Subreddit Comparison using Natural Language Processing

Joshua Wilding
General Assembly
9/10/2018

What am I comparing?

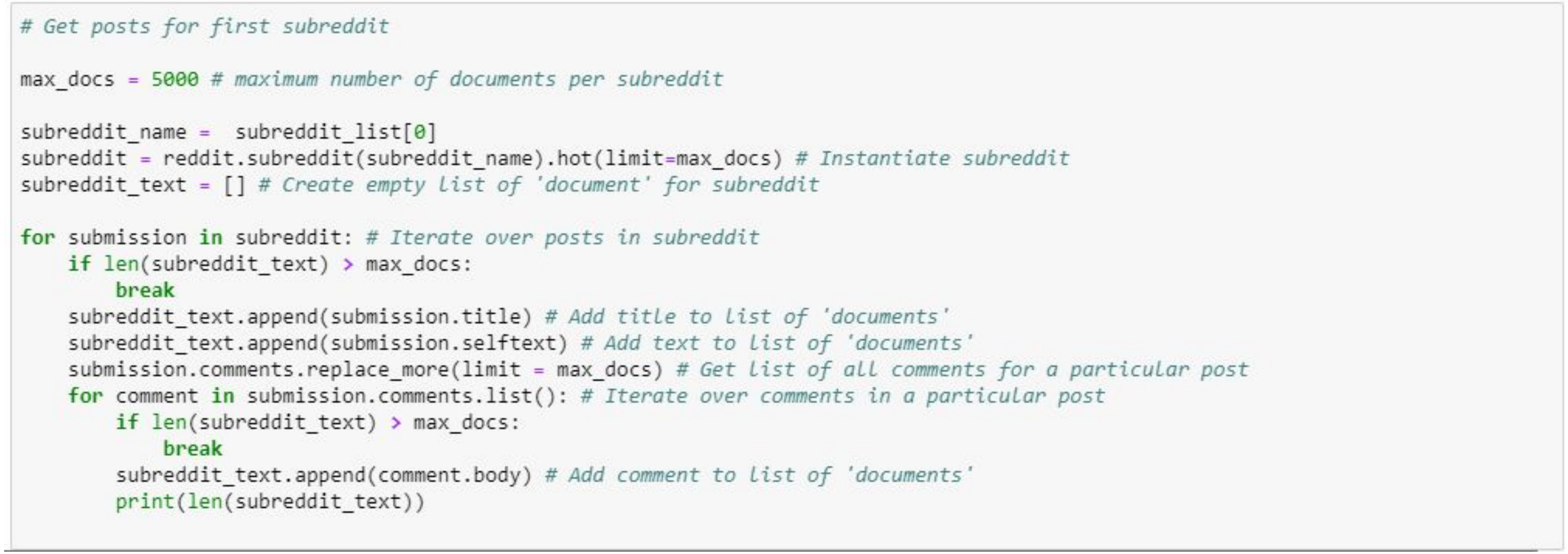
r/boston

- 96.7k subscribers
- “A reddit for the city of Boston, MA (featuring the cities of Cambridge, Somerville, Malden, Medford, Quincy, Braintree, Newton and the town of Brookline)”
- Expected a wider, more regional scope of topics

r/CambridgeMA

- 3.1k subscribers
- “For news, events, and info about the city.”
- Expecting hyper-local topics

