Real News vs The Onion

Fake News Detection
via
Natural Language Processing

The Problem of Fake News

Public opinion of our company has been negative mostly due to the claims of fake news being widespread throughout our primary platforms: Facebook and Instagram.

The Problem of Fake News

Public opinion of our company has been negative mostly due to the claims of fake news being widespread throughout our primary platforms: Facebook and Instagram.

Lawmakers have also began discussing regulation of our platforms both in the U.S. and Europe.

The Problem of Fake News

As part of an internal panel on developing a response stratagem, the Data Science department has launched multiple parallel projects to explore feasible action plans.

Our Team’s Project :

Explore the possibility of

detecting fake news articles

on posts shared by our users

What is Fake News?

- Objectively false facts?
- Technically true but maliciously misleading?
- Things believed to be true but later demonstrated false?
- Satire and parody content?

What is Fake News?

- Objectively false facts?
- Technically true but maliciously misleading?
- Things believed to be true but later demonstrated false?
- Satire and parody content?

Methodology

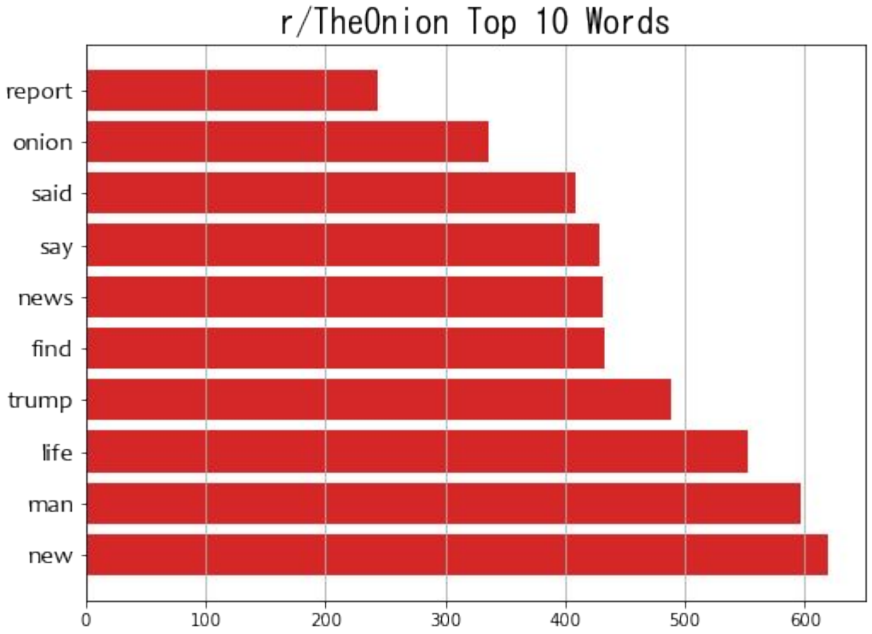
1. Scrape posts from two subreddits: *r/news* and *r/TheOnion*.
    a. Only use post titles to mimic news headlines.
    b. Investigate 2 separate situations of balanced and imbalanced classes.
2. Use *Count encoding, TF-IDF, SIA,* and *Word2Vec* to encode text as numeric data.
3. Train a discriminator to identify which posts came from where.
    a. Naive Bayes, Logistic Regression, SVM, and Neural Network.
    b. Bagged Trees, Random Forest, AdaBoost and Gradient Boosting

