Subreddit Classification: r/theonion vs r/nottheonion

Presented by: Group 6

r/theonion vs r/nottheonion

- r/theonion
  - Publishes satirical articles on international, national, and local news
  - 165K subscribers
  - Created Mar 23, 2008

- r/nottheonion
  - Publishes true news on international, national, and local news
  - 20.6m subscribers
  - Created Oct 25, 2008

Problem statement:

When performing maintenance, an engineer accidentally deleted multiple posts from r/nottheonion and r/theonion. Unfortunately, the engineer was only able to recover the titles of the lost posts.

Our goals of the project:

Using the natural language processing to build a classification model for

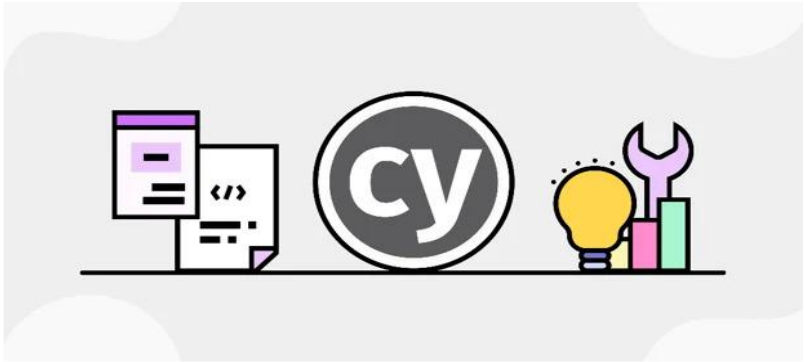
- training on posts submitted before 01 Jan 2022 to classify the recovered posts back to their respective subreddits, r/nottheonion and r/theonion, based solely on the post titles.
- as a proof of concept for the development of an automated moderator which would automatically delete posts that do not belong to the subreddit that they are posted to.
- Having automated moderators police the subreddit for spam posts would free up time for human moderators, who are volunteers, to do things that they want to do.

