6103 Final Project Report

Bradley Reardon, Kushal Ismael, Divya Parmar
The George Washington University
May 2, 2021

Introduction

The 2020 US Presidential Election caused a great deal of speculation surrounding voter-turnout, largely driven by the high-profile candidates and expanded voting methods enacted to allow voters to stay safe from COVID-19. Furthermore, there is a long-standing interest in what drives eligible voters to either vote or not vote, and who to vote for if a vote is cast. We decided to dive deeper into the latter and explore which characteristics and features of a voter drove them to choose between Donald Trump and Joe Biden in the 2020 US Presidential Election.

In this paper, we explain how we conducted EDA on a survey results dataset, determined which features highly influenced voter decisions, and trained both a random forest and a gradient boosting classifier to predict which candidate a voter will vote for based on survey answers and demographic information.

Dataset Description

The dataset used in this project contains the results from a survey conducted in September 2020 by Ipsos, a multinational market research and consulting firm, and FiveThirtyEight, an American data journalism website that focuses on opinion poll analysis, politics, economics, and sports. The survey focused on political subjects, including what it means to be a good American, how much the survey respondent agrees or disagrees with various statements regarding systemic racism, their trust and faith in the US government, how they are affected by government policy making, thoughts on voting and past voting actions, demographic information, and who they plan to vote for in the presidential election. The raw dataset contained 119 features (columns) and 5836 observations (rows), but we reduced the number of features to 95 after deciding some were unnecessary for our project. All features are categorical due the survey questions prompting predefined answers from a given set of options. The target variable, Question 23 in the survey, asked “Which presidential candidate are you planning to support?” and the given options were, “Donald Trump”, Joe Biden”, and “Unsure.”

We chose our target variable (Question 23) over our other consideration (Question 21) because Question 23 responses were more balanced than Question 21 responses. Classifier models do worse with imbalanced datasets, and that was the deciding factor when choosing between Q21 (yes/no) and Q23 (Trump/Biden).

Question 21 - Do you plan to vote in the November 2020 Election?

Option	Response Count
Yes	4,945
No	464

Unsure/Undecided	404
Refused	23

Question 23 - Which presidential candidate are you planning to support?

Option	Response Count
Donald Trump	1,907
Joe Biden	2,777
Unsure	1,043
Refused	109

Data Mining/Learning Techniques

The two machine learning techniques we decided to use are random forest and gradient boosting classifiers. Random forest and gradient boosting classifiers are similar in that they both are sets of decision trees. The two classifiers differ in two main ways: how the trees are built, and how the results are combined. In a random forest, decision trees are built independently of one another in parallel, and the results are combined at the end by averaging the results from each tree. The gradient boosting classifier builds a sequence of decision trees with each tree prioritizing weak learners from the previous tree to improve its classification ability while combining results along the way.

Exploratory Data Analysis

To begin our EDA, we renamed all of the features to better reflect what each feature is since they were originally titled, Q1, Q2, etc. Next we explored which features provided complete answers and dropped those that only had partial responses (i.e. "Which type of Republican are you?") as it warranted no response from Democrats and thus gave a null value. We then moved onto checking our dataset for balance by checking the normality of the demographic features. An additional column, Age_group, was added to check the balance of the survey takers' ages.

Pie Charts

Our first method of checking balance amongst the demographic features was through pie charts. Here we can see that Gender, Age, Education Level, and Income Category are all fairly balanced. The race of survey takers is predominately white, with 64% of survey takers identifying as white (although this matches up with the actual voter demographics in the 2020 election).

