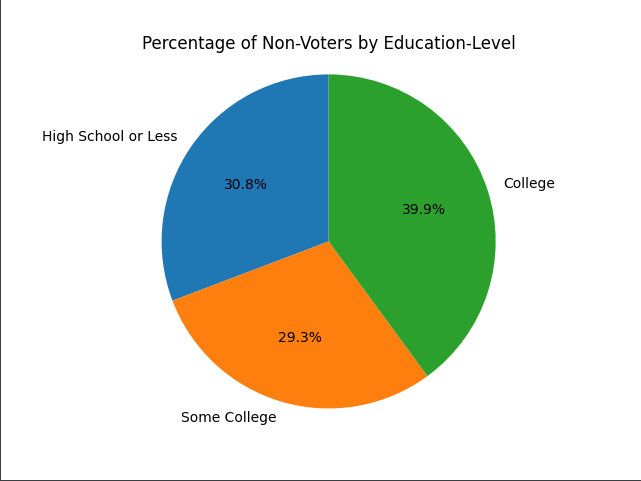
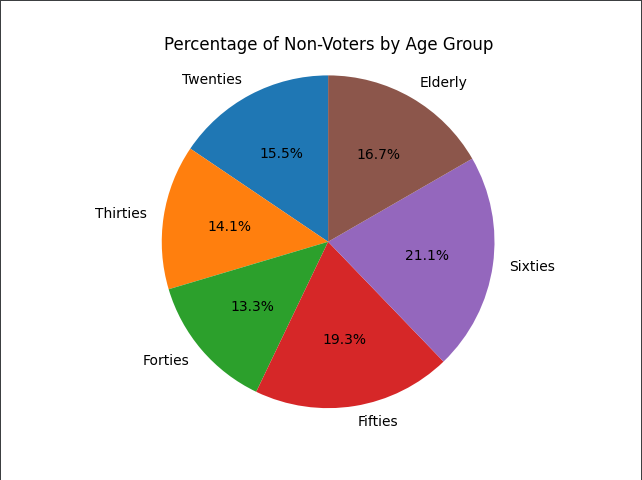
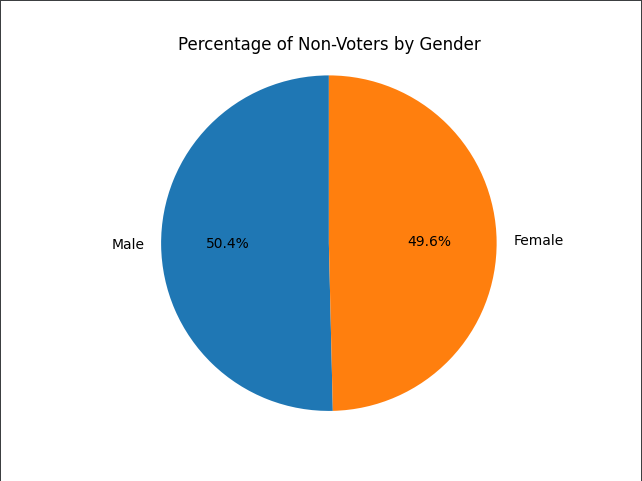
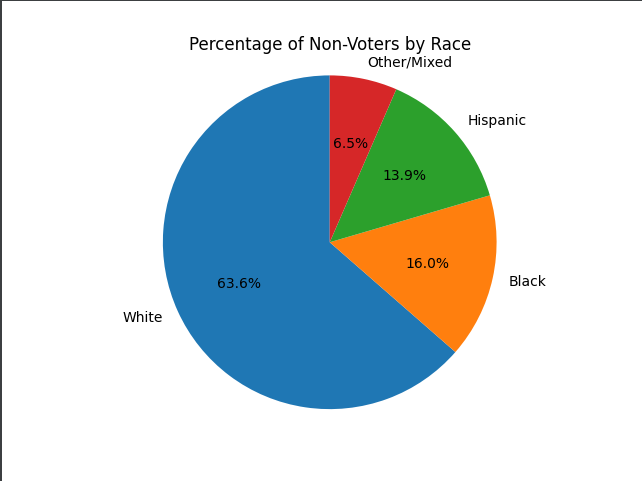
Divya Parmar