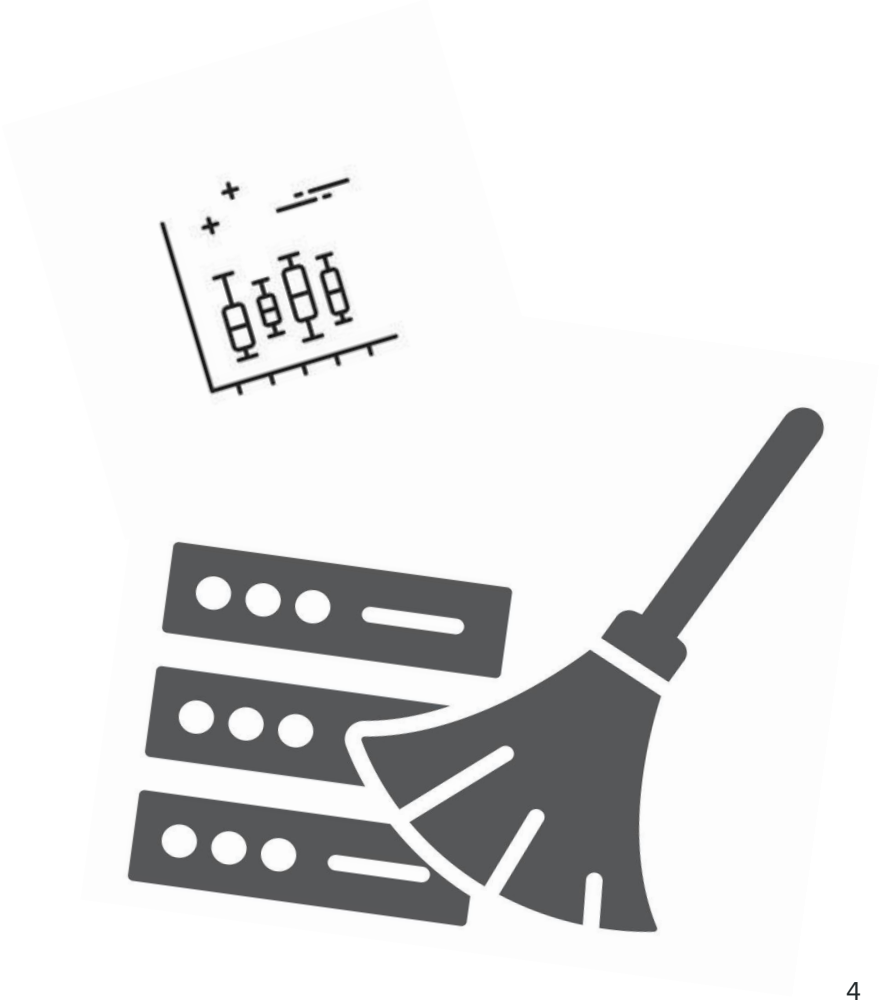
Data Cleaning and EDA

I. **Data Cleaning**
II. **Exploratory Data Analysis (EDA):**

- Bar plot for Most Active Authors
- Bar plot for Most Referenced Domains
- CountVectorizer (NLP):

❖ Unigrams & Bigrams

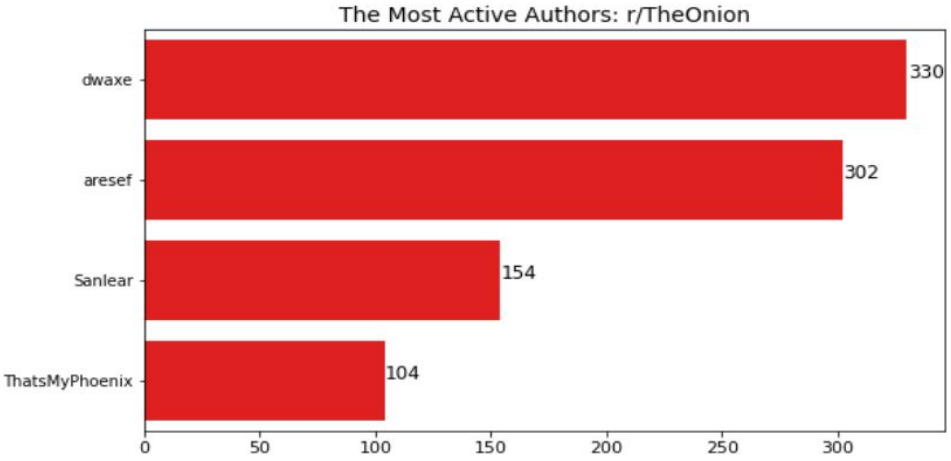




Bar Plot: Most Active Authors

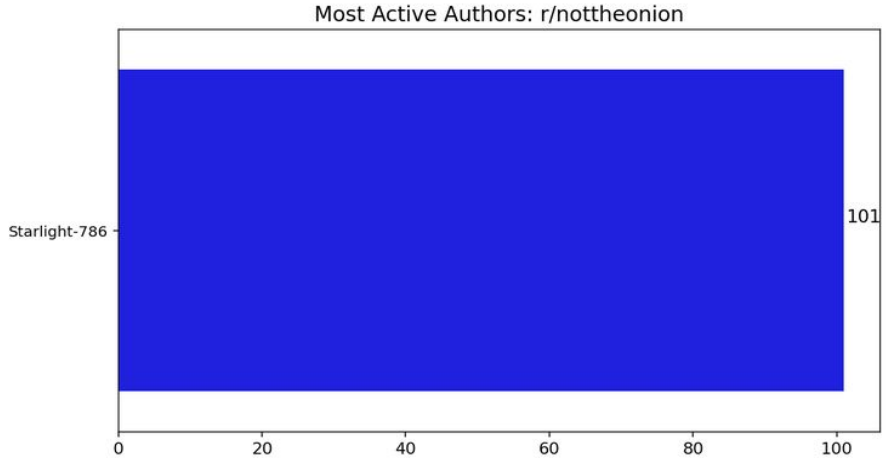
r/TheOnion:

- 4 Authors more than 100
- highest posts count : 330
- second highest count: 302



r/nottheonion:

