Investigating the Impact of Endorsements and Campaign Financing on Election Outcomes

Esha Palkar, Deborah Chang, Taylor Moore, Brendon Lin, Josh Lee, Ryan Soohoo

Berkeley
UNIVERSITY OF CALIFORNIA

Why we chose this topic:

- Personal Interest
- Real-life use cases
- Stakeholders
- Potentially useful for future elections and investigations

Datasets Used

- The datasets we used were:
- Primary Candidates 2018 - containing information on both democratic and republican primary election candidates
- Including information like candidate history, office type, election result and endorsements
- FEC Data on campaign contributions, to who, when, and how much
- **What we want to model: predicting whether or not a candidate wins based on financing metrics**

Concerns/Limitations

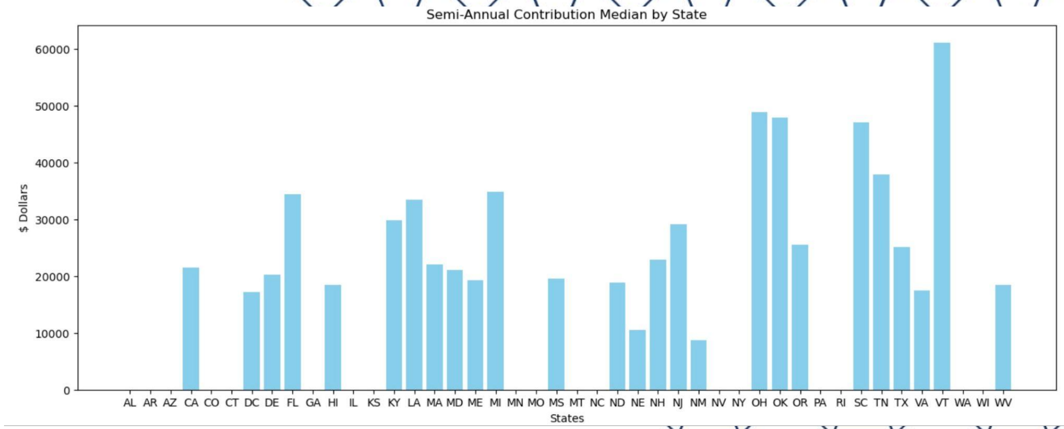
- Assumptions made during data prep with null = 0
- On privacy - both datasets are public
- Feature Selection via MDIA feature importance, for multicollinearity
- Propensity Scoring to avoid Correlation vs Causation and reduce influence of causal effects.
- Transparency in our code to avoid bias

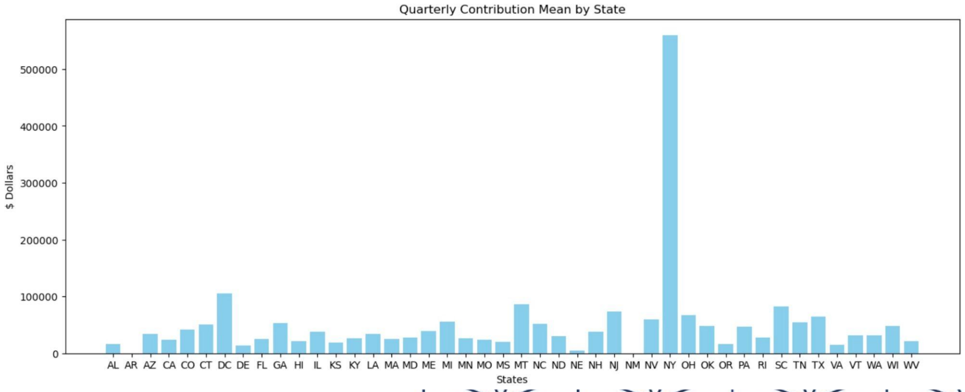
Step 1: EDA

- Unusually high sum/mean values of contributions for particular states

- Given our data, difficulty linking money to particular candidates

- Another limitation of data were lack of population data





Step 2: Data Cleaning

- We handled missing values in the datasets by filling them with zeros, based on the assumption that NaN indicated no support or contributions.
- We aggregated the contribution details by calculating the sum of the quarterly contribution and semi annual contributions columns to get a total contribution column.
- Combined the candidate datasets with the lobbyist dataset on the shared State column, with the assumption that candidates of respective states would receive state specific contributions.
- Encoded binary categorical variables (Yes, No) to 1 and 0 and applied label encoding to convert the string values to numerical values to input into the models.

