Daniel Dai, Angelo Desiderio, Taylor Kim, Khang Luu, Harry Tong
STATS 101C, Summer 2024
03 August 2024

Kaggle Classification Competition Report:

Predicting the 2020 Presidential Election County Winners - Trump vs. Biden

INTRODUCTION

This data is sourced from the US Census Bureau in regards to the 2020 presidential election with a total of 3111 US counties. Our goal with the data is to predict the winner of the county (Biden/Trump). We believe certain variables such as race and education play a crucial role. This report breaks down our decision behind the use of a stacking model. We utilize a combination of EDA graphs, preprocessing techniques and model evaluation to obtain the most effective performing model.

EXPLORATORY DATA ANALYSIS

A large number of variables are related to the ethnicities and racial identities of voters in each county. Specifically, major groups of ethnicities according to the dataframe include Asian, American Indian and Alaska Native, Hawaiian and Other Pacific Islander, and Hispanic or Latino. We will further explore the distribution of these groups below.

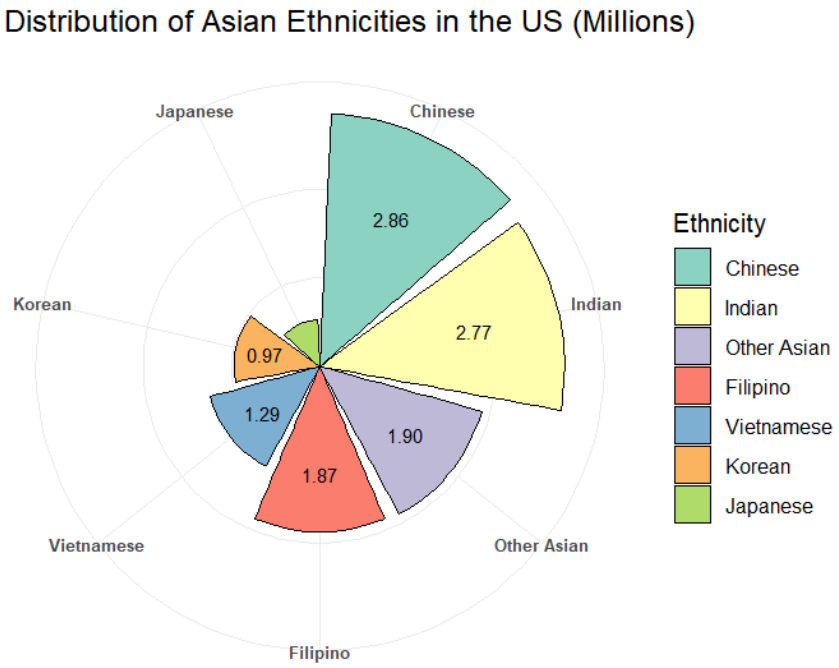

Figure 1: Distribution of Asian Ethnicities in the US. The largest Asian diasporas in the US are Chinese, Indians, Filipinos, and Vietnamese, all of which have more than one million people. These diaspora groups immigrated to the US at different times with Chinese and Japanese immigrants being some of the earliest to call the United States home.

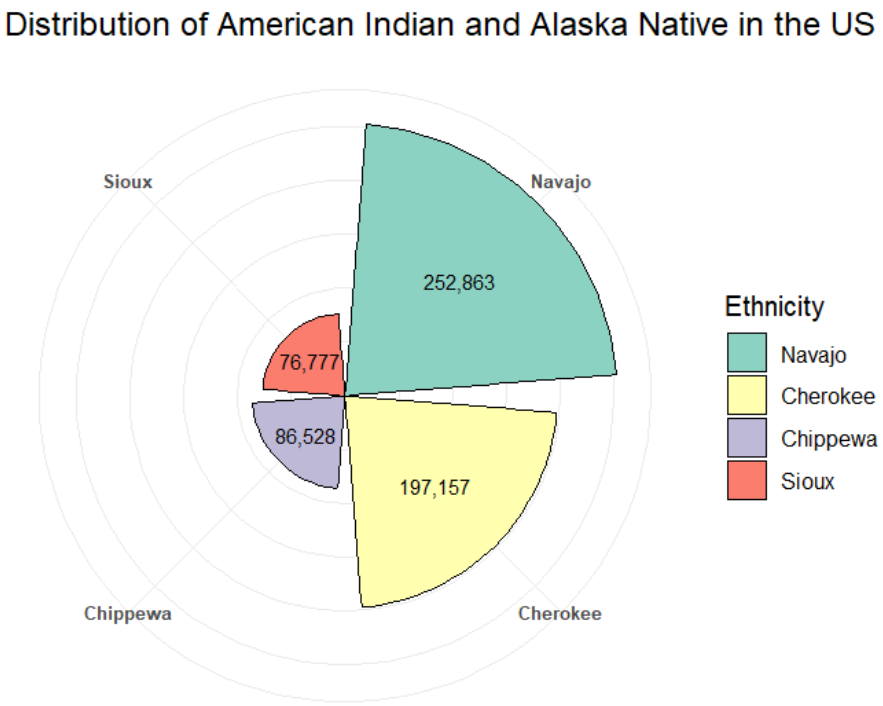

Figure 2: Distribution of American Indian and Alaska Native in the US. Native Americans are indigenous people native to North America. Unfortunately, as the United States expanded West, many tribes died out due to a variety of factors, including violent conflicts or starvations. Nowaday, most tribes live on reservations across the US.

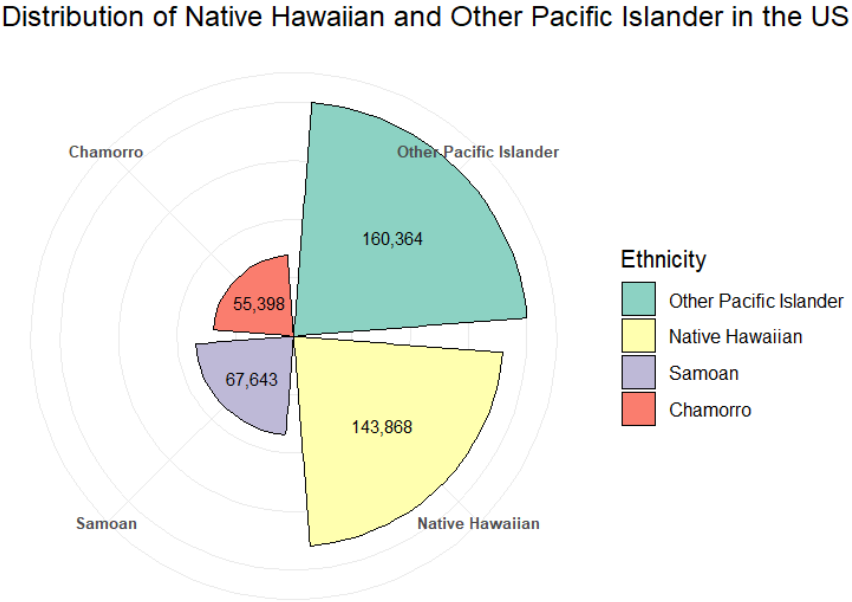

Figure 3: Distribution of Native Hawaiian and Other Pacific Islander in the US. Native Hawaiians are indigenous people native to the islands of Hawaii. The population declined dramatically after the US annexed Hawaii into a state and due to violent confrontations with the US militation. Pacific Islanders are people living on US territories in the Pacific Ocean, like Guam and American Samoa.

