FiveThirtyEight Election Prediction

Yasaman Ebrahimi

Contents

1 Data Overview

The FiveThirtyEight Election dataset is a census dataset that contains information from 811 Democrat and 774 Republican candidates from the ballot 2018 election. These candidates were in the primary and/or general elections for the Senate, House of Representatives, and Governor.

A primary election is where each political party selects candidates to run for office in the general election. Voters decide on which candidates from the primary advance to the general election. In the general election, candidates are elected to federal office.

This data was collected from Ballotpedia. Ballotpedia is a charitable nonprofit organization that serves as an encyclopedia for American politics. The goal of Ballotpedia is for their data to be "neutral, accurate, and verifiable" about elections and candidates.

Therefore, for the purpose of this project, the data is assumed to be unbiased. This means that we assume that no groups were systematically excluded from this data set, and we are not assuming any selection bias, measurement error, or convenience sampling to have occurred. Additionally, the participants are aware of the collection and use of their data as they are politicians who are running for office.

We chose two supplementary data sets for Research Question 1 that gave information about vote share between candidates in the Republican and Democrat parties. This is the US House and US Senate. These additional data sets can connect the features of each candidate during the primary election year to the vote share column, which helps us use multiple statistical analyses.

We chose a second supplementary data set for Research Question 2 that gave information about the urbanization of each voting district. This is the FiveThirtyEight District Urbanization Index 2022 dataset. This additional data was found to be helpful because it could be used to draw conclusions about the relationship between urbanization and a candidate's political party.

1.1 Data Setup

```
# Dependencies
! pip install pymc3
import pandas as pd
import numpy as np
from os.path import join
from matplotlib import pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from IPython.display import display
import re
import pymc3 as pm
from pymc3 import glm
import statsmodels.formula.api as smf
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import arviz as az
from itertools import combinations

# Loading Data
url_dem = 'https://github.com/fivethirtyeight/data/blob
/master/primary-candidates-2018/dem_candidates.csv?raw=true'
url_rep = 'https://github.com/fivethirtyeight/data/blob
/master/primary-candidates-2018/rep_candidates.csv?raw=true'

# Supplemental Data for research question 2
url_urban = 'https://github.com/fivethirtyeight/data/blob/master
/district-urbanization-index-2022/urbanization-index-2022.csv?raw=true'

cols = ['Candidate', 'State', 'District', 'Office Type', 'Race Type',
'Race Primary Election Date', 'Primary Status', 'Primary Runoff Status',
'General Status', 'Primary %', 'Won Primary']
dem = pd.read_csv(url_dem, encoding='cp1252')[cols]
rep = pd.read_csv(url_rep, encoding='cp1252')[cols]
dem['Party'] = np.array(['Democrat' for _ in range(dem.shape[0])])
rep['Party'] = np.array(['Republican' for _ in range(rep.shape[0])])

election_data = pd.concat([dem, rep],ignore_index=True)
urban = pd.read_csv(url_urban, encoding='cp1252')
urban['cd'] = urban['cd'].astype(str)
election_data.head(3)
```

1.2 Main Data Frame

	Candidate	State	District	Office Type	Race Type	Race Primary Election Date	Primary Status	Primary Runoff Status	General Status	Primary %	Won Primary	Party
0	Anthony White (Alabama)	AL	Governor of Alabama	Governor	Regular	6/5/18	Lost	None	None	3.42	No	Democrat
1	Christopher Countryman	AL	Governor of Alabama	Governor	Regular	6/5/18	Lost	None	None	1.74	No	Democrat
2	Doug "New Blue" Smith	AL	Governor of Alabama	Governor	Regular	6/5/18	Lost	None	None	3.27	No	Democrat

1.3 Main Data Granularity

- Candidate Name
- State
- District
  - Position Name
  - State
- Office Type
  - Governor
  - House
  - Senate
- Race Type
  - Regular
  - Special (when someone dies or gets removed from office)
- Race Primary Election Date
  - The date on which the primary was held
- Primary Status
  - Lost
  - Advanced
- Primary Runoff Status
  - "None" if there was no runoff election
  - "On the Ballot", "Lost", or "Advanced" if there was a runoff election
- General Status
  - "On the Ballot" if the candidate won the primary election or runoff
  - "None" if the candidate did not win the primary election or runoff
- Primary %
  - The percentage of the vote received in the primary
- Won Primary
  - "Yes" if the candidate won the primary
  - "No" if the candidate lost the primary
- Party

1.4 Supplementary Data Frame 1

```
FORECAST_FOLDER = 'forecast_data'

HOUSE_FORECAST_FILE = 'house_district_forecast.csv'

SENATE_FORECAST_FILE = 'senate_seat_forecast.csv'
election_df = election_df[election_df.year == 2018]
election_df.columns

candidates_df = candidates_df.rename({'Candidate':'candidate',
'State':'state'}, axis=1)

forecast_df = forecast_df.merge(candidates_df,
on=['candidate','state'])

house_forecast_df = forecast_df[forecast_df['office'] == 'house']
house_forecast_df['district'] = house_forecast_df['district'].astype(int)
house_forecast_df['party'] = house_forecast_df['party_x']
house_election_df = election_df[election_df.office=='US-HOUSE']

house_election_df['party'] =
house_election_df.party.map({'REPUBLICAN':'R', 'DEMOCRAT':'D'})

house_election_df['state'] = house_election_df['state_po']

house_forecast_df =
house_forecast_df.merge(house_election_df,
how='left', on=['state','district','party'])

house_forecast_df['actual_voteshare'] =
house_forecast_df['candidatevotes'] / house_forecast_df['totalvotes'] * 100

house_forecast_df =
house_forecast_df[~house_forecast_df['actual_voteshare'].isna()]

house_forecast_df['voteshare_miss'] =
house_forecast_df['voteshare'] - house_forecast_df['actual_voteshare']

house_lite_forecast_df =
house_forecast_df[house_forecast_df.model=='lite']

house_lite_forecast_df['color'] =
house_lite_forecast_df.party.map({'R':'red','D':'blue'})
```

	forecastdate	state	district	special_x	candidate_x	party_x	incumbent	model	win_probability	voteshare	...
2	2018-11-06	AL	1	NaN	Robert Kennedy Jr.	D	False	lite	0.0041	34.68	...
5	2018-11-06	AL	2	NaN	Tabitha Isner	D	False	lite	0.0030	33.51	...
8	2018-11-06	AL	3	NaN	Mallory Hagan	D	False	lite	0.0023	33.22	...
11	2018-11-06	AL	4	NaN	Lee Auman	D	False	lite	0.0000	19.99	...
14	2018-11-06	AL	5	NaN	Peter Joffrion	D	False	lite	0.0022	32.77	...

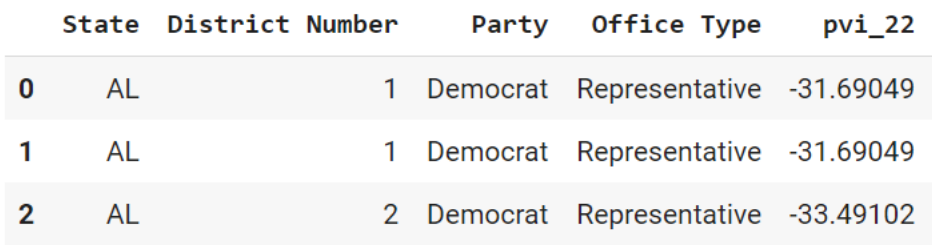
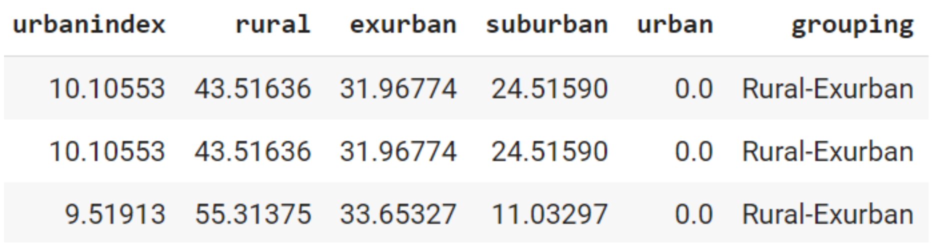
	forecastdate	state	district	special_x	candidate_x	party_x	incumbent	model	win_probability	voteshare	...
29	2018-11-06	AZ	1	NaN	Wendy Rogers	R	False	lite	0.2843	47.88	...
32	2018-11-06	AZ	2	NaN	Lea Marquez Peterson	R	False	lite	0.1291	45.62	...
35	2018-11-06	AZ	9	NaN	Steve Ferrara	R	False	lite	0.0383	40.33	...
41	2018-11-06	CA	3	NaN	Charlie Schaupp	R	False	lite	0.0039	34.10	...
47	2018-11-06	CA	7	NaN	Andrew Grant	R	False	lite	0.0319	41.87	...