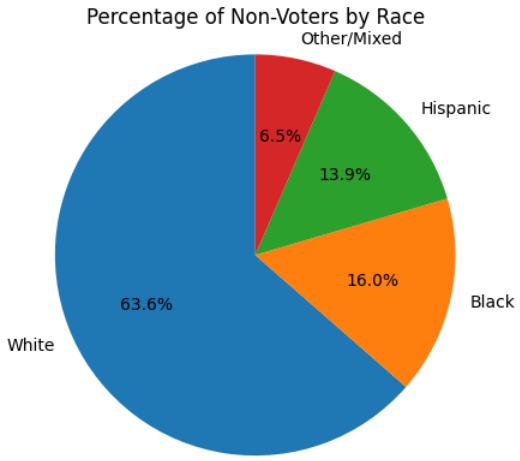

Figure 1.) Percentage of Non-Voters by Race

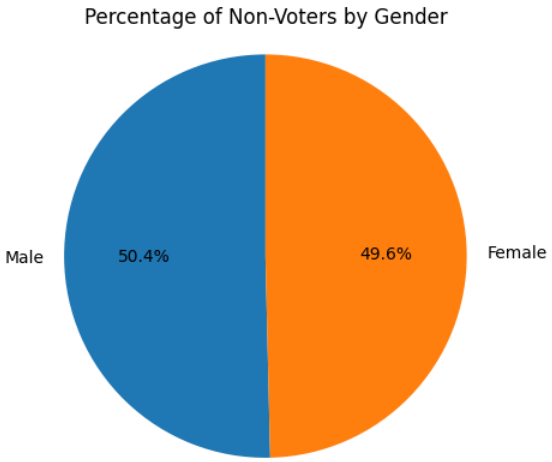

Figure 2.) Percentage of Non-Voters by Gender

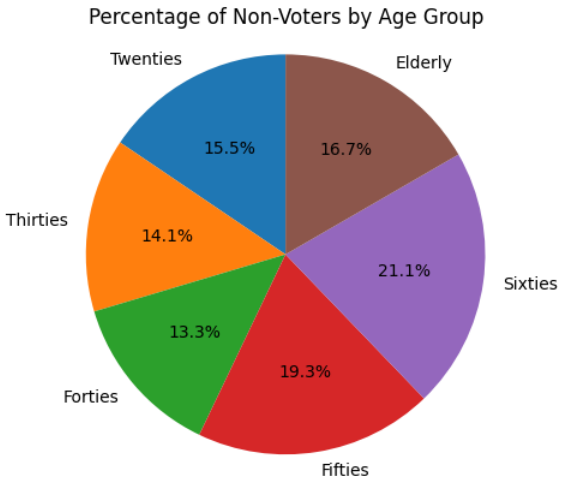

Figure 3.) Percentage of Non-Voters by Age Group

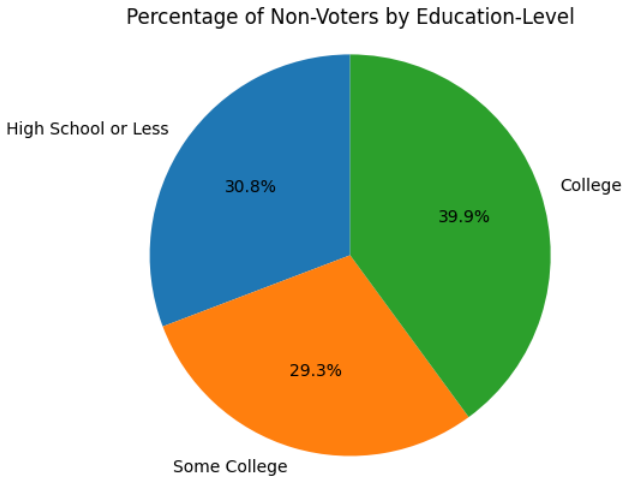

Figure 4.) Percentage of Non-Voters by Education-Level

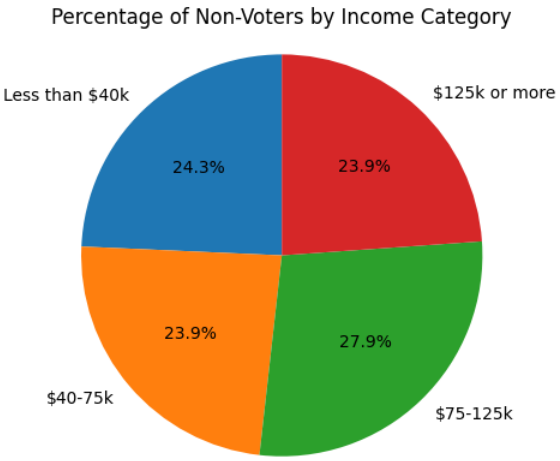

Figure 5.) Percentage of Non-Voters by Age Group

Histograms

Below are histogram views of the same data modeled in the pie charts above. The histograms allow us to better see that gender, age group, education level, and income category are all fairly balanced while race is unbalanced.