- 1 Author with posts over 100
- Starlight-786



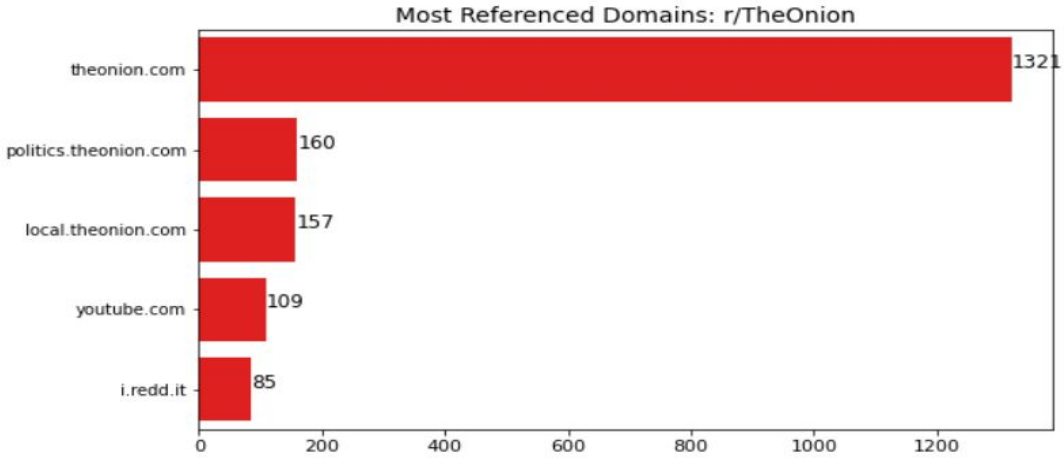
Bar Plot: Most Referenced Domains

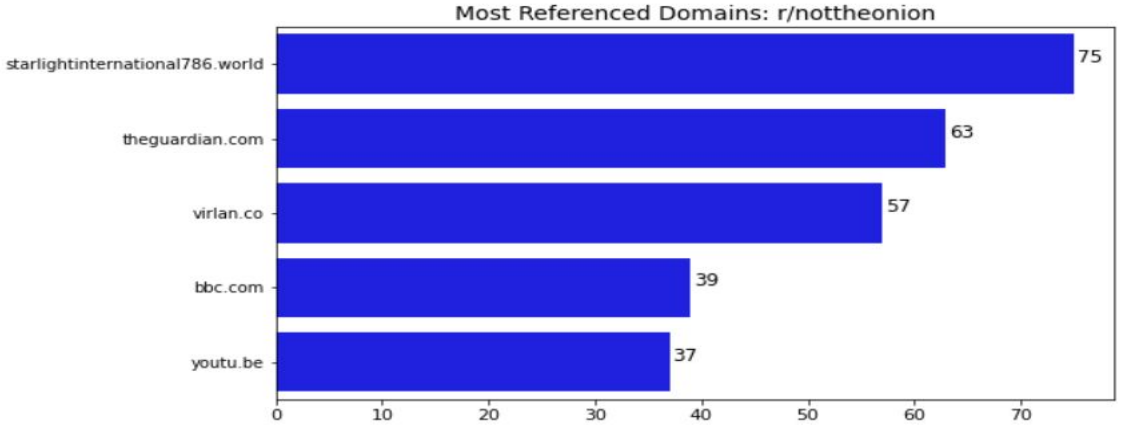
r/TheOnion:

- Top 5 Most referenced domains
- theonion.com
- Count 1321

r/nottheonion:

- Most referenced domains
- starlightinternational786.world
- Count 75



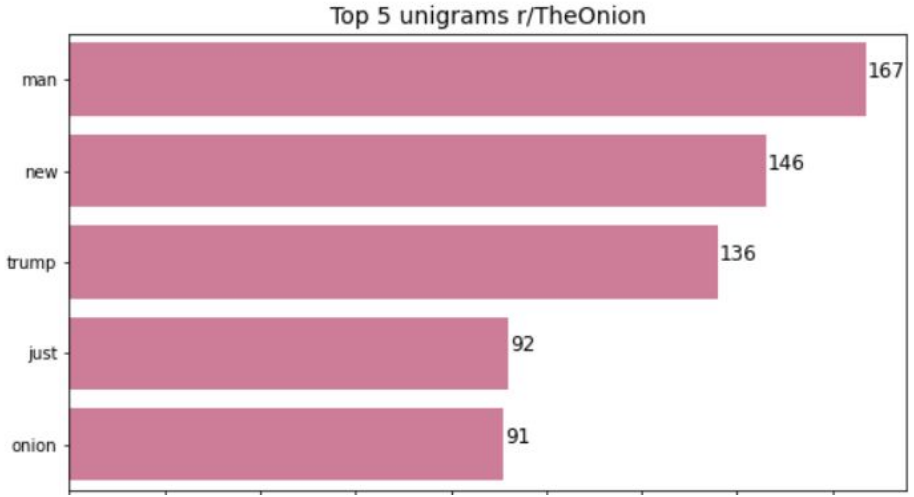


CountVectorizer

Top 5 Unigrams: `{'man'}`

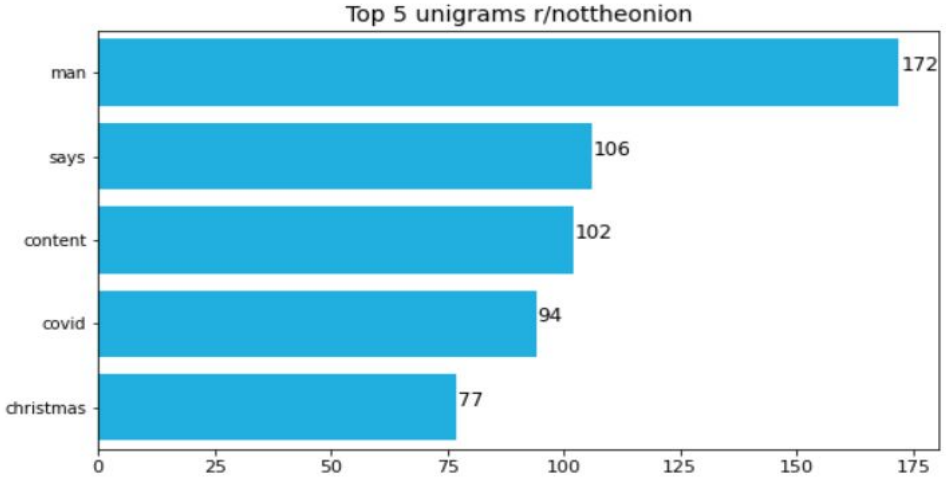
- **ngram_range = (1,1)**
- **TheOnion** subreddit is 1