May 2, 2021

DATS 6103

**Final Project Individual Write-up**

1. Introduction
   1. Using machine learning classification techniques, we predict which candidate a voter plans to vote for based on self-reported survey data.
   2. We chose this dataset because the topic is highly relevant, there was an opportunity to apply EDA and preprocessing to a large number of features, and the dataset was mostly clean and well-documented courtesy of FiveThirtyEight.
   3. The shared work included data cleaning, exploratory data analysis, preprocessing, modeling, model iteration, GUI development, the write-up, and the demo.
2. Description of Individual Work
   1. I found the non-voters dataset, suggested it to my group, and went through the documentation to understand the columns. I re-named all 100+ columns to allow for better readability.
   2. I wrote code to read in the data, label encode the variables, replace invalid feature values with a mean, and run a function to fit a random forest for a pre-specified number of features.
   3. I contributed to the group write-up and PowerPoint by adding the introduction, dataset explanation, and conclusion. I spoke to those sections in our presentation recording.
3. Description of My Portion in Detail
   1. Please see the code on appended at the end of this report.
4. Results



|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

1. Summary
   1. Using FiveThirtyEight survey data, we decided to predict who voters would vote for president based on their survey answers.
   2. We conducted exploratory data analysis to better understand the group of voters and make sure the classes were balanced.
   3. We preprocessed our data - label encoding, dropping columns and observations.
   4. We fit both random forest and gradient boosting models, and we ran on “full” and “slim” feature sets.
   5. We saw extremely high accuracy and f1 scores, as evidenced in “Results” section.
   6. Random forest did better than gradient boosting, and “full” feature models did better than “slim” feature models.
2. Percent of Code
   1. 0%. No code was copied from the internet for this. I looked up function and syntax, but all EDA/preprocessing/modeling code was written on our own.
3. References
   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

**Appendix – My Individual Code**

*'''  
  
#---------------------------------------------------------------------------------------------------------  
FiveThirtyEight Non-Voters Dataset  
#---------------------------------------------------------------------------------------------------------  
  
'''  
  
##### Import packages and data #####*import pandas as pd  
import numpy as np  
import os  
from pathlib import Path  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.preprocessing import LabelEncoder  
from sklearn.metrics import plot\_roc\_curve  
from sklearn.metrics import accuracy\_score  
from sklearn.metrics import roc\_auc\_score  
  
dir = str(Path(os.getcwd()).parents[0])  
df = pd.read\_csv(dir+**'**\\**'**+**'nonvoters\_data.csv'**, sep=**','**, header=0)  
  
*# Change directory for graphing purposes*graphing\_dir = os.path.join(dir, **'Graphs'**)  
if not os.path.exists(graphing\_dir):  
 os.mkdir(graphing\_dir)  
os.chdir(graphing\_dir)  
  
*##### Exploratory data analysis ##### --------------------------------------------------------------------------*print(df.head)  
initial\_cols = df.columns  
print([x for x in df.columns])  
  
print(df[**'Q1'**].value\_counts())  
print(df[**'ppage'**].value\_counts())  
print(df[**'educ'**].value\_counts())  
print(df[**'race'**].value\_counts())  
print(df[**'gender'**].value\_counts())  
print(df[**'income\_cat'**].value\_counts())  
print(df[**'voter\_category'**].value\_counts())  
  
*#### Data Pre-Processing to prepare for modeling ##### --------------------------------------------------------------------------*

