Data102_Final_R2

May 6, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import seaborn as sns

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import plotly.offline as py
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.figure_factory as ff
import cufflinks as cf
cf.set_config_file(offline=True, sharing=False, theme='ggplot');

from scipy.optimize import minimize

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV

%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import pymc3 as pm
from pymc3 import glm
import arviz as az
sns.set()
```

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

import pymc3 as pm
from pymc3 import glm
import statsmodels.api as sm
import arviz
```

```
/opt/conda/lib/python3.9/site-packages/geopandas/_compat.py:111: UserWarning:

The Shapely GEOS version (3.10.3-CAPI-1.16.1) is incompatible with the GEOS
version PyGEOS was compiled with (3.10.4-CAPI-1.16.2). Conversions between both
will be slow.
```

[2]:
```
dem = pd.read_csv("dem_candidates.csv")
dems = dem[['Candidate', 'State', 'Veteran?', 'LGBTQ?', 'Elected Official?',␣
 ↪'STEM?', 'Obama Alum?', 'Self-Funder?']]
dems['win'] = dem['Primary Status'].map({'Lost': 0, 'Advanced': 1})
dems['Veteran?'] = dem['Veteran?'].map({'No': 0, 'Yes': 1})
dems['LGBTQ?'] = dem['LGBTQ?'].map({'No': 0, 'Yes': 1})
dems['Elected Official?'] = dem['Elected Official?'].map({'No': 0, 'Yes': 1})
dems['STEM?'] = dem['STEM?'].map({'No': 0, 'Yes': 1})
dems['Obama Alum?'] = dem['Obama Alum?'].map({'No': 0, 'Yes': 1})
dems['Self-Funder?'] = dem['Self-Funder?'].map({'No': 0, 'Yes': 1})
```