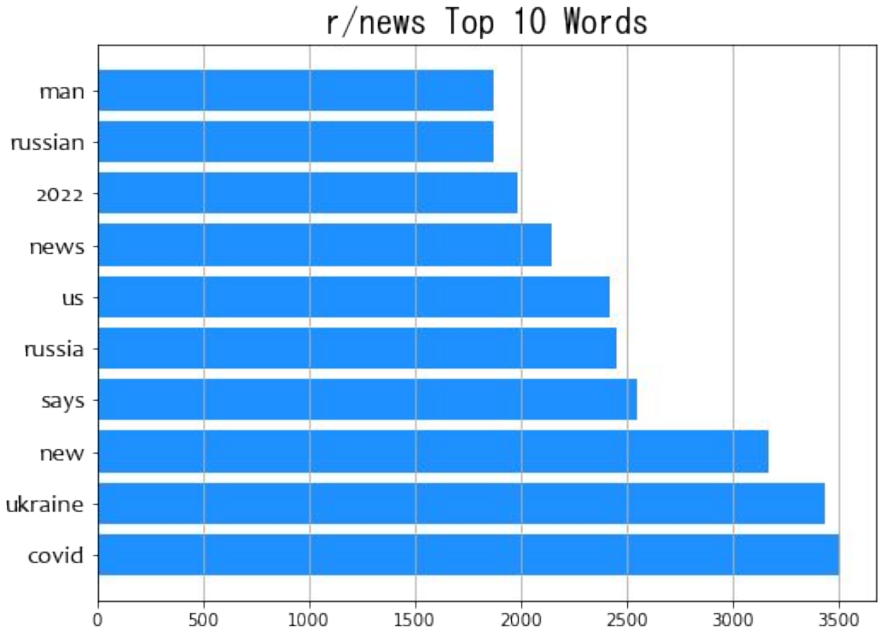
Some Data Insights

Data Size

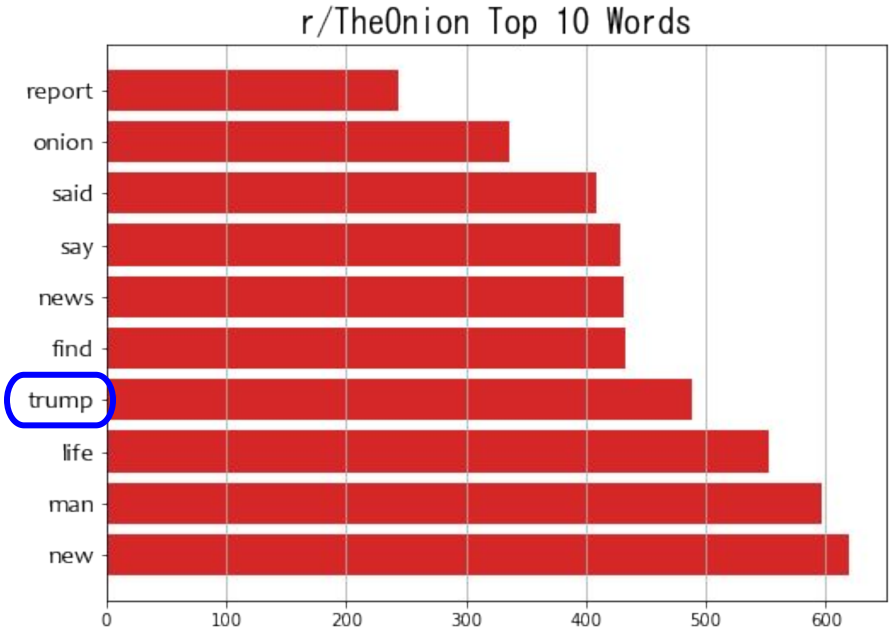
- Scraped 100,000 *r/news* posts vs 17,000 *r/TheOnion* posts.
- Purged duplicates and non-ASCII titles; 80,000 training examples remained.
- Scenario 1: treat this as an imbalanced class problem with *r/news* vs *r/TheOnion* ratio at 7:1.
- Scenario 2: reduce down to 20,000 training examples but with 50:50 split (basically equivalent to Random Under Sampling).