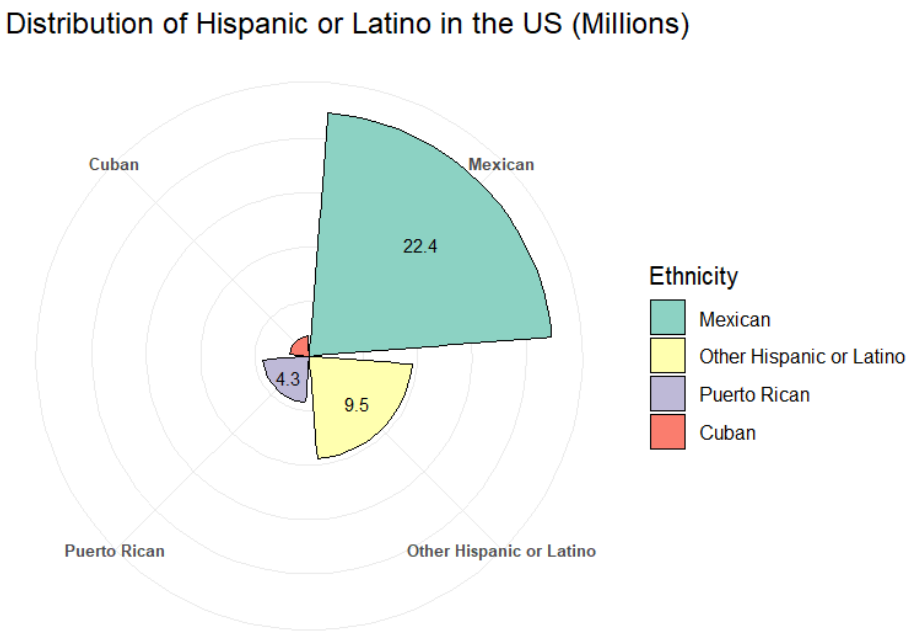

Figure 4: Distribution of Hispanic or Latino in the US. Mexicans are by far the largest ethnic group and most live in Western and Southern states near the Southern borders, like California and Texas. Puerto Ricans are from Puerto Rico, a US territory in the Caribbean.

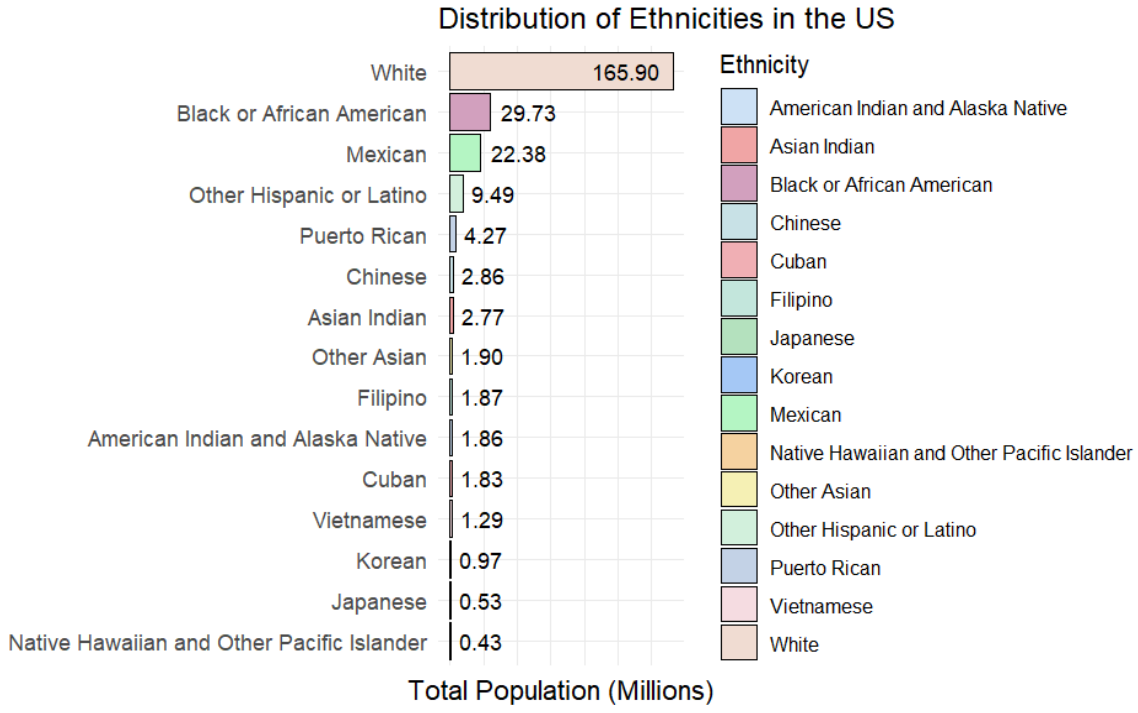

Figure 5: Ethnicity Distribution in the United States. White and African American or Black are the two largest groups in the US, followed by Mexican. Overall Asian ethnicities are a minority as no single group has more than 3 million people. Lastly, the smallest groups is Native Hawaiian and Other Pacific Islander with less than 500 thousand people.

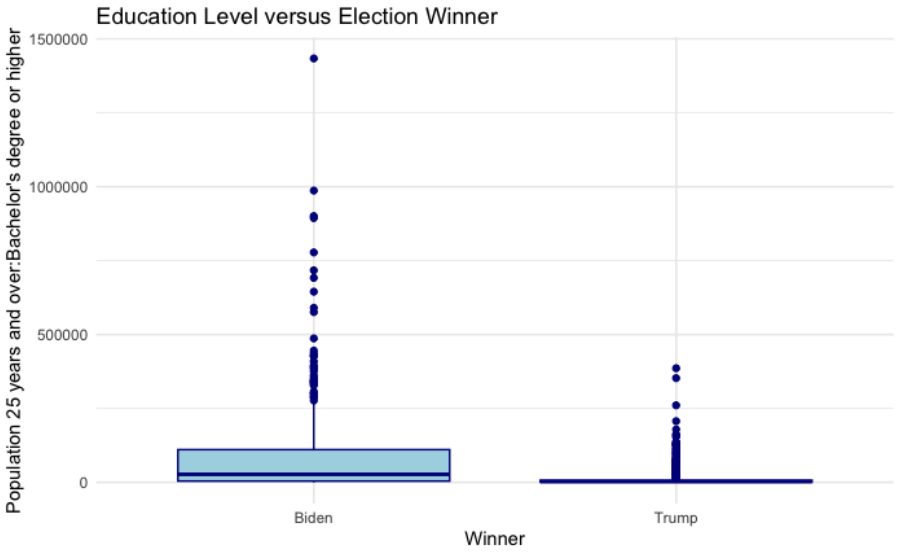

Figure 6: Distribution of the average total population of citizens aged 25 and older with a bachelor's degree or higher in each county. Due to the skewness in the data, we later transformed these variables into proportions. This boxplot epitomizes the issue of directly plotting counts of predictors by winner, a problem experienced by many raw predictors. To address this, we scaled the boxplots by converting the counts into proportions.