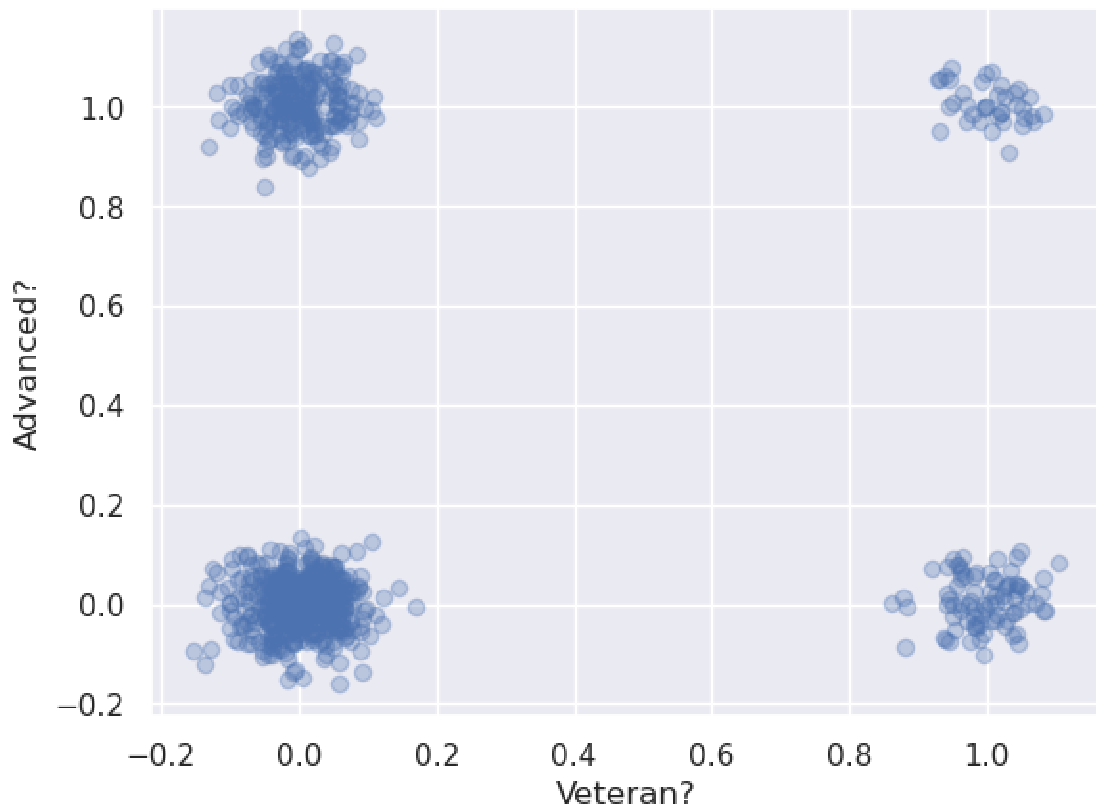
[3]:
```
dems.head()
```

[3]:
```
                   Candidate State  Veteran?  LGBTQ?  Elected Official?  STEM?  \
0  Anthony White (Alabama)    AL       1.0     0.0                0.0    0.0
1    Christopher Countryman    AL       0.0     1.0                0.0    0.0
2     Doug "New Blue" Smith    AL       1.0     0.0                0.0    0.0
3           James C. Fields    AL       1.0     0.0                1.0    0.0
4             Sue Bell Cobb    AL       0.0     0.0                1.0    0.0

   Obama Alum?  Self-Funder?  win
0          0.0             0    0
1          0.0             0    0
2          0.0             0    0
3          0.0             0    0
4          0.0             0    0
```

[4]:
```
xnoise, ynoise = np.random.normal(0, 0.05, size = 811), np.random.normal(0, 0.
↪05, size = 811)
plt.scatter(x = dems['Veteran?'] + xnoise, y = dems['win'] + ynoise, alpha = 0.
↪3)
plt.xlabel("Veteran?")
plt.ylabel("Advanced?")
f = np.mean(dems[dems['Veteran?'] == 0]['win'])
t = np.mean(dems[dems['Veteran?'] == 1]['win'])
f, t
```

[4]: (0.3353028064992615, 0.3089430894308943)

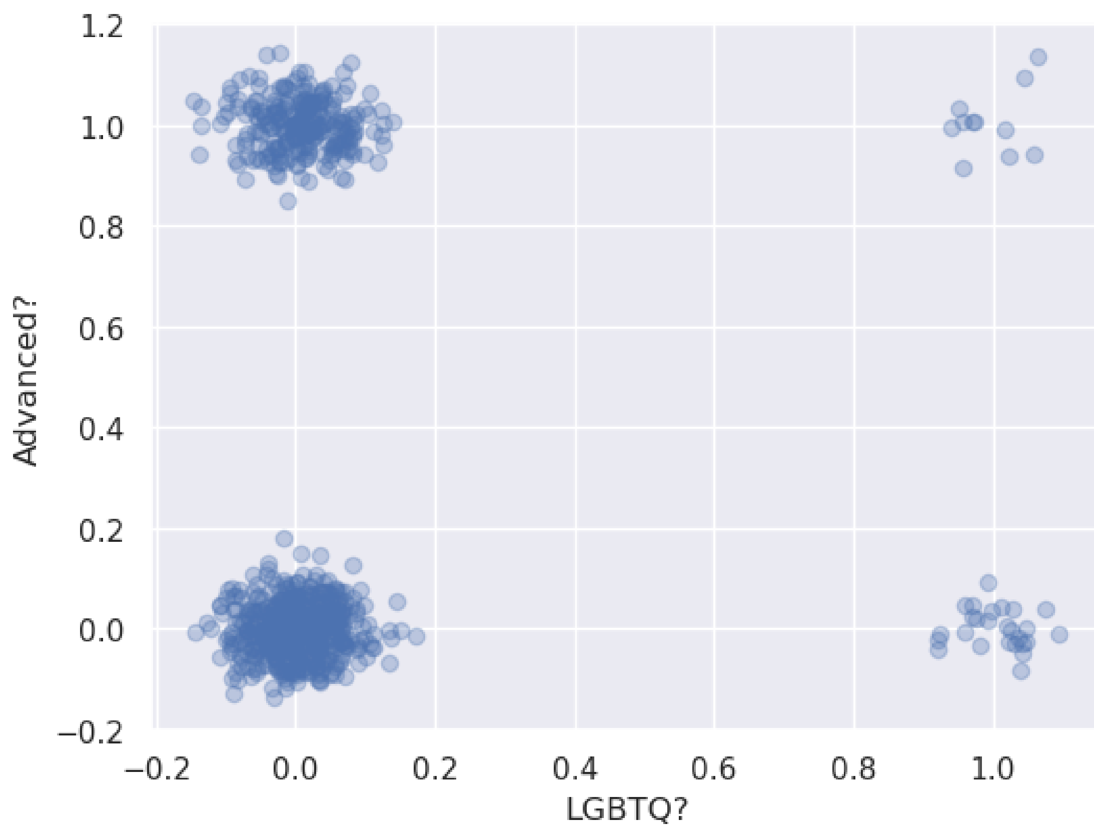


On average, being a veteran marginally reduces chances of advancing without controling for confounders, this might be worth a further analysis.

