# Stats101C Kaggle 1

***Prediction of Voters in a County that Voted for Biden***

**Group Mahjong**

**July 29th 2022**

**Yuxuan Bai: baiyuxuan1203@gmail.com**
**Yuetong Li: liyuetong.cathy@gmail.com**
**Jinghong Zou: jinghongzou@ucla.edu**
**Xiaocong Xuan: xiaocong0116@gmail.com**

**Kaggle Score: 0.05872**
**Kaggle Rank: 10th**

# 1 Introduction

The Presidential Election which happens once every four years is always a hot and thought-provoking topic. Who did you vote for? Who did they vote for? Uncovering the 'secrets' underlying the voting behavior hence becomes a worthwhile investigation orientation. In our project, we used the data of 3,111 counties with 214 various demographics like age, gender, race, education, etc., collected from the US Census Bureau by Professor Chen and attempted to predict the percentage of the voters in each county that voted for Biden during the 2020 US Presidential Election.

We believe unveiling the potential connections between features and voting behavior can generate unexpected insights from this previous election. Subjectively, some minority groups like Latino or Asian tend to vote for Biden because of his support of immigration; young people are more likely to vote for Biden due to his proposal for the improvement of student debt; and also people with more advanced degrees tend to vote for Biden. Hence, based on our background information, race, age, education level may be associated with the response variable significantly.

In addition, we believe that the machine learning model can be a great reference tool for future prediction of elections. We learned a popular algorithm called 'gradient boosted machines' and applied in our constructed model.

# 2 Exploratory Data Analysis

In order to further investigate which predictors might be the crucial components in predicting the percentage of voters that vote for Biden, we decide to make several visualizing plots like scatterplots and color plot to see the potential associations. We made multiple visualizations to investigate any potential associations between *percent_dem* and the rest of the predictors. It turned out that we found 12 predictors showing the fairly obvious association.

As shown from the scatter plots below, we can tell that *precent_dem* has association with $C02\_013E$, $C02\_015E$, $C02\_027E$, $X0037PE$, $X0044PE$, $X0046PE$, $X0038$PE, $X0064PE$, $X0065PE$, $X0067PE$, $X0077PE$, but we need further exploration on those predictors by feature selection methods and modeling process. Predictors like $X0045PE$ and $X0053PE$ may not be considered since the graphs of those two predictors show no trends or patterns.

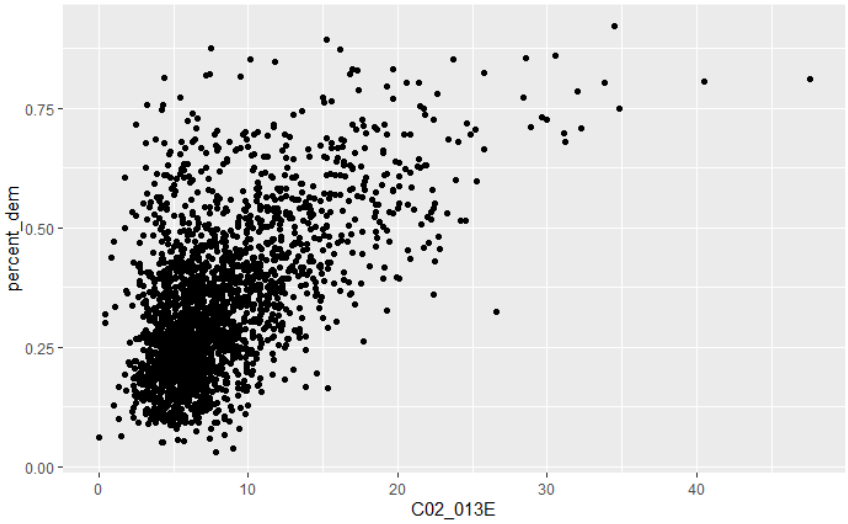



Figure 1: This scatter plot shows that there might be a potential logarithmic relationship between the *percent_dem* and $C02\_013E$. Hence, this predictor may tend to contribute considerably in future prediction. This indicates if the population that are 25 years' old and over and obtain graduate or professional degree increase, the more people in this county will vote for Biden.

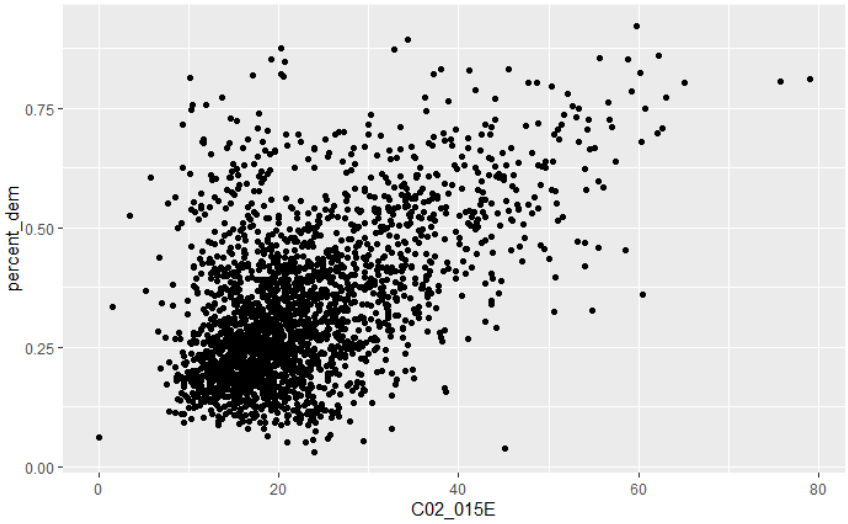



Figure 2: This scatter plot shows that there might be a linear relationship between the *percent_dem* and $C02\_015E$. Hence, this predictor may tend to contribute considerably in future prediction. The plot also indicates that the percentage of people in a county that vote for Biden will increase if there are more people who are 25 years' old and over with bachelor's degree or higher. This indeed reflects the real world situation that more educated and mature people tend to vote for Democratic Party.