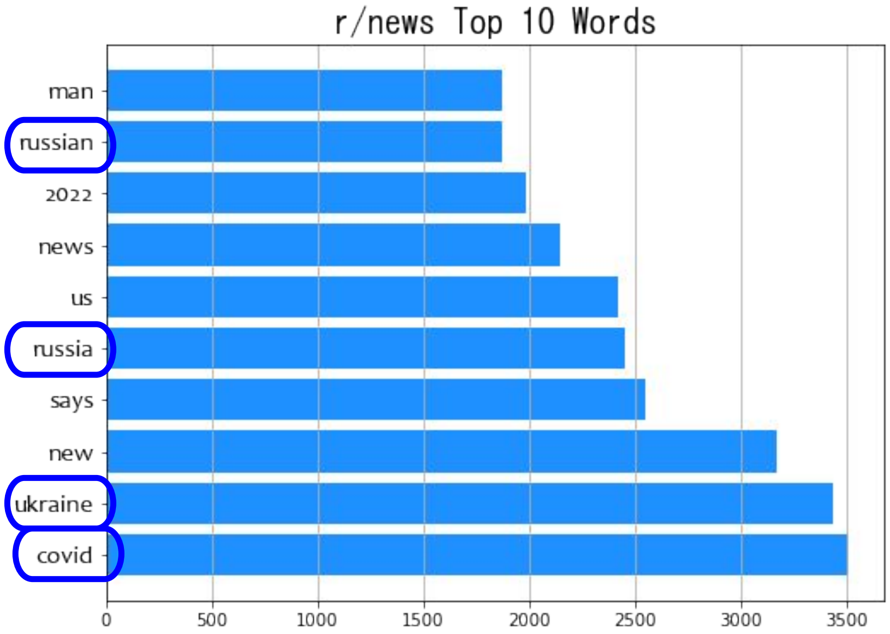
Top 10 Words


r/TheOnion Top 10 Words
report
onion
said
say
news
find
trump
life
man
new
0
100
200
300
400
500
600

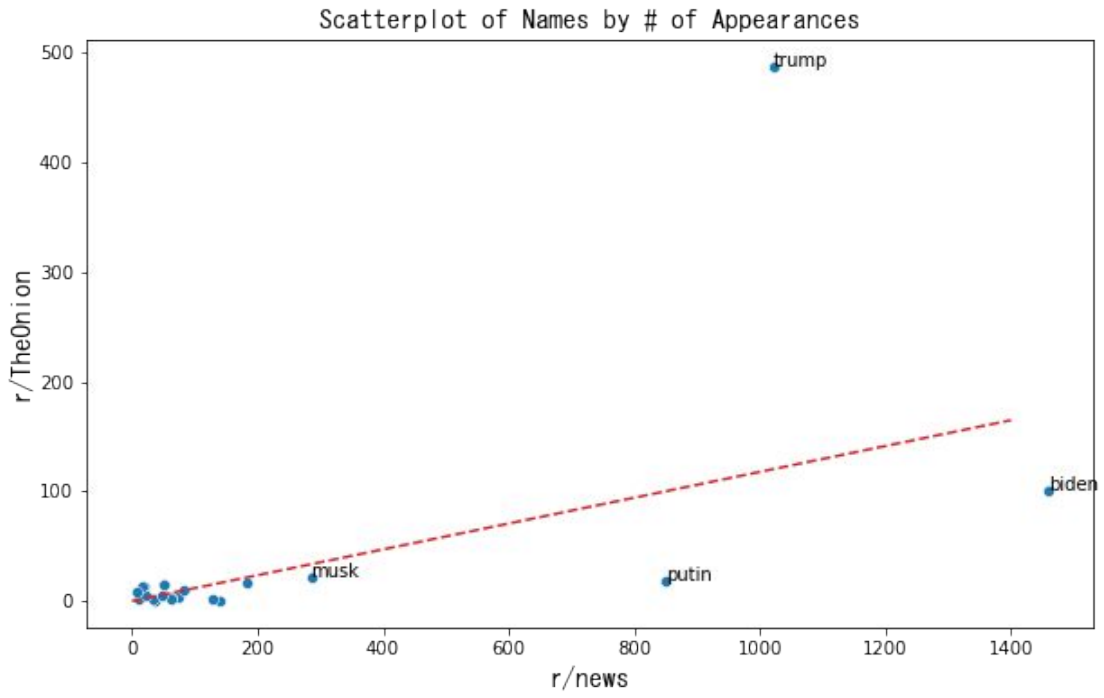

r/news Top 10 Words
man
russian
2022
news
us
russia
says
new
ukraine
covid
0
500
1000
1500
2000
2500
3000
3500

Top 10 Words


r/TheOnion Top 10 Words
report
onion
said
say
news
find
trump
life
man
new
0
100
200
300
400
500
600