1.5 Supplementary Data Granularity 1

- Year - The year of the election; dtype = int
- Candidate - The name of the candidate; dtype = string
- State - State which the candidate represents; dtype = string
- District - District which the candidate represents; dtype = string
- Party - Party which the candidate represents; dtype = string
- Candidate Votes - The number of votes the candidates received in the election; dtype = int
- Total Votes - The number of total votes in the election; dtype = int
- Vote Share - Share of votes the candidate has; dtype = float
- Vote Share Miss - Share of votes the candidate has missed; dtype = float

1.6 Supplementary Data Frame 2

```
district_num = election_data['District'].str.extract(r'([0-9]+)')
election_data['District Number'] = district_num
election_data = election_data.iloc[district_num.dropna().index]
election_urban = election_data.merge(
    urban,
    left_on=['State', 'District Number'],
    right_on=['state', 'cd']
)
numeric_cols = [
'urbanindex', 'rural', 'exurban', 'suburban', 'urban', 'grouping'
]
categorical_cols = [
'State', 'District Number', 'Party', 'Office Type'
]
election_urban = election_urban[categorical_cols + numeric_cols]
dem_urban = election_urban[election_urban['Party'] == 'Democrat']
rep_urban = election_urban[election_urban['Party'] == 'Republican']
election_urban.head(3)
```

	State	District Number	Party	Office Type	pvi_22	urbanindex	rural	exurban	suburban	urban	grouping
0	AL	1	Democrat	Representative	-31.69049	10.10553	43.51636	31.96774	24.51590	0.0	Rural-Exurban
1	AL	1	Democrat	Representative	-31.69049	10.10553	43.51636	31.96774	24.51590	0.0	Rural-Exurban
2	AL	2	Democrat	Representative	-33.49102	9.51913	55.31375	33.65327	11.03297	0.0	Rural-Exurban

1.7 Supplementary Data Granularity 2

- pvi_22 - Partisan lean for each congressional district ahead of the 2022 midterm elections
  - Positive values = Democratic lean
  - negative values = Republican lean
- Urban - The percentage of the district population that live in census tracts where 250,000 or more people within a five-mile radius.
- Suburban - The percentage of the district population that live in census tracts where between 100,000 and 249,999 people live within a five-mile radius.
- Exurban - The percentage of the district population that live in census tracts where between 25,000 and 99,999 people live within a five-mile radius.
- Rural - The percentage of the district population that live in census tracts where fewer than 25,000 people live within a five-mile radius.
- Urban Index - FiveThirtyEight's urbanization index for each congressional district. The urbanization index is the natural logarithm of the average number of people living within a five-mile radius of every census tract in a given district, based on a weighted average of the population of each census tract in the 2020 census.
- Grouping - categorization of congressional districts based on their urbanization index

2 EDA Question 1

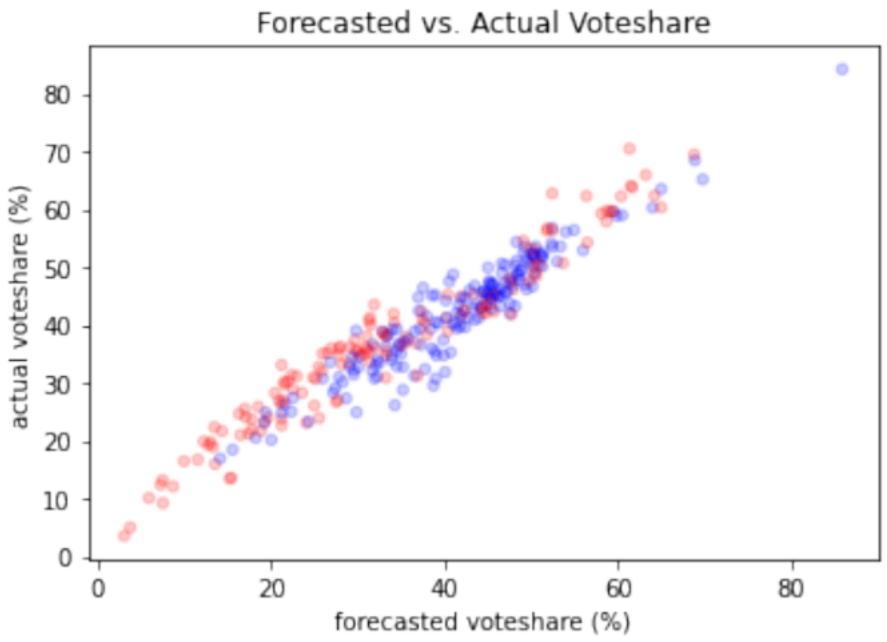
2.1 Data Processing: Voteshare Miss

In this section, we will be creating a new column called ”voteshare miss” for each candidate to represent by what percentage the forecasted model was off actual data.
The formula for voteshare miss:

[voteshare miss] = (1/[total votes])∗([actual vote share]−[forecasted vote share])

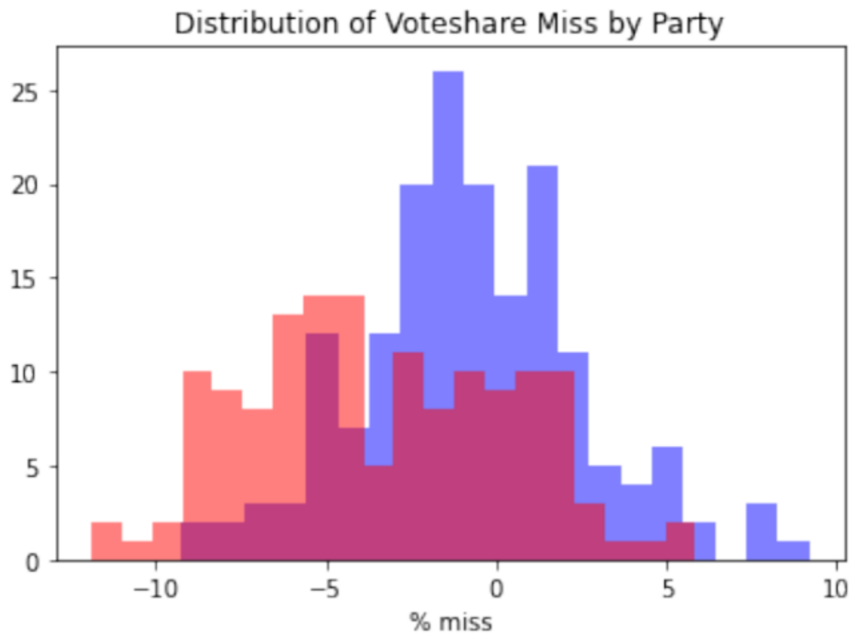
- Requirement #1:
  - 2 Categorical Variables
    - ∗ Rep Party Support?
    - ∗ Trump Endorsed?
    - ∗ and more (check below)
- Requirement #2:
  - 2 Quantitative Variables
    - ∗ Forecasted Voteshare
    - ∗ Actual Voteshare
    - ∗ and more (check below)

```
house_lite_forecast_df.plot.scatter(
    'voteshare','actual_voteshare', alpha=0.2, c='color'
)
plt.title('Forecasted vs. Actual Voteshare')
plt.xlabel('forecasted voteshare (%)')
plt.ylabel('actual voteshare (%)')
```



We can see that forecasted voteshare follows the actual voteshare very well on the scatterplot.

```
dem_vm = house_lite_forecast_df[house_lite_forecast_df['party']\
    == 'D'].voteshare_miss
rep_vm = house_lite_forecast_df[house_lite_forecast_df['party']\
    == 'R'].voteshare_miss
plt.hist(dem_vm, color='blue', alpha=.5, bins=20)
plt.hist(rep_vm, color='red', alpha=.5, bins=20)
plt.title('Distribution of Voteshare Miss by Party')
plt.xlabel('% miss')
```

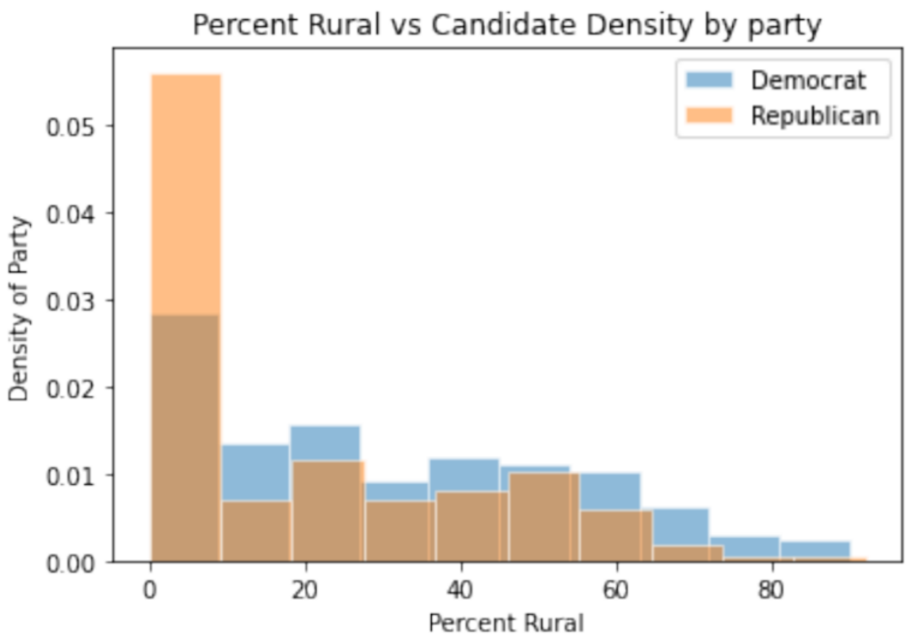


With the voteshare_miss variable normally distributed, it makes it a good variable to plot our OLS against with the available demographic features.

3 EDA Question 2

3.1 Rural Districts

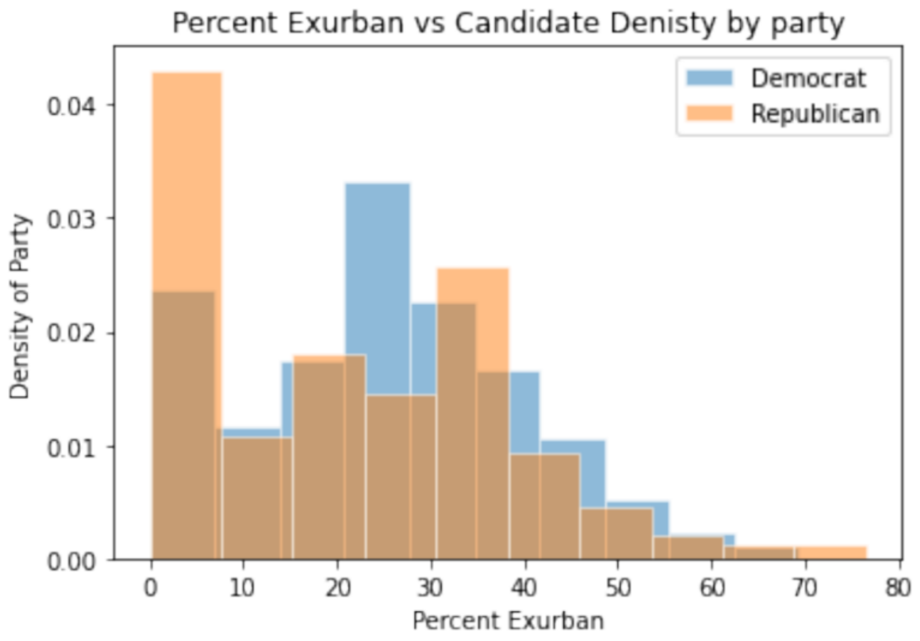
```
plt.hist(
    dem_urban['rural'], ec='white', alpha=0.5,
    label='Democrat', density=True
)
plt.hist(
    rep_urban['rural'], ec='white', alpha=0.5,
    label='Republican', density=True
)
plt.legend()
plt.xlabel('Percent Rural')
plt.ylabel('Density of Party')
plt.title('Percent Rural vs Candidate Density by party');
```



These histograms display that in districts where the over 10% of the population lives in a small town, the Democratic candidates had a higher density. This could potentially indicate that more Republican candidates come from non-rural areas than rural areas.