*# Rename columns to descriptive names*df.columns = [**'RespId'**, **'weight'**,   
 **'q1\_uscitizen'**,  
 **'q2\_important\_voting'**,**'q2\_important\_jury'**,**'q2\_important\_following'**,**'q2\_important\_displaying'**,**'q2\_important\_census'**,  
 **'q2\_important\_pledge'**,**'q2\_important\_military'**,**'q2\_important\_respect'**,**'q2\_important\_god'**,**'q2\_important\_protesting'**,  
 **'q3\_statement\_racism1'**,**'q3\_statement\_racism2'**,**'q3\_statement\_feminine'**,  
 **'q3\_statement\_msm'**,**'q3\_statement\_politiciansdontcare'**,**'q3\_statement\_besensitive'**,  
 **'q4\_impact\_officialsfed'**,**'q4\_impact\_officialsstate'**,**'q4\_impact\_officialslocal'**,  
 **'q4\_impact\_news'**,**'q4\_impact\_wallstreet'**,**'q4\_impact\_lawenforcement'**,  
 **'q5\_electionmatters'**,  
 **'q6\_officialsarelikeyou'**,  
 **'q7\_governmentdesign'**,  
 **'q8\_trust\_presidency'**,**'q8\_trust\_congress'**,**'q8\_trust\_supremecourt'**,**'q8\_trust\_cdc'**,**'q8\_trust\_electedofficials'**,  
 **'q8\_trust\_fbicia'**,**'q8\_trust\_newsmedia'**,**'q8\_trust\_police'**,**'q8\_trust\_postalservice'**,  
 **'q9\_politicalsystems\_democracy'**,**'q9\_politicalsystems\_experts'**,**'q9\_politicalsystems\_strongleader'**,**'q9\_politicalsystems\_army'**,  
 **'q10\_disability'**,**'q10\_chronic\_illness'**,**'q10\_unemployed'**,**'q10\_evicted'**,  
 **'q11\_lostjob'**,**'q11\_gotcovid'**,**'q11\_familycovid'**,  
 **'q11\_coviddeath'**,**'q11\_worriedmoney'**,**'q11\_quitjob'**,  
 **'q14\_view\_of\_republicans'**,  
 **'q15\_view\_of\_democrats'**,  
 **'q16\_how\_easy\_vote'**,  
 **'q17\_secure\_votingmachines'**,**'q17\_secure\_paperballotsinperson'**,**'q17\_secure\_paperballotsmail'**,**'q17\_secure\_electronicvotesonline'**,  
 **'q18\_votingsituations1'**,**'q18\_votingsituations2'**,**'q18\_votingsituations3'**,**'q18\_votingsituations4'**,**'q18\_votingsituations5'**,  
 **'q18\_votingsituations6'**,**'q18\_votingsituations7'**,**'q18\_votingsituations8'**,**'q18\_votingsituations9'**,**'q18\_votingsituations10'**,  
 **'q19\_get\_more\_voting1'**,**'q19\_get\_more\_voting2'**,**'q19\_get\_more\_voting3'**,**'q19\_get\_more\_voting4'**,**'q19\_get\_more\_voting5'**,  
 **'q19\_get\_more\_voting6'**,**'q19\_get\_more\_voting7'**,**'q19\_get\_more\_voting8'**,**'q19\_get\_more\_voting9'**,**'q19\_get\_more\_voting10'**,  
 **'q20\_currentlyregistered'**,  
 **'q21\_plan\_to\_vote'**,  
 **'q22\_whynotvoting\_2020'**,  
 **'q23\_which\_candidate\_supporting'**,  
 **'q24\_preferred\_voting\_method'**,  
 **'q25\_howcloselyfollowing\_election'**,  
 **'q26\_which\_voting\_category'**,  
 **'q27\_didyouvotein18'**,**'q27\_didyouvotein16'**,**'q27\_didyouvotein14'**,  
 **'q27\_didyouvotein12'**,**'q27\_didyouvotein10'**,**'q27\_didyouvotein08'**,  
 **'q28\_whydidyouvote\_past1'**,**'q28\_whydidyouvote\_past2'**,**'q28\_whydidyouvote\_past3'**,**'q28\_whydidyouvote\_past4'**,  
 **'q28\_whydidyouvote\_past5'**,**'q28\_whydidyouvote\_past6'**,**'q28\_whydidyouvote\_past7'**,**'q28\_whydidyouvote\_past8'**,  
 **'q29\_whydidyounotvote\_past1'**,**'q29\_whydidyounotvote\_past2'**,**'q29\_whydidyounotvote\_past3'**,**'q29\_whydidyounotvote\_past4'**,**'q29\_whydidyounotvote\_past5'**,  
 **'q29\_whydidyounotvote\_past6'**,**'q29\_whydidyounotvote\_past7'**,**'q29\_whydidyounotvote\_past8'**,**'q29\_whydidyounotvote\_past9'**,**'q29\_whydidyounotvote\_past10'**,  
 **'q30\_partyidentification'**,  
 **'q31\_republicantype'**,  
 **'q32\_democratictype'**,  
 **'q33\_closertowhichparty'**,  
 **'ppage'**, **'educ'**, **'race'**, **'gender'**, **'income\_cat'**, **'voter\_category'** ]  
  