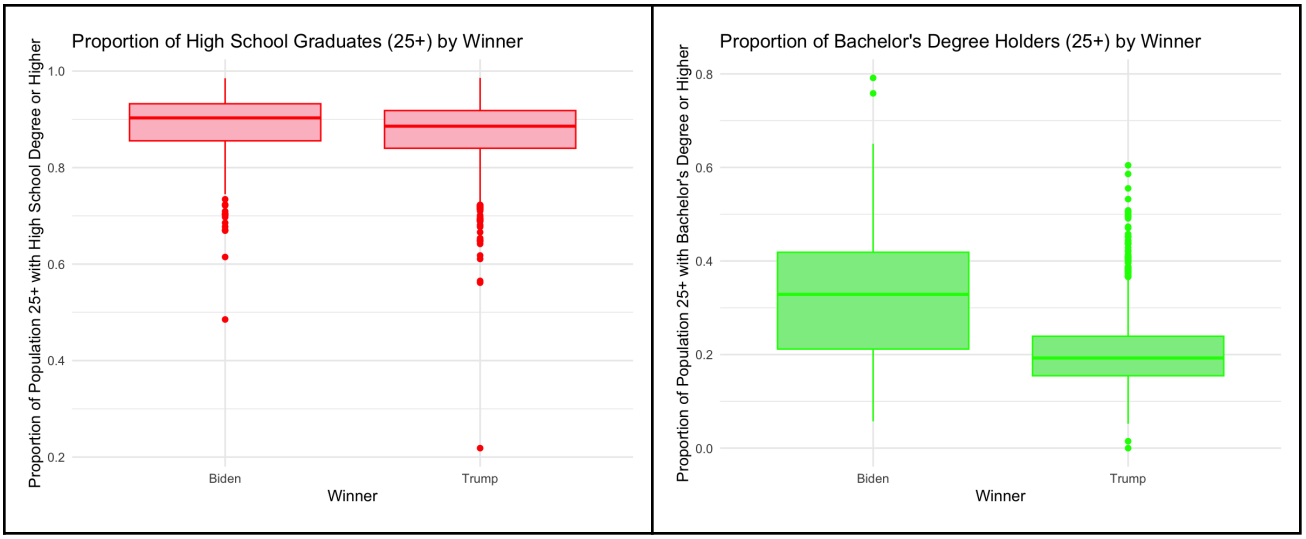

Figure 7: We created two boxplots for the 25+ age group, showing the proportions with high school and bachelor's degrees or higher, categorized by the election winner. The medians for high school education are relatively similar between the groups. However, in the bachelor's degree or higher plot, the median is noticeably higher for Biden, highlighting the significance of having a bachelor's degree or higher as a key factor. This trend holds for other age groups as well.

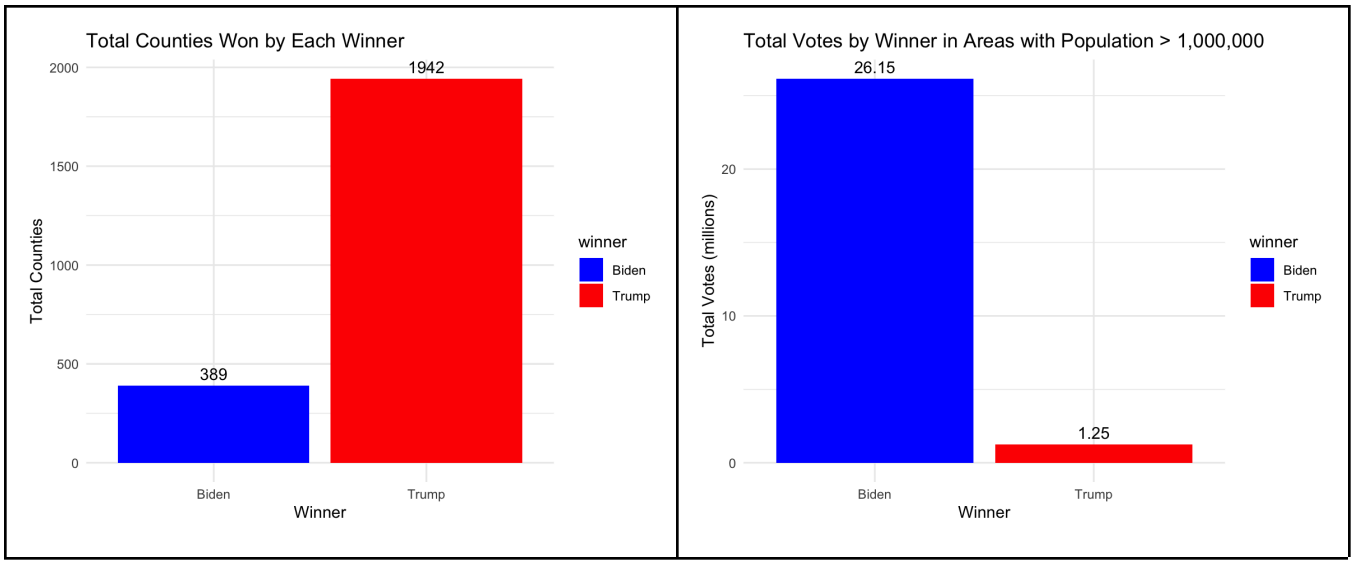

Figure 8: The left plot shows Biden won 389 counties, while Trump won 1942. Despite winning fewer counties, Biden dominated large counties, receiving over 26 million votes from counties with over a million people, making up 40% of his total votes to win the election.

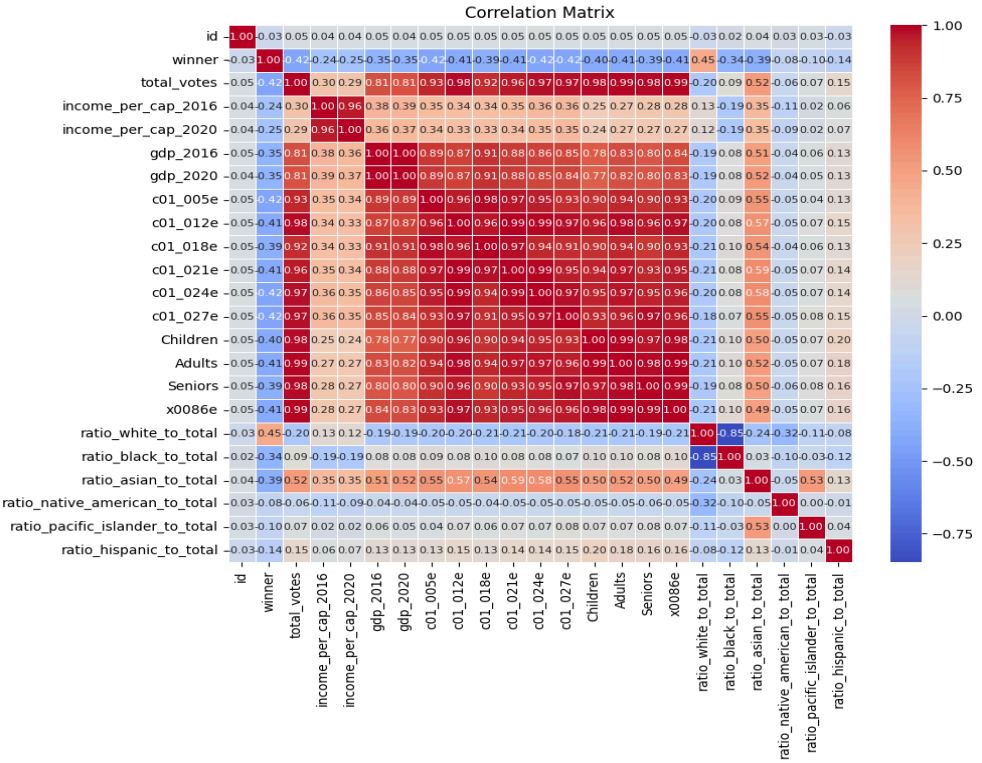

Figure 9: This matrix shows the correlation between the winner of the county and certain variables, one thing to highlight is the winner row being primarily blue colored but turns red when it reaches the ratio_white_to_total column. Which represents the amount of white voters proportional to the total population.