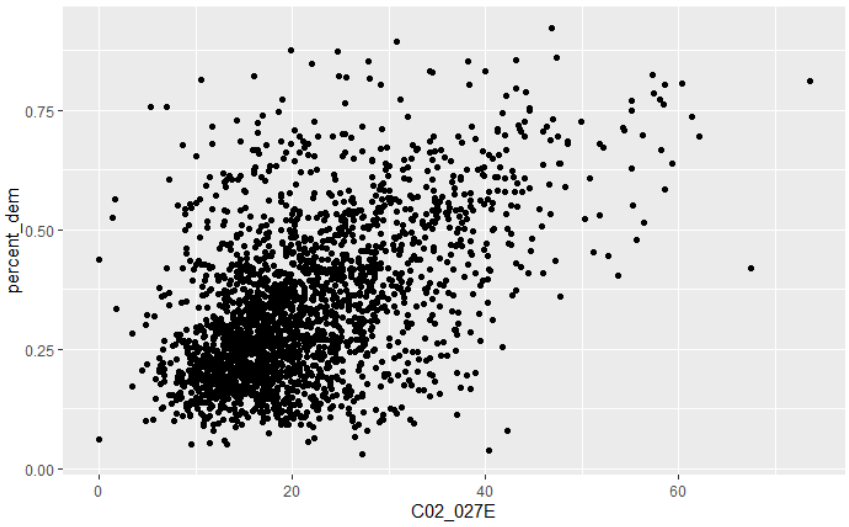



Figure 3: This scatter plot shows that there might be a linear relationship between the *percent_dem* and $C02\_027E$. Hence, this predictor may tend to contribute considerably in future prediction. The figure shows that the percentage of people that vote for Biden will increase if there are more people that are above 65 years' old and with bachelor's degree or higher. This reflects the real world situation that old and educated people are more likely to support Democratic Party.

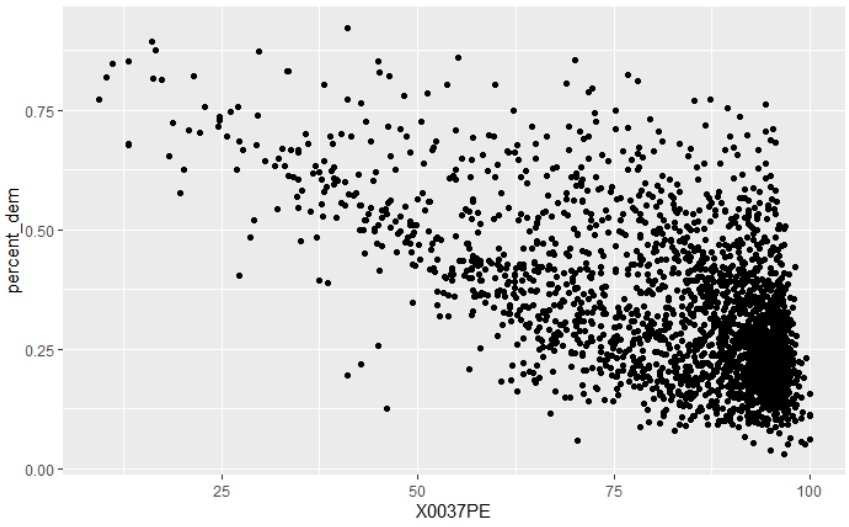



Figure 4: This scatter plot shows that there might be a linear relationship between the *percent_dem* and X0037PE, and the slope is negative. Hence, this predictor may tend to contribute considerably in future prediction. This shows that less people will vote for Biden if a county has more white population. Indeed, this is consistent with the fact that the majority of white people would vote for Trump instead of Biden.

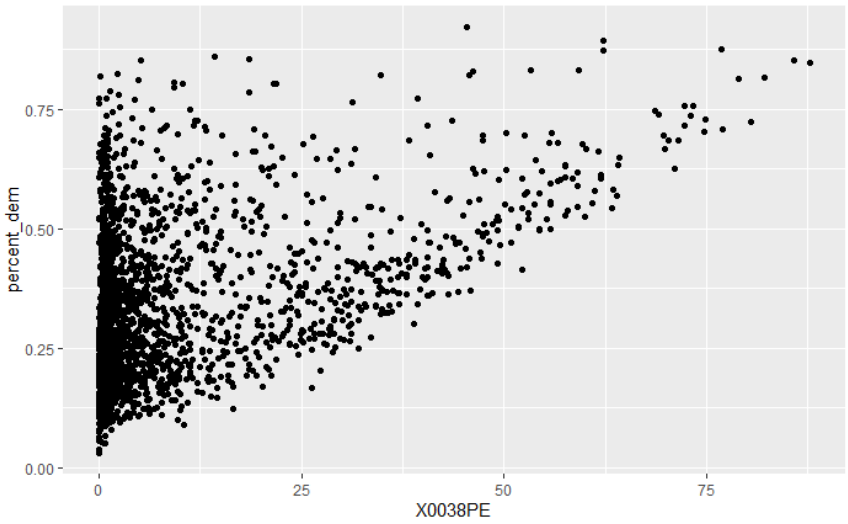



Figure 5: The scatterplot shows that there is a linear association between the *precent_dem* and X0038PE, therefore X0038PE may be considered to predict the response variable. In other words, the size of black or African American population is a crucial factor in predicting the percentage of people in a county that votes for Biden. Counties that have more African Americans will have higher rates in voting for Biden. African American population has been the solid support to Democratic Party, the plot has reenforced this fact.