3.2 Small Town District

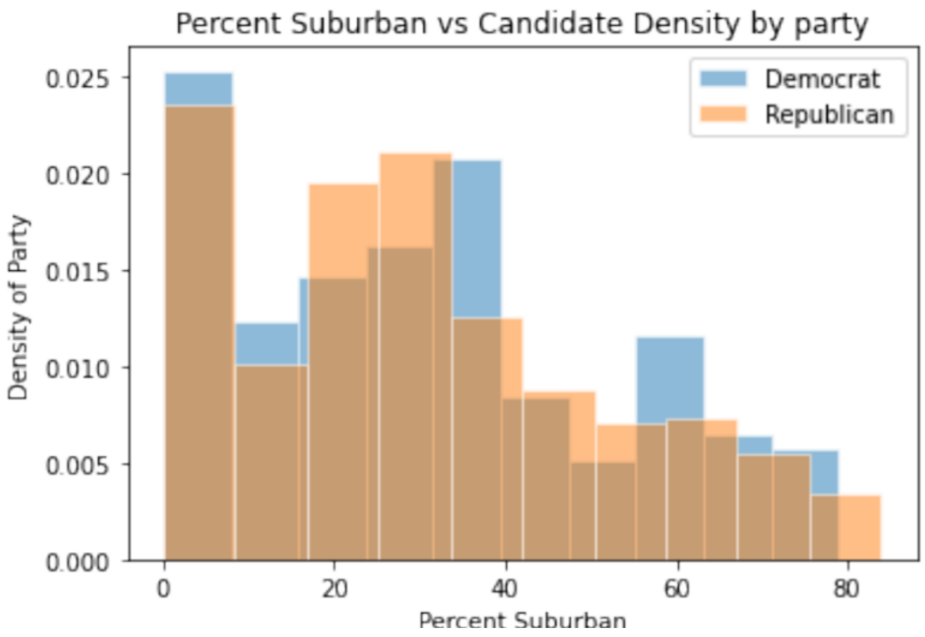
```
plt.hist(
    dem_urban['exurban'], ec='white', alpha=0.5,
    label='Democrat', density=True
)
plt.hist(
    rep_urban['exurban'], ec='white', alpha=0.5,
    label='Republican', density=True
)
plt.legend()
plt.xlabel('Percent Exurban')
plt.ylabel('Density of Party')
plt.title('Percent Exurban vs Candidate Denisty by party');
```



These histograms display that in districts where the over 10% of the population lives in a small town, the Democratic candidates had a higher density. The Republican density was slightly lower than the Democrat loss density for districts the same districts.

3.3 Suburban District

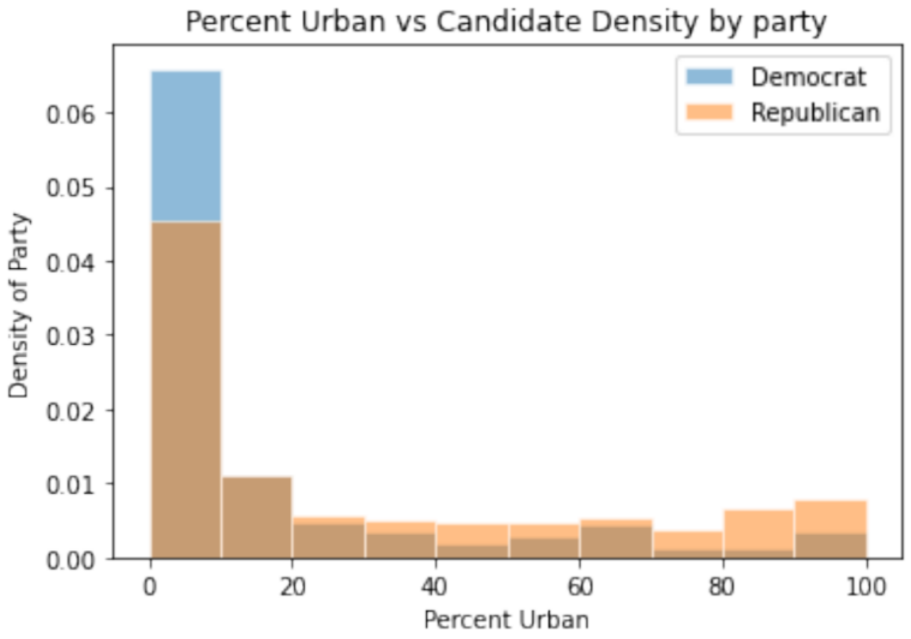
```
plt.hist(
    dem_urban['suburban'], ec='white', alpha=0.5,
    label='Democrat', density=True
)
plt.hist(
    rep_urban['suburban'], ec='white', alpha=0.5,
    label='Republican', density=True
)
plt.legend()
plt.xlabel('Percent Suburban')
plt.ylabel('Density of Party')
plt.title('Percent Suburban vs Candidate Density by party');
```



The Republican candidate density was nearly equivalent to the Democrat candidate density for all districts with any amount of suburban population. The candidate density varied throughout the percentages and no discernable trend can be found.

3.4 Urban District

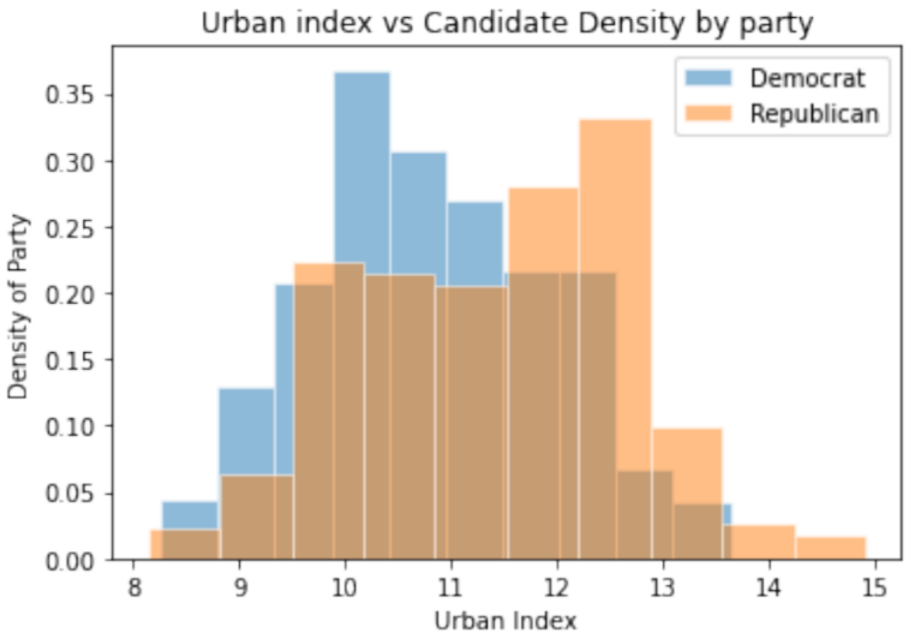
```
plt.hist(
    dem_urban['urban'], ec='white', alpha=0.5,
    label='Democrat', density=True
)
plt.hist(
    rep_urban['urban'], ec='white', alpha=0.5,
    label='Republican', density=True
)
plt.legend()
plt.xlabel('Percent Urban')
plt.ylabel('Density of Party')
plt.title('Percent Urban vs Candidate Density by party');
```



In districts with a over 10% urban populations, Republican candidates had a higher density than Democrat candidates. This could indicate more Republican candidates running from urbanized districts.

3.5 Candidate Density by Urbanization

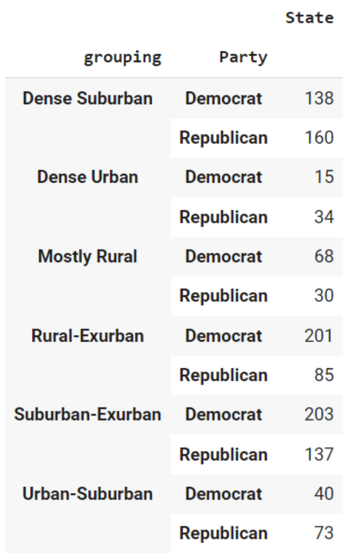
```
plt.hist(
    dem_urban['urbanindex'], ec='white', alpha=0.5,
    label='Democrat', density=True
)
plt.hist(
    rep_urban['urbanindex'], ec='white', alpha=0.5,
    label='Republican', density=True
)
plt.legend()
plt.xlabel('Urban Index')
plt.ylabel('Density of Party')
plt.title('Urban index vs Candidate Density by party');
```



Areas with a higher urban index also had a higher candidate density for Republicans. This could be because current office positions of urban areas are held by Democrats and suburban areas are held by Republicans, so less of those political parties run in those regions.

3.6 Party Density by Urbanization

```
election_urban.groupby(['grouping', 'Party']).count()[['State']]
```

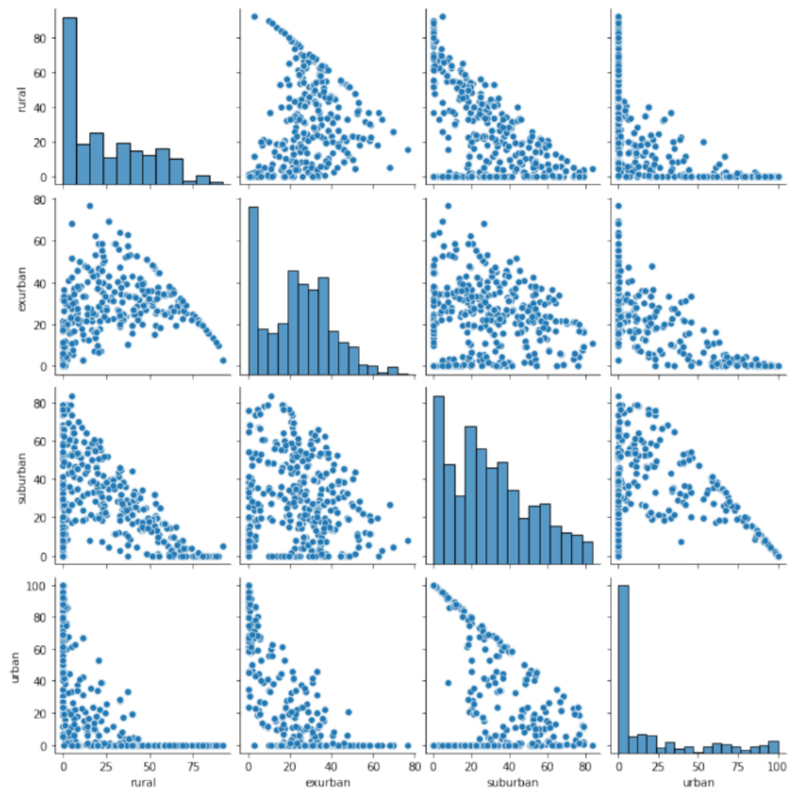


		State
grouping	Party	
Dense Suburban	Democrat	138
	Republican	160
Dense Urban	Democrat	15
	Republican	34
Mostly Rural	Democrat	68
	Republican	30
Rural-Exurban	Democrat	201
	Republican	85
Suburban-Exurban	Democrat	203
	Republican	137
Urban-Suburban	Democrat	40
	Republican	73

Most Democrat candidates came from rural-exurban, suburban-exurban, or dense suburban regions, while most Republican candidates came from suburban-exurban or dense suburban regions.