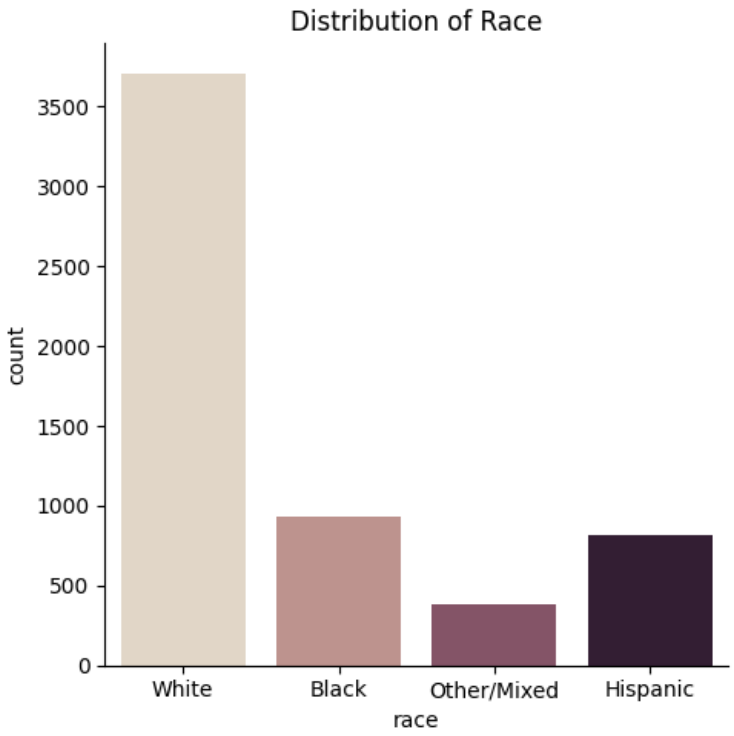

Fig 6.) Distribution of Race

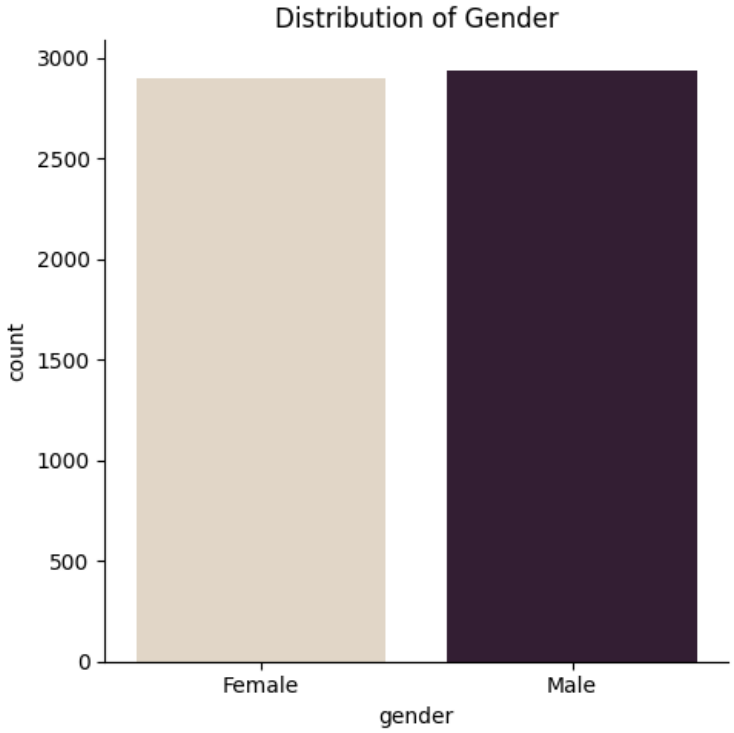

Fig 7.) Distribution of Gender

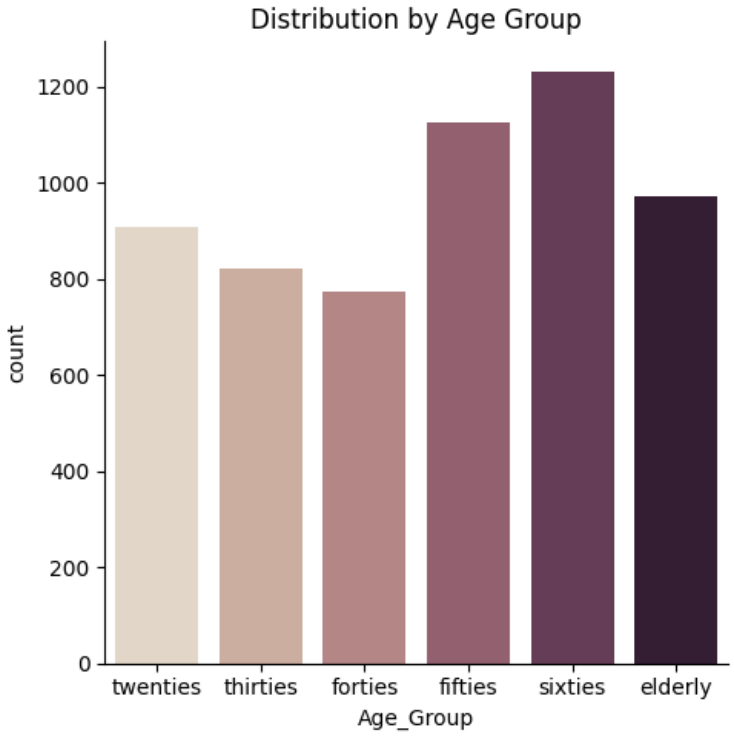

Fig 8.) Distribution of Age Group

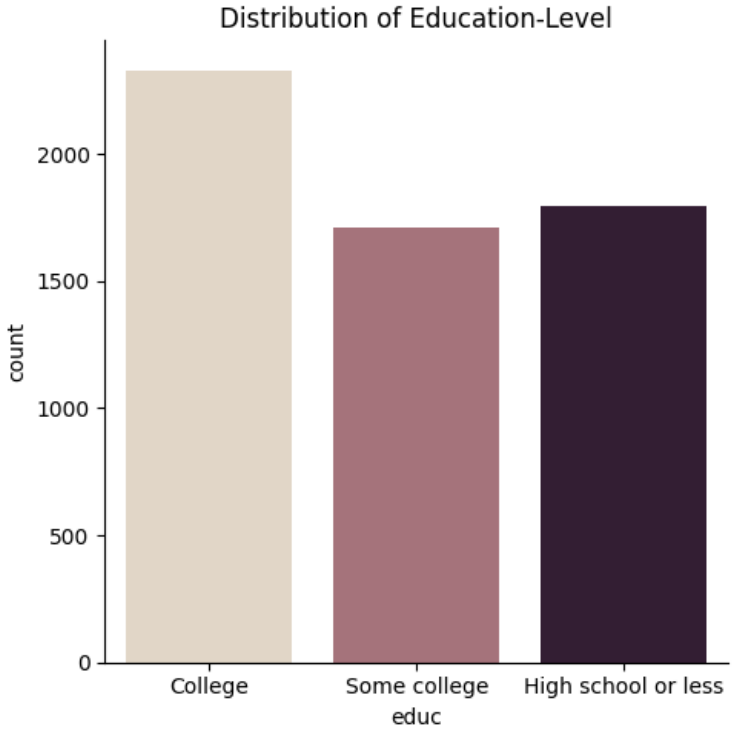

Fig 9.) Distribution of Education-Level

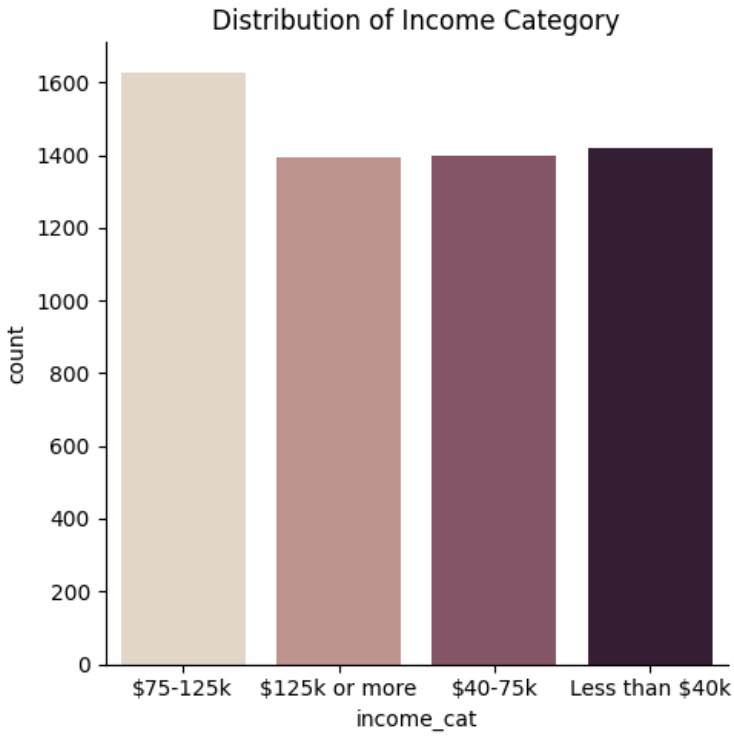

Fig 10.) Distribution of Income Category

Preprocessing

We began preprocessing by using the label encoder to transform the categorical features. This allows the machine learning models to better process the data passed in. We wanted our target variable to be binary, so we decided to drop observations where a person supported a third-party candidate or refused to provide an answer for our target variable. For the rest of the features, we replaced all null or refused values with the mean response of the individuals demographic group. Next we calculated feature importance and created a slimmed dataset where we dropped the features that were very highly correlated with party affiliation and therefore were a direct proxy for our target variable.

Implementation of Models

