Predicting the Origin of a Tweet

Davis Busteed
ECON 484

Question

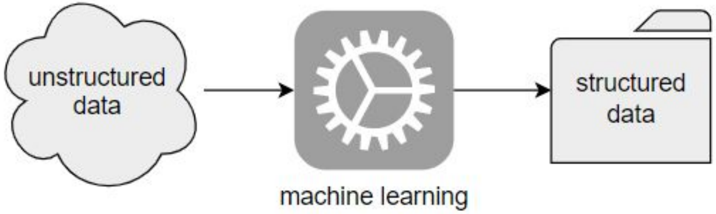
Is it possible to accurately determine the country of origin of a tweet?

Can we classify tweets by just looking at the text? (without using metadata)

Why? Not all Twitter users have location services enabled, which limits the research that can be done with Twitter data

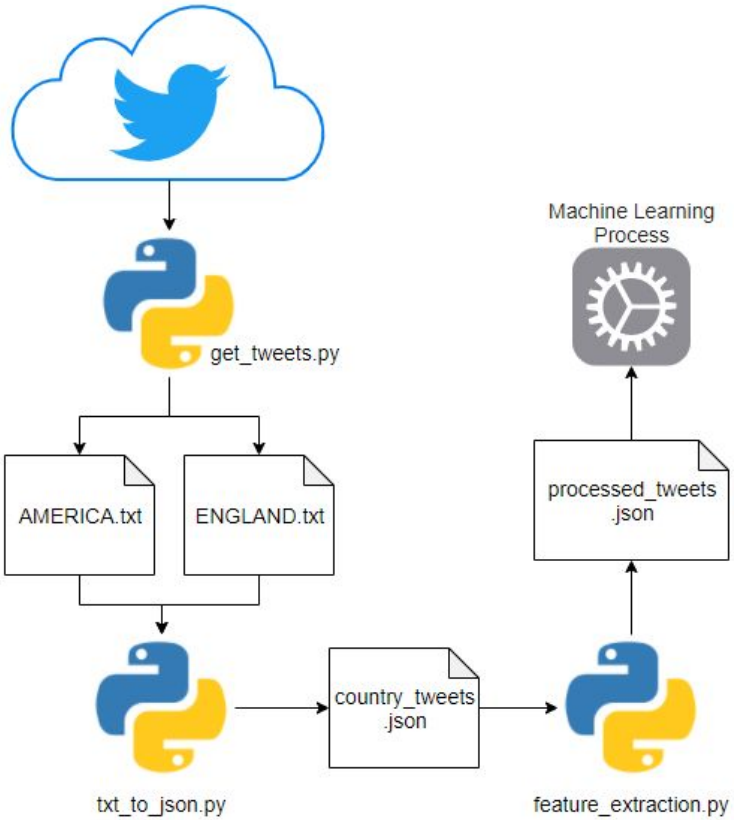
Goal

Create a predictive model that can be used to add structure (country labels) to otherwise unstructured data (tweets)



Data

- Used Python + Tweepy (Twitter API)
  - Filtered tweets using latitude/longitude coordinates
- Combined raw text files into JSON format
  - Used RegEx to normalize tweets
- Extracted features from tweets
  - Number of hashtags, number of user mentions, etc
- Final dataset: n = 40000
  - 20000 American tweets
  - 20000 English tweets



Data Summary

Column Names	Type	Data Type	Count (Obs)	Mean	S.D.
country	Outcome, Y	string	40000	-	-
at_count	Predictor, X	int	40000	0.843875	1.255524
hash_count	Predictor, X	int	40000	0.23225	0.77645
emoji_count	Predictor, X	int	40000	0.48205	1.485726
has_url	Predictor, X	int / bool	40000	0.368525	0.482411
text_len	Predictor, X	int	40000	57.86865	35.468604
word_count	Predictor, X	int	40000	11.969275	6.779608
sentiment	Predictor, X	float	40000	0.12939	0.408243
raw_text	Intermediary	string	40000	-	-
clean_text	Intermediary	string	40000	-	-

Machine Learning Methods

- Preliminary data discovery left me with no direction/hints about which algorithm(s) would perform best
- Used different approaches and compared results:
    a. Feature extraction w/ common classification algorithms
        - Trees, LogReg, SVM, Neural Network, etc.

    b. Term frequency–inverse document frequency

    c. MALLET

Feature Extraction

- Used previously extracted features as the predictors
- Carried out steps 2 thru 8 of the machine learning process
  - Split data into train and test sets
  - Tested several algorithms
  - Used GridSearchCV to choose tuning parameters
  - Compared out-of-sample accuracy for all models

	at_count	hash_count	emoji_count	has_url	text_len	word_count	sentiment
0	1	0	0	0	24	6	0.0000
1	0	0	2	0	7	2	0.4588
2	2	0	2	0	30	8	0.6391
3	1	0	0	0	21	5	0.0000
4	0	0	0	0	28	5	0.0000
...	...	...	...	...	...	...	...
39995	1	0	0	1	36	9	-0.5423
39996	0	0	2	0	56	13	0.8286
39997	0	3	3	0	55	10	0.0000
39998	0	1	1	0	75	17	0.0000
39999	2	0	0	0	30	6	0.0000