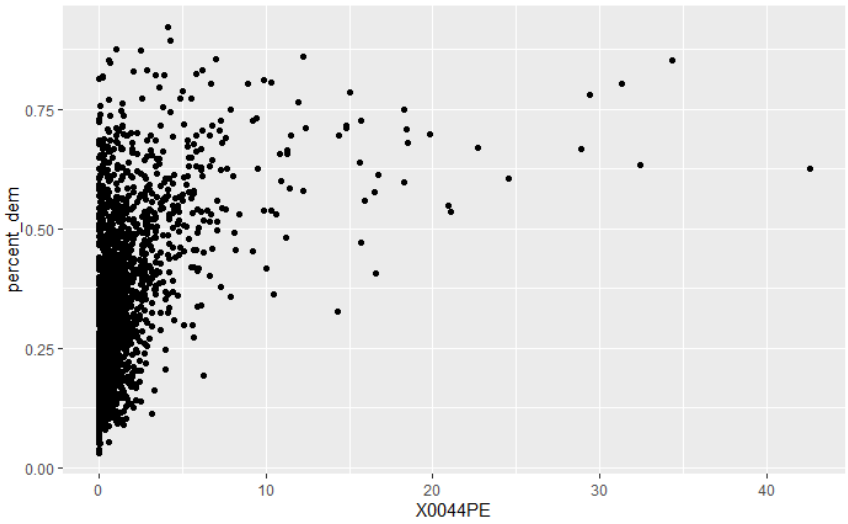



Figure 6: This scatter plot shows that there might be a linear relationship between the *percent_dem* and X0044PE, but the association is not quite strong. Hence, this predictor may not tend to contribute considerably in future prediction. This shows that a county that has more Asian population will have a higher rate of voting for Biden. Asian community has been supporting Biden or Democratic Party for their advocacy in moderate immigration policy.

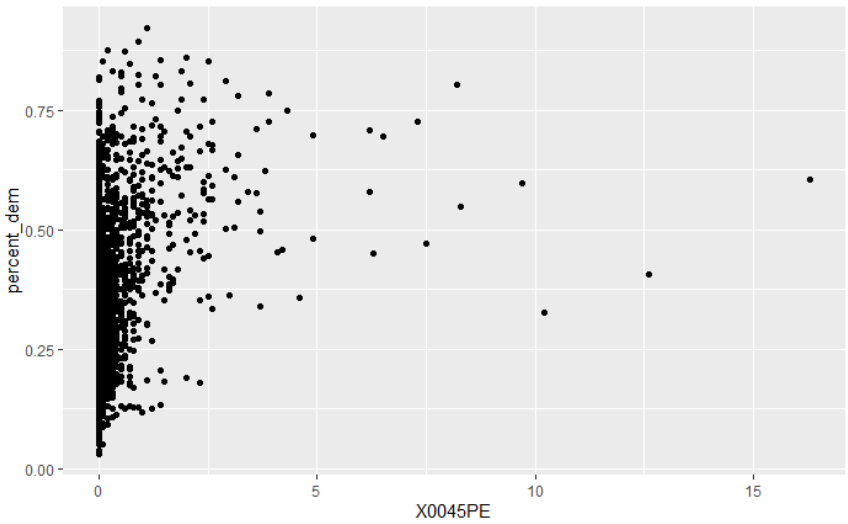



Figure 7: Here is an example of showing a data with no pattern between *precent_dem* and the predictor. The scatterplot shows that there is no obvious pattern, therefore X0045PE might not be considered to predict the response variable. In other words, Asian Indian population is not a significant predictor for percentage of voting for Biden.

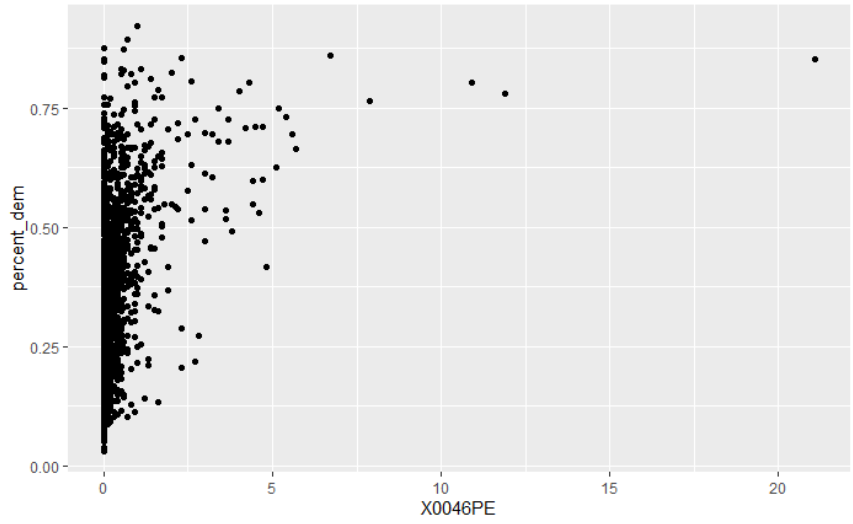



Figure 8: This scatter plot shows that there might be a logarithmic relationship between the *percent_dem* and X0046PE, but the association is not quite strong. Hence, this predictor may tend to contribute considerably in future prediction. This shows that if a county has more Chinese population, the percentage of population in a county that vote for Biden will increase. This also reflects the real world situation that Chinese community support Biden for his immigration policy.

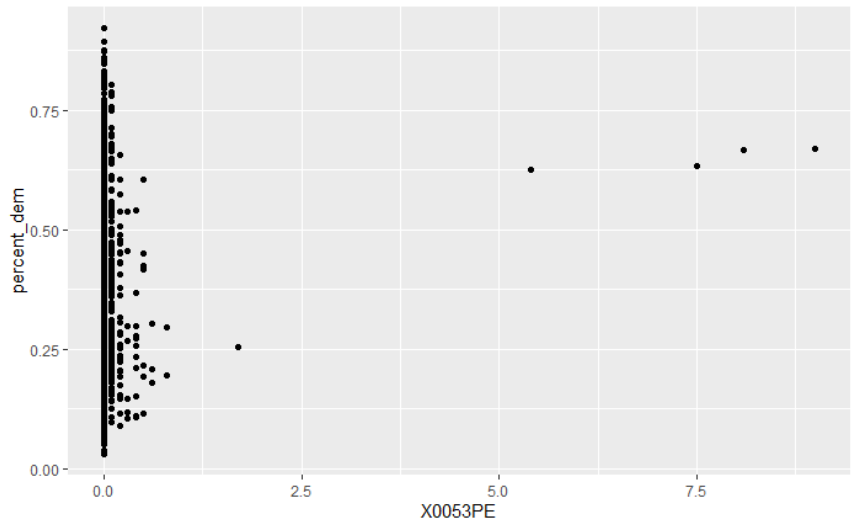



Figure 9: Here is an example of showing a data with no pattern between *precent_dem* and the predictor. Therefore, X0053PE may shows less significance for further prediction. In other words, Native Hawaiian and Other Pacific Islander is not significant in predicting the percentage of population that vote for Biden.