*# Drop irrelevant fields (US Citizen, responder ID, observation weight)  
# Drop questions that were not asked to all participants (i.e. "why did you vote" to non-voters, "Republican type" for Democrats)*df.drop([**'q1\_uscitizen'**,**'q22\_whynotvoting\_2020'**,  
 **'q28\_whydidyouvote\_past1'**,**'q28\_whydidyouvote\_past2'**,**'q28\_whydidyouvote\_past3'**,**'q28\_whydidyouvote\_past4'**,  
 **'q28\_whydidyouvote\_past5'**,**'q28\_whydidyouvote\_past6'**,**'q28\_whydidyouvote\_past7'**,**'q28\_whydidyouvote\_past8'**,  
 **'q29\_whydidyounotvote\_past1'**,**'q29\_whydidyounotvote\_past2'**,**'q29\_whydidyounotvote\_past3'**,**'q29\_whydidyounotvote\_past4'**,**'q29\_whydidyounotvote\_past5'**,  
 **'q29\_whydidyounotvote\_past6'**,**'q29\_whydidyounotvote\_past7'**,**'q29\_whydidyounotvote\_past8'**,**'q29\_whydidyounotvote\_past9'**,**'q29\_whydidyounotvote\_past10'**,  
 **'q31\_republicantype'**,  
 **'q32\_democratictype'**,  
 **'q33\_closertowhichparty'**,  
 **'q21\_plan\_to\_vote'**,  
 **'q22\_whynotvoting\_2020'**,  
 **'RespId'**,  
 **'weight'** ], axis=1, inplace=True)  
  
  
  
*# Replace "refused" answers (value of -1) with the demographic average for each group  
# Step 1 - Replace -1 in certain columns with NaN  
# Step 2 - Replace NaN with demographic average using groupby  
  