Since our dataset was comprised entirely of categorial features, we chose to implement the random forest **(RandomForestClassifier()**) and gradient boosting (**GradientBoostingClassifier()**) classifiers from the sci-kit learn package with the goal of determining if a boosting or bagging ensemble classifier would perform better with our dataset. Some of the survey questions were direct proxy for our target variable ("What is your view of Republicans?", "Do you trust the presidency?"), so we decided to create a subset of our main data where we dropped said questions and used this subset to train additional models. We dubbed the original models as "full models" since they were trained on a set containing all features and the additional models as "slim models" since they were trained on a slimmed down version of the dataset. The models were all trained and tested on a 75:25 split of their respective datasets and had the number of estimators set to 500 to increase their performance rate. Because our dataset was small, we were able to set the number of estimators to a high value without the model becoming too computationally expensive. The gradient boosting classifiers had an additional parameter of learning rate which was set to 0.05. Boosting classifiers are prone to overfitting by learning too quickly, so we set the learning rate to a low level in an attempt to prevent such from happening. The performance of the models was determined by the accuracy and F1 scores produced by the **accuracy_score()** and **classification_report()** functions.

Accuracy score: the measure of all the correctly identified cases. It is most used when all the classes are equally important. See the below figure for how to calculate the accuracy score.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{(\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative})}$$

F1-score: the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases than the Accuracy Metric. See the below figure for how to calculate the F1-score.

$$\text{F1-score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

Results

Since our dataset consists entirely of categorical features, we determined it would be best to train a random forest classifier to predict which candidate someone will vote for. During our experimentation, we trained and tested both a bagging random forest classifier and a gradient boosting classifier to see which would handle our data best. For each type of classifier, we trained and tested a model on all of the features in the dataset and dubbed them "full models." Then we created an additional model for each classifier type, but this time used the slimmed version of our data that does not contain the features that are a direct proxy to our target feature, which we dubbed "slim models."