[5]:
```
xnoise, ynoise = np.random.normal(0, 0.05, size = 811), np.random.normal(0, 0.
↪05, size = 811)
plt.scatter(x = dems['LGBTQ?'] + xnoise, y = dems['win'] + ynoise, alpha = 0.3)
plt.xlabel("LGBTQ?")
plt.ylabel("Advanced?")
f = np.mean(dems[dems['LGBTQ?'] == 0]['win'])
```

```
t = np.mean(dems[dems['LGBTQ?'] == 1]['win'])
f, t
```

[5]: (0.3328964613368283, 0.2972972972972973)



On average, being an LGBTQ member marginally reduces chances of advancing without controling for confounders, this might be worth further analyzing.

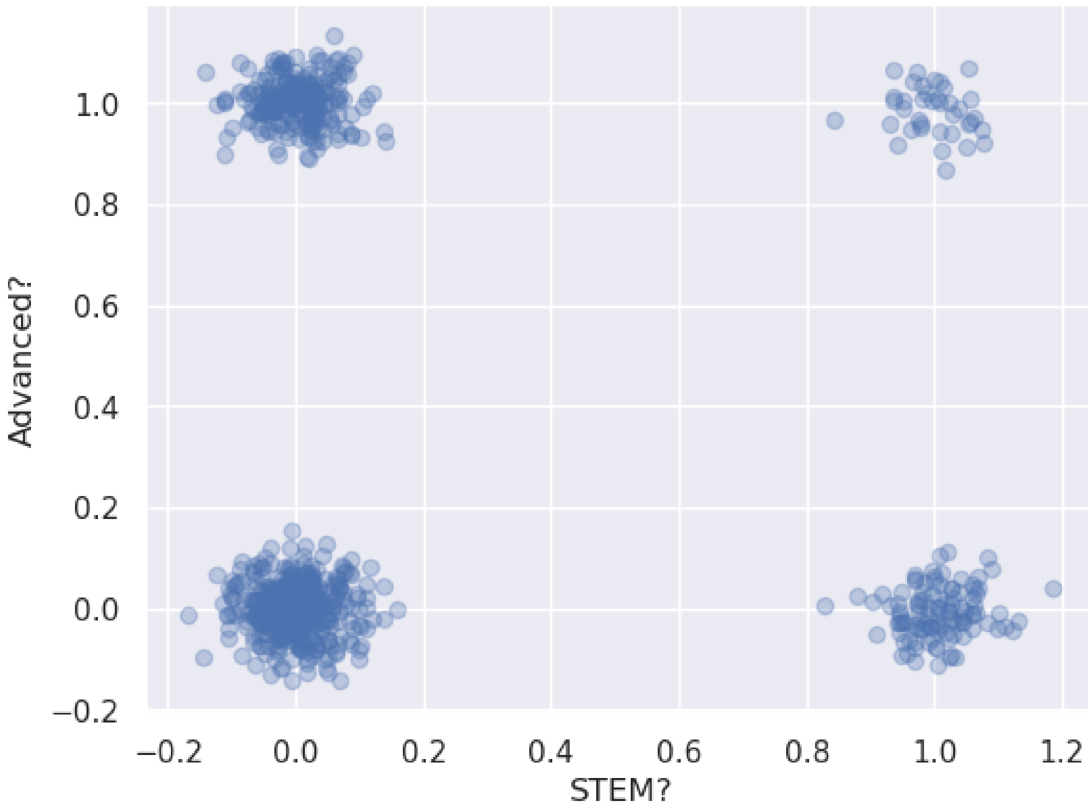
[6]:
```
xnoise, ynoise = np.random.normal(0, 0.05, size = 811), np.random.normal(0, 0.
↪05, size = 811)
plt.scatter(x = dems['Elected Official?'] + xnoise, y = dems['win'] + ynoise,␣
↪alpha = 0.3)
plt.xlabel("Elected official?")
plt.ylabel("Advanced?")
f = np.mean(dems[dems['Elected Official?'] == 0]['win'])
t = np.mean(dems[dems['Elected Official?'] == 1]['win'])
f, t
```

[6]: (0.3073463268365817, 0.45112781954887216)

On average being an Elected official increases the chances of advancement without controling for confounders, this is worth a further exploration in phase 2.