`(2324, 7541)`



- **ngram_range = (1,1)**
- **nottheonion** subreddit is 0

`(1920, 5820)`

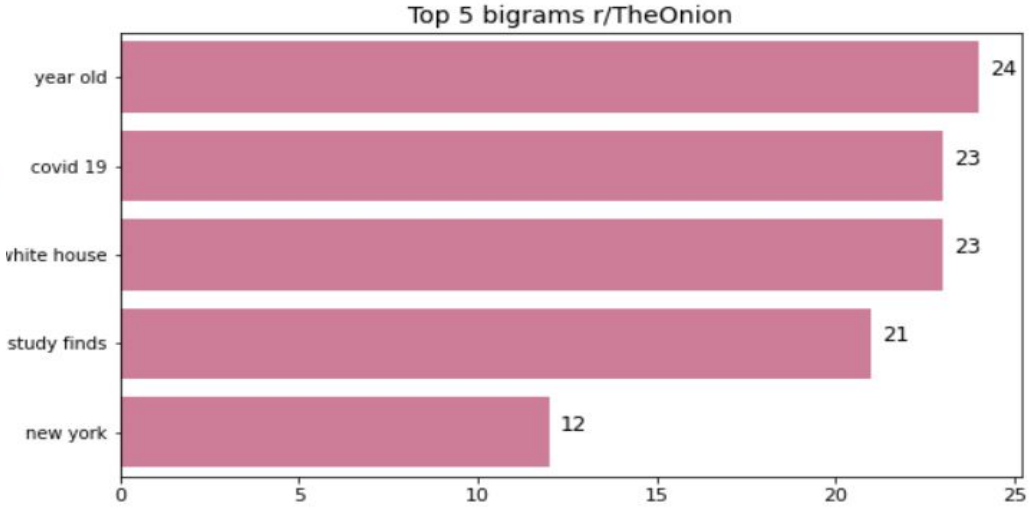


CountVectorizer

Top 5 Bigrams: `{'covid 19', 'year old'}`

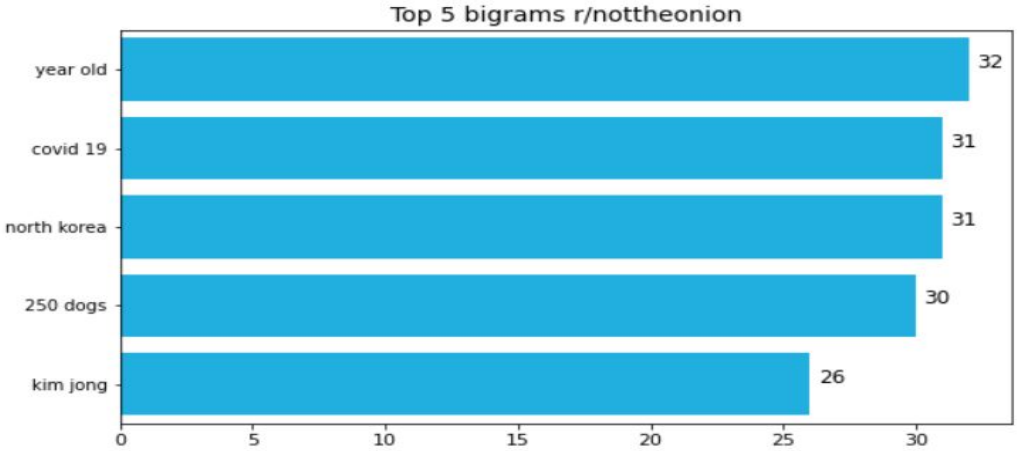
- **ngram_range = (2,2)**
- **TheOnion** subreddit is 1

`(2324, 15783)`



- **ngram_range = (2,2)**
- **nottheonion** subreddit is 0

`(1920, 11476)`



Modelling - Preprocessing

1) Binary labels

0	Not The Onion (genuine news)
1	The Onion (spinoff)

2) Further cleaning: lemmatizing, remove duplicate rows, punctuation etc.

3)

Feature X	Title of post
Predict y	Subreddit thread

4) Train test split

Modelling - Vectorizer selection

Logistic Regression

```
CountVectorizer + Logistic Regression scores:
Train score: 0.989
Test score: 0.773
Cross val score: 0.7256666666666666
=====
TfidfVectorizer + Logistic Regression scores:
Train score: 0.954
Test score: 0.779
Cross val score: 0.7226666666666667
```

Random Forest

```
CountVectorizer + Random Forest scores:
Train score: 0.959
Test score: 0.779
Cross val score: 0.7323333333333333
=====
TfidfVectorizer + Random Forest scores:
Train score: 0.961
Test score: 0.787
Cross val score: 0.7343333333333334
```

Insignificant difference between implementing CountVectorizer and TfidfVectorizer.

Modelling - Hyperparameter tuning

Logistic Regression

Parameters kept constant

```
pipe_tvec_lr_params = {'tvec__max_features':[None,1000,2000],
                       'tvec__stop_words':[None,'english'],
                       'tvec__ngram_range': [(1,1),(1,2),(1,3)],
                       'lr__penalty':['l1','l2'],
                       'lr__C':[0.01,0.1,1]}
```

Best parameters

```
{'lr__C': 1,
 'lr__penalty': 'l2',
 'tvec__max_features': None,
 'tvec__ngram_range': (1, 3),
 'tvec__stop_words': None}
```