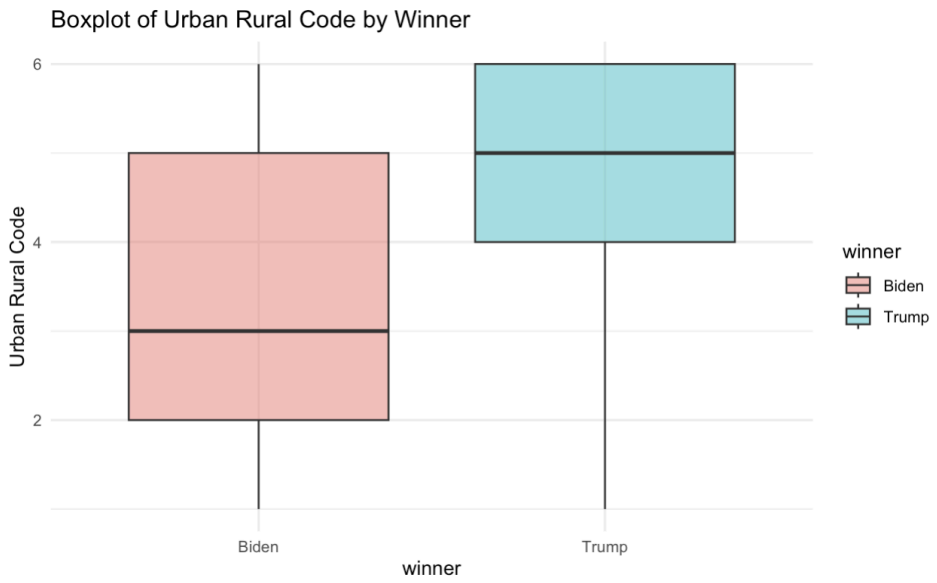

Figure 10: The Urban Rural Code ranges from 1 to 6, with 1 representing the most urban areas and 6 representing the most rural areas. The significant difference in the box plot indicates that Trump won more rural counties, while Biden won more urban areas.

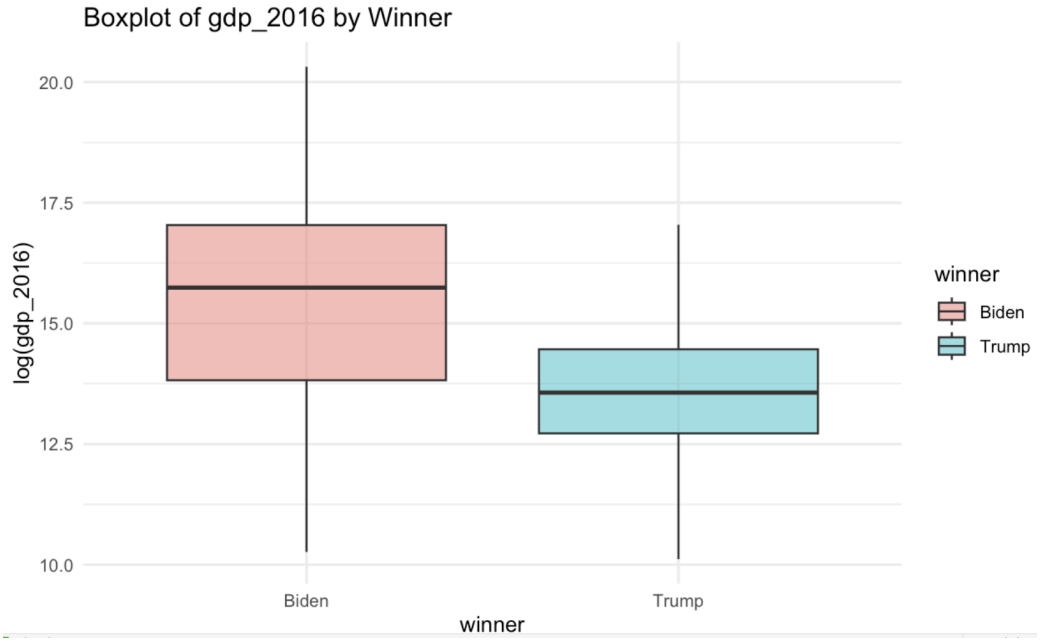

Figure 11: The GDP data is also skewed, so we applied a logarithmic transformation. This reveals that Biden won counties with substantially higher GDP than those won by Trump. This trend has been consistent from 2016 to 2020.

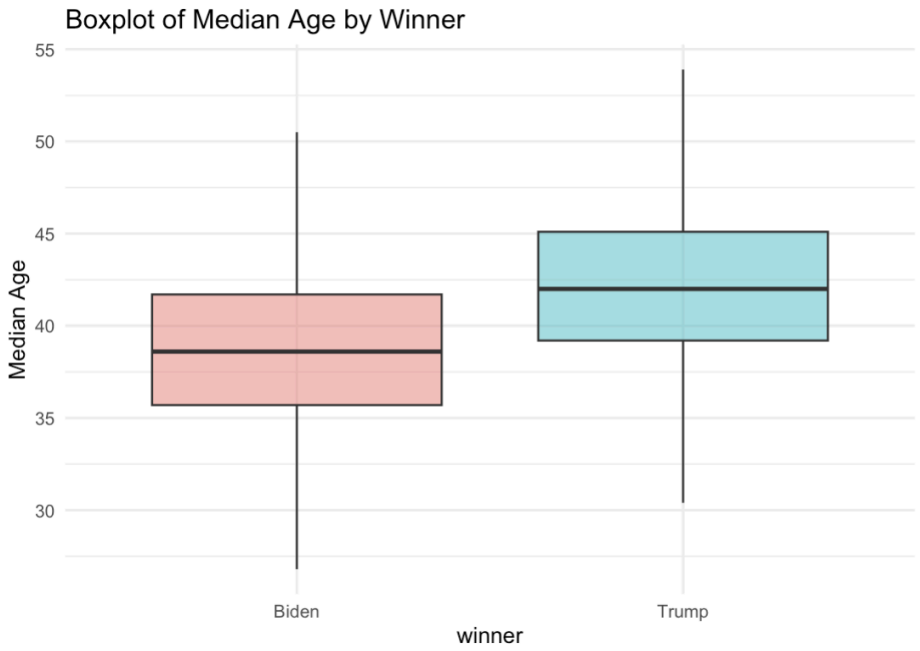

Figure 12: Trump won counties with a higher median age, while Biden won counties with a slightly younger median age. This trend is evident when examining the proportions of specific age groups. Biden won counties with higher proportions of people in the 15-19, 20-24, and 35-44 age categories, while Trump won counties with higher proportions of people in the 55-59, 60-64, and 65-74 age categories, as well as those older than 74.

