Media bias

Michael Kuhn

Herman Schaumburg

May 9, 2020

Goals

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

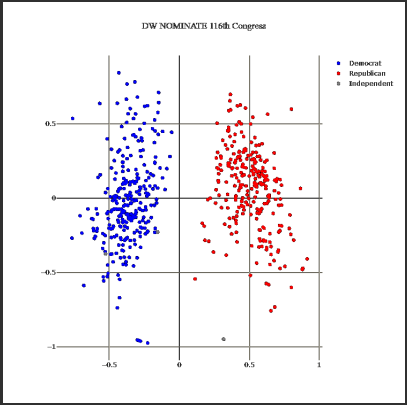
Secondary: Estimate the political bias of a given tweet.

Secondary: Heroku deployment.

Objective Media Bias Rankings

www.mediabias.herokuapp.com

DW-NOMINATE scores



Methodology

1. Gathered tweets from Democratic and Republican law makers (1,266,104) from Alex Litel github repository into SQL database. From media orgs 54,833.

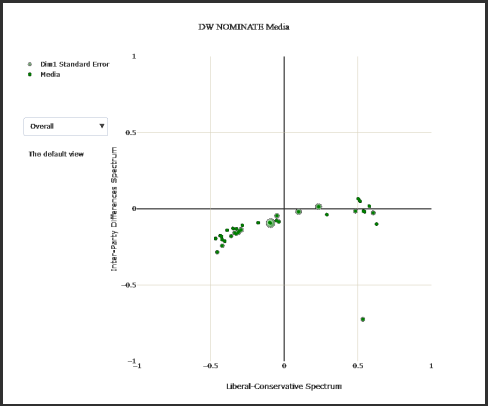
   https://github.com/alexlitel/congresstweets

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

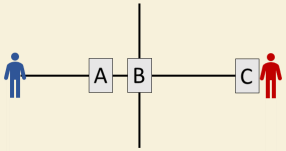


Method diagram

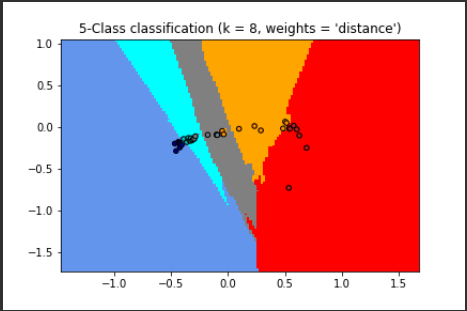
$$A = \frac{(1,0) + (-1,0) + (-1,0)}{3} = \frac{(-1,0)}{3} = (\frac{-1}{3}, 0)$$

$$B = \frac{(1,0) + (1,0) + (-1,0) + (-1,0)}{4} = \frac{(0,0)}{4} = (0,0)$$

$$C = \frac{(1,0)}{1} = \frac{(1,0)}{1} = (1,0)$$



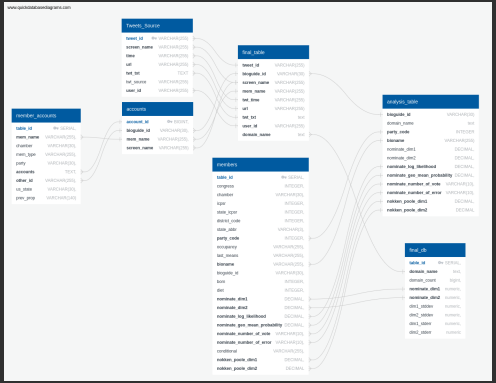
Bias Ranking



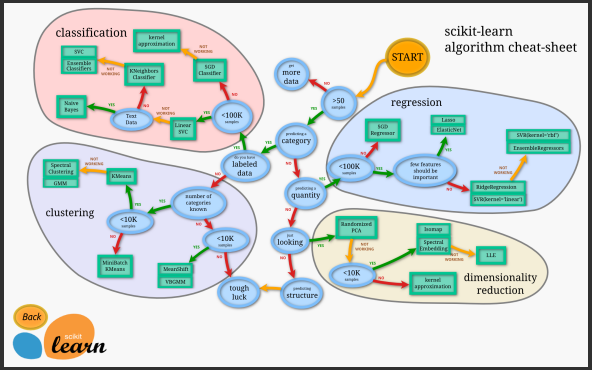
K-Nearest neighbors to classify media domains:

Left, Left-leaning, Moderate, Right-leaning, Right

SQL Schema


www.quickdatabasediagrams.com
tweets_source
tweet_id PK VARCHAR(255)
screen_name VARCHAR(255)
time VARCHAR(255)
url VARCHAR(255)
twt_txt TEXT
twt_source VARCHAR(255)
user_id VARCHAR(255)
accounts
account_id PK SERIAL
blogpulse_id VARCHAR(255)
mom_name VARCHAR(255)
screen_name VARCHAR(255)
final_table
tweet_id VARCHAR(255)
blogpulse_id VARCHAR(36)
screen_name VARCHAR(255)
mom_name VARCHAR(255)
twt_time VARCHAR(255)
url VARCHAR(255)
twt_txt text
user_id VARCHAR(255)
domain_name text
member_accounts
table_id PK SERIAL
mem_name VARCHAR(255)
chamber VARCHAR(255)
mem_type VARCHAR(255)
party VARCHAR(255)
comments TEXT
other_id VARCHAR(255)
cc_state VARCHAR(2)
prev_prop VARCHAR(240)
members
table_id PK SERIAL
congress INTEGER
chamber VARCHAR(30)
type INTEGER
state_icpsr INTEGER
district_code INTEGER
state_abbr VARCHAR(10)
party_code INTEGER
occupancy VARCHAR(255)
last_means VARCHAR(255)
bioname VARCHAR(255)
bioguide_id VARCHAR(255)
born INTEGER
died INTEGER
nominate_dim1 DECIMAL
nominate_dim2 DECIMAL
nominate_log_likelihood DECIMAL
nominate_geo_mean_probability DECIMAL
nominate_number_of_vote VARCHAR(10)
nominate_number_of_error VARCHAR(10)
conditional VARCHAR(255)
noklean_posle_dim1 DECIMAL
noklean_posle_dim2 DECIMAL
analysis_table
blaguide_id VARCHAR(99)
domain_name text
party_code INTEGER
bioname VARCHAR(255)
nominate_dim1 DECIMAL
nominate_dim2 DECIMAL
nominate_log_likelihood DECIMAL
nominate_number_of_vote VARCHAR(10)
nominate_number_of_error VARCHAR(10)
noklean_posle_dim1 DECIMAL
noklean_posle_dim2 DECIMAL
final_db
table_id PK SERIAL
domain_name text
domain_count bigint
nominate_dim1 numeric
nominate_dim2 numeric
dim1_tilden numeric
dim2_tilden numeric
dim1_obiret numeric
dim2_obiret numeric

ML Model Selection



ML Model

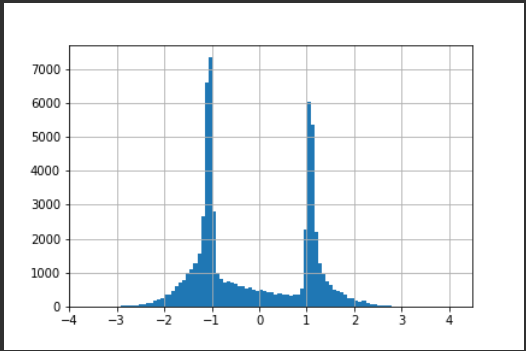
1. Use nltk to remove stopwords from tweets.
2. Tweets are vectorized using several schemes (used ~15K tweets).
3. Best performing scheme is selected for fine tuning.
4. End result is Stochastic Gradient Decent Model with 0.8482 score on large selection of tweets (~64K tweets).
5. Empirical CDF of errors in large test used to estimate probability of making an error.

Vectorized Tweets

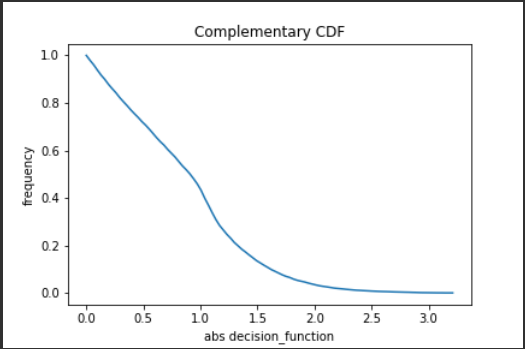
```
  (0, 599005) 0.13724572483536815
  (0, 598951) 0.10656565688654346
  (0, 597099) 0.07127021892848737
  (0, 596894) 0.0696688776808445
  (0, 596846) 0.0835728977646812
  (0, 596744) 0.07270567427770662
  (0, 581647) 0.08857473689547032
  (0, 581646) 0.08857473689547032
  (0, 568348) 0.21930758305074324
  (0, 568331) 0.1444337225102922
  (0, 548752) 0.08863406696726873
```

⋮

Test for normality failed - p-value near zero (couldn’t use prediction interval)



Empirical Complementary CDF



Passing user input to flask app

```
<div class="col">
    <h2 class="headers">Party Predictor</h2>
    <p>
        Type in a tweet to get a prediction of tweeters political party.
    </p>
    <form method="POST">
        <input name="text">
        <input type="submit">
    </form>
    {{party_prediction}}
</div>
```

Second model attempted

Attempted to use Stochastic Gradient Decent Regression Model to get dw nominate score of tweeter.

Training $R^2 = 0.95$ and test $R^2 = 0.5$.

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:

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

SQL Query

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

jsonToCSV

```python
def jsonToCSV(json_path, csv_path):
    import json
    import csv
    merged_csv = []
    with open(json_path, encoding='utf-8') as ref:
        data = json.load(ref)
        headers = list(data[0].keys())
        csv_row = []
        for item in data:
            item_ls = []
            for col in headers:
                try:
                    item_ls.append(item[col])
                except:
                    item_ls.append(None)
            csv_row.append(item_ls)
        merged_csv += csv_row

    with open(csv_path, 'w', newline='', encoding='utf-8') as csvfile:
        spamwriter = csv.writer(csvfile)
        for row in merged_csv:
            spamwriter.writerow(row)
```

Questions