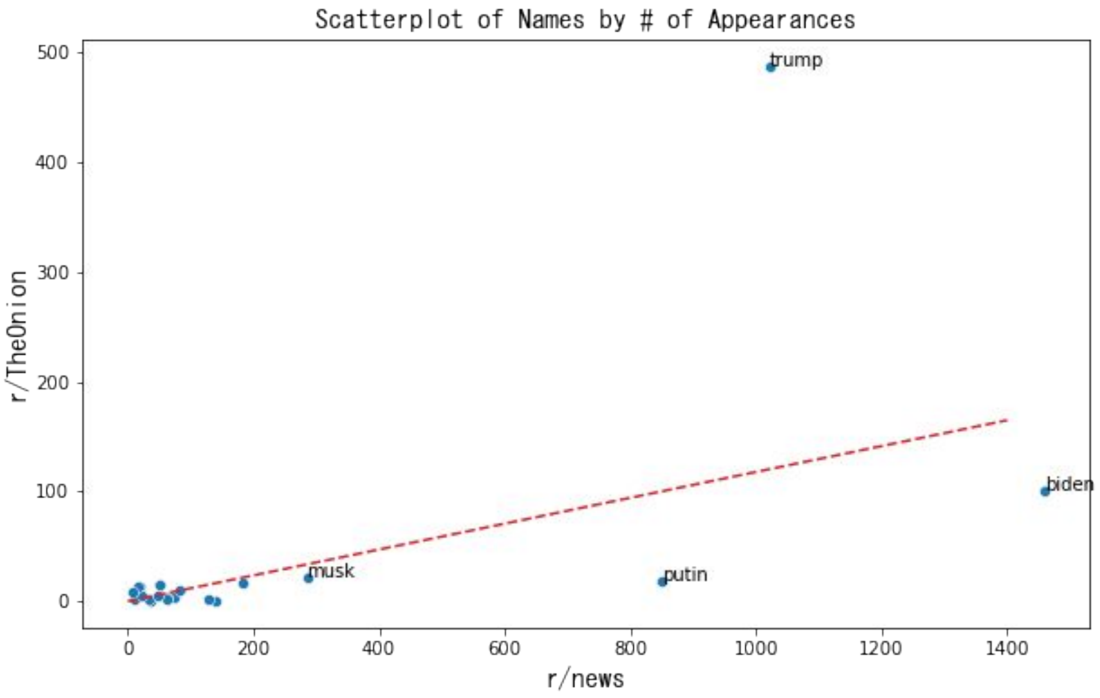

r/news Top 10 Words
man
russian
2022
news
us
russia
says
new
ukraine
covid
0
500
1000
1500
2000
2500
3000
3500

Name Mentions



Name	r/TheOnion	r/news
“trump”	488	1021
“biden”	101	1458
“putin”	19	849
“macron”	1	37
“trudeau”	3	73
“kardashian”	2	61
“kanye”	10	81
“bezos”	15	50
“gates”	15	52
“musk”	22	285