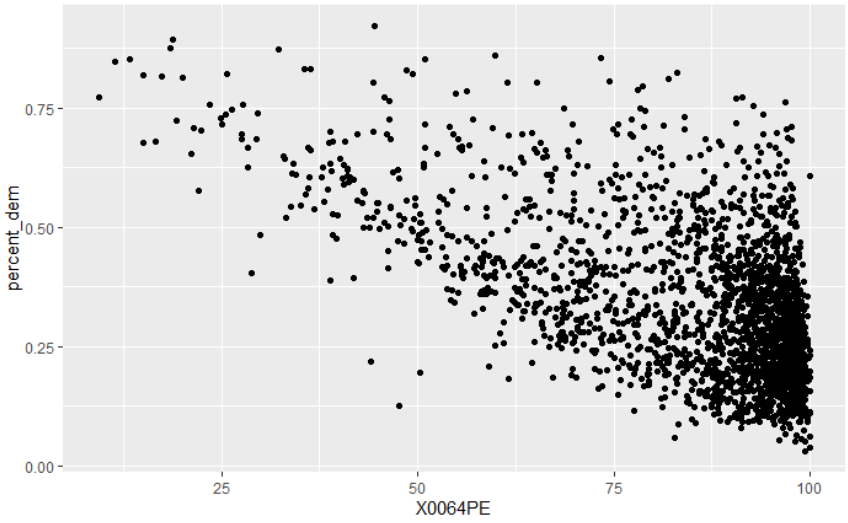



Figure 10: This scatter plot shows that there might be a linear relationship between the *percent_dem* and X0064PE, and the slope is negative. Hence this predictor may tend to contribute considerably in future prediction. This shows that a county that has more white people or in combination with one or more other races will have lower rate in voting for Biden. This is consistent with what we analyzed previously that white people tend to vote for Trump instead of Biden.

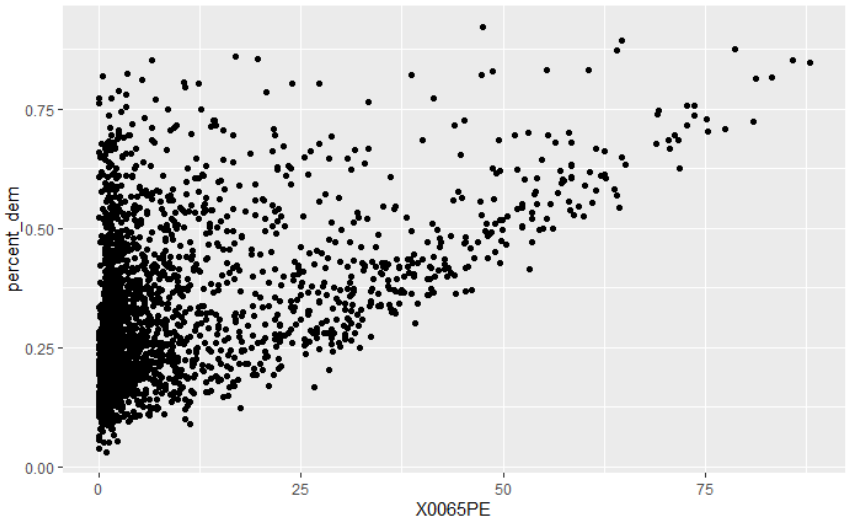



Figure 11: This scatter plot shows that there might be a linear relationship between the *percent_dem* and X0065PE, and the association is quite strong. Hence this predictor may tend to contribute considerably in future prediction. This shows that a county will have a higher rate in voting for Biden if there are more African Americans or in combination with one or more other races. This is consistent with what we analyzed earlier that African Americans are more likely to support Biden.

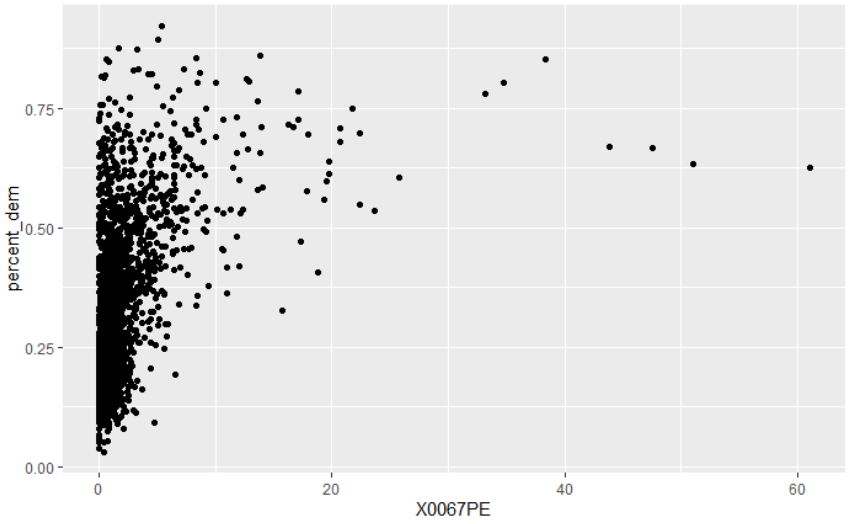



Figure 12: This scatter plot shows that there might be a potential logarithmic relationship between the *percent_dem* and X0067PE. Hence, this predictor may tend to contribute considerably in future prediction. This shows that if a county has more Asian or in combination of one or more other races population, the rate of voting for Biden in that county will be higher. This is consistent with what we analyzed earlier that Asian community is more likely to support Biden.