Multinomial NB

```
pipe_tvec_nb_params = {'tvec__max_features':[None,1000,2000],
                       'tvec__stop_words':[None,'english'],
                       'tvec__ngram_range': [(1,1),(1,2),(1,3)],
                       'nb__alpha':[1,10,100]}
```

```
{'nb__alpha': 1,
 'tvec__max_features': None,
 'tvec__ngram_range': (1, 2),
 'tvec__stop_words': None}
```

Modelling - Hyperparameter tuning

Random Forest

```
pipe_tvec_rf_params = {'tvec__max_features':[1000,2000,3000],
                       'tvec__stop_words':[None,'english'],
                       'tvec__ngram_range': [(1,1),(1,2)], [1,3]
                       'rf__n_estimators':[100,200]}
```

```
{'rf__n_estimators': 200,
 'tvec__max_features': 3000,
 'tvec__ngram_range': (1, 1),
 'tvec__stop_words': None}
```

SVC

```
pipe_tvec_svc_params = {'tvec__max_features':[None,1000,2000],
                        'tvec__stop_words':[None,'english'],
                        'tvec__ngram_range': [(1,1),(1,2),(1,3)],
                        'svc__C':[1,0.1],
                        'svc__degree':[2,3]}
```

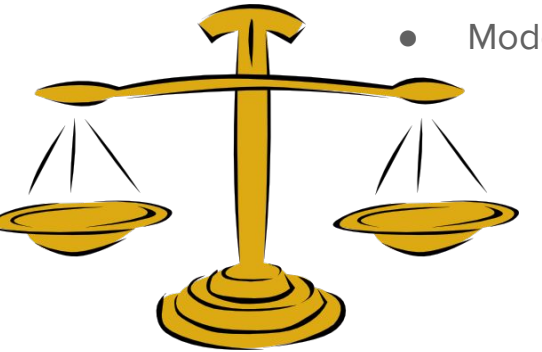
```
{'svc__C': 1,
 'svc__degree': 2,
 'tvec__max_features': None,
 'tvec__ngram_range': (1, 2),
 'tvec__stop_words': None}
```

Modelling - Comparison

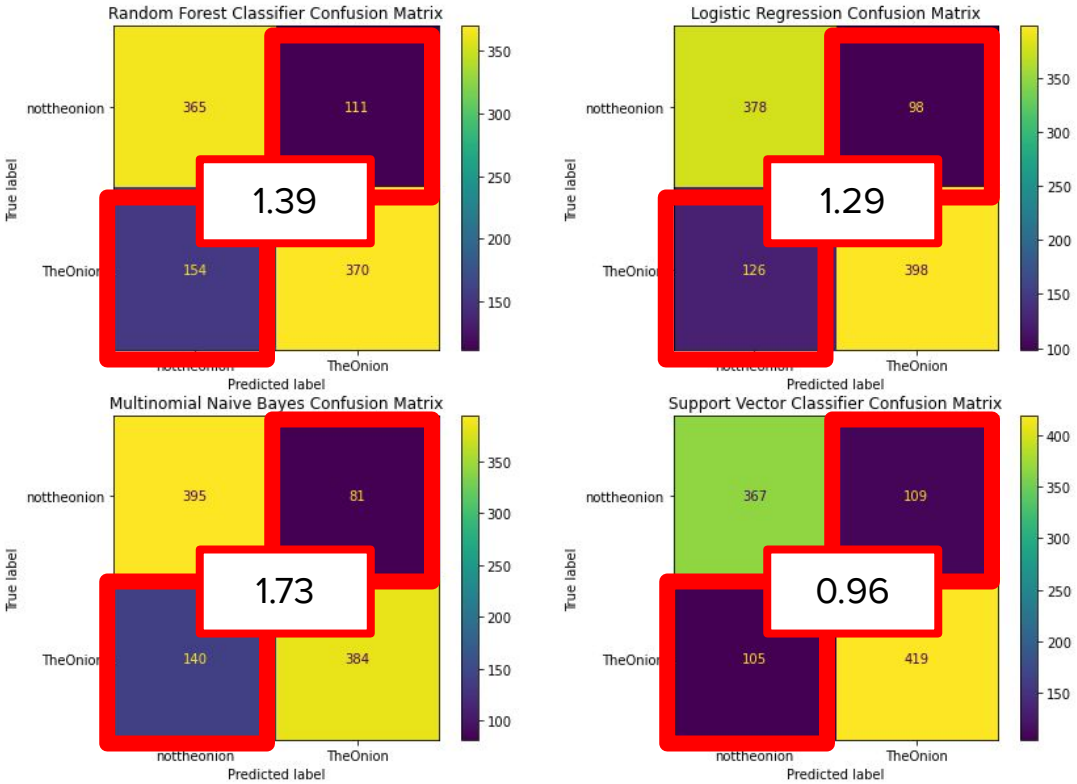
	Train accuracy	Test accuracy	Cross val score	F1 score
Baseline model (Dummy Classifier)	0.508	0.476	0.508	-
Random Forest Classifier	0.995	0.747	0.712	0.749
Logistic Regression	0.996	0.776	0.732	0.780
Multinomial Naive Bayes	0.995	0.779	0.740	0.777
Support Vector Classifier	0.999	0.786	0.742	0.797