3.7 Relationship between population densities in regions

```
sns.pairplot(election_urban[[
    'rural', 'exurban', 'suburban', 'urban'
]])
plt.show()
```



This KDE plot displays that the more urban a region gets, the less likely it is to have small town regions in the district.

4 Research Questions

4.1 Question 1

Question: Do some demographic features enable an over-performance against polls (e.g. celebrity status in the 2022 primaries)?

This question will be answered using Option A (Multiple Hypothesis Testing/Decision Making) and Option C (Prediction with GLMs and non-parametric methods, and uses Bayesian Hierarchical Modeling). The multiple p-values for the hypothesis testing will be calculated using the test statistic and the p-values will be accepted or rejected using different error control rates. The GLM we will be choosing to test feature performance would be Logistic based on the normally distributed vote share we achieve from the combined data set.

The real-world application of this question would be how different features help us make quantitative decisions during primary elections.

4.2 Question 2

Question: Is the urbanization/sub-urbanization of the district predictive of the political party of the candidate?

This question will be answered using Option C: Prediction with GLMs and non-parametric methods, and uses Bayesian Hierarchical Modeling. The GLM we are choosing is Logistic because this is a classification problem. The link function is logit, the inverse link function is sigmoid, and the likelihood is Bernoulli.

The real-world application of this question would be that it could help a candidate know what type of competition to expect when running in the primary elections of a certain region.

5 Inference and Decisions Question 1

5.1 Multiple Hypothesis Testing

In this question, we will try to test whether particular features, such as being endorsed by a political group, have an association with the final vote share. We will do this by running hypothesis tests for each feature, with the null hypothesis being that there is no correlation between the feature and vote share and the alternative hypothesis being that there is some non-zero correlation between the two (whether that be positive or negative). A correlation test makes sense here since we have binary features and want to see how they potentially may have an association with the resulting votes.
To conduct this testing, we used the t-test, with a test statistic of

$$t^* = \frac{r\sqrt{n-2}}{\sqrt{1-R^2}}$$

Where n is the number of samples and r is the correlation coefficient. The degrees of freedom thus is $n-2$. We then can calculate the p-value by plugging the test statistic into the scipy.stats.t.sf function. The following are different methods of analyzing these p-values.

5.1.1 Simple Alpha Threshold

With this method, we simply make a conclusion comparing the p-value to the alpha level (in this case, we chose alpha to be 5%). At this level, we found 8/13 discoveries for the Republican features and 3/12 discoveries for the Democrat features.

5.1.2 Bonferroni

With this method, we set our FWER to be 0.15 and made decisions with the Bonferroni method, which computes an alpha threshold based on our FWER and the number of decisions to be made. This controls the probability that we have at least one false discovery to be 0.15. At this FWER, we found only 4 discoveries for Republican features and 2 for Democrat features.

5.1.3 Benjamini-Hochberg

With this method, we set out FDR to be 0.15 and made decisions with the Benjamini-Hochberg method, which finds a certain p-value from the tests to be the threshold based on the alpha level and the number of decisions to be made. This controls the expected number of false positives proportional to the total number of positives. At this FDR, we found 10 discoveries for Republican features and 3 for Democrat features.

We can summarize our results for the different methods of discovery in the following tables.

5.1.4 Feature Selection

For the data on Republicans, the features we will use are:

- Rep Party Support?
- Trump Endorsed?
- Bannon Endorsed?
- Chamber Endorsed?
- Great America Endorsed?
- Right to Life Endorsed?
- Tea Party Endorsed?
- Susan B. Anthony Endorsed?
- NRA Endorsed?
- House Freedom Support?
- Main Street Endorsed?
- Club for Growth Endorsed?
- Kock Support?

For the data on Democratss, the features we will use are:

- Veteran?
- LGBTQ?
- Elected Official?
- Self-Funder?
- STEM?
- Obama Alum?
- Party Support?
- Emily Endorsed?
- Guns Sense Candidate?
- WFP Endorsed?
- Indivisible Endorsed?
- Justice Dems Endorsed?

5.1.5 Hypothesis

Null Hypothesis: No Correlation between the voteshare and the features we've chosen

Alternative Hypothesis: There's a correlation between the voteshare and the features we've chosen

```
dem_hyp_data = house_lite_forecast_df\
    [house_lite_forecast_df['party'] == 'D']
rep_hyp_data = house_lite_forecast_df\
    [house_lite_forecast_df['party'] == 'R']
rep_features = [
    "Rep Party Support?", "Trump Endorsed?",
    "Bannon Endorsed?", "Chamber Endorsed?",
    "Great America Endorsed?", "Right to Life Endorsed?",
    "Tea Party Endorsed?", "Susan B. Anthony Endorsed?",
    "NRA Endorsed?", "House Freedom Support?",
    "Main Street Endorsed?",
    "Club for Growth Endorsed?", "Koch Support?"
]

dem_features = [
    "Veteran?", "LGBTQ?", "Elected Official?", "Self-Funder?",
    "STEM?", "Obama Alum?", "Party Support?", "Emily Endorsed?",
    "Guns Sense Candidate?", "WFP Endorsed?",
    "Indivisible Endorsed?", "Justice Dems Endorsed?"
]
```

5.1.6 P-value and Rejection Functions

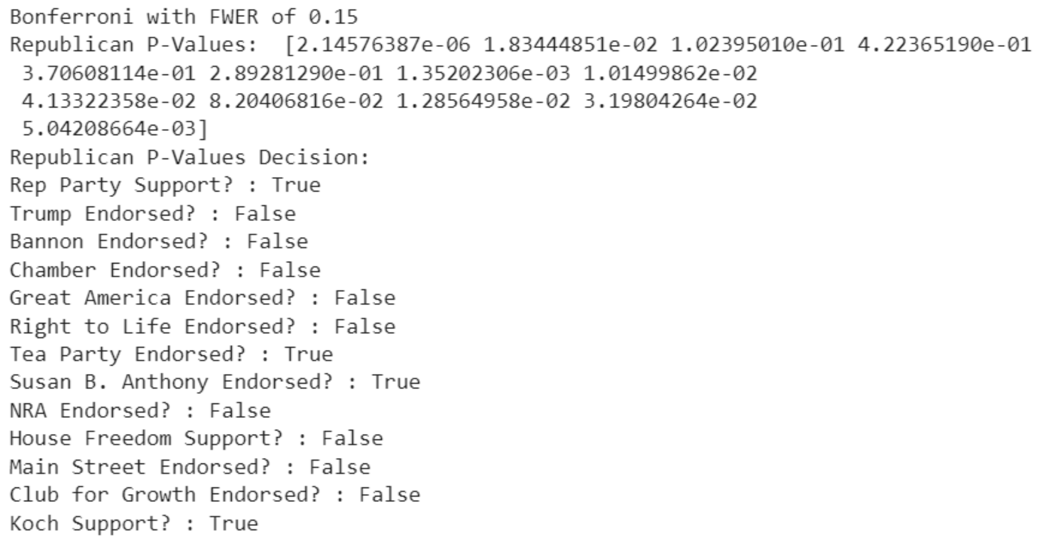
```
def pvalue(feature):
    if feature in rep_features:
        binary = rep_hyp_data[feature] == "Yes"
        r = rep_vm.corr(binary)
    else:
        binary = dem_hyp_data[feature] == "Yes"
        r = dem_vm.corr(binary)

    n = len(binary)
    t = r*((n - 2) ** 0.5) / ((1 - r**2) ** 0.5)
    return scipy.stats.t.sf(abs(t), df=n-2)

def rejections(alpha=0.05, display=False):
    outcome = [
        "Fail to Reject the Null Hypothesis",
        "Reject the Null Hypothesis"
    ]
    resultRep = []
    resultDem = []
    print("Republican:")
    print()
    for feature in rep_features:
        pval = pvalue(feature)
        rejected = 0 if pval > alpha else 1
        resultRep += [(feature, pval, rejected)]
        if display:
            print(feature, pval, outcome[rejected])
            print()
    print()
    print("Democrats:")
    print()
    for feature in dem_features:
        pval = pvalue(feature)
        rejected = 0 if pval > alpha else 1
        resultDem += [(feature, pval, rejected)]
        if display:
            print(feature, pval, outcome[rejected])
            print()
    return resultRep, resultDem

resultRep1, resultDem1 = rejections(display=True)
```

```
Republican:

Rep Party Support? 2.1457638700749487e-06 Reject the Null Hypothesis

Trump Endorsed? 0.01834448506793082 Reject the Null Hypothesis

Bannon Endorsed? 0.1023950102763866 Fail to Reject the Null Hypothesis

Chamber Endorsed? 0.4223651899131726 Fail to Reject the Null Hypothesis

Great America Endorsed? 0.37060811354831513 Fail to Reject the Null Hypothesis

Right to Life Endorsed? 0.2892812897522239 Fail to Reject the Null Hypothesis

Tea Party Endorsed? 0.001352023055384846 Reject the Null Hypothesis

Susan B. Anthony Endorsed? 0.01014998622543684 Reject the Null Hypothesis

NRA Endorsed? 0.0413322357696588 Reject the Null Hypothesis

House Freedom Support? 0.0820406815520466 Fail to Reject the Null Hypothesis

Main Street Endorsed? 0.012856495750998995 Reject the Null Hypothesis

Club for Growth Endorsed? 0.03198042640191035 Reject the Null Hypothesis

Koch Support? 0.005042086635543937 Reject the Null Hypothesis
```