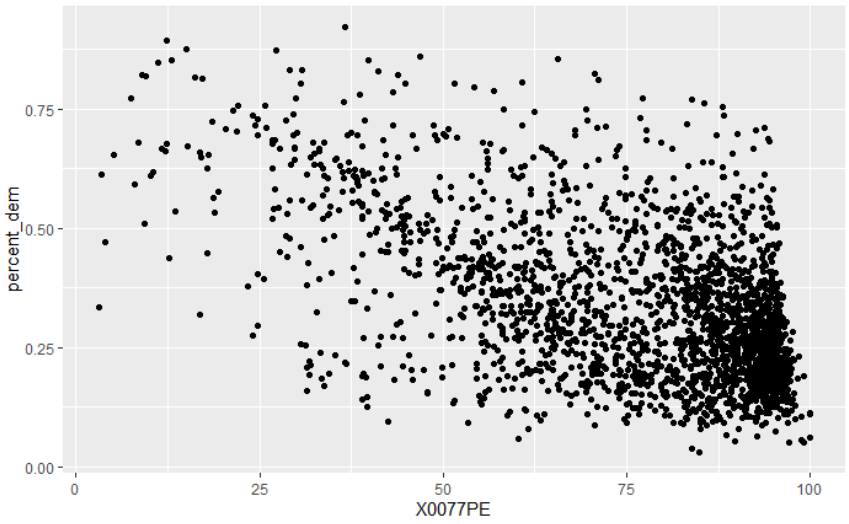
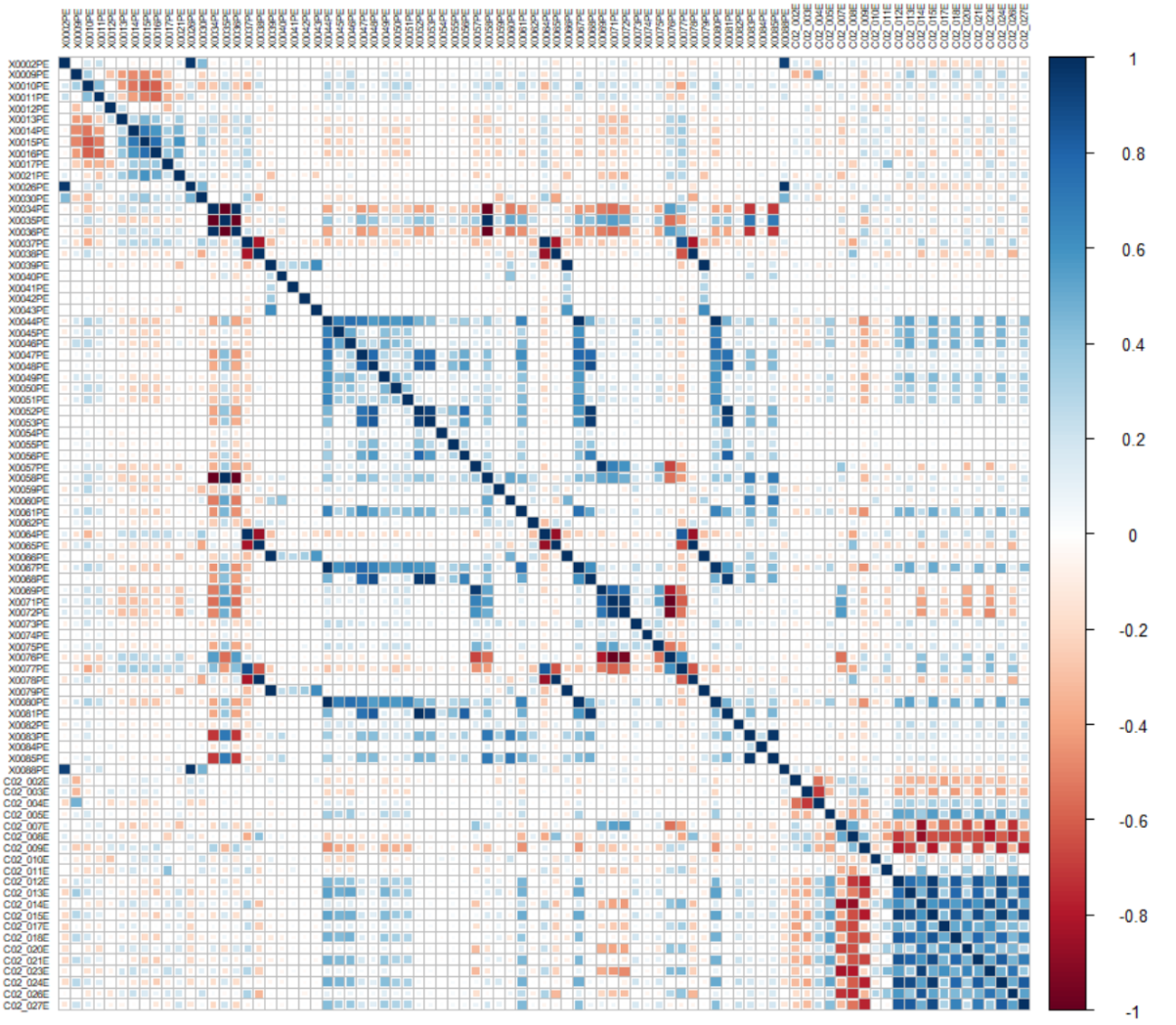



Figure 13: This scatter plot shows that there might be a linear relationship between the *percent_dem* and X0077PE, and the slope is negative. Hence, this predictor may tend to contribute considerably in future prediction. This shows that the rate of voting for Biden will be lower if there is more population that is not Hispanic or Latino (White alone). This is consistent with what we analyzed previously that white people are less likely to vote for Biden.