Evaluation

- In addition to just the Test Accuracy Score, to dive deeper into the predictions
- We are trying to train an automoderator as members of Reddit's data science team
    - This automoderator will reflect Reddit as a company
    - Test Accuracy Score has to be sufficiently better than baseline model
    - Model's predictions should not favor 1 subreddit over the other

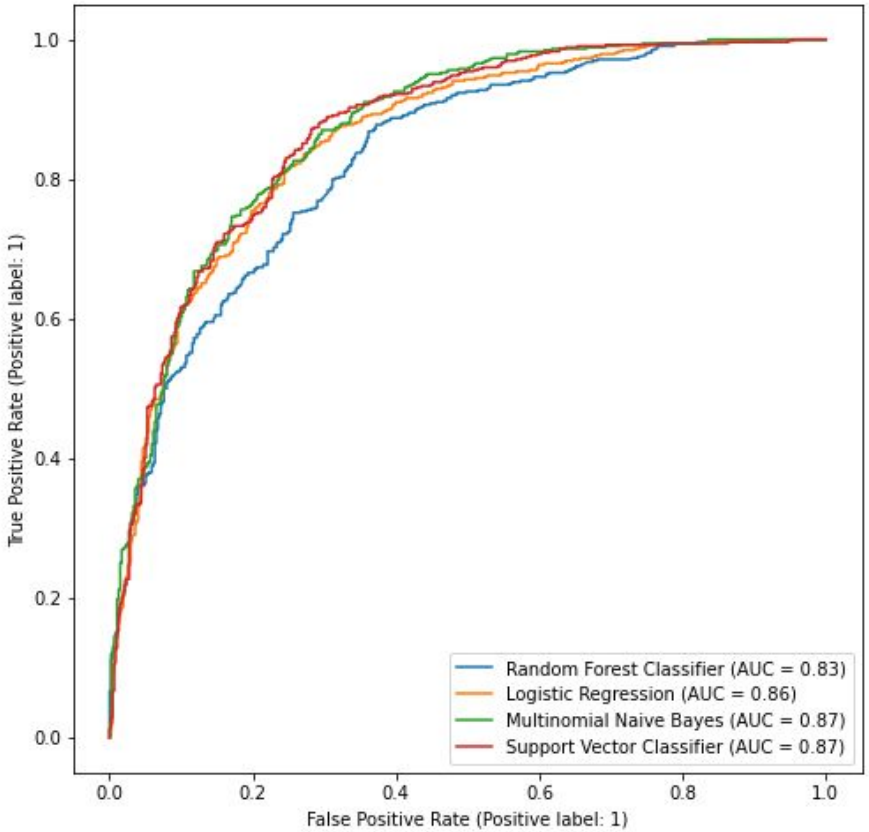


Confusion Matrix



- Our objective is to develop an automoderator that reduces workload for the human moderators
- Conversely, wrong predictions will increase their workloads instead
- As the Reddit data science team, we want to be impartial towards moderators of each subreddit
- Hence, we want to pick the model with the more balanced false predictions
  - I.e. ratio of false negatives to false positives = 1 ideally