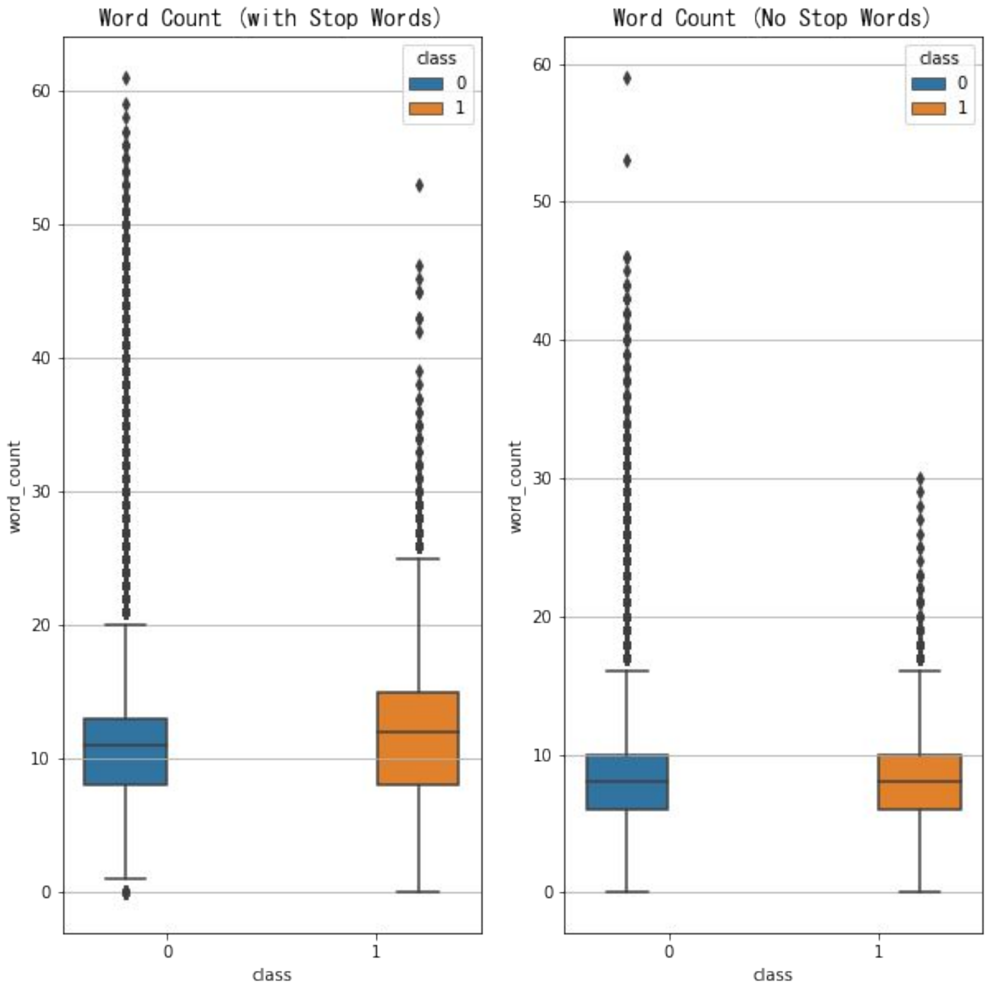
Name Mentions



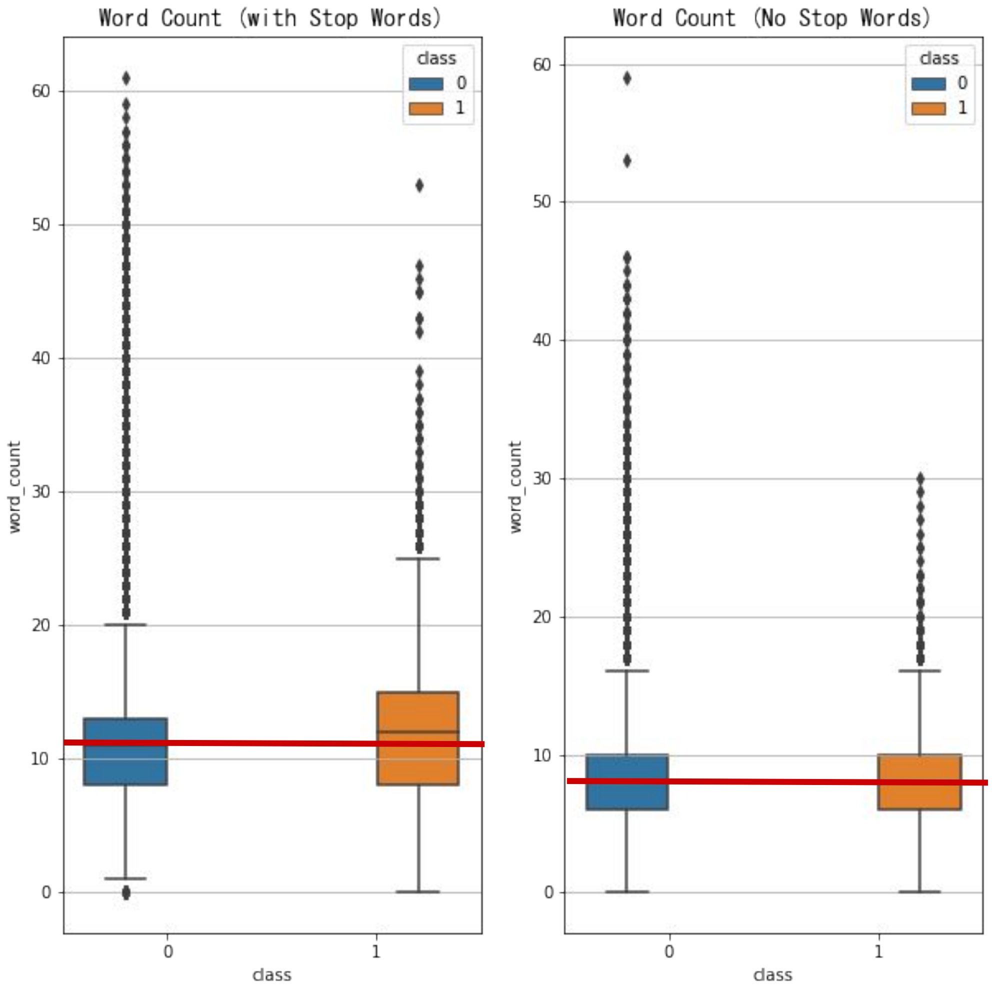
Red line = 7:1 ratio of *r/news* to *r/TheOnion*.