Model 1

Our first model is our "full model" random forest classifier. This model is built with the standard random forest classifier found in the sklearn package and is trained on our full dataset. We set the number of estimators parameter to 500 since a high number of estimators often increases computation time, and we did not have to worry about computation time since our dataset is small. From our classification report, we see that this model performed very well, with an F1 score of .98 and an accuracy score of .97. Our ROC curve shows a perfect performance with an Area Under Curve (AUC) of 1.

```
Results Using All Features:

Classification Report:
              precision    recall  f1-score   support

           1       0.98      0.96      0.97       491
           2       0.97      0.98      0.98       680

    accuracy                           0.97      1171
   macro avg       0.97      0.97      0.97      1171
weighted avg       0.97      0.97      0.97      1171
```

Fig 11.) Classification report for Model 1

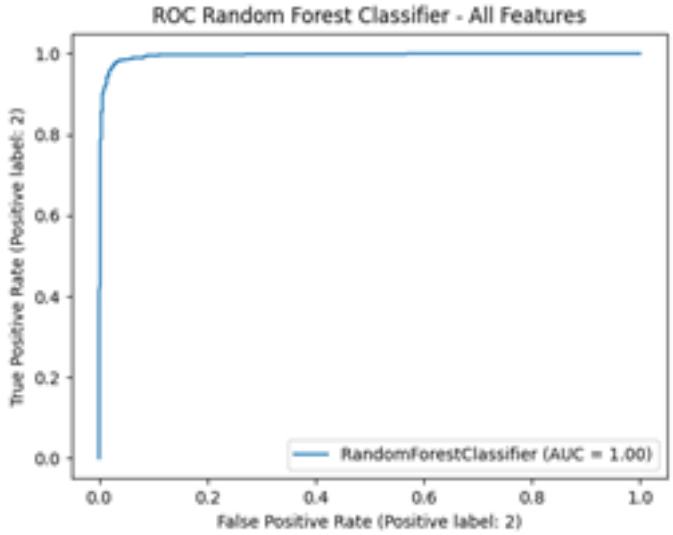

Fig 12.) ROC-AUC Graph for Model 1

Model 2

Our second model is our "slim model" random forest classifier. The only difference between this model and the previous "full model" random forest classifier is the dataset it was trained and tested on, this model being trained and tested on the slimmed dataset. From our classification report, we see that this model also performed well with an F1 score and accuracy score of .93 and an AUC of .98, which means it did not perform quite as well as Model 1.

```
Results For Slim Model:

Classification Report:
              precision    recall  f1-score   support

           1       0.93      0.90      0.91       449
           2       0.94      0.96      0.95       722

    accuracy                           0.93      1171
   macro avg       0.93      0.93      0.93      1171
weighted avg       0.93      0.93      0.93      1171
```