# Create list of columns that need answer cleaning  
# This isn't all the columns (some columns only had values of -1 and 1, which is fine)*replace\_neg\_one = [  
 **'q2\_important\_voting'**,**'q2\_important\_jury'**,**'q2\_important\_following'**,**'q2\_important\_displaying'**,**'q2\_important\_census'**,  
 **'q2\_important\_pledge'**,**'q2\_important\_military'**,**'q2\_important\_respect'**,**'q2\_important\_god'**,**'q2\_important\_protesting'**,  
 **'q3\_statement\_racism1'**,**'q3\_statement\_racism2'**,**'q3\_statement\_feminine'**,  
 **'q3\_statement\_msm'**,**'q3\_statement\_politiciansdontcare'**,**'q3\_statement\_besensitive'**,  
 **'q4\_impact\_officialsfed'**,**'q4\_impact\_officialsstate'**,**'q4\_impact\_officialslocal'**,  
 **'q4\_impact\_news'**,**'q4\_impact\_wallstreet'**,**'q4\_impact\_lawenforcement'**,  
 **'q5\_electionmatters'**,  
 **'q6\_officialsarelikeyou'**,  
 **'q7\_governmentdesign'**,  
 **'q8\_trust\_presidency'**,**'q8\_trust\_congress'**,**'q8\_trust\_supremecourt'**,**'q8\_trust\_cdc'**,**'q8\_trust\_electedofficials'**,  
 **'q8\_trust\_fbicia'**,**'q8\_trust\_newsmedia'**,**'q8\_trust\_police'**,**'q8\_trust\_postalservice'**,  
 **'q9\_politicalsystems\_democracy'**,**'q9\_politicalsystems\_experts'**,**'q9\_politicalsystems\_strongleader'**,**'q9\_politicalsystems\_army'**,  
 **'q10\_disability'**,**'q10\_chronic\_illness'**,**'q10\_unemployed'**,**'q10\_evicted'**,  
 **'q11\_lostjob'**,**'q11\_gotcovid'**,**'q11\_familycovid'**,  
 **'q11\_coviddeath'**,**'q11\_worriedmoney'**,**'q11\_quitjob'**,  
 **'q14\_view\_of\_republicans'**,  
 **'q15\_view\_of\_democrats'**,  
 **'q16\_how\_easy\_vote'**,  
 **'q17\_secure\_votingmachines'**,**'q17\_secure\_paperballotsinperson'**,**'q17\_secure\_paperballotsmail'**,**'q17\_secure\_electronicvotesonline'**,  
 **'q18\_votingsituations1'**,**'q18\_votingsituations2'**,**'q18\_votingsituations3'**,**'q18\_votingsituations4'**,**'q18\_votingsituations5'**,  
 **'q18\_votingsituations6'**,**'q18\_votingsituations7'**,**'q18\_votingsituations8'**,**'q18\_votingsituations9'**,**'q18\_votingsituations10'**,  
 **'q20\_currentlyregistered'**,  
 **'q24\_preferred\_voting\_method'**,  
 **'q25\_howcloselyfollowing\_election'**,  
 **'q26\_which\_voting\_category'**,  
 **'q27\_didyouvotein18'**,**'q27\_didyouvotein16'**,**'q27\_didyouvotein14'**,  
 **'q27\_didyouvotein12'**,**'q27\_didyouvotein10'**,**'q27\_didyouvotein08'**,  
 **'q30\_partyidentification'** ]  
  
*# Step 1 - Replace -1 or -1.0 values with NaN  
# Values might be stored as int or float, so account for both*df[replace\_neg\_one] = df[replace\_neg\_one].replace(-1, np.nan)  
df[replace\_neg\_one] = df[replace\_neg\_one].replace(-1.0, np.nan)  
  
*# Step 2 - Replace NaN with demographic mean*for x in replace\_neg\_one:  
 df[x] = df[x].fillna(df.groupby(by=[**'educ'**, **'race'**, **'gender'**, **'income\_cat'**])[x].transform(**'mean'**))  
  
*# Transform non-numeric categorical variables into numeric for model processing*le = LabelEncoder()  
df[**'educ'**] = le.fit\_transform(df[**'educ'**])  
df[**'race'**] = le.fit\_transform(df[**'race'**])  
df[**'gender'**] = le.fit\_transform(df[**'gender'**])  
df[**'income\_cat'**] = le.fit\_transform(df[**'income\_cat'**])  
df[**'voter\_category'**] = le.fit\_transform(df[**'voter\_category'**])  
  
*# Identify values of the target variable*print(df[**'q23\_which\_candidate\_supporting'**].value\_counts())  
  
*# For q23\_which\_candidate\_supporting, value of 1 is Trump and value of 2 is Biden  
# Drop unsure (value of 3) and refused to answer (value of -1) to set up our two-way classification*df\_mod = df[(df[**'q23\_which\_candidate\_supporting'**] == 1) | (df[**'q23\_which\_candidate\_supporting'**] == 2)]  
  
  
*##### Random Forest Model - Full Model with All Features ##### --------------------------------------------------------------------------  
  
# Create features dataframe that doesn't contain the target variable*X = df\_mod.drop([**'q23\_which\_candidate\_supporting'**], axis=1)  
*# Create target variable*y = df\_mod[**'q23\_which\_candidate\_supporting'**]  
  
*# Split data into train and test*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=419)  
  