[7]:
```
xnoise, ynoise = np.random.normal(0, 0.05, size = 811), np.random.normal(0, 0.
↪05, size = 811)
plt.scatter(x = dems['STEM?'] + xnoise, y = dems['win'] + ynoise, alpha = 0.3)
plt.xlabel("STEM?")
plt.ylabel("Advanced?")
f = np.mean(dems[dems['STEM?'] == 0]['win'])
t = np.mean(dems[dems['STEM?'] == 1]['win'])
f, t
```

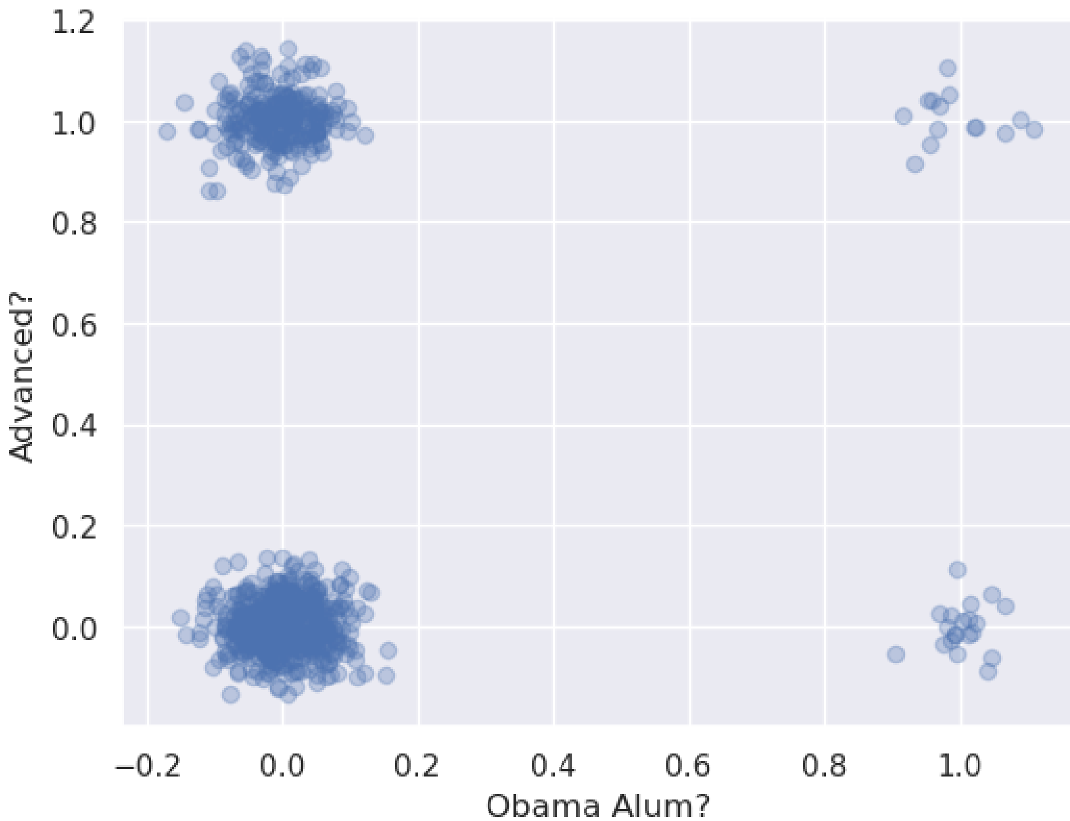
[7]:
```
(0.3470948012232416, 0.2602739726027397)
```



On average, being an STEM significantly reduces the chances of advancing without controling for confounders, this is surely worth exploring in phase 2. Maybe non-STEM degrees builds skills required for politicians.

[8]:
```
xnoise, ynoise = np.random.normal(0, 0.05, size = 811), np.random.normal(0, 0.
 ↪05, size = 811)
plt.scatter(x = dems['Obama Alum?'] + xnoise, y = dems['win'] + ynoise, alpha =␣
 ↪0.3)
plt.xlabel("Obama Alum?")
plt.ylabel("Advanced?")
f = np.mean(dems[dems['Obama Alum?'] == 0]['win'])
t = np.mean(dems[dems['Obama Alum?'] == 1]['win'])
f, t
```