Step 3: Model Building

- We started by building a baseline model, which in our case was always predicting that the candidate with the most amount of money donated would win the election
- Next, we built a logistic regression model, in which you can see the results here:

Step 3 : Building Model for Baseline and LR

```
merged_dem = pd.merge(dem_candidates, lobbyist_bundle[['Committee_Election_State', 'Total_Contribution']], left_on='State', right_on='Committee_Election_State', how='left')
merged_rep = pd.merge(rep_candidates, lobbyist_bundle[['Committee_Election_State', 'Total_Contribution']], left_on='State', right_on='Committee_Election_State', how='left')


merged_dem.drop('Committee_Election_State', axis=1, inplace=True)
merged_rep.drop('Committee_Election_State', axis=1, inplace=True)

combined_candidates = pd.concat([merged_dem, merged_rep], axis=0)


selected_features = ['Partisan Lean', 'Party Support?', 'Total_Contribution', 'Won Primary']
final_data = combined_candidates[selected_features]

final_data.fillna(0, inplace=True)

yes_no_mapping = {'Yes': 1, 'No': 0}
final_data = final_data.applymap(lambda x: yes_no_mapping.get(x, x))

X = final_data.drop(['Won Primary'], axis=1)
y = final_data['Won Primary']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 3: Baseline Model

```python
# Baseline Model
baseline_model = DummyClassifier(strategy="stratified", random_state=42)
baseline_model.fit(X_train, y_train)
y_baseline_pred = baseline_model.predict(X_test)

# Baseline model evaluation
print('Baseline Model:')
print(f'Accuracy: {accuracy_score(y_test, y_baseline_pred)}')
print(f'Confusion Matrix:\n{confusion_matrix(y_test, y_baseline_pred)}')
print(f'Classification Report:\n{classification_report(y_test, y_baseline_pred)}')
```

```
Baseline Model:
Accuracy: 0.5712906784335355
Confusion Matrix:
[[3416 1547]
 [1562  727]]
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.69      0.69      4963
           1       0.32      0.32      0.32      2289

    accuracy                           0.57      7252
   macro avg       0.50      0.50      0.50      7252
weighted avg       0.57      0.57      0.57      7252
```

Accuracy:

0.5712

Step 3: Logistic Regression Model

```
# Logistic Regression Model
log_reg_model = LogisticRegression(random_state=42)
log_reg_model.fit(X_train, y_train)
y_log_reg_pred = log_reg_model.predict(X_test)

# LR model evaluation
print('\nLogistic Regression Model:')
print(f'Accuracy: {accuracy_score(y_test, y_log_reg_pred)}')
print(f'Confusion Matrix:\n{confusion_matrix(y_test, y_log_reg_pred)}')
print(f'Classification Report:\n{classification_report(y_test, y_log_reg_pred)}')
```

Accuracy:

0.6771

```
Logistic Regression Model:
Accuracy: 0.6770546056260341
Confusion Matrix:
[[4878   85]
 [2257   32]]
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.98      0.81      4963
           1       0.27      0.01      0.03      2289

    accuracy                           0.68      7252
   macro avg       0.48      0.50      0.42      7252
weighted avg       0.55      0.68      0.56      7252
```

Step 3: RF Approach

```python
# Define the hyperparameter grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
}

clf = RandomForestClassifier()