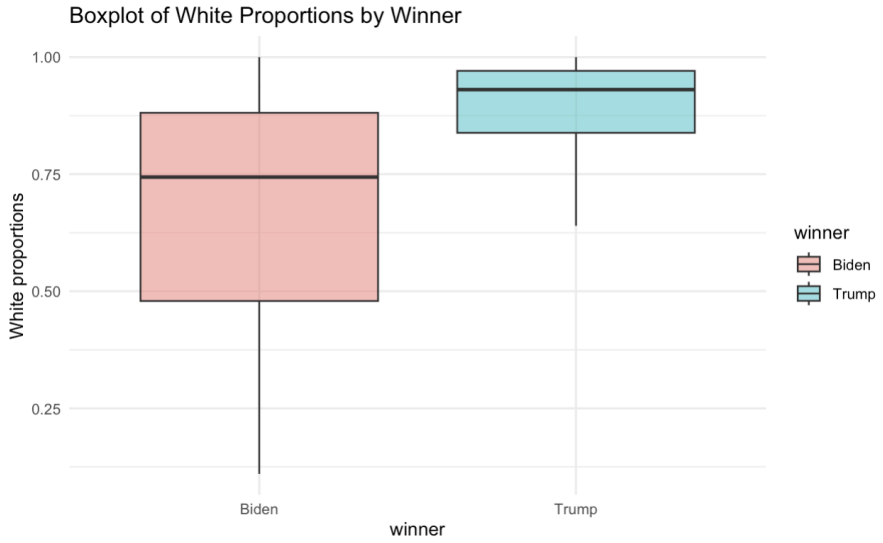

Figure 13: Trump won counties with higher proportions of white residents, with a striking median difference of almost 20 percent. It's important to note that even in terms of proportions, the distributions are slightly skewed. In our model, this variable, along with the proportion of black residents, emerged as one of the strongest predictors.

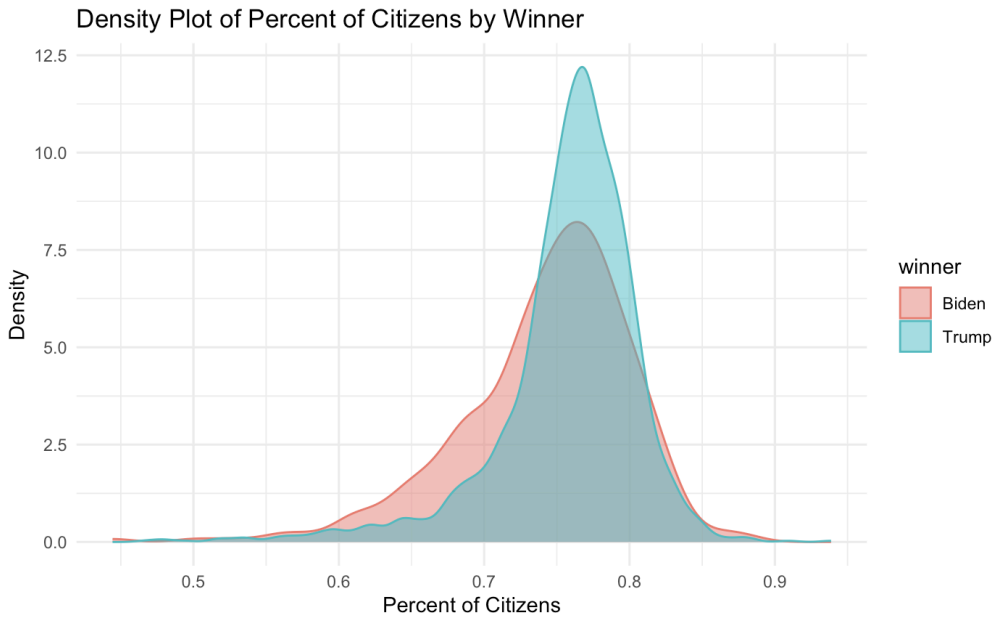

Figure 14: The density plot shows that counties with lower percentages of citizens to total population are more likely to vote for Biden. So it is important to transform the citizens' variables to percent of citizens over total population and keep it as a predictor.

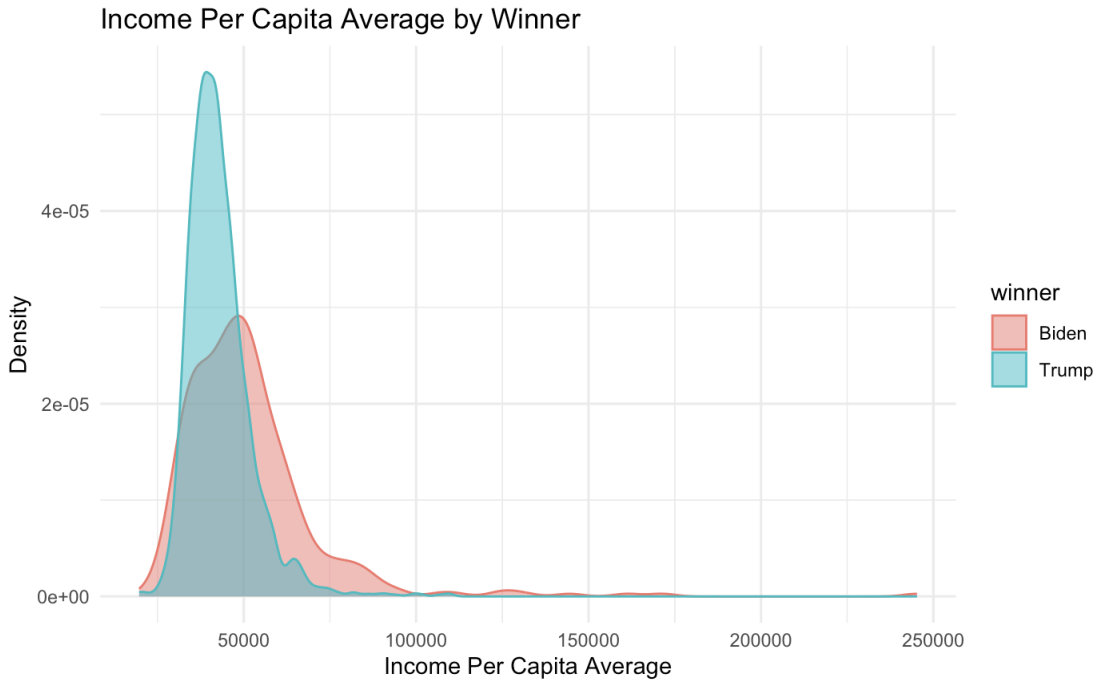

Figure 15: The density plot shows that counties with higher income per capita during the past years are more likely to vote for Biden, and it shows that income per capita is related to the classification of winners, thus we should keep it as our predictors.

PREPROCESSING/RECIPES

We create two major recipes, and cross them with 6 different models to create a total of 12 models.

Recipe 1: The main focus of this recipe is to transform total counts into proportions to address skewness in the boxplot by winner.

Step_impute_median(): Certain counties are missing GDP and income per capita data. As these data are heavily skewed, we use the median rather than the mean.

Step_mutate(): This step serves two purposes:

1. Transform counts into proportions:
    a. Age Section: We divide all age category counts by the total population.
    b. Race Section: We divide each race count by the respective total counts they belong to. For example, total whites are divided by the total one-race population. For different Asian ethnic groups, we divide by the total Asian population rather than the total one-race population.
    c. Education Section: We divide the counts of individuals with a bachelor's degree or higher for each age group by the total population of that age group.
2. Create new variables
    a. Proportion of Citizens: We divide the number of citizens by the total population.
    b. Mean Household Size: We divide the total population by the total number of housing units.

Step_interact(): Since race counts (columns 37e to 57e) and non-Hispanic race counts (columns 76e to 82e) describe similar metrics, we create interaction terms between these variables.

Step_log(): Given the skewness of the GDP data and the absence of a baseline like total population for scaling, we apply a logarithmic transformation.