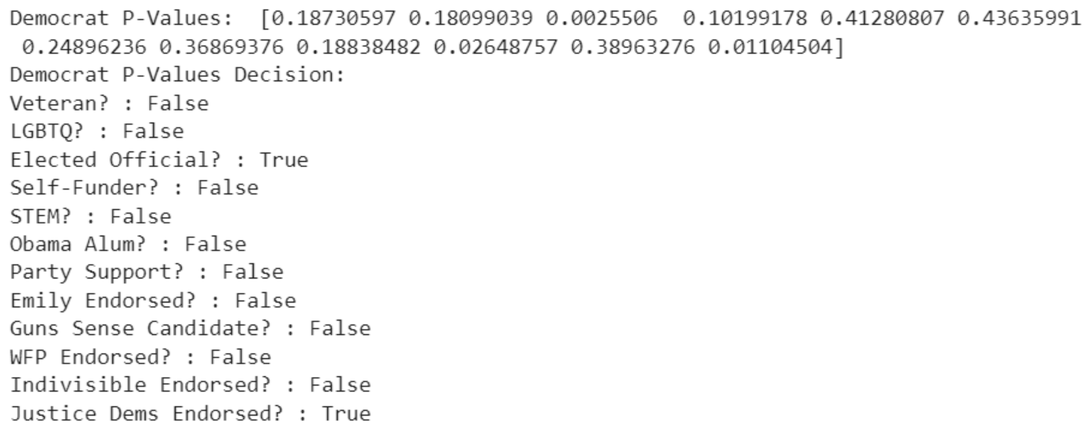
```
Democrats:

Veteran? 0.1873059715640597 Fail to Reject the Null Hypothesis

LGBTQ? 0.1809903944292694 Fail to Reject the Null Hypothesis

Elected Official? 0.0025506043676521636 Reject the Null Hypothesis

Self-Funder? 0.1019917809084675 Fail to Reject the Null Hypothesis

STEM? 0.41280806777280954 Fail to Reject the Null Hypothesis

Obama Alum? 0.4363599109896423 Fail to Reject the Null Hypothesis

Party Support? 0.2489623621859655 Fail to Reject the Null Hypothesis

Emily Endorsed? 0.36869375749428257 Fail to Reject the Null Hypothesis

Guns Sense Candidate? 0.1883848199690159 Fail to Reject the Null Hypothesis

WFP Endorsed? 0.026487573690039696 Reject the Null Hypothesis

Indivisible Endorsed? 0.38963275980963563 Fail to Reject the Null Hypothesis

Justice Dems Endorsed? 0.011045041836795437 Reject the Null Hypothesis
```

5.1.7 Bonferroni Function

```
def bonferroni(p_values, alpha_total):
    """
    Returns decisions on p-values using the Bonferroni correction.

    Inputs:
        p_values: array of p-values
        alpha_total: desired family-wise error rate

    Returns:
        decisions: binary array of same length as p-values,
        where `decisions[i]` is 1 if `p_values[i]` is deemed
        significant, and 0 otherwise
    """
    n = len(p_values)
    alpha_star = alpha_total/n
    decisions = p_values <= alpha_star
    return decisions

print("Bonferroni with FWER of 0.15")

alpha = 0.15
repArray = np.array([])
for rep in resultRep1:
    repArray = np.append(repArray, rep[1])
print("Republican P-Values: ", repArray)
repDecisions = bonferroni(repArray, alpha)
print("Republican P-Values Decision: ")
for i in range(len(repDecisions)):
    print(resultRep1[i][0], ":", repDecisions[i])
print()

alpha = 0.15
demArray = np.array([])
for dem in resultDem1:
    demArray = np.append(demArray, dem[1])
print("Democrat P-Values: ", demArray)
demDecisions = bonferroni(demArray, alpha)
print("Democrat P-Values Decision: ")
for i in range(len(demDecisions)):
    print(resultDem1[i][0], ":", demDecisions[i])
resultRep2 = [(rep_features[i], repArray[i],    repDecisions[i]*1)\
    for i in range(len(rep_features))]
resultDem2 = [(dem_features[i], demArray[i], demDecisions[i]*1)\
    for i in range(len(dem_features))]
```



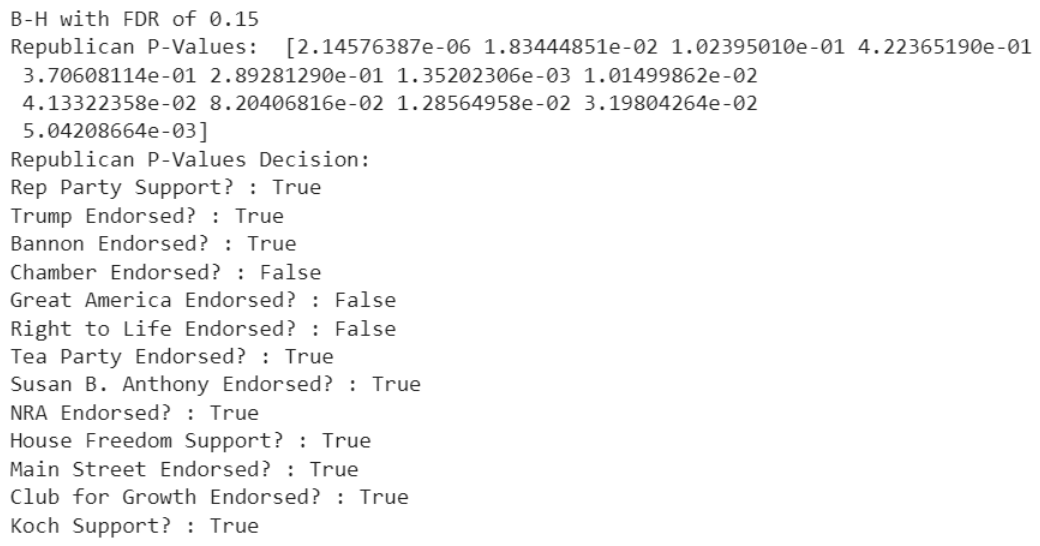
```
Bonferroni with FWER of 0.15
Republican P-Values:  [2.14576387e-06 1.83444851e-02 1.02395010e-01 4.22365190e-01
 3.70608114e-01 2.89281290e-01 1.35202306e-03 1.01499862e-02
 4.13322358e-02 8.20406816e-02 1.28564958e-02 3.19804264e-02
 5.04208664e-03]
Republican P-Values Decision:
Rep Party Support? : True
Trump Endorsed? : False
Bannon Endorsed? : False
Chamber Endorsed? : False
Great America Endorsed? : False
Right to Life Endorsed? : False
Tea Party Endorsed? : True
Susan B. Anthony Endorsed? : True
NRA Endorsed? : False
House Freedom Support? : False
Main Street Endorsed? : False
Club for Growth Endorsed? : False
Koch Support? : True
```



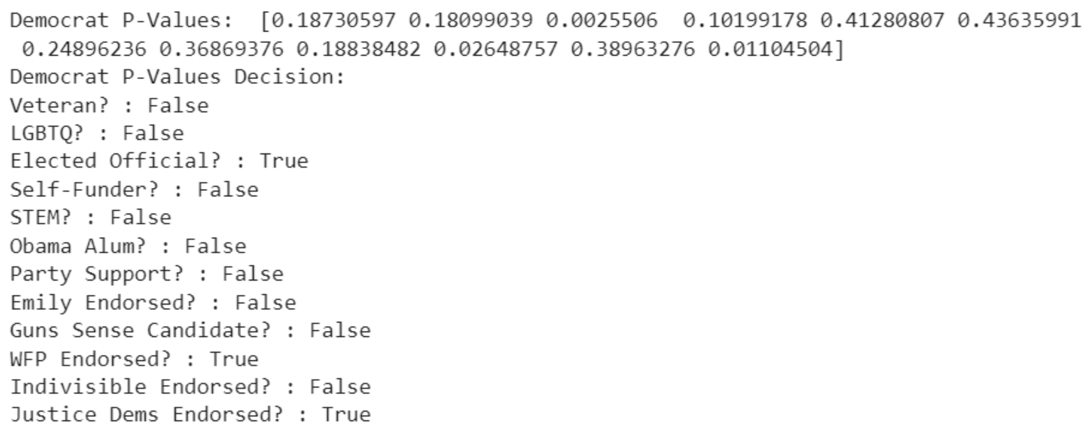
```
Democrat P-Values:  [0.18730597 0.18099039 0.0025506  0.10199178 0.41280807 0.43635991
 0.24896236 0.36869376 0.18838482 0.02648757 0.38963276 0.01104504]
Democrat P-Values Decision:
Veteran? : False
LGBTQ? : False
Elected Official? : True
Self-Funder? : False
STEM? : False
Obama Alum? : False
Party Support? : False
Emily Endorsed? : False
Guns Sense Candidate? : False
WFP Endorsed? : False
Indivisible Endorsed? : False
Justice Dems Endorsed? : True
```