Figure 14: This color plot shows the correlation coefficient matrix in the form of color distribution in red and blue, where red indicates negative correlation coefficient and blue indicates positive correlation coefficient. As the color getting darker, the correlation coefficient is closer to one. As shown from the color plot, we can tell that there are several dark spots at the right-lower corner, middle, and boundary. That means our predictors has collinearity issues.



# 3 Preprocessing

*(1). Select the variable with percentage type*
In exploratory data analysis, we found that the variance of variables is considerably large and different in scale, so we chose to only use the percentage type data in the 'train' dataset in order to standardize the data.

*(2). Remove variables*
a. based on the election rules:
According to the election rules, U.S. citizens under the age of 18 do not have the right to vote, so we removed all variables that are categorized by ages under 18.
b. all variables categorized as 'female':
The collection of data only considers two categories of sex (male and female). We chose only to keep one type in our data because they are complementary, one significant, the other would not, and vice versa.
c. remove wrong column:
Due to the mistake in dataset collection, the variable 'X0033PE' does not follow the percentage. We chose to remove this column to achieve standardized data for better analysis.

*(3). Remove the collinearity*
The severe collinearity against the assumption of independency between variables does not support us to construct an ideal linear regression model. Hence, we use the method of 'Lasso' to make penalties on variables. Based on its variable selection, we decided to retain variables with coefficients greater than 0.001 for simplicity.

*(4). Add interactions*
Dependent on the analysis of the color map in part 2, we believe that our variables contain interactions. Since the number of variables is relatively large, we decided to explore interactions with four levels.

# 4 Candidate models

| Model identifier | Type of model | Engine | Hyperparameters |
|---|---|---|---|
| lasso_model | regression | glmnet | lambda |
| rf_model | random forest | random forest | mtry, ntree, min_n |
| gbm_model_1 | gradient boosting machines | dismo, gbm | n.trees, shrinkage, interaction.depth |
| gbm_model_2 | gradient boosting machines | dismo, gbm | n.trees, shrinkage, interaction.depth |

Table 1: **Candidate models attempted**

Due to the number of variables being large, we list below the lasso model with individual variables (not including their four-level interactions in the fitted model), and the variables that contribute the most for the random forest model and gbm model.

| Model identifier | Variables |
|---|---|
| lasso_model | X0015PE,X0016PE,X0021PE,X0026PE,X0037PE,X0040PE,X0049PE,X0051PE, X0057PE,X0060PE,X0061PE,X0073PE,X0077PE,X0081PE,C02_008E,C02_010E, C02_011E,C02_013E,C02_023E |
| rf_model | C02_013E, X0037PE,X0064PE,C02_015E,X0077PE,X0067PE,X0065PE,X0044PE, C02_027E,X0080PE (top 10 contribution variable) |
| gbm_model_1 | C02_013E,X0037PE,X0077PE,X0067PE,X0065PE,C02_027E,X0021PE,C02_011E, X0064PE,C02_015E (top 10 contribution variable) |
| gbm_model_2 | C02_013E,X0037PE,X0077PE,X0067PE,X0065PE,X0064PE,C02_027E,X0021PE, C02_005E,X0046PE (top 10 contribution variable) |

Table 2: **Candidate models (listing of variables)**

**Candidate model 1: Lasso**
Due to the high dependency between variables, we chose to use the ‘lasso‘ method. Specifically, ‘Lasso‘ performs regularization and variable selection on a given model. It decreases the variance between variables significantly with only a little increase in bias. Hence, in order to maximize the strategy of the method of ‘Lasso‘, we use the ‘tune()‘ function to figure out the best penalty coefficient ‘lambda‘ based on the minimum rmse. The tuned ‘lambda‘ is 0.001176812, which is the penalty value we added to the following model. We updated the best ‘lambda‘ in our linear model with the dataset finalized in part 3 and only retained the variables with coefficients provided by ‘lasso‘ greater than 0.001. Then, we included the four-level interactions in our model. Due to not all the interactions being significant and the complexity of 19 variables with four-level interactions, we use the ‘leaps‘ package and execute ‘forward‘ method to choose the most influential variable. The ‘forward‘ method only adds one variable at a time and checks its significance level. We used BIC as our standardized metric, and selected the best model which contained the minimum value of BIC. Lastly, we checked the assumption of regression model and concluded it as one of our our candidates models.

**Candidate model 2: Random Forest**

Random forest is one of the most accurate learning algorithms available and can produce a high predicting accuracy in a large dataset. Random forest considers all possible splits on a random sample of a small subset of all variables. Hence, the random forest is able to perform a stronger prediction accuracy. Specifically, it avoids the stronger biased predictor impact. Besides, it is easy to implement the model of random forest and it can estimate the importance of the variables in the regression analysis process. Before fitting the model to our dataset, the first step was tuning the model for the best hyperparameters, which gives us the minimum rmse. Then, we applied the best hyperparameters to our final model construction and generated predictions.

**Candidate model 3: Gradient Boosting Machines 1**
Boosting is an idea that combines weak previous predictions into the next strong prediction. In other words, boosting will create a sequence of trees and each one builds on the previous one with the previous one's residuals. Based on this, the boosting model will try its best to explain the information that it can't explain in the previous steps (it will 'learn' something new from the previous prediction). To slow down the learning process, we used the parameter called 'shrinkage'. For this problem, we use 'gbm.step' from the 'dismo' package to assess the optimal number of boosting trees using v-fold cross validation. We set 'gbm.x' as the top 16 most contributed variables due to the large number of variables. In detail, we tuned the 'shrinkage' and decided that its optimal value is 0.04. Another important parameter is the interaction depth (the maximum depth of each tree) which we tuned and decided that the optimal value is 11. In addition, we tuned the number of trees as well and decided that the optimal value is 450. After finding the optimal hyperparameters which have minimum residual deviance, we used the 'gbm' function from the 'gbm' package to fit the model.