*# Fit model on train data*clf = RandomForestClassifier(n\_estimators=500)  
clf.fit(X\_train, y\_train)  
*# Predict on test data  
# Categorical predictions are for accuracy and probabilities are for ROC score*y\_pred = clf.predict(X\_test)  
y\_pred\_probs = clf.predict\_proba(X\_test)  
  
*# Get accuracy and ROC values*roc\_auc\_full = roc\_auc\_score(y\_test, y\_pred\_probs[:, 1])  
accuracy\_full = accuracy\_score(y\_test, y\_pred)  
print(**f'The full model AUC is** {roc\_auc\_full} **and the accuracy is** {accuracy\_full}**.'**)  
  
*# Plot ROC curve  
# More area under the curve indicates the model has skill in finding true positives and avoiding false positives*plot\_roc\_curve(clf, X\_test, y\_test)  
plt.savefig(**'roc\_curve\_full\_model.png'**, dpi=300, bbox\_inches=**'tight'**)  
plt.show()  
  
*# Get feature importances and plot them*importances = clf.feature\_importances\_  
feat\_imp = pd.Series(importances, X\_train.columns)  
feat\_imp.sort\_values(ascending=False, inplace=True)  
feat\_imp.plot(x=**'Features'**, y=**'Importance'**, kind=**'bar'**, figsize=(16, 9), rot=90, fontsize=15)  
plt.tight\_layout()  
plt.savefig(**'feature\_importances\_full\_model.png'**, dpi=300, bbox\_inches=**'tight'**)  
plt.show()  
  
*##### Feature Importance Analysis ##### --------------------------------------------------------------------------  
  
# Get top 20 features*top20 = feat\_imp.index[0:20]  
  
*# Plot correlation matrix of top 20 features against the target variable (for all records)*df20 = df[top20]  
df20[**'y'**] = df[y.name]  
  
plt.figure(figsize=(16,16))  
plt.tight\_layout()  
sns.set(font\_scale=1)  
corr\_heatmap = sns.heatmap(df20.corr(), vmin=-1, vmax=1, annot=True, cbar=False)  
corr\_heatmap.set\_title(**'Correlation of Top 20 Features and Target Variable'**)  
corr\_heatmap.set\_xticklabels(labels=df20.columns, rotation=30, fontsize=9, ha=**'right'**)  
plt.savefig(**'heatmap\_top20\_features.png'**, dpi=300, bbox\_inches=**'tight'**)  
plt.show()  
  
*##### Random Forest Model - Slim model without the top features ##### --------------------------------------------------------------------------  
  
# Run another model without top features such as party identification and trust of presidency  
# These variables are very highly correlated with view of Trump, GOP, Dems, etc.*X\_slim = df\_mod.drop([**'q23\_which\_candidate\_supporting'**, **'q30\_partyidentification'**,**'q8\_trust\_presidency'**,  
 **'q14\_view\_of\_republicans'**, **'q15\_view\_of\_democrats'**], axis=1)  
  
*# Train test split for this new model*X\_slim\_train, X\_slim\_test, y\_slim\_train, y\_slim\_test = train\_test\_split(X\_slim, y, test\_size=0.25, random\_state=125)  
  
*# Fit model*clf2 = RandomForestClassifier(n\_estimators=500)  
clf2.fit(X\_slim\_train, y\_slim\_train)  
*# Predict*y\_slim\_pred = clf2.predict(X\_slim\_test)  
y\_slim\_pred\_probs = clf2.predict\_proba(X\_slim\_test)  
  
*# Get accuracy and ROC values*roc\_auc\_slim = roc\_auc\_score(y\_slim\_test, y\_slim\_pred\_probs[:, 1])  
accuracy\_slim = accuracy\_score(y\_slim\_test, y\_slim\_pred)  
print(**f'The slim model AUC is** {roc\_auc\_slim} **and the accuracy is** {accuracy\_slim}**.'**)  
  
*# Plot ROC curve*plot\_roc\_curve(clf2, X\_slim\_test, y\_slim\_test)  
plt.savefig(**'roc\_curve\_slim\_model.png'**, dpi=300, bbox\_inches=**'tight'**)  
plt.show()  
  