40000 rows × 7 columns

TF-IDF

- Term frequency–inverse document frequency
    - Used to show how “important” a word is in a given document

	00	000	007	009	00in	00pm	01	01273660506	0141z	0142z	...	*London*	BABY	Christmas	DEAD	Night	ZEDS	be	must	nice	with
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2 rows × 30078 columns

- Each word becomes a column in the matrix, so I used PCA to reduce p≈30000 to p=100
- Used this matrix of predictors with previously tested models

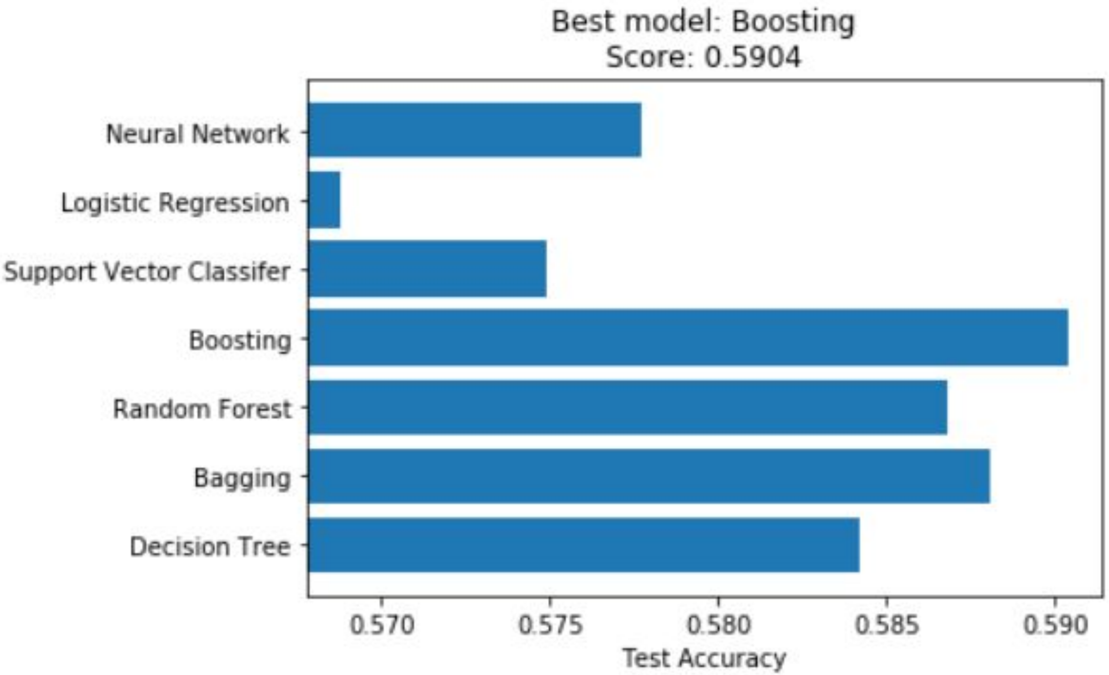
MALLET

- **MA**chine **L**earning for **L**anguag**E** **T**oolkit
  - http://mallet.cs.umass.edu/
- Took a step away from Scikit-learn and experimented with ML tools specifically made for classifying language
- Quickly reorganized data to be MALLET compatible
- Followed similar ML process
  - Split data in train and test sets
  - Tested several algorithms
    - Maximum Entropy Classifier
    - Naive Bayes Classifier
  - Used cross validation to choose tuning parameters
  - Compared out-of-sample accuracy for all models

```
|--data/
     |--AMERICA/
     |    |--0.txt
     |    |--1.txt
     |    ...
     |    |--19998.txt
     |    |--19999.txt
     |
     |--ENGLAND/
          |--0.txt
          |--1.txt
          ...
          |--19998.txt
          |--19999.txt
```