Name	r/TheOnion	r/news
"trump"	488	1021
"biden"	101	1458
"putin"	19	849
"macron"	1	37
"trudeau"	3	73
"kardashian"	2	61
"kanye"	10	81
"bezos"	15	50
"gates"	15	52
"musk"	22	285

Title Lengths



Title Lengths



Scenario 1: Imbalanced Classes

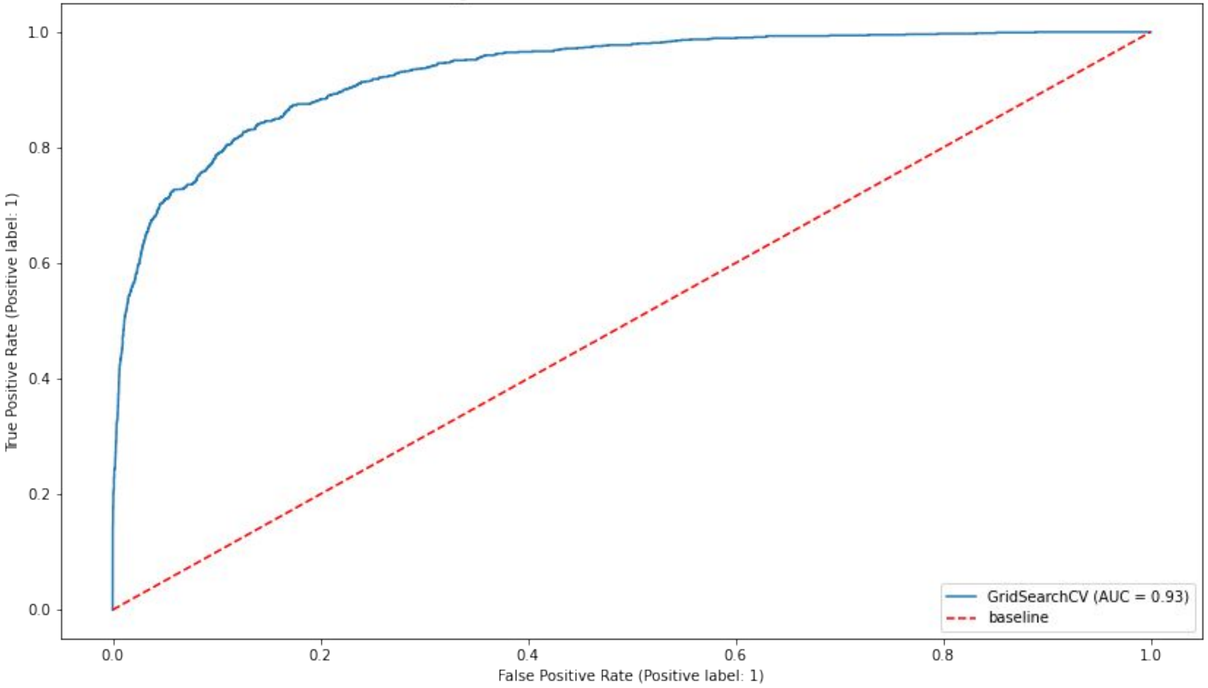
Naive Bayes

Encoding	Accuracy	Recall	Precision	F1	AUC
Baseline	87%	-	-	-	0.5
CountVectorizer	93%	59%	78%	67%	0.93
TF-IDF	88%	5%	100%	10%	0.86
SIA	87%	8%	13%	1%	0.55