Update_role(): Some variables are duplicated:

1. Literal Duplicates: Columns like the population of individuals aged 18 or younger appear more than once.
2. Redundant Variables: Variables such as total male and total female are redundant because knowing one allows us to infer the other.

Step_dummy(): The x2013_code ranges from 1 to 6 and reflects the ruralness or advanceness of the county. We treat it as a categorical variable and create dummy variables to represent its categories.

Recipe 2: The main focus for this recipe was to use key ethnic and demographic predictors and add other variables that increased the accuracy. A total of 20 variables were used in this model.

Step_log(): handle skewed distribution of predictor variables relating to education and ethnicity.

Step_impute_mean(): Dataset contains numerous missing values that needed to be imputed so that they could be used as predictors, especially if they were found to be variables of interest.

MODEL EVALUATION

Model Identifier	Model Type	Engine	Recipe	Hyperparameters
xgboost_rec1	Boosted Tree	xgboost	Recipe 1: Step_mutate() Step_impute_median() Step_log() Step_interact() Step_dummy() Update_role()	Trees = 436 Min_n = 7 Tree_depth = 7 Learn_rate = 0.0592 Loss_reduction = 0.00000000236 Sample_size = 0.865 Stop_iter = 18

knn_rec1	Nearest Neighbor	kknn	Recipe 1	Neighbors = 15
glm_rec1	Logistic Regression	glm	Recipe 1	NA
rf_rec1	Random Forest	ranger	Recipe 1	Trees = 249 Min_n = 5
svm_rec1	Support Vector Machine	kernlab	Recipe 1	Cost = 2.33 Rbf_sigma = 0.000820 Margin = 0.0374
dt_rec1	Decision Tree	rpart	Recipe 1	cost_complexity = 0.00000000144 tree_depth = 4 min_n = 31
xgboost_rec2	Boosted Tree	xgboost	Recipe 2: step_log() step_impute_median()	Trees = 436 Min_n = 7 Tree_depth = 7 Learn_rate = 0.0592 Loss_reduction = 0.00000000236 Sample_size = 0.865 Stop_iter = 18
knn_rec2	Nearest Neighbor	kknn	Recipe 2	Neighbors = 15
glm_rec2	Logistic Regression	glm	Recipe 2	NA
rf_rec2	Random Forest	ranger	Recipe 2	Trees = 249 Min_n = 5
svm_rec2	Support Vector Machine	kernlab	Recipe 2	Cost = 2.33 Rbf_sigma = 0.000820 Margin = 0.0374
dt_rec2	Decision Tree	rpart	Recipe 2	cost_complexity = 0.00000000144 tree_depth = 4 min_n = 31

In this model, we include 6 models, and cross them with 2 recipes to create a total of 12 models.

Decision Tree: It splits the data into subsets based on the value of input features, with each node representing a feature. But without pruning, there's a higher risk of overfitting. Key hyperparameters include cost_complexity(trade-off between the tree's complexity and its fit to the training data), tree_depth, and minimum number of samples required at each split.

Random Forest: an ensemble of decision trees created by bootstrapping data and randomly selecting variables for each tree, which reduces overfitting. Key parameters include the number of tree and min_n.

Boosted Tree: Boosting involves sequentially adding decision trees to adjust errors made by the previous models. In addition to decision tree hyperparameters, key hyperparameters also include learn_rate(step size at each iteration), loss_reduction(Minimum loss reduction required to make a further partition), stop_iter(Number of iterations).

Nearest Neighbor: It classifies a data point based on the majority class of its k nearest neighbors in the feature space, which may be hard to define given a complex dataset.

Logistic Regression: estimates the probability that a given input belongs to a particular class, which is perfect for binary classification problems in this project..

Support Vector Machine (SVM): find the hyperplane that best separates the data into classes.

Model Identifier	Metric Score (Accuracy)	SE of metric
Boosted Tree	0.939	0.00478
Random Forest	0.936	0.00541
Support Vector Machine	0.925	0.00510
Logistic Regression	0.922	0.00646
Decision Tree	0.919	0.00543
Nearest Neighbor	0.909	0.00365

Table shows accuracy scores for each model. Note that Recipe 1 outperforms Recipe 2, so we present the best-performing models using Recipe 1. By creating a workflow set with a grid of 10, the best-tuned models all exceed 90% accuracy. Boosted tree and random forest models perform the best, each exceeding 93.5% accuracy.

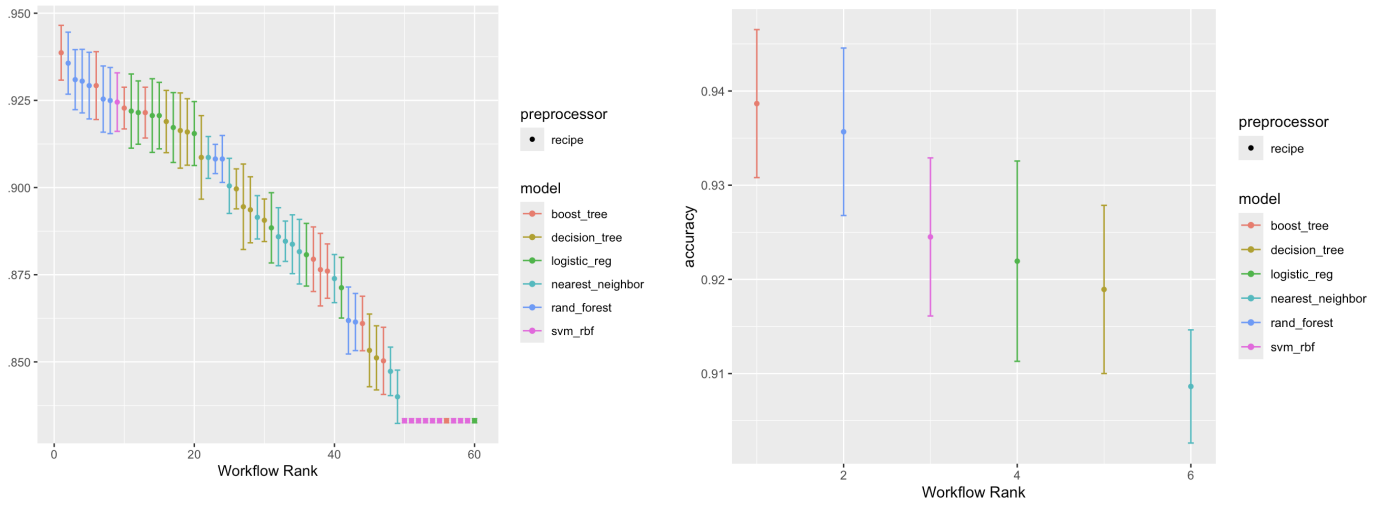

Figure 16: the best performing models for recipe 1 were boosted tree, random forest, support vector machine, logistic regression, decision tree, and nearest-neighbor. Their accuracy was above 90 percent.