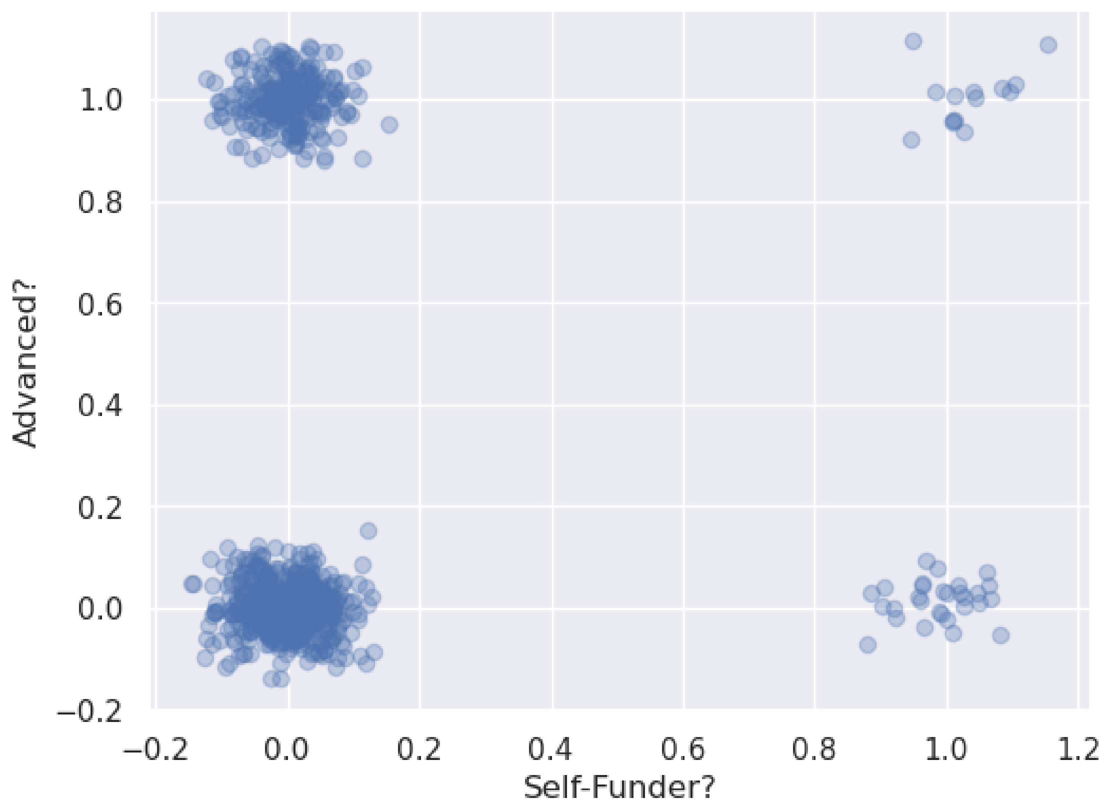
[8]:
```
(0.3234536082474227, 0.4117647058823529)
```



On average, being an Obama Alum significantly increases the chances of advancing without controling for confounders, this is surely worth exploring in phase 2. Maybe Obama alum have skills required for successful advancement.

[9]:
```
xnoise, ynoise = np.random.normal(0, 0.05, size = 811), np.random.normal(0, 0.
↪05, size = 811)
plt.scatter(x = dems['Self-Funder?'] + xnoise, y = dems['win'] + ynoise, alpha␣
↪= 0.3)
plt.xlabel("Self-Funder?")
plt.ylabel("Advanced?")
f = np.mean(dems[dems['Self-Funder?'] == 0]['win'])
t = np.mean(dems[dems['Self-Funder?'] == 1]['win'])
f, t
```