# Create the GridSearchCV object
grid_search = GridSearchCV(clf, param_grid, cv=5, scoring='accuracy')

# Fit the GridSearchCV object to the data
grid_search.fit(X_train, y_train)

# Retrieve the best model and hyperparameters
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_

# Make predictions on the test set
y_pred = best_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Best Hyperparameters: {best_params}')
print(f'Accuracy: {accuracy}')
```

```
Best Hyperparameters: {'max_depth': 10, 'min_samples_split': 5, 'n_estimators': 100}
Accuracy: 0.7361963190184049
```

Random Forest model w/ Cross-Validation using GridSearchCV

```python
[17] # Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_classifier.fit(X, y)

# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)
```

```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Display the results
print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{class_report}')
```

```
Accuracy: 0.9079754601226994
```

Random Forest model test predictions

Democratic model

```
selected_features = [
    'Total_Contribution',
    'Partisan Lean',
    'Self-Funder?',
    'Obama Alum?',
    'Party Support?',
    'Emily Endorsed?',
    'Guns Sense Candidate?',
    'Biden Endorsed?',
    'Warren Endorsed? ',
    'Sanders Endorsed?',
    'Our Revolution Endorsed?',
    'Justice Dems Endorsed?',
    'PCCC Endorsed?',
    'Indivisible Endorsed?',
    'WFP Endorsed?',
    'VoteVets Endorsed?',
    'No Labels Support?',
    'Won Primary'  # Target variable
]
```

Republican model

```
rep_features = [
    'Total_Contribution',
    'Rep Party Support?',
    'Trump Endorsed?',
    'Bannon Endorsed?',
    'Great America Endorsed?',
    'NRA Endorsed?',
    'Right to Life Endorsed?',
    'Susan B. Anthony Endorsed?',
    'Club for Growth Endorsed?',
    'Koch Support?',
    'House Freedom Support?',
    'Tea Party Endorsed?',
    'Main Street Endorsed?',
    'Chamber Endorsed?',
    'No Labels Support?',
    'Won Primary'  # Target variable
]
```

RF Results
(Democratic candidates)

- RF w/ GridSearchCV:
  - Best Hyperparameters: max_depth=10, min_samples_split=5, n_estimators=100
  - **Accuracy: 0.73619319018404**
- RF w/o CV: *overfitting concern
  - **Accuracy: 0.90797546011226994**
  - Precision, recall
  - Confusion Matrix

	precision	recall	f1-score
0	0.89	0.99	0.94
1	0.97	0.73	0.84