5.1.8 Benjamini Hochberg Function

```
def benjamini_hochberg(p_values, alpha):
    """
    Returns decisions on p-values using Benjamini-Hochberg.

    Inputs:
        p_values: array of p-values
        alpha: desired FDR

    Returns:
        decisions: binary array of same length as p-values,
        where `decisions[i]` is 1 if `p_values[i]` is deemed
        significant, and 0 otherwise
    """
    p_sorted = sorted(p_values)
    n = len(p_sorted)
    for k in range(n):
        if p_sorted[k] <= ((k+1)*alpha/n):
            threshold = p_sorted[k]
    decisions = p_values <= threshold
    return decisions

print("B-H with FDR of 0.15")

alpha = 0.15
repArray = np.array([])
for rep in resultRep1:
    repArray = np.append(repArray, rep[1])
print("Republican P-Values: ", repArray)
repDecisions = benjamini_hochberg(repArray, alpha)
print("Republican P-Values Decision: ")
for i in range(len(repDecisions)):
    print(resultRep1[i][0], ":", repDecisions[i])
print()

alpha = 0.15
demArray = np.array([])
for dem in resultDem1:
    demArray = np.append(demArray, dem[1])
print("Democrat P-Values: ", demArray)
demDecisions = benjamini_hochberg(demArray, alpha)
print("Democrat P-Values Decision: ")
for i in range(len(demDecisions)):
    print(resultDem1[i][0], ":", demDecisions[i])
```



```
B-H with FDR of 0.15
Republican P-Values:  [2.14576387e-06 1.83444851e-02 1.02395010e-01 4.22365190e-01
 3.70608114e-01 2.89281290e-01 1.35202306e-03 1.01499862e-02
 4.13322358e-02 8.20406816e-02 1.28564958e-02 3.19804264e-02
 5.04208664e-03]
Republican P-Values Decision:
Rep Party Support? : True
Trump Endorsed? : True
Bannon Endorsed? : True
Chamber Endorsed? : False
Great America Endorsed? : False
Right to Life Endorsed? : False
Tea Party Endorsed? : True
Susan B. Anthony Endorsed? : True
NRA Endorsed? : True
House Freedom Support? : True
Main Street Endorsed? : True
Club for Growth Endorsed? : True
Koch Support? : True
```



```
Democrat P-Values:  [0.18730597 0.18099039 0.0025506  0.10199178 0.41280807 0.43635991
 0.24896236 0.36869376 0.18838482 0.02648757 0.38963276 0.01104504]
Democrat P-Values Decision:
Veteran? : False
LGBTQ? : False
Elected Official? : True
Self-Funder? : False
STEM? : False
Obama Alum? : False
Party Support? : False
Emily Endorsed? : False
Guns Sense Candidate? : False
WFP Endorsed? : True
Indivisible Endorsed? : False
Justice Dems Endorsed? : True
```

5.1.9 Results

In the hypothesis testing section, we first used the test statistics formula to calculate the p-values for different features (13 in total) and used three different methods to reject or accept the hypothesis. From the naive methodology where we decide on a fixed threshold, we discovered 8 of the 13 features affected the vote share for republican and 3 of the 12 features affected the vote share for democrats.
When we use the Bonferroni error correction with an FWER of 0.15, we discovered that only 4 out of 13 features affected the vote share for republican features and 2 out of 12 for democrat.
Lastly, for B-H error correction with an FWER of 0.15, we discovered 10 of the 13 features affected the vote share for republican, while only 3 features out of the 12 features affected the vote share for democrats.

```
finalRepResults = [(rep_features[i], resultRep1[i][1], \
    resultRep1[i][2], resultRep2[i][2], resultRep3[i][2]) \
    for i in range(len(rep_features))]
finalDemResults = [(dem_features[i], resultDem1[i][1], \
    resultDem1[i][2], resultDem2[i][2], resultDem3[i][2]) \
    for i in range(len(dem_features))]

tableRep = pd.DataFrame.from_records(finalRepResults,\
    columns = ["Feature", "P-Value", "Simple Threshold Discovery",\
    "Bonferroni Discovery", "Benjamini Hochberg Discovery"])
tableDem = pd.DataFrame.from_records(finalDemResults,\
    columns = ["Feature", "P-Value", "Simple Threshold Discovery",\
    "Bonferroni Discovery", "Benjamini Hochberg Discovery"])

tableRep = tableRep.sort_values("P-Value")
display(tableRep)

tableDem = tableDem.sort_values("P-Value")
display(tableDem)
```

Republican Feature Table

	Feature	P-Value	Simple Threshold Discovery	Bonferroni Discovery	Benjamini Hochberg Discovery
0	Rep Party Support?	0.000002	1	1	1
6	Tea Party Endorsed?	0.001352	1	1	1
12	Koch Support?	0.005042	1	1	1
7	Susan B. Anthony Endorsed?	0.010150	1	1	1
10	Main Street Endorsed?	0.012856	1	0	1
1	Trump Endorsed?	0.018344	1	0	1
11	Club for Growth Endorsed?	0.031980	1	0	1
8	NRA Endorsed?	0.041332	1	0	1
9	House Freedom Support?	0.082041	0	0	1
2	Bannon Endorsed?	0.102395	0	0	1
5	Right to Life Endorsed?	0.289281	0	0	0
4	Great America Endorsed?	0.370608	0	0	0
3	Chamber Endorsed?	0.422365	0	0	0

Democrat Feature Table

	Feature	P-Value	Simple Threshold Discovery	Bonferroni Discovery	Benjamini Hochberg Discovery
2	Elected Official?	0.002551	1	1	1
11	Justice Dems Endorsed?	0.011045	1	1	1
9	WFP Endorsed?	0.026488	1	0	1
3	Self-Funder?	0.101992	0	0	0
1	LGBTQ?	0.180990	0	0	0
0	Veteran?	0.187306	0	0	0
8	Guns Sense Candidate?	0.188385	0	0	0
6	Party Support?	0.248962	0	0	0
7	Emily Endorsed?	0.368694	0	0	0
10	Indivisible Endorsed?	0.389633	0	0	0
4	STEM?	0.412808	0	0	0
5	Obama Alum?	0.436360	0	0	0

For Republican features, over half resulted in a discovery across multiple tests, leading to the conclusion that there is some notable association between these features and a candidates vote share. However for Democrat features, there were very few discoveries across any test, potentially showing that there is less association with the different features and the candidates vote share when compared to the Republican features.

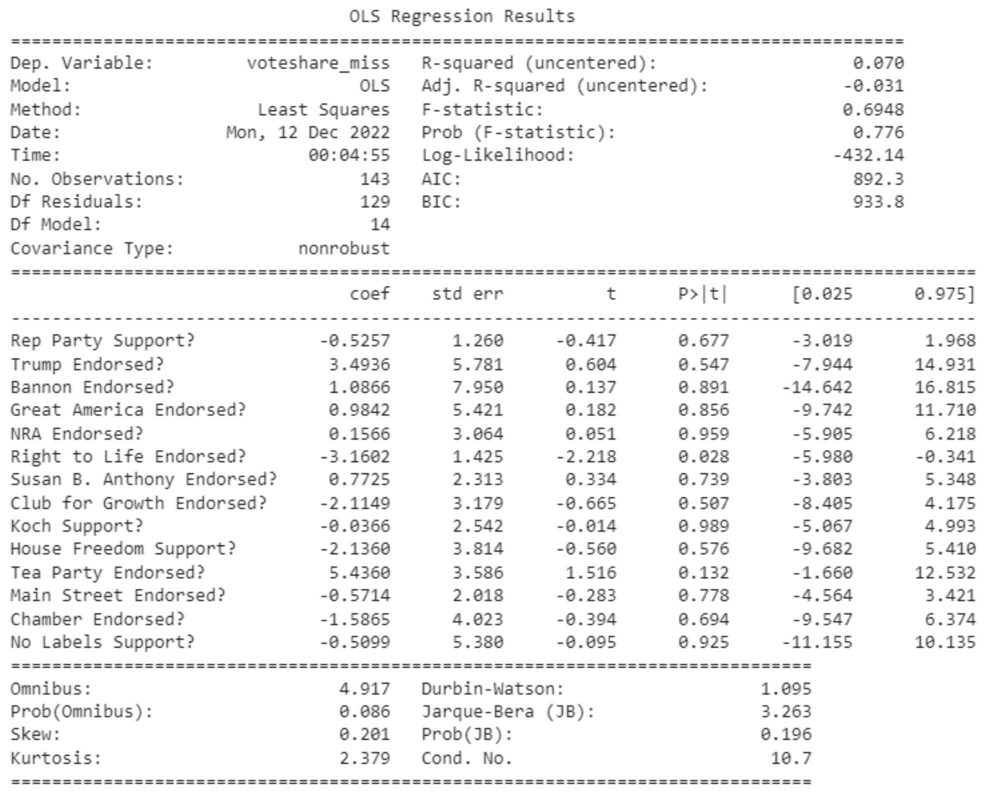
5.2 Prediction with GLMS and nonparametric methods

This section will look into how selected features actually perform against Republican voteshare and Democrat voteshare.