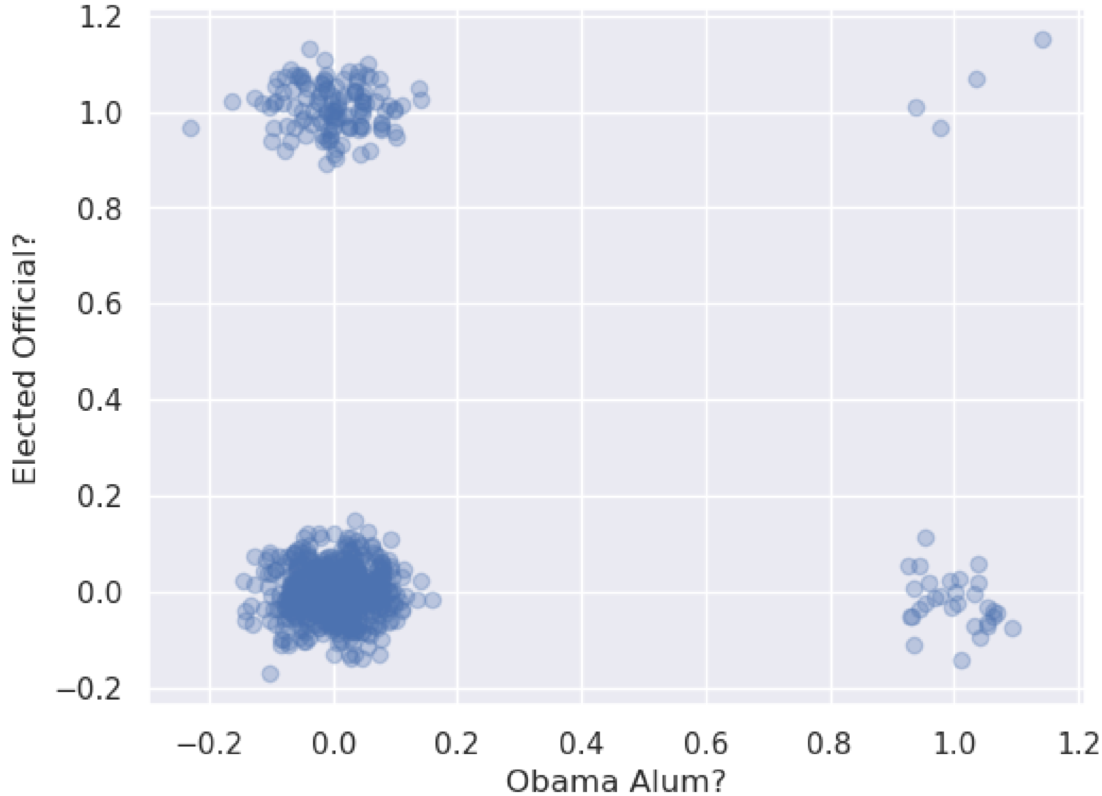
[9]:
```
(0.3268229166666667, 0.32558139534883723)
```



On average, being an self funded doesn't affect the chances of advancing without controling for confounders, this might be worth exploring in phase 2. It makes sense logically that more money implies larger campaign reach.

[10]:
```
xnoise, ynoise = np.random.normal(0, 0.05, size = 811), np.random.normal(0, 0.
↪05, size = 811)
plt.scatter(x = dems['Obama Alum?'] + xnoise, y = dems['Elected Official?'] +␣
↪ynoise, alpha = 0.3)
plt.xlabel("Obama Alum?")
plt.ylabel("Elected Official?")
f = np.mean(dems[dems['Obama Alum?'] == 0]['Elected Official?'])
t = np.mean(dems[dems['Obama Alum?'] == 1]['Elected Official?'])
f, t
```

[10]:
```
(0.16840731070496084, 0.11764705882352941)
```



On average, being an Obama Alum affects the chances of being elected, this is also worth exploring in phase 2. This would mean that the two are confounding variables in predicting a win because they affect win rates and also affect each other.

[11]:
```
rep = pd.read_csv("rep_candidates.csv", encoding='latin-1')
rep.head()
```

[11]:
```
         Candidate State           District Office Type Race Type  \
0    Mike Dunleavy    AK  Governor of Alaska    Governor   Regular
1  Michael Sheldon    AK  Governor of Alaska    Governor   Regular
2   Mead Treadwell    AK  Governor of Alaska    Governor   Regular
3     Darin Colbry    AK  Governor of Alaska    Governor   Regular
4    Thomas Gordon    AK  Governor of Alaska    Governor   Regular

  Race Primary Election Date Primary Status Primary Runoff Status  \
0                    8/21/18       Advanced                  None
1                    8/21/18           Lost                  None
2                    8/21/18           Lost                  None
3                    8/21/18           Lost                  None
4                    8/21/18           Lost                  None
```

```
  General Status  Primary %  … NRA Endorsed? Right to Life Endorsed?  \
0  On the Ballot       61.8  …           NaN                     NaN
1           None        2.2  …           NaN                     NaN
2           None       31.9  …           NaN                     NaN
3           None        0.6  …           NaN                     NaN
4           None        1.3  …           NaN                     NaN

  Susan B. Anthony Endorsed? Club for Growth Endorsed? Koch Support?  \
0                        NaN                       NaN           NaN
1                        NaN                       NaN           NaN
2                        NaN                       NaN           NaN
3                        NaN                       NaN           NaN
4                        NaN                       NaN           NaN

  House Freedom Support? Tea Party Endorsed? Main Street Endorsed?  \
0                    NaN                 NaN                   NaN
1                    NaN                 NaN                   NaN
2                    NaN                 NaN                   NaN
3                    NaN                 NaN                   NaN
4                    NaN                 NaN                   NaN

  Chamber Endorsed? No Labels Support?
0               NaN                NaN
1               NaN                NaN
2               NaN                NaN
3               NaN                NaN
4               NaN                NaN

[5 rows x 25 columns]
```

After initial graphic, we can see several potential confounding effects and realized the lack of demographic necessary information to negate this confounding effect within the republicans dataset. Hence, we chose to limit our anlaysis to the democrats dataset for this project.