```
Confusion Matrix:
[[110   1]
 [ 14  38]]
```

RF Results
(Republican candidates)

- RF w/ GridSearchCV:
    - Best Hyperparameters: max_depth=None, min_samples_split=2, n_estimators=50
    - **Accuracy: 0.709677419**
- RF w/o CV:
    - **Accuracy: 0.709677419**
    - Precision, recall
    - Confusion Matrix

	precision	recall	f1-score
0	0.76	0.87	0.81
1	0.48	0.30	0.37

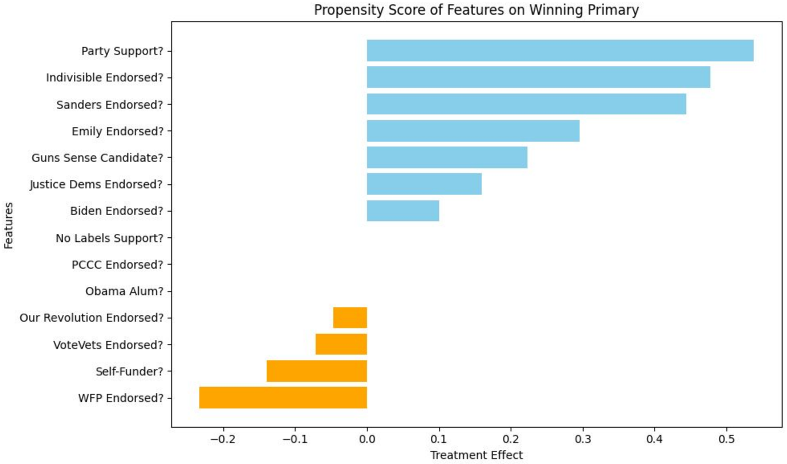
```
Confusion Matrix:
[[97 14]
 [31 13]]
```

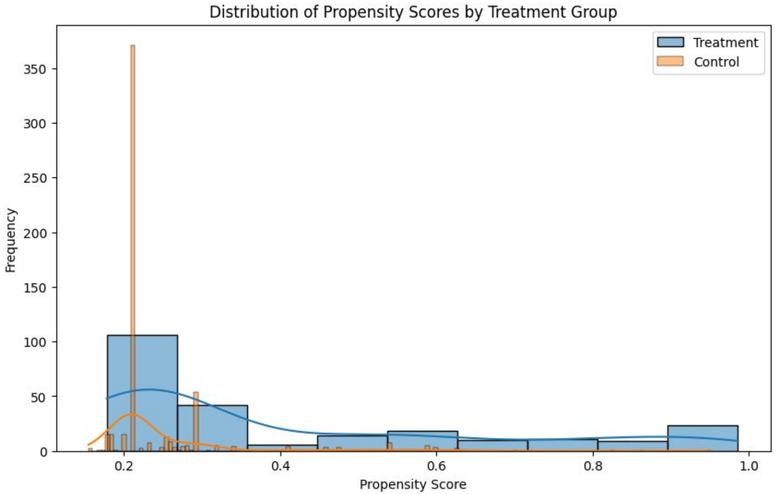
Propensity Scoring

Objective: Identify the quality of features.

Used: Logistic Regression propensity

Results: Key Features to include in model





- Biden Endorsed

  0.4353 coef

  0.214 std error

- R-squared = 0.014

  (easy to overfitting)

- No Labels Support

  -0.4390 coef

  0.334 p-value

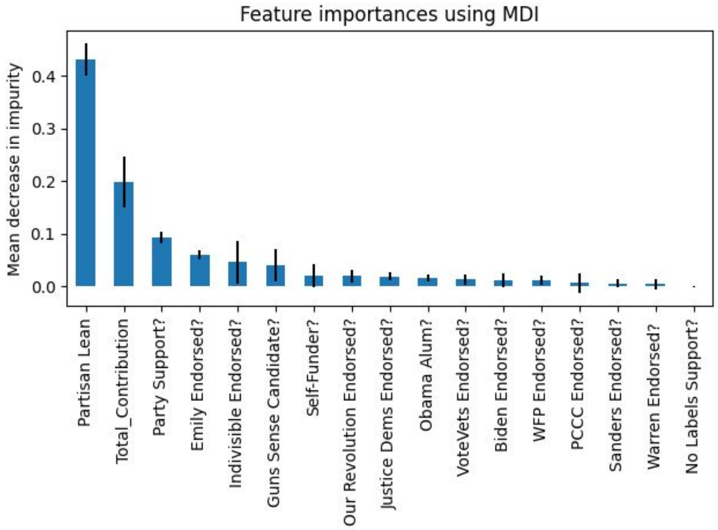
```
                            WLS Regression Results
==============================================================================
Dep. Variable:            Won Primary   R-squared:                       0.014
Model:                            WLS   Adj. R-squared:                 -0.004
Method:                 Least Squares   F-statistic:                    0.7877
Date:                Sun, 10 Dec 2023   Prob (F-statistic):              0.683
Time:                        21:14:19   Log-Likelihood:                -644.34
No. Observations:                 811   AIC:                             1319.
Df Residuals:                     796   BIC:                             1389.
Df Model:                          14
Covariance Type:            nonrobust
==================================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                         0.4916      0.022     21.953      0.000       0.448       0.536
Self-Funder?                  0.0197      0.077      0.254      0.799      -0.132       0.172
Obama Alum?                   0.0073      0.091      0.080      0.937      -0.172       0.187
Party Support?                0.1392      0.107      1.304      0.193      -0.070       0.349