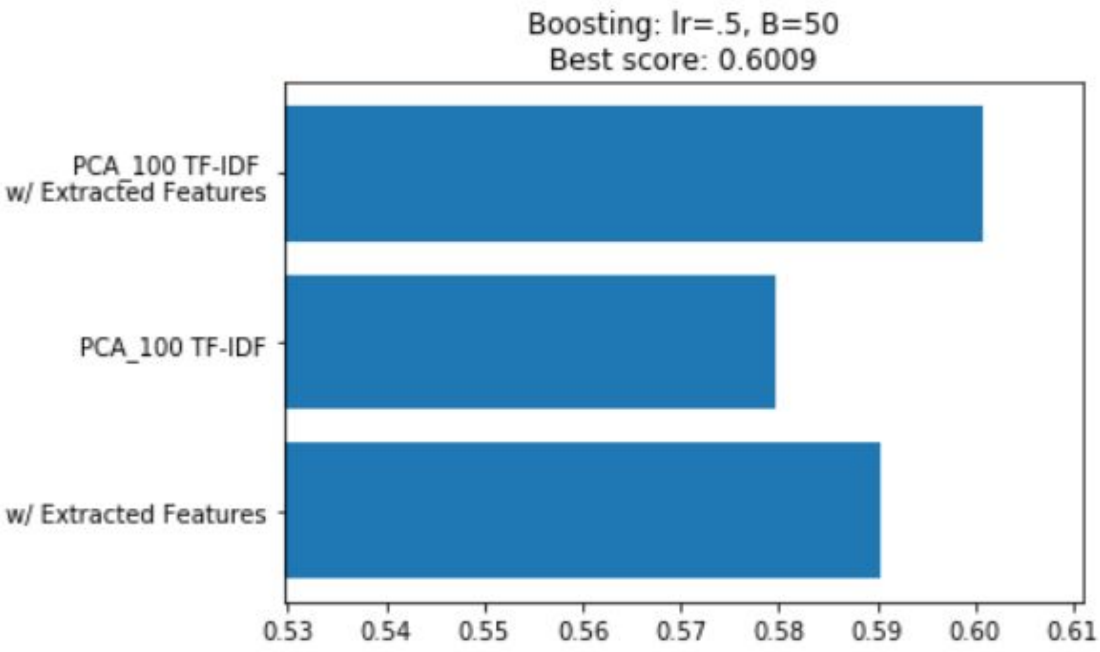
Code Walkthrough

Results — only extracted features

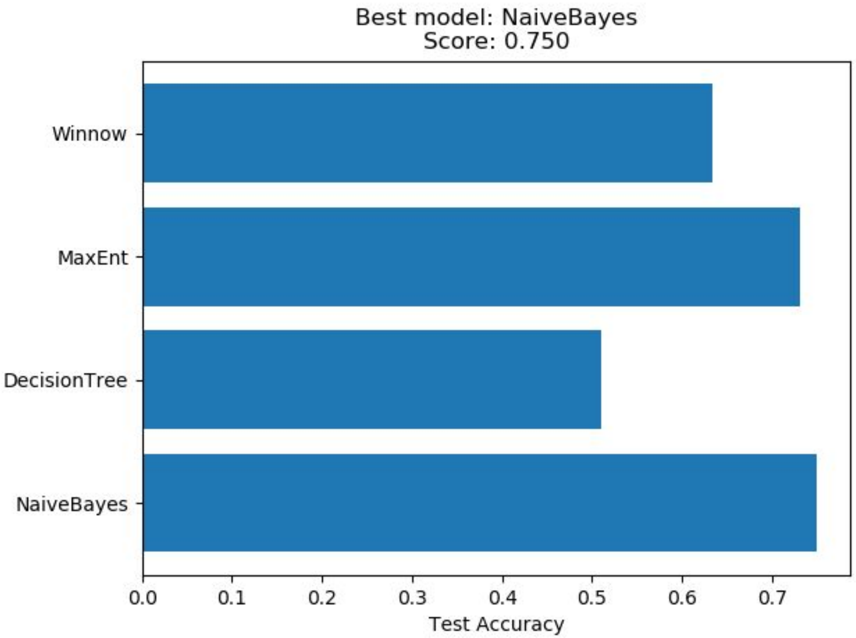


Gradient Boosting				
Test accuracy: 0.5904		PREDICTED		
		AMERICA	ENGLAND	TOTAL
TRUE	AMERICA	2971	2029	5000
	ENGLAND	2067	2933	5000
	TOTAL	5038	4962	10000

Results — extracted features w/ TF-IDF



Results — MALLET



NaïveBayes Classifier (MALLET)					
Test accuracy: 0.7516		PREDICTED			
		AMERICA	ENGLAND	TOTAL	
TRUE	AMERICA	3797	1173	4970	
	ENGLAND	1311	3719	5030	
	TOTAL	5108	4892	10000	

Conclusion

- Moving forward, I would select the Naive Bayes model (MALLET)
- If restricted to using Scikit-learn, I would select the Gradient Boosting model that uses the extracted features
  - Boosting model with TF-IDF adds 1% to test accuracy, but increases training time by ~20 minutes
- To increase the overall accuracy of the classification, more knowledge in the field of computational linguistics would be necessary

Questions?