Area under ROC



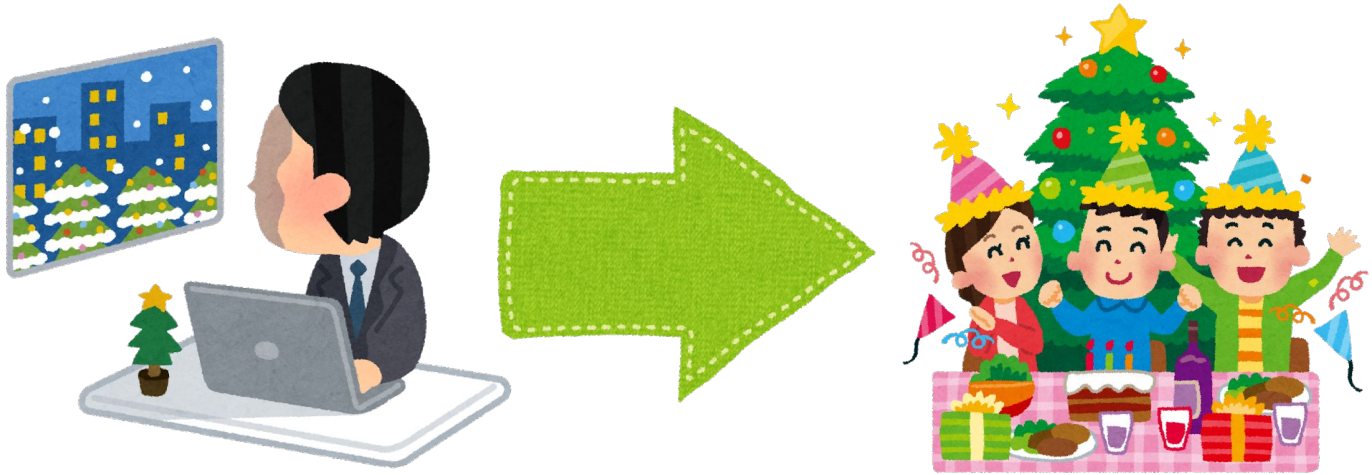
- “A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.”
- Basically, the higher the area (max. of 1.0), the more separate class 0 is from class 1
- Generally similar performance across the 4 models

Overall Evaluation

Model	Test Accuracy Score	False Neg. vs. Pos.	Area under ROC
Random Forest	0.747	1.39	0.83
Logistic Regression	0.776	1.29	0.86
Multinomial NB	0.779	1.73	0.87
Support Vector	0.786	0.96	0.87

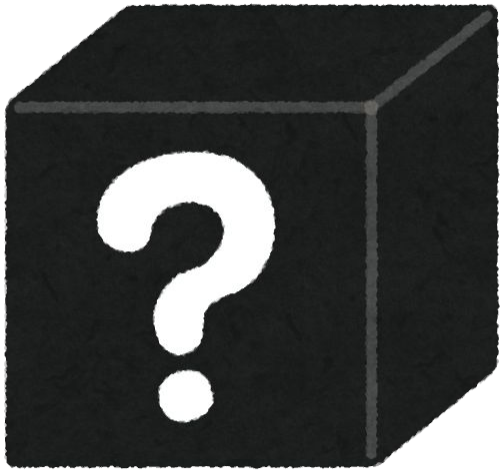
Conclusion and Recommendations

- Model sufficiently good to classify posts - 78.6% accuracy
  - Almost 80% of lost data will be recovered correctly
- Machine learning models feasible for automated moderation
  - Human moderators (who are volunteers) will have more free time



Limitations

- Blackbox
  - Not able to tell which words make a post more likely to be classified to be from one subreddit than the other



Future improvements

- More data to train model

- Examine text more closely
  - Remove words that do not provide additional information

- Engineer metadata as additional features
  - Character count, word count, sentiment analysis etc