Emily Endorsed?              -0.0397      0.076     -0.520      0.603      -0.190       0.110
Guns Sense Candidate?        -0.0120      0.042     -0.285      0.776      -0.095       0.071
Biden Endorsed?               0.4353      0.214      2.035      0.042       0.015       0.855
Sanders Endorsed?             0.0691      0.178      0.388      0.698      -0.281       0.419
Our Revolution Endorsed?     -0.0150      0.066     -0.228      0.820      -0.144       0.114
Justice Dems Endorsed?        0.0030      0.082      0.036      0.971      -0.158       0.164
PCCC Endorsed?                0.0624      0.159      0.393      0.694      -0.249       0.374
Indivisible Endorsed?         0.0605      0.078      0.772      0.440      -0.093       0.214
WFP Endorsed?                 0.0364      0.100      0.364      0.716      -0.160       0.233
VoteVets Endorsed?           -0.1032      0.098     -1.050      0.294      -0.296       0.090
No Labels Support?           -0.4390      0.454     -0.967      0.334      -1.330       0.452
==============================================================================
Omnibus:                      127.720   Durbin-Watson:                   0.134
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              146.084
Skew:                           0.988   Prob(JB):                     1.90e-32
Kurtosis:                       2.355   Cond. No.                         27.3
==============================================================================
```

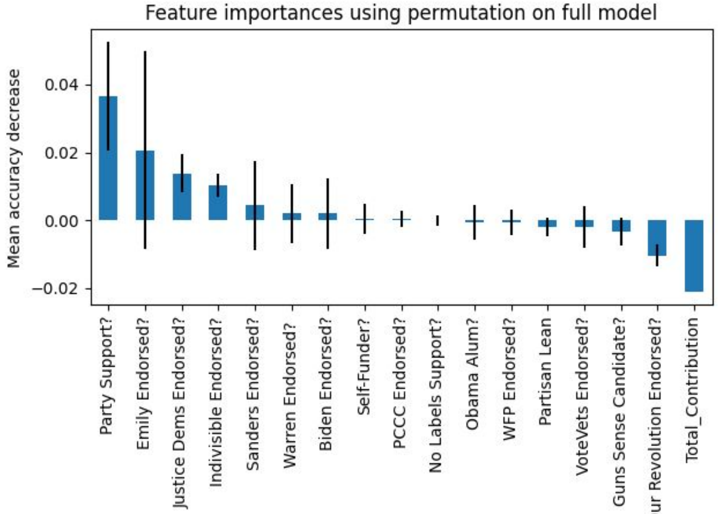
Feature Selection

Objective: Reduce Multicollinearity and Overfitting

Used: MDI (Mean Decrease in Impurity)

Results: Reduces model complexity, features to include in model





Step 3: Model Training

Objective: Create model for election Prediction using new found features

Used: MLP binary classification with sigmoid activation

Hyperparameters Tested: # of layers, Neurons per layer, activation function, epochs, batch size, k-fold

Results: Accuracy Matrix results based on the model of found features.