*# Get feature importances*importances2 = clf2.feature\_importances\_  
feat\_imp2 = pd.Series(importances2, X\_slim\_train.columns)  
feat\_imp2.sort\_values(ascending=False, inplace=True)  
feat\_imp2.plot(x=**'Features'**, y=**'Importance'**, kind=**'bar'**, figsize=(16, 9), rot=90, fontsize=15)  
plt.tight\_layout()  
plt.savefig(**'feature\_importances\_slim\_model.png'**, dpi=300, bbox\_inches=**'tight'**)  
plt.show()  
  
  
*##### IGNORE CODE BELOW ##### --------------------------------------------------------------------------***'''  
  
print(df['q21\_plan\_to\_vote'].value\_counts())  
print(df['q30\_partyidentification'].value\_counts())  
  
  
df.drop(['q1\_uscitizen','q20\_currentlyregistered','q22\_whynotvoting\_2020',  
 'q23\_which\_candidate\_supporting','q26\_which\_voting\_category',  
 'q27\_didyouvotein18','q27\_didyouvotein16','q27\_didyouvotein14',  
 'q27\_didyouvotein12','q27\_didyouvotein10','q27\_didyouvotein08',  
 'q28\_whydidyouvote\_past1','q28\_whydidyouvote\_past2','q28\_whydidyouvote\_past3','q28\_whydidyouvote\_past4',  
 'q28\_whydidyouvote\_past5','q28\_whydidyouvote\_past6','q28\_whydidyouvote\_past7','q28\_whydidyouvote\_past8',  
 'q29\_whydidyounotvote\_past1','q29\_whydidyounotvote\_past2','q29\_whydidyounotvote\_past3','q29\_whydidyounotvote\_past4','q29\_whydidyounotvote\_past5',  
 'q29\_whydidyounotvote\_past6','q29\_whydidyounotvote\_past7','q29\_whydidyounotvote\_past8','q29\_whydidyounotvote\_past9','q29\_whydidyounotvote\_past10',  
 'q31\_republicantype',  
 'q32\_democratictype',  
 'q33\_closertowhichparty',  
 'voter\_category',  
 'RespId',  
 'weight'  
 ], axis=1, inplace=True)  
  
  
# Fit the random forest  
# Get feature  
rf = RandomForestClassifier()  
cv = cross\_validate(rf, X, y, cv=10)  
print(cv)  
  
# Get feature importances  
rf2 = RandomForestClassifier()  
rf2.fit(X=X, y=y, sample\_weight=None)  
feat\_imp = list(zip(rf2.feature\_importances\_, X.columns))  
print(sorted(feat\_imp, reverse=True))  
  
# Try feature selection with SelectFromModel  
select = SelectFromModel(RandomForestClassifier(n\_estimators=20))  
select.fit(X\_train, y\_train)  
# Select.get\_support returns True or False for each feature  
# Take only the true values for features and look at our accuracy  
print(select.get\_support())  
feature\_inclusion\_array = select.get\_support()  
print(X\_train.columns[feature\_inclusion\_array])  
inclusion\_cols = X\_train.columns[feature\_inclusion\_array]  
X\_train\_skinny = X\_train[inclusion\_cols]  
  
rf3 = RandomForestClassifier()  
cv\_skinny = cross\_validate(rf3, X\_train\_skinny, y\_train, cv=10)  
print(cv\_skinny)  
  
plt.tight\_layout()  
#plt.subplots\_adjust(top = 3, bottom = 2, right = 3, left = 2,  
# hspace = 0.1, wspace = 0.1)  
#corr\_heatmap.xaxis.labelpad = 0  
#corr\_heatmap.title.labelpad = 0  
  
'''**

*# Write a function that takes  
# X, y, percent test in train-test, number of features in model  
# Return model accuracy metrics, confusion matrix, feature importance, roc curve*def rf\_model\_visualize(df: pd.DataFrame, num\_features: int, test\_percent: float):  
  
 *'''  
  
 :param df: Dataframe of all observations (train and test) to build model.  
 :param num\_features: The number of features to include in the model (all variables except target).  
 :param test\_percent: Percent of data to use in test (i.e. 0.3 means 70% train, 30% test).  
  
