

# Media bias

Michael Kuhn

Herman Schaumburg

May 9, 2020

## Goals

Primary: Find objective way to score bias in 36 selected news organizations.

Secondary: Estimate the political bias of a given tweet.

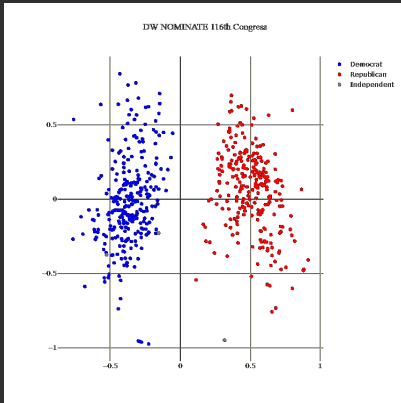
Secondary: Heroku deployment.

# Objective Media Bias Rankings

[www.mediabias.herokuapp.com](http://www.mediabias.herokuapp.com)

Left	Left-Leaning	Moderate	Right-Leaning	Right
      	       	      	       	      

# DW-NOMINATE scores



## Methodology

1. Gathered tweets from Democratic lawmakers than Republicans (38,491 tweets vs 15,826 tweets) from Alex Litel github repository into SQL database.

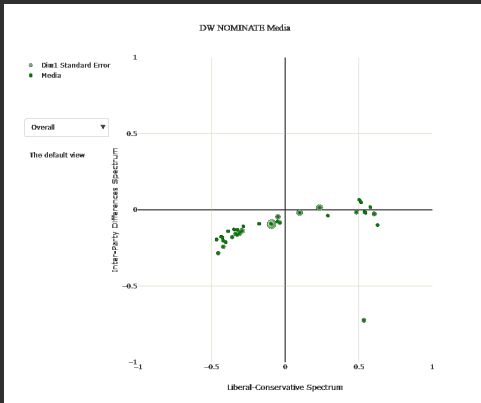
<https://github.com/alexlitel/congresstweets>

2. DW-NOMINATE scores of congress people from voteview

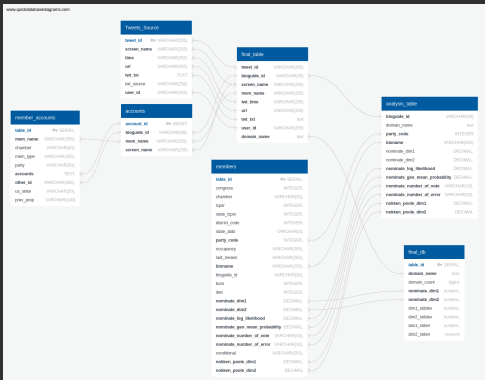
<https://voteview.com/data>

3. Queried tweets of congresspeople for the media domains they tweet from and used the congressperson DW-NOMINATE score to score the media domain.

# Media domain scores



## SQL Schema

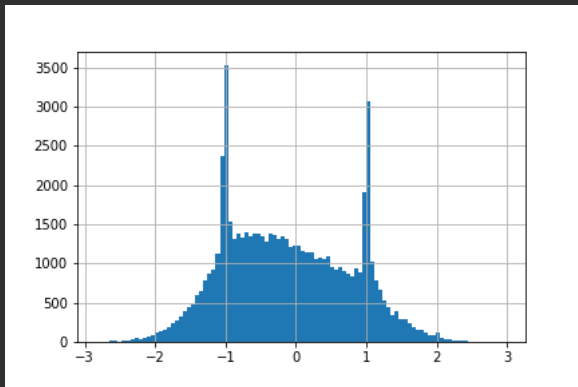


## ML Model

1. Use nltk to remove stopwords from tweets.
2. Tweets are vectorized using several schemes.
3. Best performing scheme is selected for fine tuning.
4. End result is Stochastic Gradient Decent Model with X score.
5. Empirical CDF of errors in large test used to estimate probability of making an error.



Test for normality failed (couldn't use prediction interval)



## Empirical CDF

# Passing user input to flask app

```
1 <div class="col">
2   <h2 class="headers">Party Predictor</h2>
3   <p>
4     Type in a tweet to get a prediction of tweeters political party.
5   </p>
6   <form method="POST">
7     <input name="text">
8     <input type="submit">
9   </form>
10   {{party_prediction}}
11 </div>
```

# Heroku deployment features

- About page - interactive plots.
- ML models page - text box to predict party of tweeter.
- Search Media Scores Page - text box to search SQL data base for media domain.

# Heroku difficulty

Flask app, `app.py`, is in a separate directory from heroku config files. This made it difficult to load the scikit learn models into the app. The issue was resolved using by printing information to the page on the current working directory using the `os` library if the models failed to run:

```
1  if loading_error:
2      def list_tostring(input_list):
3          return ' '.join(input_list)
4      party_result = os.getcwd()+' loaded '+str(num_loaded)+' '+list_tostring(os.listdir())
5  else:
```

# SQL Query

```
1 select domain_name, domain_count, nominate_dim1, nominate_dim2, dim1_stddev,  
2 dim2_stddev, round((dim1_stddev/sqrt(domain_count-1))::numeric,3) as dim1_stderr,  
3 round((dim2_stddev/sqrt(domain_count-1))::numeric,3) as dim2_stderr  
4 into final_db  
5 from tenthousand_db  
6 where domain_count>1  
7 group by domain_name, domain_count, nominate_dim1, nominate_dim2, dim1_stddev, dim2_stddev  
8 order by domain_count desc
```

# jsonToCSV

```
1 def jsonToCSV(json_path, csv_path):
2     import json
3     import csv
4     merged_csv = []
5     with open(json_path, encoding='utf-8') as ref:
6         data = json.load(ref)
7         headers = list(data[0].keys())
8         csv_row = []
9         for item in data:
10             item_ls = []
11             for col in headers:
12                 try:
13                     item_ls.append(item[col])
14                 except:
15                     item_ls.append(None)
16             csv_row.append(item_ls)
17         merged_csv += csv_row
18
19     with open(csv_path, 'w', newline='', encoding='utf-8') as csvfile:
20         spamwriter = csv.writer(csvfile)
21         for row in merged_csv:
22             spamwriter.writerow(row)
```