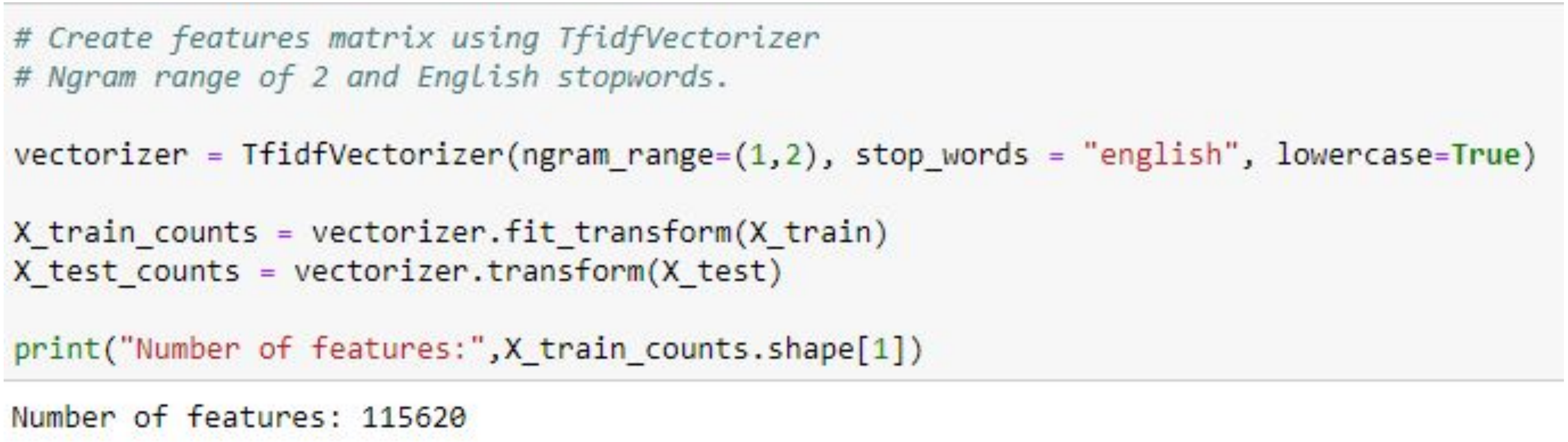
Gathering Data using PRAW



```python
# Get posts for first subreddit

max_docs = 5000 # maximum number of documents per subreddit

subreddit_name =  subreddit_list[0]
subreddit = reddit.subreddit(subreddit_name).hot(limit=max_docs) # Instantiate subreddit
subreddit_text = [] # Create empty list of 'document' for subreddit

for submission in subreddit: # Iterate over posts in subreddit
    if len(subreddit_text) > max_docs:
        break
    subreddit_text.append(submission.title) # Add title to list of 'documents'
    subreddit_text.append(submission.selftext) # Add text to list of 'documents'
    submission.comments.replace_more(limit = max_docs) # Get list of all comments for a particular post
    for comment in submission.comments.list(): # Iterate over comments in a particular post
        if len(subreddit_text) > max_docs:
            break
        subreddit_text.append(comment.body) # Add comment to list of 'documents'
        print(len(subreddit_text))
```

PRAW = Python Reddit API Wrapper

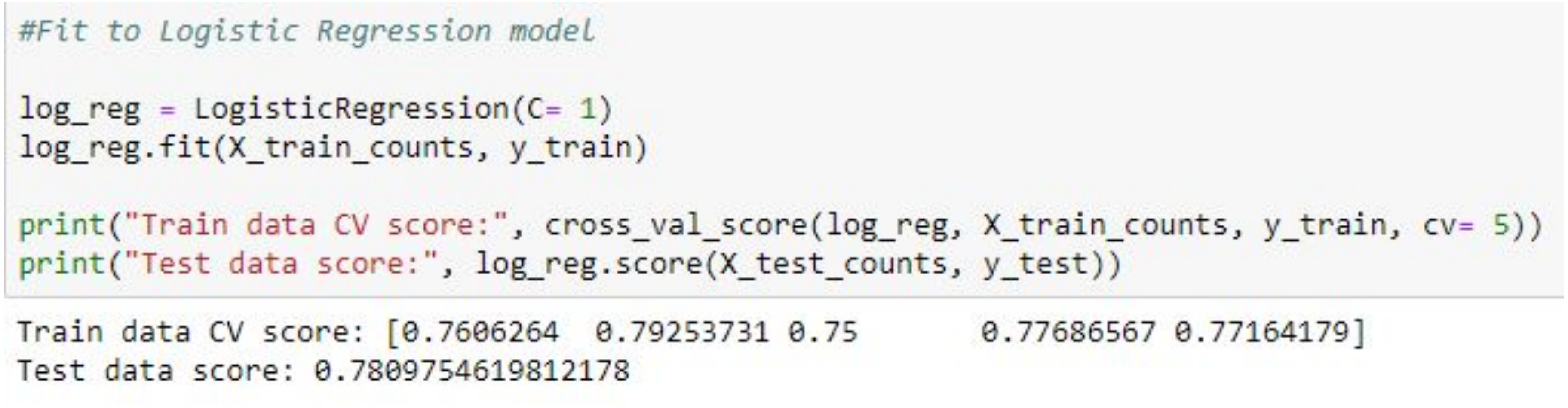
Feature Engineering



```
# Create features matrix using TfidfVectorizer
# Ngram range of 2 and English stopwords.

vectorizer = TfidfVectorizer(ngram_range=(1,2), stop_words = "english", lowercase=True)

X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

print("Number of features:",X_train_counts.shape[1])
```

```
Number of features: 115620
```