**Candidate model 3: Gradient Boosting Machines 2**
We set 'gbm.x' as all the variables (85 in total) in the dataset. In detail, we tuned the 'shrinkage' and decided that its optimal value is 0.06. Another important parameter is the interaction depth (the maximum depth of each tree) which we tuned and decided that the optimal value is 12. In addition, we tuned the number of trees as well and decided that the optimal value is 1000. After finding the optimal hyperparameters which have minimum residual deviance, we used the 'gbm' function from the 'gbm' package to fit the model.

# 5 Model evaluation and tuning

**Tuning of hyperparameters:**

**1. Lasso (lasso_model):**
In order to maximize the strategy of the method of 'Lasso', we used the 'tune()' function to figure out the best penalty coefficient 'lambda' based on the minimum rmse. We set the size of cross-validation as twenty with detailed strata on our response 'percent_dem' and the level of 'grid_regular ' is 100. The tuned 'lambda' is 0.001176812, which is the penalty value we added to the following model. We updated the best 'lambda' in our linear model and only retained the variable with coefficients provided by 'lasso' that are greater than 0.001. To measure the performance of this model, we used v-fold cross validation with setting: 'set.seed(123)', fold number (v=10), and the mean of rmse is 0.07156774.

**2. Random Forest (rf_model):**
We tuned the selected hyperparameters: mtry, ntree, and min n. Here, 'min_n' denotes an integer for the minimum number of data points in a node that is required for the node to be split further; 'ntree' denotes an integer for the number of trees contained in the ensemble; and 'mtry' represents an integer for the number of predictors that will be randomly sampled at each split when creating the tree models. The tuning result gives us that the best 'min_n' should be 2, 'mtry' should be 50, and 'trees' should be 462. Gaining the best value for these three hyperparameters is able to help us construct a final random forest model by using the tidymodels. To measure the performance of this model, we used v-fold cross validation with setting: 'set.seed(123)', fold number (v=10), and the mean of rmse is 0.0728.

**3. Gradient Boosting Machines (gbm_model_1):**
We used 'gbm.step' from the 'dismo' package to assess the optimal number of boosting trees using v-fold cross validation. We set 'gbm.x' as the top 16 most contributed variables due to the large number of variables. In detail, we tuned the 'shrinkage' and decided that its optimal value is 0.04. Another important parameter is the interaction depth (the maximum depth of each tree) which we tuned and decided that the optimal value is 11. In addition, we tuned the number of trees as well and decided that the optimal value is 450. After finding the optimal hyperparameters which have minimum residual deviance, we used the 'gbm' function from the 'gbm' package to fit the model. To measure the performance of this model, we used v-fold cross validation with setting: 'set.seed(123)', fold number (v=10), and the mean of rmse is 0.06550481.

**4. Gradient Boosting Machines (gbm_model_2):**
We set 'gbm.x' as all the variables (85 in total) in the dataset. In detail, we tuned the 'shrinkage' and decided that its optimal value is 0.06. Another important parameter is the

interaction depth (the maximum depth of each tree) which we tuned and decided that the optimal value is 12. In addition, we tuned the number of trees as well and decided that the optimal value is 1000. After finding the optimal hyperparameters which have minimum residual deviance, we used the 'gbm' function from the 'gbm' package to fit the model. To measure the performance of this model, we used v-fold cross validation with setting: 'set.seed(123)', fold number (v=10), and the mean of rmse is 0.06580415.

| Model identifier | hyperparameters |
|---|---|
| lasso_model | lambda: 0.001176812 |
| rf_model | mtry: 50, ntree: 462, min_n: 2 |
| gbm_model_1 | n.trees:450, shrinkage:0.04, interaction.depth:11 |
| gbm_model_2 | n.trees:1000, shrinkage:0.06, interaction.depth:12 |

Table 3: **Tuning of hyperparameters**

| Model identifier | rmse |
|---|---|
| lasso_model | 0.07156774 |
| rf_model | 0.0728 |
| gbm_model_1 | 0.06550481 |
| gbm_model_2 | 0.06580415 |

Table 4: **The performance of each model**

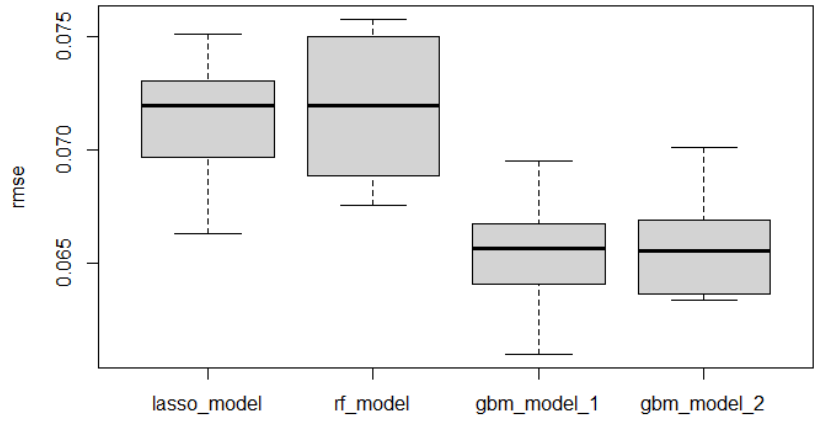



Figure 15: The distribution of rmse of the Four Models

According to Figure 15, we can conclude that 'lasso_model' and 'rf_model' have higher quantiles of rmse. It means models that are constructed by gradient boosting machines perform better. Hence, we decided to use 'gbm_model_1' and 'gbm_model_2' as our final models.

# 6 Discussion of final model

For the final two models, we chose both gradient boosting machines. We used different settings of predictors in the search for the optimal parameters for these two models to construct trees with various layers. In 'gbm_model_1', we used the top 16 most contributed variables and 'gbm_model_2' was tuned by all the variables. In detail, considering the size of variables is large and the need for multiple attempts, 'gbm_model_1' was motivated by fewer variables and more flexibility in finding the optimal parameters; and 'gbm_model_2' was motivated by the possibility of covering the most comprehensive set of variable possibilities and correlations for keeping most relationships. According to the final Kaggle results, 'gbm_model_1' performs a better prediction with the lower value of rmse. Hence, finding the most optimal parameters for gradient boosting machines is not an easy task. The best model is the result of a balance and we still have space to improve it.