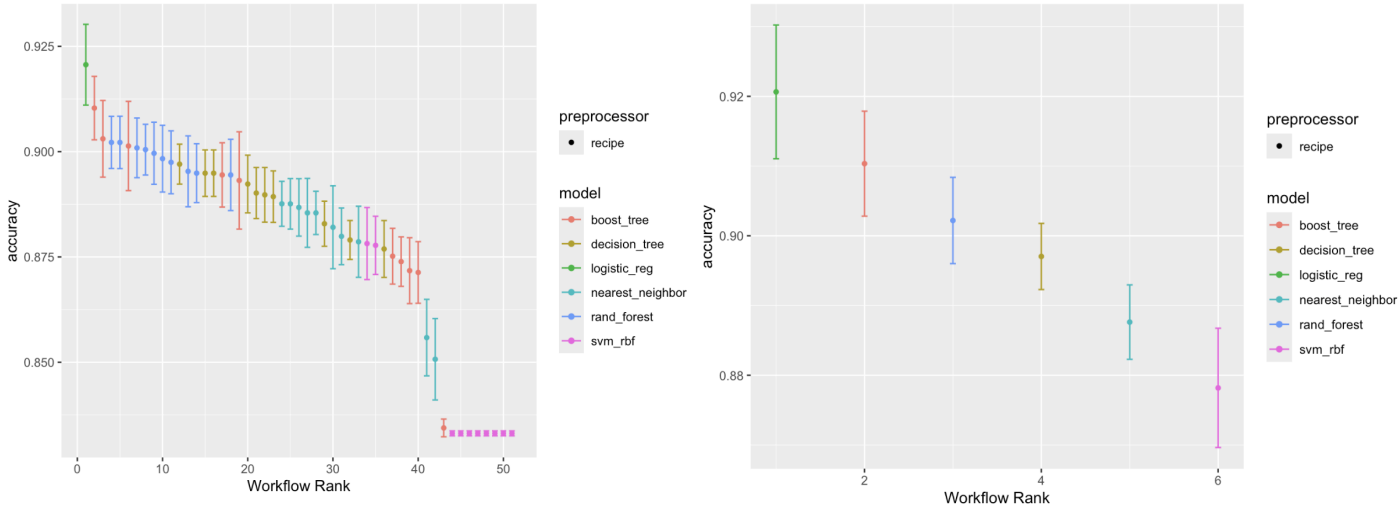

Figure 17: For recipe 2, the top three models were logistic regression, boosted tree, and random forest. Notably, recipe 1 performed better than recipe 2 because all 6 models in recipe 1 had accuracy over 90 percent whereas decision tree, nearest-neighbor, and support vector machine models had accuracy less than 90 percent for recipe 2. These results led us to use boosted tree and random forest models in the final model.

DISCUSSION OF FINAL MODEL

Given that all six models perform exceptionally well, we decided to combine them using a stacking model. Since the boosted tree and random forest models performed particularly well, we included the three best-tuned versions of each. Combined with the other four models, this resulted in a total of ten candidate models. Stacking involves comparing and blending the results of these models by assigning weights to each. The figure shows that boosting received significant weight, followed by logistic regression and random forest. By stacking the models together, we improved prediction accuracy and reduced overfitting, achieving a 5th place public score and a 4th place private score.

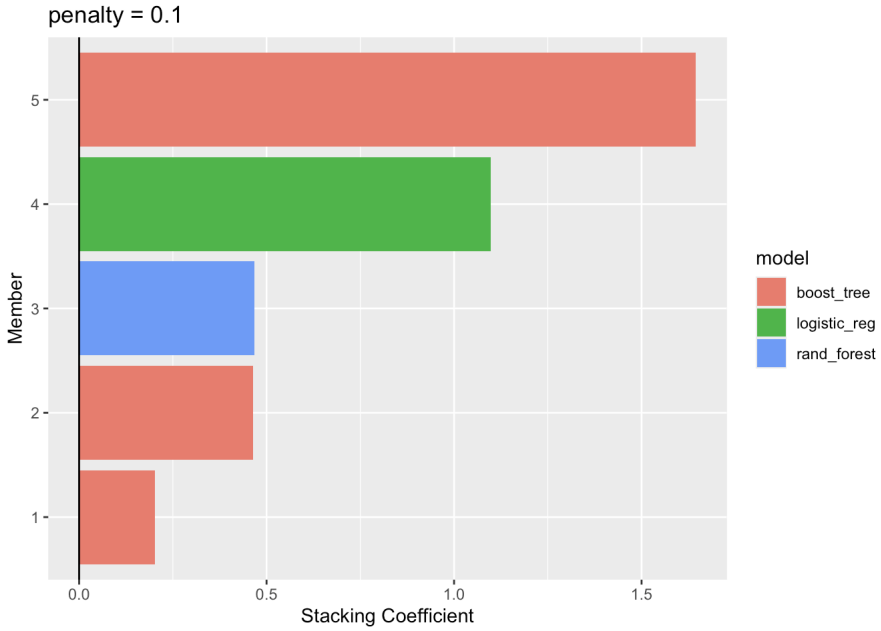

Figure 18: Weights of Different Models in the Stacking Ensemble

Choosing the right recipe is crucial for this project. The key step we took was converting counts into proportions, which properly scaled the boxplots. This revealed that the proportions of whites and African Americans in a county significantly influence the selection of the winner. Unlike our previous regression project that solely used random forest, stacking diverse models also improved the robustness of our predictions.

However, a weakness lies in our inability to remove redundant variables. EDA and domain knowledge indicate that certain variables (e.g., Native American ethnic group population, high school or higher education population, population aged 5 years or older) are less significant. Removing them, however, decreases predictability, forcing us to retain them and resulting in an overly complex model, hindering our ability to interpret the most important factors.

Potential improvements include better domain knowledge and more thorough EDA. Improved research on what influences elections the most would help us see if these patterns are represented in our data. Obtaining additional data, such as race data by education or age, could reveal significant patterns. This is because our current model shows that race significantly affects outcomes, so combining race with other variables could uncover important insights.