Logistic Regression Classifier

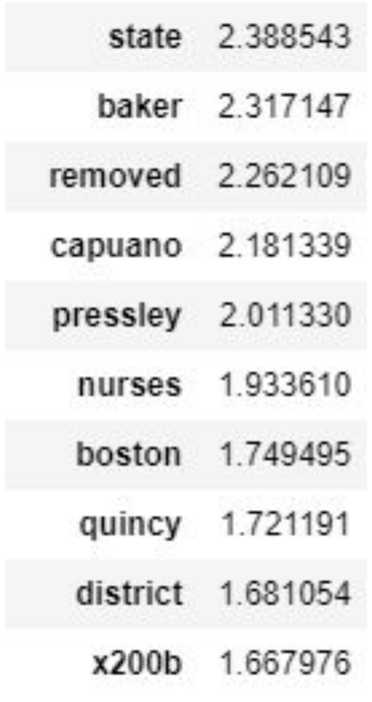


```
#Fit to Logistic Regression model

log_reg = LogisticRegression(C= 1)
log_reg.fit(X_train_counts, y_train)

print("Train data CV score:", cross_val_score(log_reg, X_train_counts, y_train, cv= 5))
print("Test data score:", log_reg.score(X_test_counts, y_test))
```

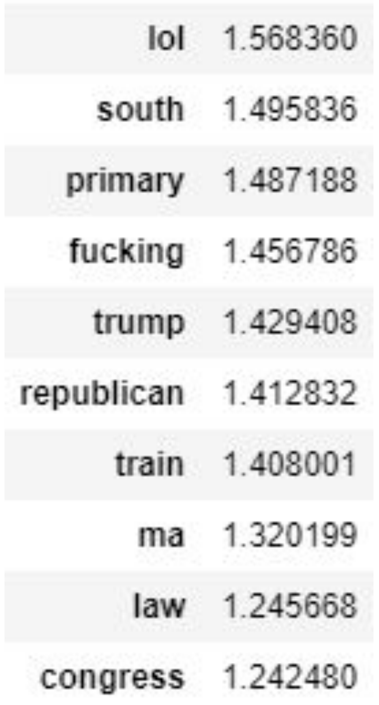
```
Train data CV score: [0.7606264  0.79253731 0.75       0.77686567 0.77164179]
Test data score: 0.7809754619812178
```

Top 20 Boston features

- State and national politics
- Cities other than Boston



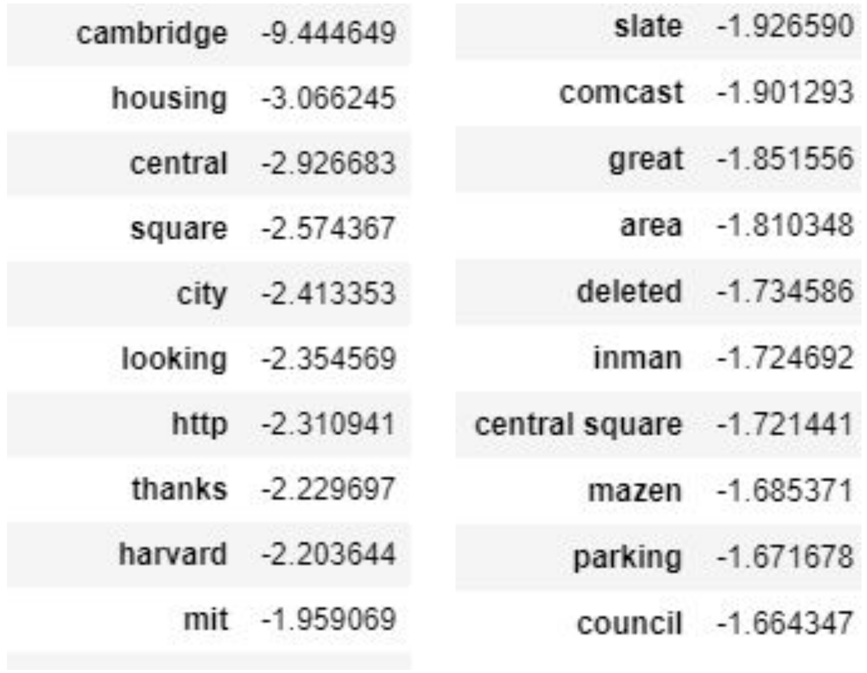
state	2.388543
baker	2.317147
removed	2.262109
capuano	2.181339
pressley	2.011330
nurses	1.933610
boston	1.749495
quincy	1.721191
district	1.681054
x200b	1.667976



lol	1.568360
south	1.495836
primary	1.487188
fucking	1.456786
trump	1.429408
republican	1.412832
train	1.408001
ma	1.320199
law	1.245668
congress	1.242480

Top 20 Cambridge features

- Local issues
- Friendlier tone



cambridge	-9.444649
housing	-3.066245
central	-2.926683
square	-2.574367
city	-2.413353
looking	-2.354569
http	-2.310941
thanks	-2.229697
harvard	-2.203644
mit	-1.959069
slate	-1.926590
comcast	-1.901293
great	-1.851556
area	-1.810348
deleted	-1.734586
inman	-1.724692
central square	-1.721441
mazen	-1.685371
parking	-1.671678
council	-1.664347

Things to consider

- Weighting based on upvote count
- Fancy graphs
- Who posted what?
- Spend more time tuning