In general, the advantages of gradient boosting machines are faster training speed and higher efficiency. It can optimize different loss functions and provide several hyperparameter tuning options that make the function fit flexibly. Specifically, we tried different settings of predictors 'gbm.x' in 'gbm.step' due to the large size of variables. 'gbm_model_1' contains fewer variables for more flexibility when building the trees. 'gbm_model_2' covers all the variables to minimize the losses. However, there is no such thing as a perfect model. The disadvantage of gradient boosting machines is the decision tree growth strategy, which treats leaves at the same level indiscriminately, and introduces a lot of unnecessary overhead. In fact, many of the leaves have low splitting gain and there is no need for searching and splitting.

To predict the percentage of voters in a county that voted for Biden in the 2020 US Presidential Election more accurately, we can try the method of Xgboost for further improvement. Xgboost is flexible, using the power of parallel processing, and supporting regularization, which is beneficial for prediction with the large size of variables and complex relationships. Besides, additional data could be useful for the purpose of accuracy. We think that collecting data on income and religious beliefs will be helpful. Biden proposes to raise taxes on rich people, so it is highly possible that high-income people are not likely to vote for Biden. Moreover, religion is also an important factor because there has been conflicts regarding the beliefs between the Democratic Party and the Republican Party.

# 7 Appendix: Final annotated script

```
```{r message=FALSE}
library(tree)
library(randomForest)
library(dismo)
library(gbm)
library(tidyverse)
```

1. Load data
```{r}
train <- read.csv("train.csv")
test <- read.csv("test.csv")
descriptions <- read.csv("column_descriptions.csv")
sample <- read.csv("sample_submission.csv")
```

2. Preprocessing
```{r}
# use `percent` data in `train` dataset
names <- colnames(train)
percent <- str_detect(names, "P")
percent.train <- train[which(percent==T)]
percent.train <- cbind(train[,c(2,3)], percent.train,
train[195:215])

# remove total_votes, under 18-year-old, `female`, X0033PE
percent.train <- percent.train[,-c(2,5, 6, 7, 8, 18, 19, 21, 22,
23, 24, 27, 4, 26, 29, 82, 30)]

# top 16 most contributed variable
top_contri <- c(76,18,44,78,56,47,45,25,86,59,84,19,27,12,74,57)
```

3. Final model 1 (gbm_model_1)
```{r message=FALSE}
# a. tuning of hyperparameters
# gbm.step: assess the optimal number of boosting trees using
v-fold cross validation
# gbm.x: top 16 most contributed variable
# gbm.y: percent_dem
# tree.complexity: sets the complexity of individual trees
# learning.rate: sets the weight applied to inidividual trees
set.seed(173)
gbm_cv <- gbm.step(data = percent.train, gbm.x =top_contri, gbm.y
= 1, family = 'gaussian', tree.complexity = 11, learning.rate =
0.04)

# b. fit the model
# extract the best hyperparameters in gbm.step: minimum residual
deviance
# n.trees (450): specifying the total number of trees to fit
# shrinkage (0.04): step-size reduction
```
```

```
# interaction.depth (11): specifying the maximum depth of each
tree (the highest level of variable interactions allowed)
gbm_model_1 <- gbm(percent_dem~., data = percent.train, n.trees =
gbm_cv$n.trees, shrinkage = gbm_cv$shrinkage, interaction.depth =
gbm_cv$interaction.depth)

# c. prediction
gbm_predict <- predict(gbm_model_1, test)
ans <- bind_cols(Id=test$id, Predicted=gbm_predict)
write.csv(ans,"result_1.csv", row.names = F)
```

3. Final model 2 (gbm_model_2)
```{r message=FALSE}
# a. tuning of hyperparameters
# gbm.step: assess the optimal number of boosting trees using
v-fold cross validation
# gbm.x: all the variables (85 in total)
# gbm.y: percent_dem
# tree.complexity: sets the complexity of individual trees
# learning.rate: sets the weight applied to inidividual trees
set.seed(173)
gbm_cv <- gbm.step(data = percent.train, gbm.x =2:86, gbm.y = 1,
family = 'gaussian', tree.complexity = 12, learning.rate = 0.06)

# b. fit the model
# extract the best hyperparameters in gbm.step: minimum residual
deviance
# n.trees (1000): specifying the total number of trees to fit
# shrinkage (0.06): step-size reduction
# interaction.depth (12): specifying the maximum depth of each
tree (the highest level of variable interactions allowed)
gbm_model_2 <- gbm(percent_dem~., data = percent.train, n.trees =
gbm_cv$n.trees, shrinkage = gbm_cv$shrinkage, interaction.depth =
gbm_cv$interaction.depth)

# c. prediction
gbm_predict <- predict(gbm_model_2, test)
ans <- bind_cols(Id=test$id, Predicted=gbm_predict)
write.csv(ans,"result_2.csv", row.names = F)
```

# 8 Appendix: Team member contributions

**Yuxuan Bai**:
1. explore the background information of the Presidential Election
2. write the report Part 1 and Part 2
3. use v-fold cross validation to measure the performance of the candidate models

**Yuetong Li**:
1. construct the linear regression model
2. figure out the significance of interactions between variables
3. write the report Part 6

**Jinghong Zou**:
1. construct the random forest model
2. tune the hyperparameters
3. write the report Part 4 and Part 5

**Xiaocong Xuan**:
1. construct the gradient boosting machines
2. tune the hyperparameters
3. write the report Part 3, Part 4, and Part 5

# 9 Reference

UC Business Analytics R Programming Guide: *Gradient Boosting Machines*

`http://uc-r.github.io/gbm_regression#:~:text=and%20efficient%20productionalization.-,gbm,original%20R%20implementation%20of%20GBM.`