 :return: accuracy\_score\_value: The accuracy of the RF model with the parameters passed above. (TP + FN)/ (TP + FP + TN + FN)  
 :return: conf: Confusion matrix of RF model. This classifies the true positives, false positives, true negatives, false negatives.  
 :return: auc\_graph: Graph of AUC (area under curve) of the RF model.  
 :return: auc\_score\_value: AUC (area under curve) score. Random guessing is 0.5, and closer to 1 means smarter model.  
 :return: feature\_importance\_plot: Importance of the num\_features chosen. Higher importance means it greater reduces entropy in classification.  
  
 '''  
  
 # Create empty variables to return if the user passes in invalid parameters* auc\_null = np.nan  
 conf\_null = np.zeros((2,2), dtype=int)  
 auc\_\_null\_graph = plt.plot()  
 auc\_score\_null = np.nan  
 feature\_importance\_plot\_null = plt.plot()  
  
 *# There are only 92 features available* if (num\_features < 1) or (num\_features > 92):  
 return auc\_null, conf\_null, auc\_\_null\_graph, auc\_score\_null, feature\_importance\_plot\_null  
 *# We cannot test on 0 or 100 percent of our data* if (test\_percent < 0.01) or (test\_percent > 0.99):  
 return auc\_null, conf\_null, auc\_\_null\_graph, auc\_score\_null, feature\_importance\_plot\_null  
  
 *# Go through modeling steps in this function  
 # Start with getting X, y, and train-test split* Xpre = df.drop(columns=[**'q23\_which\_candidate\_supporting'**], axis=1)  
 ypre = df[**'q23\_which\_candidate\_supporting'**]  
  
 X\_pre\_train, X\_pre\_test, y\_pre\_train, y\_pre\_test = train\_test\_split(Xpre, ypre, test\_size=test\_percent, random\_state=1918)  
  
 *# Fit the model* rf\_pre = RandomForestClassifier()  
 rf\_pre.fit(X\_pre\_train, y\_pre\_train)  
  
 *# Get the most important features* importances = rf\_pre.feature\_importances\_  
 feat\_imp = pd.Series(importances, X\_pre\_train.columns)  
 feat\_imp.sort\_values(ascending=False, inplace=True)  
 features\_to\_keep = feat\_imp.index[0:num\_features]  
  
 *# Re-fit with the slimmed down list* X = Xpre[features\_to\_keep]  
 y = ypre  
  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=test\_percent, random\_state=1918)  
  
 *# Fit the model* rf = RandomForestClassifier()  
 rf.fit(X\_train, y\_train)  
 y\_pred = rf.predict(X\_test)  
 y\_pred\_proba = rf.predict\_proba(X\_test)  
  
 *# Output: accuracy metrics* accuracy\_score\_value = accuracy\_score(y\_test, y\_pred)  
  
 *# Output: confusion matrix* conf = confusion\_matrix(y\_test, y\_pred)  
  
 *# Output: ROC Curve* auc\_graph = plot\_roc\_curve(rf, X\_test, y\_test)  
  
 *# Output: ROC score* auc\_score\_value = roc\_auc\_score(y\_test, y\_pred\_proba[:, 1])  
  
 *# Output Feature importance* imp\_final = rf.feature\_importances\_  
 feat\_imp\_final = pd.Series(imp\_final, X\_train.columns)  
 feat\_imp\_final.sort\_values(ascending=False, inplace=True)  
 feature\_importance\_plot = plt.bar(x=feat\_imp\_final.index, height=feat\_imp\_final.values)  
  
 return accuracy\_score\_value, conf, auc\_graph, auc\_score\_value, feature\_importance\_plot  
  
  
accuracy\_score\_value, conf, auc\_graph, auc\_score\_value, feature\_importance\_plot = rf\_model\_visualize(df=df,  
 num\_features=25,  
 test\_percent=0.25)