Fig 13.) Classification report for Model 2

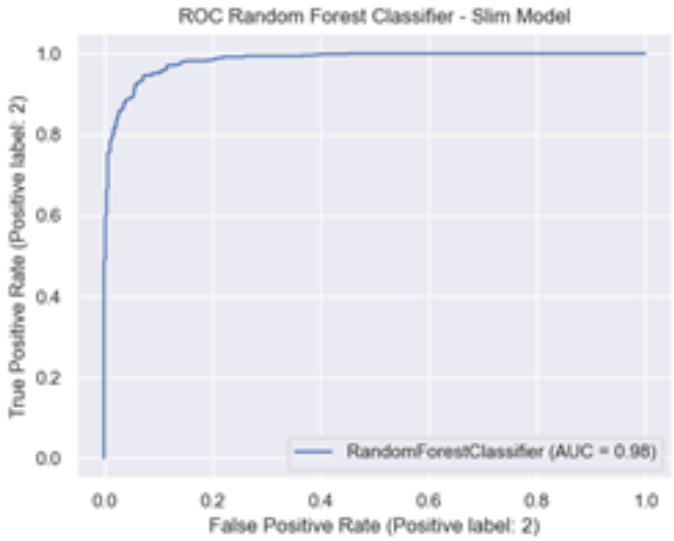

Fig 14.) ROC-AUC Graph for Model 2

Model 3

Our third model is our "full model" gradient boosting classifier. This model is built on the standard gradient boosting classifier found in the sklearn package and is trained and tested on our full dataset. Again, we set the number of estimators to a high value of 500 since we do not have to worry about computation time due to working with a small dataset. This model contains an additional parameter, learning rate. We set the learning rate to a low value of 0.05 since boosting classifiers are prone to overfitting by learning too quickly, and setting the learning rate to a low value helps to prevent such from happening. Like the previous models, this model performed well with an F1 score and an accuracy score of .97, and an AUC of .99.

```
Results Using Gradient Boosting & All Features:

Classification Report:
              precision    recall  f1-score   support

           1       0.97      0.97      0.97       491
           2       0.97      0.97      0.97       680

    accuracy                           0.97      1171
   macro avg       0.97      0.97      0.97      1171
weighted avg       0.97      0.97      0.97      1171
```

Fig 15.) Classification report for Model 3

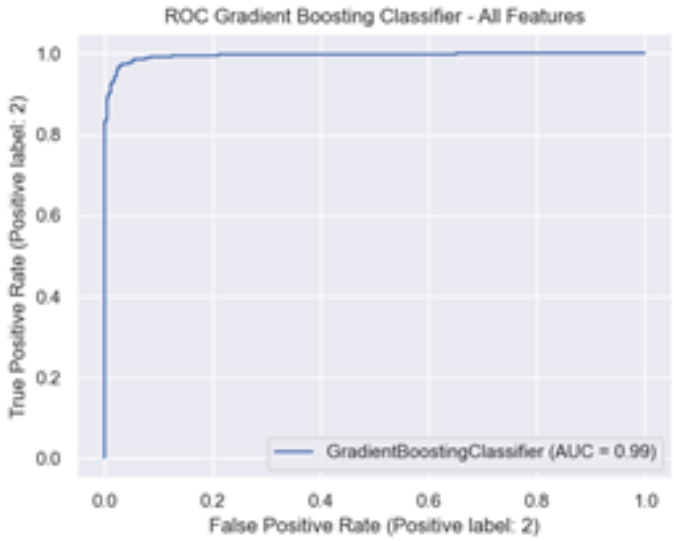

Fig 16.) ROC-AUC Graph for Model 3

Model 4

Our forth model is our "slim model" gradient boosting classifier. The only difference between this model and the previous "full model" gradient boosting classifier is the dataset it was trained and tested on, this one being trained and tested on the slimmed dataset. From our classification report, we see that this model performed badly with an F1 score of .5 and accuracy score of .51. This means the combination of the gradient boosting classifier and our slimmed dataset does not do well with predicting which candidate someone will vote for. We suspect this model suffered from overfitting on the training dataset and thus was unable to perform well with predicting the target feature values in the test dataset.

```
Results Using Gradient Boosting & Slim Dataframe:

Classification Report:
              precision    recall  f1-score   support

           1       0.41      0.37      0.39       491
           2       0.58      0.62      0.60       680

    accuracy                           0.51      1171
   macro avg       0.49      0.49      0.49      1171
weighted avg       0.51      0.51      0.51      1171
```