- Methods – Describe what you're trying to predict, and what features you're using. Justify your choices.

We are trying to predict whether a candidate advances utilizing demographic information. We believe there is a certain priveliges that certain backgrounds allow, either through experiences or the subconscious mind of the voters and in turn the party voters. We will not be grouping by state. We noted that different sates have different demographic information but states doesn't affect our dependent variable which is a democrat advancing in this case, given that they all belong to the same party - i.e. this is a democractic only dataset.

– Describe the GLM you'll be using, justifying your choice. Describe any assumptions being made by your modeling choice. If you are using a prior for the coefficients of the Bayesian GLM, explain why you chose the prior you did.

We are using a logistic regression using independent variables of Obama alum, being elected, STEM, LGBTQ, and veteran.

– Describe the nonparametric method(s) you'll be using, justifying your choice. Describe any assumptions being made by your modeling choice.

Decision tree clustering will be used as the underlying model. Given the nature of the data being binary, a range of yes/no questions being asked will lead to potentially higher accuracy on test sets as compared to that of parametric methods.

– How will you evaluate each model's performance?

We will test for accuracy: by segmenting data into train and test sets. Furthermore the uncertainty in our GLM will be evaluated vs. the explanability of our decision tree.

- Results – Summarize and interpret the results from your models.

– Estimate any uncertainty in your GLM predictions, providing

[89]:
```
#clean the dataset utilizing only variables of value.
df = dems[['Candidate', 'Veteran?', 'LGBTQ?', 'Elected Official?', 'STEM?',␣
↪'Obama Alum?', 'win']].dropna()
```

[90]:
```
df.head()
```

[90]:
```
                  Candidate  Veteran?  LGBTQ?  Elected Official?  STEM?  \
0  Anthony White (Alabama)       1.0     0.0                0.0    0.0
1   Christopher Countryman       0.0     1.0                0.0    0.0
2    Doug "New Blue" Smith       1.0     0.0                0.0    0.0
3          James C. Fields       1.0     0.0                1.0    0.0
4            Sue Bell Cobb       0.0     0.0                1.0    0.0

   Obama Alum?  win
0          0.0    0
1          0.0    0
2          0.0    0
3          0.0    0
4          0.0    0
```

[91]:
```
#Split the data set into train and test sets
from sklearn.model_selection import train_test_split
data_tr, data_te = train_test_split(df, test_size=0.10, random_state=42)
print("Training Data Size: ", data_tr.shape)
print("Test Data Size: ", len(data_te))

# X, Y are training data
X = data_tr[['Veteran?', 'LGBTQ?', 'Elected Official?', 'STEM?', 'Obama Alum?']]
Y = data_tr['win']
```

```
Training Data Size:  (720, 7)
Test Data Size:  80
```

[15]:
```
X
```

[15]:

```
      Veteran?  LGBTQ?  Elected Official?  STEM?  Obama Alum?
434        0.0     0.0                0.0    1.0          0.0
137        0.0     0.0                0.0    0.0          0.0
72         0.0     0.0                0.0    0.0          0.0
77         0.0     0.0                0.0    0.0          0.0
518        0.0     0.0                0.0    0.0          0.0
..         …       …               …     …          …
71         0.0     0.0                0.0    0.0          0.0
106        0.0     0.0                0.0    0.0          0.0
272        0.0     0.0                0.0    0.0          0.0
441        0.0     0.0                0.0    0.0          0.0
102        0.0     0.0                1.0    0.0          0.0

[720 rows x 5 columns]
```