5.2.1 Republican Voteshare Miss

We are forecasting the Republican Voteshare Miss using Ordinary Least Squares Regression (OLS). From this OLS build, the influence of each demographic feature on the voteshare can be witnessed for the Republican group. From the relationship to the voteshare_miss variable, the most influential demographic feature seems to be the candidate's ability to be endorsed by the Tea Party.

```
house_rep_lite_forecast_df = house_lite_forecast_df\
    [house_lite_forecast_df.party=='R']
rep_endorse_cols = rep_candidates_df.columns[11:-1]
house_rep_lite_forecast_df[rep_endorse_cols] = \
    house_rep_lite_forecast_df[rep_endorse_cols]\
    .fillna(0).replace('Yes',1).replace('No',0)
X = house_rep_lite_forecast_df[rep_endorse_cols]
sm.add_constant(X)
y = house_rep_lite_forecast_df['voteshare_miss']
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```



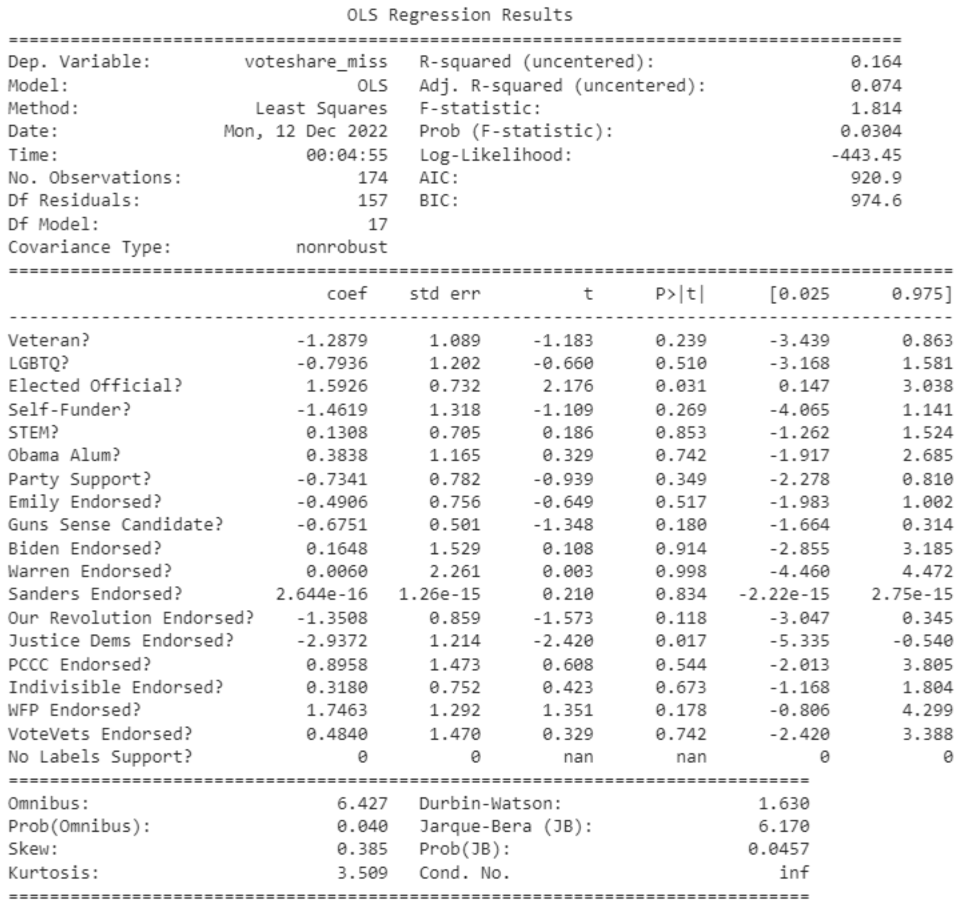
```
                            OLS Regression Results
=======================================================================================
Dep. Variable:         voteshare_miss   R-squared (uncentered):                   0.070
Model:                            OLS   Adj. R-squared (uncentered):             -0.031
Method:                 Least Squares   F-statistic:                             0.6948
Date:                Mon, 12 Dec 2022   Prob (F-statistic):                       0.776
Time:                        00:04:55   Log-Likelihood:                         -432.14
No. Observations:                 143   AIC:                                      892.3
Df Residuals:                     129   BIC:                                      933.8
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
Rep Party Support?            -0.5257      1.260     -0.417      0.677      -3.019       1.968
Trump Endorsed?                3.4936      5.781      0.604      0.547      -7.944      14.931
Bannon Endorsed?               1.0866      7.950      0.137      0.891     -14.642      16.815
Great America Endorsed?        0.9842      5.421      0.182      0.856      -9.742      11.710
NRA Endorsed?                  0.1566      3.064      0.051      0.959      -5.905       6.218
Right to Life Endorsed?       -3.1602      1.425     -2.218      0.028      -5.980      -0.341
Susan B. Anthony Endorsed?     0.7725      2.313      0.334      0.739      -3.803       5.348
Club for Growth Endorsed?     -2.1149      3.179     -0.665      0.507      -8.405       4.175
Koch Support?                 -0.0366      2.542     -0.014      0.989      -5.067       4.993
House Freedom Support?        -2.1360      3.814     -0.560      0.576      -9.682       5.410
Tea Party Endorsed?            5.4360      3.586      1.516      0.132      -1.660      12.532
Main Street Endorsed?         -0.5714      2.018     -0.283      0.778      -4.564       3.421
Chamber Endorsed?             -1.5865      4.023     -0.394      0.694      -9.547       6.374
No Labels Support?            -0.5099      5.380     -0.095      0.925     -11.155      10.135
==============================================================================
Omnibus:                        4.917   Durbin-Watson:                   1.095
Prob(Omnibus):                  0.086   Jarque-Bera (JB):                3.263
Skew:                           0.201   Prob(JB):                        0.196
Kurtosis:                       2.379   Cond. No.                         10.7
==============================================================================
```

The most influential feature for a Republican candidate turned out to be whether the candidate was Tea Party endorsed. The feature that varied/deviated the most for the Republicans was whether the candidate was Bannon endorsed.

5.2.2 Democrat Voteshare Miss

We are forecasting Democrat's Voteshare Miss with OLS. From this OLS build, the influence of each demographic feature on the voteshare can be witnessed for the Democrat group. From the relationship to the voteshare_miss variable, the most influential demographic feature seems to be whether the candidate is Justice Dems Endorsed.

```
house_dem_lite_forecast_df = house_lite_forecast_df\
    [house_lite_forecast_df.party=='D']
dem_endorse_cols = dem_candidates_df.columns[13:-1]
house_dem_lite_forecast_df[dem_endorse_cols] = \
    house_dem_lite_forecast_df[dem_endorse_cols]\
    .fillna(0).replace('Yes',1).replace('No',0)
X = house_dem_lite_forecast_df[dem_endorse_cols]
sm.add_constant(X)
y = house_dem_lite_forecast_df['voteshare_miss']
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```



```
                            OLS Regression Results
=======================================================================================
Dep. Variable:          voteshare_miss   R-squared (uncentered):                  0.164
Model:                             OLS   Adj. R-squared (uncentered):             0.074
Method:                  Least Squares   F-statistic:                             1.814
Date:                 Mon, 12 Dec 2022   Prob (F-statistic):                     0.0304
Time:                         00:04:55   Log-Likelihood:                        -443.45
No. Observations:                  174   AIC:                                     920.9
Df Residuals:                      157   BIC:                                     974.6
Df Model:                           17
Covariance Type:             nonrobust
============================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
Veteran?                   -1.2879      1.089     -1.183      0.239      -3.439       0.863
LGBTQ?                     -0.7936      1.202     -0.660      0.510      -3.168       1.581
Elected Official?           1.5926      0.732      2.176      0.031       0.147       3.038
Self-Funder?               -1.4619      1.318     -1.109      0.269      -4.065       1.141
STEM?                       0.1308      0.705      0.186      0.853      -1.262       1.524
Obama Alum?                 0.3838      1.165      0.329      0.742      -1.917       2.685
Party Support?             -0.7341      0.782     -0.939      0.349      -2.278       0.810
Emily Endorsed?            -0.4906      0.756     -0.649      0.517      -1.983       1.002
Guns Sense Candidate?      -0.6751      0.501     -1.348      0.180      -1.664       0.314
Biden Endorsed?             0.1648      1.529      0.108      0.914      -2.855       3.185
Warren Endorsed?            0.0060      2.261      0.003      0.998      -4.460       4.472
Sanders Endorsed?        2.644e-16   1.26e-15      0.210      0.834   -2.22e-15    2.75e-15
Our Revolution Endorsed?   -1.3508      0.859     -1.573      0.118      -3.047       0.345
Justice Dems Endorsed?     -2.9372      1.214     -2.420      0.017      -5.335      -0.540
PCCC Endorsed?              0.8958      1.473      0.608      0.544      -2.013       3.805
Indivisible Endorsed?       0.3180      0.752      0.423      0.673      -1.168       1.804
WFP Endorsed?               1.7463      1.292      1.351      0.178      -0.806       4.299
VoteVets Endorsed?          0.4840      1.470      0.329      0.742      -2.420       3.388
No Labels Support?               0          0        nan        nan           0           0
==============================================================================
Omnibus:                         6.427   Durbin-Watson:                   1.630
Prob(Omnibus):                   0.040   Jarque-Bera (JB):                6.170
Skew:                            0.385   Prob(JB):                       0.0457
Kurtosis:                        3.509   Cond. No.                         inf
==============================================================================
```

By calculating the vote share miss, we were able to get a normal distribution which allowed us to immediately perform Ordinary Least Squares on the feature performance against our vote share. The most influential feature for a Democratic candidate turned out to be whether the candidate was Justice Dems Endorsed. The feature that varied/deviated the most for the Democrats was whether the candidate was Warren endorsed.

6 Inference and Decisions Question 2

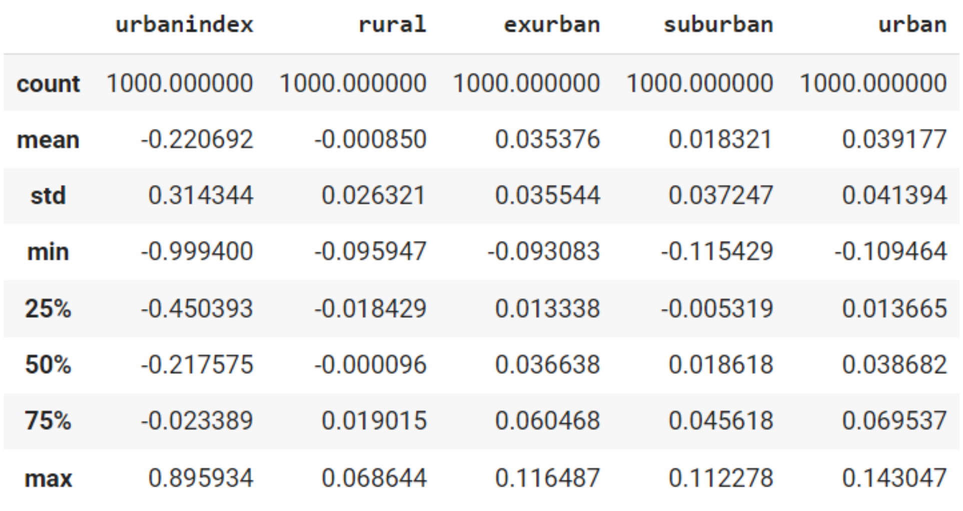
6.1 Bayesian Logistic Model

In this question, we will try to predict whether a candidate is Republican or Democrat based off of the ”urbanindex”, ”rural”, ”exurban”, ”suburban”, and ”urban” features of ‘election_data‘. As seen in the data graphs above, all five of these features are correlated with political party and could be indicative of what a candidate’s political party is for the primary elections of that district.

