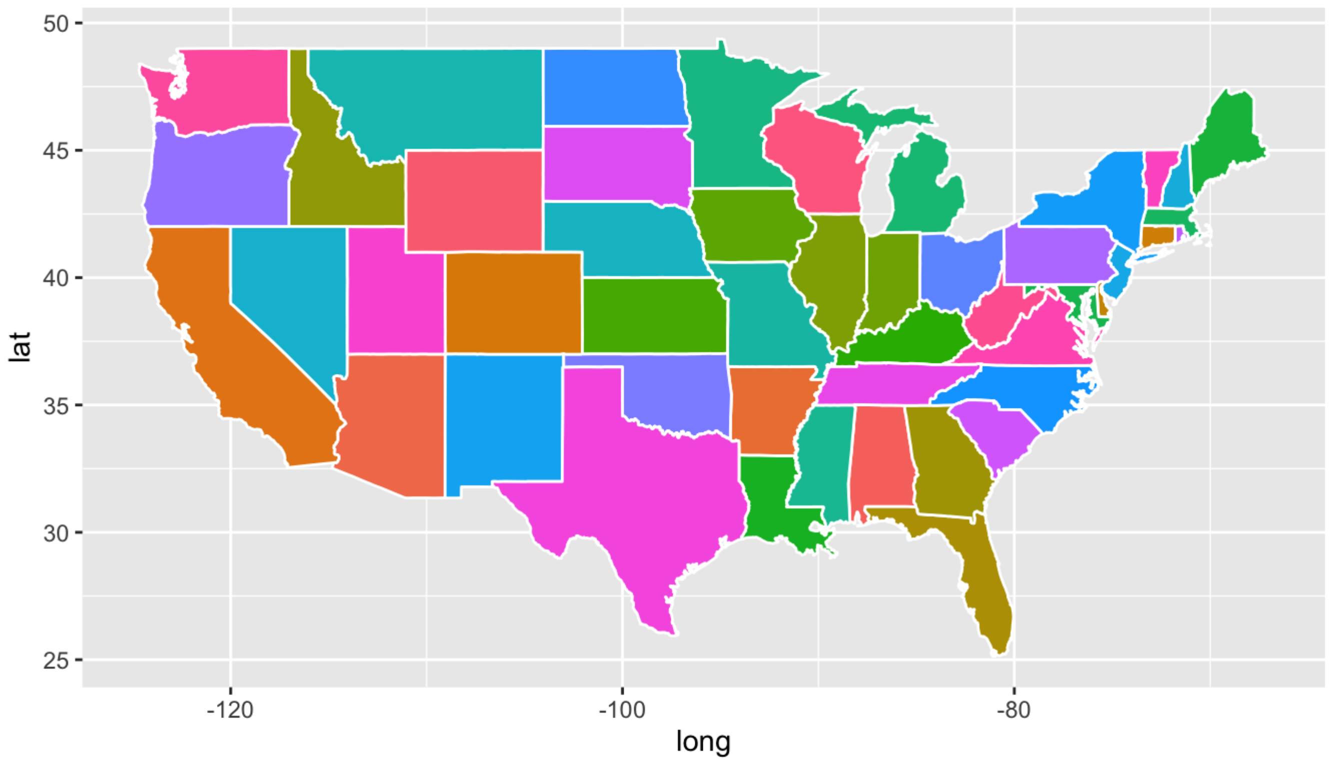
5. (6 pts) Create data sets `county.winner` and `state.winner` by taking the candidate with the highest proportion of votes in both county level and state level. Hint: to create `county.winner` , start with `election.raw` , group by `county` , compute `total` votes, and `pct = votes/total` as the proportion of votes. Then choose the highest row using `top_n` (variable `state.winner` is similar).

Visualization

Visualization is crucial for gaining insight and intuition during data mining. We will map our data onto maps.

The R package `ggplot2` can be used to draw maps. Consider the following code.

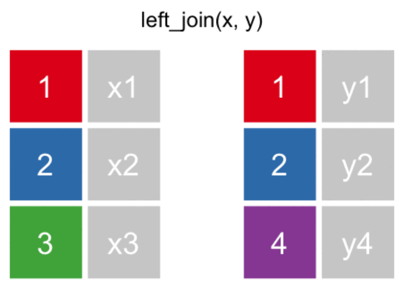
```
states <- map_data("state")

ggplot(data = states) + 
  geom_polygon(aes(x = long, y = lat, fill = region, group = group),
               color = "white") + 
  coord_fixed(1.3) +
  guides(fill=FALSE)  # color legend is unnecessary and takes too long
```



The variable `states` contain information to draw white polygons, and fill-colors are determined by `region` .