```
top_features = [
    'Party Support?',
    'Indivisible Endorsed?',
    'Our Revolution Endorsed?',
    'Sanders Endorsed?',
    'Emily Endorsed?',
    'Guns Sense Candidate?',
    'Justice Dems Endorsed?'
]
Best Parameters:
{'neurons_per_layer': 10,
'num_layers': 1, 'activation':
'relu'}
```

```
# Parameter grid
neuron_numbers = [10, 20, 30]#neurons per layer
layer_numbers = [1, 2, 3, 4, 5] #hidden layers
activations = ['relu', 'tanh'] #activation func

best_accuracy = 0
best_params = {}

#goes through param grid
for neurons in neuron_numbers:#neurons
    for _layers in layer_numbers:#layers
        for activation in activations:#activation
        #train_evaluate_model(dem_candidates, _features, kfold=True, k=
            cm, cr, accuracy = my_model( #calls model WITH KFOLD
                dem_candidates, features, kfold=True,
                neurons=neurons, layers=_layers, activation=activation,
                epochs=100, batch=10, k=5
```

ALL Features

BEST Features

K-Fold

```
6/6 [==============================] - 0s 3ms/step
6/6 [==============================] - 0s 3ms/step
6/6 [==============================] - 0s 3ms/step
6/6 [==============================] - 0s 2ms/step
6/6 [==============================] - 0s 6ms/step
[[104.4   9.8]
 [ 28.2  19.6]]
[{'0': {'precision': 0.7874015748031497, 'recall':
0.7654321074485779
------------------------------------------------------
6/6 [==============================] - 0s 3ms/step
6/6 [==============================] - 0s 3ms/step
6/6 [==============================] - 1s 5ms/step
6/6 [==============================] - 0s 2ms/step
6/6 [==============================] - 0s 2ms/step
[[108.4   5.8]
 [ 33.4  14.4]]
[{'0': {'precision': 0.7737226277372263, 'recall':
0.7580246806144715
```

No K-Fold

```
8/8 [==============================] - 0s 2ms/step
[[149  18]
 [ 43  34]]
              precision    recall  f1-score   support

           0       0.78      0.89      0.83       167
           1       0.65      0.44      0.53        77

    accuracy                           0.75       244
   macro avg       0.71      0.67      0.68       244
weighted avg       0.74      0.75      0.73       244

0.75
------------------------------------------------------
8/8 [==============================] - 0s 2ms/step
[[159   8]
 [ 48  29]]
              precision    recall  f1-score   support

           0       0.77      0.95      0.85       167
           1       0.78      0.38      0.51        77

    accuracy                           0.77       244
   macro avg       0.78      0.66      0.68       244
weighted avg       0.77      0.77      0.74       244

0.7704917788505554
```

How do all of the models we've built so far compare?

- Baseline

Baseline Model:
Accuracy: 0.5712906784335355
Confusion Matrix:
[[3416 1547]
 [1562 727]]
Classification Report:

	precision	recall	f1-score	support
0	0.69	0.69	0.69	4963
1	0.32	0.32	0.32	2289
accuracy			0.57	7252
macro avg	0.50	0.50	0.50	7252
weighted avg	0.57	0.57	0.57	7252

- Logistic Regression

Logistic Regression Model:
Accuracy: 0.6770546056260341
Confusion Matrix:
[[4878   85]
 [2257   32]]
Classification Report:

	precision	recall	f1-score	support
0	0.68	0.98	0.81	4963
1	0.27	0.01	0.03	2289
accuracy			0.68	7252
macro avg	0.48	0.50	0.42	7252
weighted avg	0.55	0.68	0.56	7252

- Random Forest with CV
  - Accuracy: 0.73619319018404 (democratic)

	precision	recall	f1-score
0	0.89	0.99	0.94
1	0.97	0.73	0.84

Confusion Matrix:
[[110   1]
 [14   38]]

  - Accuracy: 0.709677419

	precision	recall	f1-score
0	0.76	0.87	0.81
1	0.48	0.30	0.37

Confusion Matrix:
[[97 14]
 [31 13]]

- Neural Network

[[159   8]
 [48   29]]

	precision	recall	f1-score	support
0	0.77	0.95	0.85	167
1	0.78	0.38	0.51	77
accuracy			0.77	244
macro avg	0.78	0.66	0.68	244
weighted avg	0.77	0.77	0.74	244

0.7704917788505554

What would we do if we had more time to work on future iterations?

- Based on our current results we would likely try more hyper parameters in cross validation to improve accuracy
- Bigger diversity of models, try gradient boosting, etc.
- Ask different questions

Thank you very much for listening!