Therefore, the formula for the Bayesian model with no intercept is

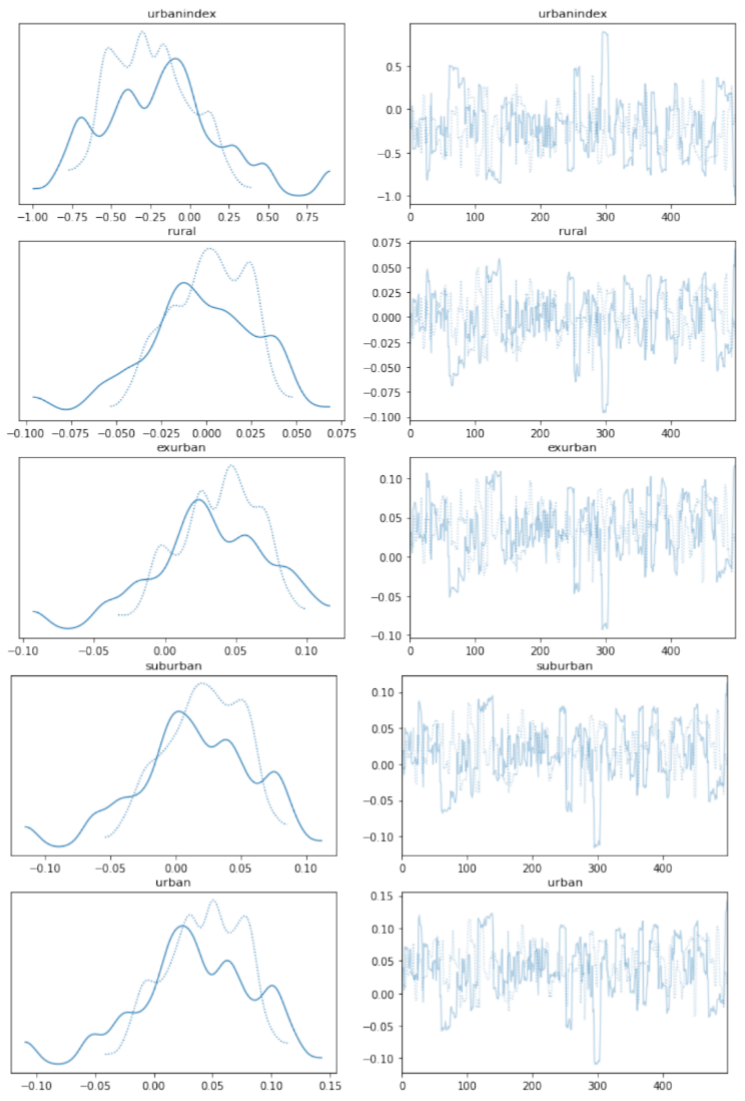
$$\text{Party} \sim 0 + \text{urbanindex} + \text{rural} + \text{exurban} + \text{suburban} + \text{urban}$$

```
formula = 'Party ~ 0 + urbanindex + rural + exurban +
suburban + urban'
with pm.Model() as bayesian_model:
    pm.glm.GLM.from_formula(
        formula, election_urban,
        family=pm.glm.families.Binomial()
    )
    bayesian_trace = pm.sample(
        500, cores=1, target_accept=0.95,
        progressbar=False, random_seed=42,
        return_inferencedata=True
    )
    bayesian_posterior = bayesian_trace.posterior.to_dataframe()
    bayesian_posterior.describe()
```



	urbanindex	rural	exurban	suburban	urban
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	-0.220692	-0.000850	0.035376	0.018321	0.039177
std	0.314344	0.026321	0.035544	0.037247	0.041394
min	-0.999400	-0.095947	-0.093083	-0.115429	-0.109464
25%	-0.450393	-0.018429	0.013338	-0.005319	0.013665
50%	-0.217575	-0.000096	0.036638	0.018618	0.038682
75%	-0.023389	0.019015	0.060468	0.045618	0.069537
max	0.895934	0.068644	0.116487	0.112278	0.143047

```
az.plot_trace(bayesian_trace, figsize=(12, 18));
```



```
features = bayesian_posterior.columns
for feature in features:
    s = ''
    if feature in ['rural', 'exurban', 'urban']:
        s = '\t'
    print(feature + '\t' + s,
    np.round(bayesian_posterior[feature].mean(), 6))
```

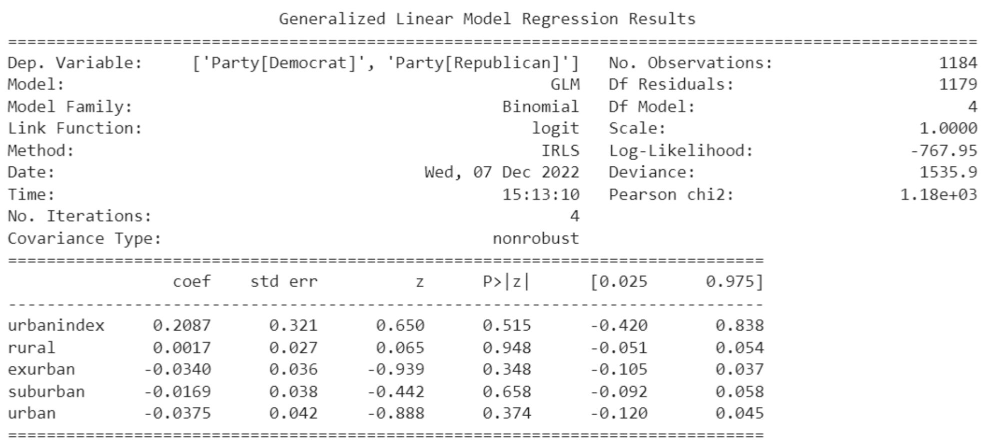
```
urbanindex          −0.220692
rural               −0.00085
exurban             0.035376
suburban            0.018321
urban               0.039177
```

Each individual urbanization factor did not play much of a role for predicting the political party of the candidate.

However, the overall ”urbanindex” feature coefficient showed that the higher the urbanization of a district, the less likely it is that a Democratic candidate is running in the primary election.

6.2 Frequentist Logistic Model

```
frequentist_model = smf.glm(
    formula, election_urban, family=sm.families.Binomial()
)
results = frequentist_model.fit()
print(results.summary())
```



```
                         Generalized Linear Model Regression Results
==================================================================================================
Dep. Variable:     ['Party[Democrat]', 'Party[Republican]']   No. Observations:               1184
Model:                                                 GLM   Df Residuals:                   1179
Model Family:                                     Binomial   Df Model:                          4
Link Function:                                       logit   Scale:                        1.0000
Method:                                               IRLS   Log-Likelihood:              -767.95
Date:                                     Wed, 07 Dec 2022   Deviance:                     1535.9
Time:                                             15:13:10   Pearson chi2:               1.18e+03
No. Iterations:                                          4
Covariance Type:                                 nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
urbanindex     0.2087      0.321      0.650      0.515      -0.420       0.838
rural          0.0017      0.027      0.065      0.948      -0.051       0.054
exurban       -0.0340      0.036     -0.939      0.348      -0.105       0.037
suburban      -0.0169      0.038     -0.442      0.658      -0.092       0.058
urban         -0.0375      0.042     -0.888      0.374      -0.120       0.045
==============================================================================
```

According to the freqentist logistical regression, a similar result from the Bayesian logistical regression is achieved. Each individual urbanization factor did not play much of a role for predicting the political party of the candidate.

However, the overall "urbanindex" feature coefficient showed that the higher the urbanization of a district, the less likely it is that a Democratic candidate is running in the primary election.

6.3 Running a frequentist Logistic Regression using sklearn

```
logistic_model = LogisticRegression(random_state=42)
X = election_urban[[
    'urbanindex', 'rural', 'exurban', 'suburban', 'urban'
]]
y = election_urban['Party']
X_train, X_test, Y_train, Y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
logistic_model.fit(X_train, Y_train)
preds = logistic_model.predict(X_test)
logistic_accuracy = accuracy_score(preds, Y_test)
print('Accuracy:', logistic_accuracy)
```

```
Accuracy: 0.6751054852320675
```

7 Conclusions

7.1 Question 1

The key finding from Question 1 was the influence of each feature on the candidate's vote share in the primary elections from the hypothesis testing and the GLMs performed.

In the hypothesis testing section, we first used the test statistics formula to calculate the p-values for different features (13 in total) and used three different methods to reject or accept the hypothesis. From the naive methodology where we decide on a fixed alpha threshold, we discovered 6 of the 13 features affected the vote share for Republicans and 3 of the 13 features affected the vote share for democrats.

When we use the Bonferroni error correction with an FWER of 0.15, we discovered that only 2 out of 13 features affected the vote share of each republican and democrat.

Lastly, for B-H error correction with an FWER of 0.15, we discovered 10 of the 13 features affected the vote share for Republicans, while only 3 features out of the 13 features affected the vote share for democrats.

By calculating the vote share miss, we were able to get a normal distribution which allowed us to immediately perform Ordinary Least Squares on the feature performance against our vote share. The most influential feature for Republicans turned out to be whether the candidate was tea party endorsed, while for the Democrats, the most influential feature turned out to be whether the candidate was Justice Dems Endorsed.

The feature that varied/deviated the most for the republicans was whether the candidate was Bannon endorsed, while for the democrats, it was whether the candidate was Warren endorsed.

Overall, the relationship between these two statistical analyses was that the features that were shown to be the most influential in GLMs turned out to have a correlation with the vote share when using the Hypothesis Testing.

In the future, we can bring in other features which we can test against the candidate's vote share and even expand the question to answering forecasting of vote share using available features. In such a case, we can try using Decision Trees and Random Forests we've learned from class to output interesting results.

7.2 Question 2

The key finding from Question 2 was that a candidate's political party in the primary elections is correlated with the amount of urbanization.

Currently, more Democrats hold office positions in urbanized areas and more Republicans hold office positions in rural areas. This map shows the current distribution of political parties over districts.

Therefore, more of the opposing party runs in the primary elections to compete for office positions. The data showed that Democrats run more frequently in primary elections in small towns and Republicans run more frequently in urban areas.

The Logistic regression model for both the frequentist and the Bayesian methods showed that the urbanization index of a region is linked to the density of the political party in that region. When a Logistic model was trained with the data, it was able to predict the correct political party of a candidate at %67.5 accuracy.

These findings are generalizable to any voting district in the United States because the training data included all candidates from all voting districts. Therefore, these findings are broad and can be applied to any city with any varying measure of urbanization.

One potential call to action based on these results would be to encourage candidates from the current political party in the office of a region to still run in the primary elections. This data showed that less candidates of the party in power run in the primaries, so it would be useful to have more candidates to have a wider variety of standpoints represented.

This conclusion was drawn by merging the Primary election data set with the Urbanization index data set. The benefit of this merge was that more information was gained on the type of voting district each candidate was from that the original data set did not contain. However, the limitation of this data set that could not be accounted for was verifying that all candidates who ran were accounted for. Because this data comes from Ballotopedia, which is a trustworthy source, we made the assumption that the data was unbiased. However, it is impossible to verify all of the candidates individually, and special circumstances may have been left unaccounted for.

In the future, this model could be developed to predict on more data than just the urbanization of a district. For example, racial or gender distributions can also be taken into account per district.