6. (4 pts) Use similar code to above to draw county-level map by creating `counties = map_data("county")` . Color by county.

7. (6 pts) Now color the map by the winning candidate for each state. First, combine `states` variable and `state.winner` we created earlier using `left_join()` . Note that `left_join()` needs to match up values of states to join the tables. A call to `left_join()` takes all the values from the first table and looks for matches in the second table. If it finds a match, it adds the data from the second table; if not, it adds missing values:



Here, we'll be combing the two data sets based on state name. However, the state names in `states` and `state.winner` can be in different formats: check them! Before using `left_join()` , use certain transform to make sure the state names in the two data sets: `states` (for map drawing) and `state.winner` (for coloring) are in the same formats. Then `left_join()` . Your figure will look similar to New York Times map (https://www.nytimes.com/interactive/2020/11/03/us/elections/results-president.html).

8. (6 pts) Color the map of the state of California by the winning candidate for each county.

9. (4 pts) (Open-ended) Create a visualization of your choice using `census` data. Many exit polls noted that demographics played a big role in the election (https://fivethirtyeight.com/features/demographics-not-hacking-explain-the-election-results/). Use this Washington Post article (https://www.washingtonpost.com/graphics/politics/2016-election/exit-polls/) and this R graph gallery (https://www.r-graph-gallery.com/) for ideas and inspiration.

10. The `census` data contains county-level census information. In this problem, we clean and aggregate the information as follows.

- *(4 pts) Clean county-level census data `census.clean`* : start with `census` , filter out any rows with missing values, convert { `Men` , `Employed` , `VotingAgeCitizen` } attributes to percentages, compute `Minority` attribute by combining {Hispanic, Black, Native, Asian, Pacific}, remove these variables after creating `Minority` , remove { `IncomeErr` , `IncomePerCap` , `IncomePerCapErr` , `Walk` , `PublicWork` , `Construction` }.
  Many columns are perfectly colineared, in which case one column should be deleted.
- *(1 pts) Print the first 5 rows of `census.clean`* :

Dimensionality reduction

11. Run PCA for the cleaned county level census data (with `State` and `County` excluded). (2 pts) Save the first two principle components PC1 and PC2 into a two-column data frame, call it `pc.county` . (2 pts) Discuss whether you chose to center and scale the features before running PCA and the reasons for your choice. (2 pts) What are the three features with the largest absolute values of the first principal component? (2 pts) Which features have opposite signs and what does that mean about the correlation between these features?

12. (2 pts) Determine the number of minimum number of PCs needed to capture 90% of the variance for the analysis. (2 pts) Plot proportion of variance explained (PVE) and cumulative PVE.

Clustering

13. (2 pts) With `census.clean` (with `State` and `County` excluded), perform hierarchical clustering with complete linkage. (2 pts) Cut the tree to partition the observations into 10 clusters. (2 pts) Re-run the hierarchical clustering algorithm using the first 2 principal components from `pc.county` as inputs instead of the original features. (2 pts) Compare the results and comment on your observations. For both approaches investigate the cluster that contains *Santa Barbara County*. (2 pts) Which approach seemed to put Santa Barbara County in a more appropriate clusters? Comment on what you observe and discuss possible explanations for these observations.

Classification

We start considering supervised learning tasks now. The most interesting/important question to ask is: *can we use census information in a county to predict the winner in that county?*

In order to build classification models, we first need to combine `county.winner` and `census.clean` data. This seemingly straightforward task is harder than it sounds. For simplicity, the following code makes necessary changes to merge them into `election.cl` for classification.

```
# we move all state and county names into lower-case
tmpwinner <- county.winner %>% ungroup %>%
  mutate_at(vars(state, county), tolower)

# we move all state and county names into lower-case
# we further remove suffixes of "county" and "parish"
tmpcensus <- census.clean %>% mutate_at(vars(State, County), tolower) %>%
  mutate(County = gsub(" county|  parish", "", County))

# we join the two datasets
election.cl <- tmpwinner %>%
  left_join(tmpcensus, by = c("state"="State", "county"="County")) %>%
  na.omit

# drop levels of county winners if you haven't done so in previous parts
election.cl$candidate <- droplevels(election.cl$candidate)

## save meta information
election.meta <- election.cl %>% select(c(county, party, CountyId, state, total_votes, p
ct, total))

## save predictors and class labels
election.cl = election.cl %>% select(-c(county, party, CountyId, state, total_votes, pc
t, total))
```