APPENDIX I: FINAL ANNOTATED SCRIPT

```
set.seed(2024)      #Set seed
library(tidymodels) # load necessary libraries
library(tidyverse)
library(stacks)

train <- read_csv("train_class.csv") #Read in training and test set
train <- train %>%
          select(-name)
test <- read.csv("test_class.csv")

train_folds <- vfold_cv(train, v = 10, strata = winner) #Create folds

ctrl_grid <- control_stack_grid() #Set control settings
ctrl_res <- control_stack_resamples()

# Create 3 random forest models with top 3 hyperparameters settings
rf1 <- rand_forest(trees = 249, min_n = 5) %>%
  set_engine("ranger") %>%
  set_mode("classification")
rf2 <- rand_forest(trees = 703, min_n = 11) %>%
  set_engine("ranger") %>%
  set_mode("classification")
rf3 <- rand_forest(trees = 538, min_n = 5) %>%
  set_engine("ranger") %>%
  set_mode("classification")

# Create a knn model
knn <- nearest_neighbor(neighbors = 15) %>%
  set_engine("kknn") %>%
  set_mode("classification")

# Create a logistic regression model
glm <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

# Create 3 boosted tree models with top 3 hyperparameters settings
xgboost1 <- boost_tree(trees = 436, min_n = 7, tree_depth = 7, learn_rate =
0.0592, loss_reduction = 0.00000000236, sample_size = 0.865, stop_iter = 18) %>%
  set_engine("xgboost") %>%
  set_mode("classification")
xgboost2 <- boost_tree(trees = 249, min_n = 4, tree_depth = 4, learn_rate =
0.0332, loss_reduction = 0.268, sample_size = 0.769, stop_iter = 20) %>%
  set_engine("xgboost") %>%
  set_mode("classification")
xgboost3 <- boost_tree(trees = 421, min_n = 4, tree_depth = 8, learn_rate =
0.140, loss_reduction = 0.00145, sample_size = 0.833, stop_iter = 5) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

# Create a decision tree model with tuned parameters
```

```
dt <- decision_tree(cost_complexity = 0.00000000144, tree_depth = 4, min_n = 31)
%>%
  set_engine("rpart") %>%
  set_mode("classification")

# Create support vector machine model using tuned parameters
svm <- svm_rbf(cost = 2.33, rbf_sigma = 0.000820, margin = 0.0374) %>%
  set_engine("kernlab") %>%
  set_mode("classification")

# Create the recipe
recipe <- recipe(winner ~ ., data = train) %>%
  # Impute missing values using median
  step_impute_median(all_numeric_predictors()) %>%
  # Convert age data into proportions
  step_mutate(across(c(x0002e, x0003e, x0005e, x0006e, x0007e, x0008e, x0009e,
x0010e, x0011e, x0012e, x0013e, x0014e, x0015e, x0016e, x0017e, x0019e, x0020e,
x0021e, x0022e, x0023e, x0024e), ~ . / x0001e)) %>%
  step_mutate(across(c(x0026e, x0027e), ~ . / x0025e)) %>%
  step_mutate(across(c(x0030e, x0031e), ~ . / x0029e)) %>%
  step_mutate(across(c(x0034e, x0035e), ~ . / x0033e)) %>%
  # Convert race data into proportions
  step_mutate(across(c(x0040e, x0041e, x0042e, x0043e), ~ . / (x0039e + 0.1))) %>%
# Convert Indian American data into props
  step_mutate(across(c(x0045e, x0046e, x0047e, x0048e, x0049e, x0050e, x0051e), ~ .
/ (x0044e + 0.1))) %>%  #Calculate proportions of Asian groups by dividing each
group by total Asians
  step_mutate(across(c(x0053e, x0054e, x0055e, x0056e), ~ . / (x0052e + 0.1))) %>%
# Calculate proportions of Pacific Islander ethnic groups by dividing each group
by total Pacific Islanders
  step_mutate(across(c(x0037e, x0038e, x0039e, x0044e, x0052e, x0057e), ~ . /
x0036e)) %>%  # Calculate proportions of each race group by diving by total one
race population
  step_mutate(across(c(x0077e, x0078e, x0079e, x0080e, x0081e, x0082e, x0083e,
x0084e, x0085e), ~ . / x0076e)) %>%  # Calculate proportions of Non-Hispanic
ethnic groups by dividing each group by total Non-Hispanics
  step_mutate(across(c(x0072e, x0073e, x0074e, x0075e), ~ . / (x0071e + 0.1))) %>%
# Calculate proportions of Hispanic ethnic groups by dividing each group by total
Hispanics

# Education section: Calculate the proportion of individuals with a bachelor's
degree or higher for each age group
    step_mutate(
    c1824 = c01_005e / (c01_001e + 0.1),
    c2534 = c01_018e / c01_016e,
    c3544 = c01_021e / c01_019e,
    c4564 = c01_024e / c01_022e,
    c65over = c01_027e / c01_025e) %>%
 # Log transformation of GDP variables
  step_mutate_at(all_of(c("gdp_2016", "gdp_2017", "gdp_2018", "gdp_2019",
"gdp_2020")), fn = ~log(. + 1)) %>%

 # Interaction terms: Between one race and non-Hispanic proportion
  step_interact(terms = ~ x0037e:x0077e) %>%
```

```
 # Create dummy variable
  step_mutate(x2013_code = as.factor(x2013_code)) %>%
  step_dummy(x2013_code) %>%
 # Create new variables: mean household size & citizen proportion
  step_mutate(citi_prop = x0087e / x0025e) %>%
  step_mutate(avg_household = x0001e / x0086e) %>%
 # Remove duplicated variables from predictors
  update_role(c("id", "c01_001e", "c01_006e", "c01_016e", "c01_019e", "c01_022e",
"c01_025e", "x0003e", "x0025e", "x0029e", "x0035e", "x0033e", "x0036e", "x0058e"),
new_role = "id")

# Fitting Resamples: Create and fit resamples to knn model
knn_wflow <-
  workflow() %>%
  add_model(knn) %>%
  add_recipe(recipe)
knn_res <- knn_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
# Create and fit resamples to logistic regression model
glm_wflow <-
  workflow() %>%
  add_model(glm) %>%
  add_recipe(recipe)
glm_res <- glm_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
# Create and fit resamples to 3 random forest models
rf1_wflow <-
  workflow() %>%
  add_model(rf1) %>%
  add_recipe(recipe)
rf1_res <- rf1_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
rf2_wflow <-
  workflow() %>%
  add_model(rf2) %>%
  add_recipe(recipe)
rf2_res <- rf2_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
rf3_wflow <-
  workflow() %>%
  add_model(rf3) %>%
  add_recipe(recipe)
rf3_res <- rf3_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
# Create and fit resamples to 3 boosted tree models
xgboost1_wflow <-
  workflow() %>%
  add_model(xgboost1) %>%
  add_recipe(recipe)
xgboost1_res <- xgboost1_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
xgboost2_wflow <-
  workflow() %>%
  add_model(xgboost2) %>%
```

```
  add_recipe(recipe)
xgboost2_res <- xgboost2_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
xgboost3_wflow <-
  workflow() %>%
  add_model(xgboost3) %>%
  add_recipe(recipe)
xgboost3_res <- xgboost3_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
# Create and fit resamples to svm model
svm_wflow <-
  workflow() %>%
  add_model(svm) %>%
  add_recipe(recipe)
svm_res <- svm_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)
# Create and fit resamples to decision tree model
dt_wflow <-
  workflow() %>%
  add_model(dt) %>%
  add_recipe(recipe)
dt_res <- dt_wflow %>%
  fit_resamples(resamples = train_folds, control = ctrl_res)

# Stacking: add 10 ensemble models into stack
stack_model <-
  stacks() %>%
  add_candidates(knn_res) %>%
  add_candidates(glm_res) %>%
  add_candidates(svm_res) %>%
  add_candidates(rf1_res) %>%
  add_candidates(rf2_res) %>%
  add_candidates(rf3_res) %>%
  add_candidates(xgboost1_res) %>%
  add_candidates(xgboost2_res) %>%
  add_candidates(xgboost3_res) %>%
  add_candidates(dt_res)

# Fit the stack using blend_predictions
stack_model <-
  stack_model %>%
  blend_predictions()
# Fit the coefficients back onto entire training set
stack_model <-
  stack_model %>%
  fit_members()
# Create predictions for the test set
model_predictions <- predict(stack_model, new_data = test)

# Bind columns and output as csv file
test <- test %>% select(id) %>% bind_cols(model_predictions)
test <- test %>% rename(winner = .pred_class)
write_csv(test, "verify_predictions.csv")
```

APPENDIX II: TEAM MEMBER CONTRIBUTION

Member	Contribution
Daniel Dai	EDA(Figure 10-13), Recipe 1, Model Evaluation, Create stacks as final model, Annotated Script
Angelo Desiderio	Introduction. Examined and performed EDA on age columns. Created rf model for comparison purposes.
Taylor Kim	EDA for county population and education columns, team management.
Khang Luu	Focus on EDA for race columns. Created the second recipe using key predictors, model performance plot analysis.
Harry Tong	Explored alternative candidate models/parameters, provided suggestions for handling data.

WORKS CITED

Miles Chen. (2024). UCLA Stats 101C 2024 Summer - Classification. Kaggle. https://kaggle.com/competitions/ucla-stats-101-c-2024-summer-classification