[18]:
```
# Fit Logistic GLM model on training set
freq_model = sm.GLM(Y, sm.add_constant(X), family=sm.families.Binomial())
freq_res = freq_model.fit()
print(freq_res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    win   No. Observations:                  720
Model:                            GLM   Df Residuals:                      714
Model Family:                Binomial   Df Model:                            5
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -448.18
Date:                Sat, 06 May 2023   Deviance:                       896.37
Time:                        18:48:20   Pearson chi2:                     720.
No. Iterations:                     4   Pseudo R-squ. (CS):            0.02003
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const                -0.7347      0.108     -6.794      0.000      -0.947
-0.523
Veteran?             -0.2154      0.223     -0.964      0.335      -0.653
0.223
LGBTQ?               -0.0696      0.377     -0.185      0.853      -0.808
0.669
Elected Official?     0.6371      0.207      3.071      0.002       0.231
1.044
```

```
STEM?                 -0.3504      0.217     -1.614      0.106      -0.776
0.075
Obama Alum?            0.1089      0.409      0.267      0.790      -0.692
0.910
================================================================================
=====
```

[59]:
```
# Fit Logistic GLM model on training set excluding LGBTQ data since it's␣
↪parameter is centered around 0.
X = data_tr[['Veteran?', 'Elected Official?', 'STEM?', 'Obama Alum?']]
freq_model = sm.GLM(Y, sm.add_constant(X), family=sm.families.Binomial())
freq_res = freq_model.fit()
print(freq_res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    win   No. Observations:                  720
Model:                            GLM   Df Residuals:                      715
Model Family:                Binomial   Df Model:                            4
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -448.20
Date:                Sat, 06 May 2023   Deviance:                       896.40
Time:                        19:14:02   Pearson chi2:                     720.
No. Iterations:                     4   Pseudo R-squ. (CS):            0.01998
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const                -0.7382      0.107     -6.930      0.000      -0.947
-0.529
Veteran?             -0.2156      0.223     -0.965      0.335      -0.654
0.222
Elected Official?     0.6369      0.207      3.071      0.002       0.230
1.043
STEM?                -0.3501      0.217     -1.613      0.107      -0.776
0.075
Obama Alum?           0.1124      0.408      0.275      0.783      -0.688
0.912
================================================================================
=====
```

[95]:
```
# use parameters to predict Y values for test set
X_test = data_te[['Veteran?', 'Elected Official?', 'STEM?', 'Obama Alum?']]
```

```
Y_prob = 1 / (1 + (1/np.exp(X_test['Veteran?']*(-0.2156) + X_test['Elected␣
↪Official?']*0.6369 + X_test['STEM?']*(-0.3501)
                     + X_test['Obama Alum?']*0.1124 - 0.7382)))
```

[97]:
```
#accuracy of parametric model on test set at 0.5 threshold
Y_test = data_te['win']
Y_pred = (Y_prob >= 0.5)
accuracy = np.mean(Y_test == Y_pred)
print(f"Accuracy on test set for parametric case: {accuracy}")
```

```
Accuracy on test set for parametric case: 0.6375
```

[101]:
```
#calculating logistic regression cross entropy loss
ce_loss = -(np.average((Y_test * np.log(Y_prob)) + (1 - Y_test) * np.
↪log(Y_prob)))
ce_loss
```

[101]: 1.0842372055740508

[102]:
```
#non parametric classifier - decision tree

X = data_tr[['Veteran?', 'LGBTQ?', 'Elected Official?', 'STEM?', 'Obama Alum?']]
X_test = data_te[['Veteran?', 'LGBTQ?', 'Elected Official?', 'STEM?', 'Obama␣
↪Alum?']]

from sklearn.tree import DecisionTreeClassifier

model_tree = DecisionTreeClassifier()
model_tree.fit(X, Y)
probs = model_tree.predict_proba(X_test)[:, 1]
y_hat_tree = (probs > 0.5).astype(np.int64)

accuracy = np.mean(Y_test == y_hat_tree)
print(f"Accuracy on test set for non-parametric case: {accuracy}")
```

```
Accuracy on test set for non-parametric case: 0.6375
```

[103]:
```
probs_null = model_tree.predict_proba(X)[:, 1]
y_null = (probs_null > 0.5).astype(np.int64)

accuracy = np.mean(Y == y_null)
print(f"Accuracy on train set for non-parametric case: {accuracy}")

#explanation for not 100% accuracy - since we have binary RV, we can only split␣
↪once on
#each feature within each vertical path.
```

```
Accuracy on train set for non-parametric case: 0.675
```

[104]:

```python
#interpretability and explanability of why model makes choices, best attempt␣
↪below.
#It appears classifier splitting down the middle for each train feature and
#rounding up the average of outcome variables to train itself, applying the␣
↪same rule to test set.

from sklearn.tree import plot_tree

plt.figure(figsize=(16, 12))
plot_tree(model_tree, fontsize=12, filled=True);
```