14. Understand the code above. (3 pts) Why do we need to exclude the predictor `party` from `election.cl`?

Classification

Using the following code, partition data into 80% training and 20% testing:

```
set.seed(10)
n <- nrow(election.cl)
idx.tr <- sample.int(n, 0.8*n)
election.tr <- election.cl[idx.tr, ]
election.te <- election.cl[-idx.tr, ]
```

Use the following code to define 10 cross-validation folds:

```
set.seed(20)
nfold <- 10
folds <- sample(cut(1:nrow(election.tr), breaks=nfold, labels=FALSE))
```

Using the following error rate function. And the object `records` is used to record the classification performance of each method in the subsequent problems.

```
calc_error_rate = function(predicted.value, true.value){
  return(mean(true.value!=predicted.value))
}
records = matrix(NA, nrow=3, ncol=2)
colnames(records) = c("train.error","test.error")
rownames(records) = c("tree","logistic","lasso")
```

15. Decision tree: (2 pts) train a decision tree by `cv.tree()`. (2 pts) Prune tree to minimize misclassification error. Be sure to use the `folds` from above for cross-validation. (2 pts) Visualize the trees before and after pruning. (1 pts) Save training and test errors to `records` object. (2 pts) Interpret and discuss the results of the decision tree analysis. (2 pts) Use this plot to tell a story about voting behavior.

16. (2 pts) Run a logistic regression to predict the winning candidate in each county. (1 pts) Save training and test errors to `records` variable. (1 pts) What are the significant variables? (1 pts) Are they consistent with what you saw in decision tree analysis? (2 pts) Interpret the meaning of a couple of the significant coefficients in terms of a unit change in the variables.

17. You may notice that you get a warning `glm.fit: fitted probabilities numerically 0 or 1 occurred`. As we discussed in class, this is an indication that we have perfect separation (some linear combination of variables *perfectly* predicts the winner). This is usually a sign that we are overfitting. One way to control overfitting in logistic regression is through regularization.

(3 pts) Use the `cv.glmnet` function from the `glmnet` library to run a 10-fold cross validation and select the best regularization parameter for the logistic regression with LASSO penalty. Set `lambda = seq(1, 50) * 1e-4` in `cv.glmnet()` function to set pre-defined candidate values for the tuning parameter λ.

(1 pts) What is the optimal value of λ in cross validation? (1 pts) What are the non-zero coefficients in the LASSO regression for the optimal value of λ? (1 pts) How do they compare to the unpenalized logistic regression? (1 pts) Comment on the comparison. (1 pts) Save training and test errors to the `records` variable.

18. (6 pts) Compute ROC curves for the decision tree, logistic regression and LASSO logistic regression using predictions on the test data. Display them on the same plot. (2 pts) Based on your classification results, discuss the pros and cons of the various methods. (2 pts) Are the different classifiers more appropriate for answering different kinds of questions about the election?

Taking it further

This part will be worth up to a 20% of your final project grade!

19. (9 pts) Explore additional classification methods. Consider applying additional *two* classification methods from KNN, LDA, QDA, SVM, random forest, boosting, neural networks etc. (You may research and use methods beyond those covered in this course). How do these compare to the tree method, logistic regression, and the lasso logistic regression?

20. (9 pts) Tackle *at least* one more interesting question. Creative and thoughtful analysis will be rewarded! Some possibilities for further exploration are:

- Explore using census data to predict mid-term election results. For this purpose, feel free to search for the mid-term election dataset, and make any necessary references.

- Consider a regression problem! Use linear regression models to predict the total votes for each candidate by county. Compare and contrast these results with the classification models. Which do you prefer and why? How might they complement one another?
- Conduct an exploratory analysis of the "purple" counties – the "battle ground" / "swing counties": which the models predict Biden and Trump were roughly equally likely to win. What is it about these counties that make them hard to predict?
- Instead of using the native attributes (the original features), we can use principal components to create new (and lower dimensional) set of features with which to train a classification model. This sometimes improves classification performance. Compare classifiers trained on the original features with those trained on PCA features.

21. (9 pts) (Open ended) Interpret and discuss any overall insights gained in this analysis and possible explanations. Use any tools at your disposal to make your case: visualize errors on the map, discuss what does/doesn't seems reasonable based on your understanding of these methods, propose possible directions (collecting additional data, domain knowledge, etc).