Fig 17.) Classification report for Model 4

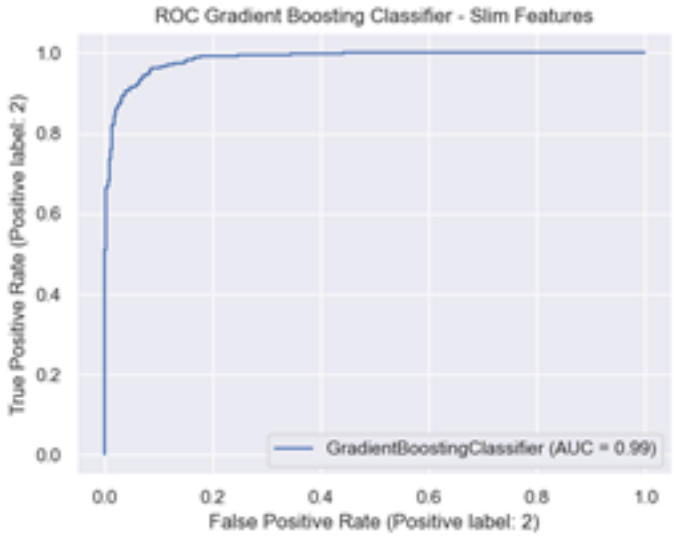

Fig 18.) ROC-AUC Graph for Model 4

Model Comparison

Below we see that the models trained and tested on the full dataset outperformed their counterpart models trained and tested on the slimmed dataset. Similarly, we see that the random forest classifier outperformed the gradient boosting classifier in both instances as well.

Model 1 Random Forest - Full Model	Model 2 Random Forest - Slim Model	Model 3 Gradient Boosting - Full Model	Model 4 Gradient Boosting - Slim Model
F1-score: 0.98 Accuracy score: 0.97	F1-score: 0.93 Accuracy score: 0.93	F1-score: 0.97 Accuracy score: 0.97	F1-score: 0.50 Accuracy score: 0.51

Fig 19.) Table comparing F1-scores and accuracy scores of each model.

Overall, our "full model" random forest classifier was the best performing model since it had the highest F1-score and accuracy score out of all the models.

Conclusion

Using FiveThirtyEight survey data, we decided, as our research question, to predict which presidential candidate a voter would vote for based on their survey answers. We chose this question over another question we initially considered (whether someone would turn out to vote) because the candidate selection variable was more balanced than the turnout variable.

We conducted exploratory data analysis with the intent to make sure the demographic groups were balanced and representative of the electorate. We found that most variables were well-balanced and generally aligned with the proportions of the electorate at large (based on election exit polls).

After preprocessing, we fit both random forest and gradient boosting models. We fit both models on the “full” and “slim” feature sets (removing features that were a direct proxy for the president), for a total of four models. We found that random forest did better than gradient boosting, and “full” features did better than “slimmed down” features. The order for our accuracy and F1 score metrics was (1) full random forest, (2) full gradient boosting, (3) slimmed random forest, and (4) slimmed gradient boosting.

The top three models all did very well, with accuracy and F1 scores above 0.9, whereas the slimmed gradient boosting did very poorly, with accuracy and F1 score around 0.5. This last model was surprisingly poor considering how self-contained our dataset was, and how many relevant features we had to work with.

Although we do not have direct evidence for this, we believe that the gradient boosting algorithm did worse overall because it was overfitting. With a large number of features to work with, it’s possible that the gradient boosting model was memorizing the training dataset to reduce error, meaning that it learned too quickly to be an effective predictor of unseen data.

If this work were to continue, we would look into the issue of overfitting and explore other classification models which could be effective at predicting. But this is not as critical because our random forest model worked well in this application.

Although our models had high accuracy, the impact of our work is limited. Knowing how someone votes based on their opinion of Democrats and Republicans, view of police, view of systemic racism, etc. is not a novel finding. Predicting elections involves deciding how individuals will vote, while having much less information than we do here, as well as trying to predict turnout rates.

Another relevant question we could have considered is voter turnout. We could have set the target variable as whether someone planned to vote or not (Question 21), then analyzed it by political affiliation, demographics, etc. This information could be applied to pollsters trying to forecast a future election, as they need to guess turnout rates among sub-populations and aggregate together to estimate overall election results. If this work were to continue, we could speak to pollsters and forecasters to understand these dynamics further.

Citations

1. https://scikit-learn.org
2. https://numpy.org/
3. https://pandas.pydata.org/
4. https://pypi.org/project/PyQt5/

5. https://medium.com/analytics-vidhya/evaluating-a-random-forest-model-9d165595ad56
6. https://www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained#:~:text=Like%20random%20forests%2C%20gradient%20boosting,one%20tree%20at%20a%20time
7. https://morningconsult.com/opinions/to-persuade-or-to-turn-out-voters-is-that-the-question/
8. https://www.bloomberg.com/graphics/2020-us-election-results/methodology
9. https://www.nytimes.com/interactive/2020/11/03/us/elections/exit-polls-president.html