Naive Bayes

Encoding	Accuracy	Recall	Precision	F1	AUC
Baseline	87%	-	-	-	0.5
CountVectorizer	93%	59%	78%	67%	0.93
TF-IDF	88%	5%	100%	10%	0.86
SIA	87%	8%	13%	1%	0.55

Naive Bayes with CountVectorizer

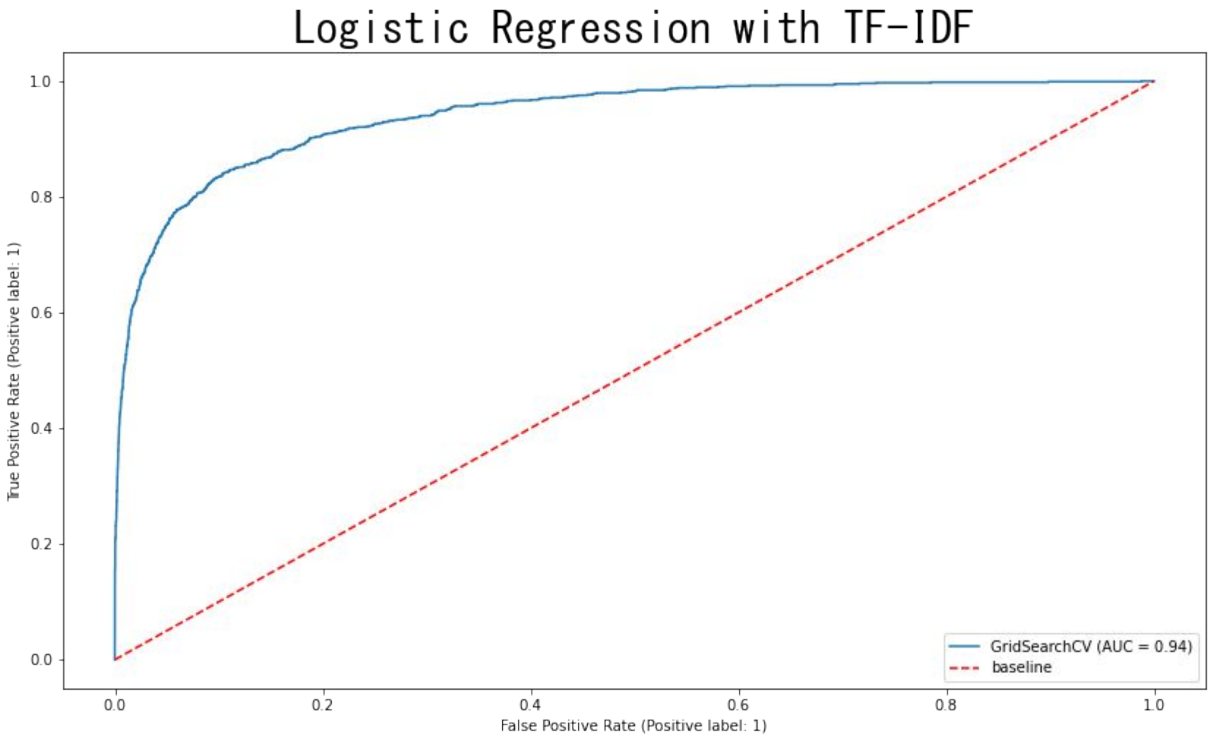


Logistic Regression

Encoding	Accuracy	Recall	Precision	F1	AUC
Baseline	87%	-	-	-	0.5
CountVectorizer	93.2%	55%	80%	67%	0.94
TF-IDF	93.3%	57%	85%	69%	0.94
SIA	87%	0%	0	0	0.56

Logistic Regression

Encoding	Accuracy	Recall	Precision	F1	AUC
Baseline	87%	-	-	-	0.5
CountVectorizer	93.2%	55%	80%	67%	0.94
TF-IDF	93.3%	57%	85%	69%	0.94
SIA	87%	0%	0	0	0.56



Logistic Regression with TF-IDF
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)
GridSearchCV (AUC = 0.94)
baseline

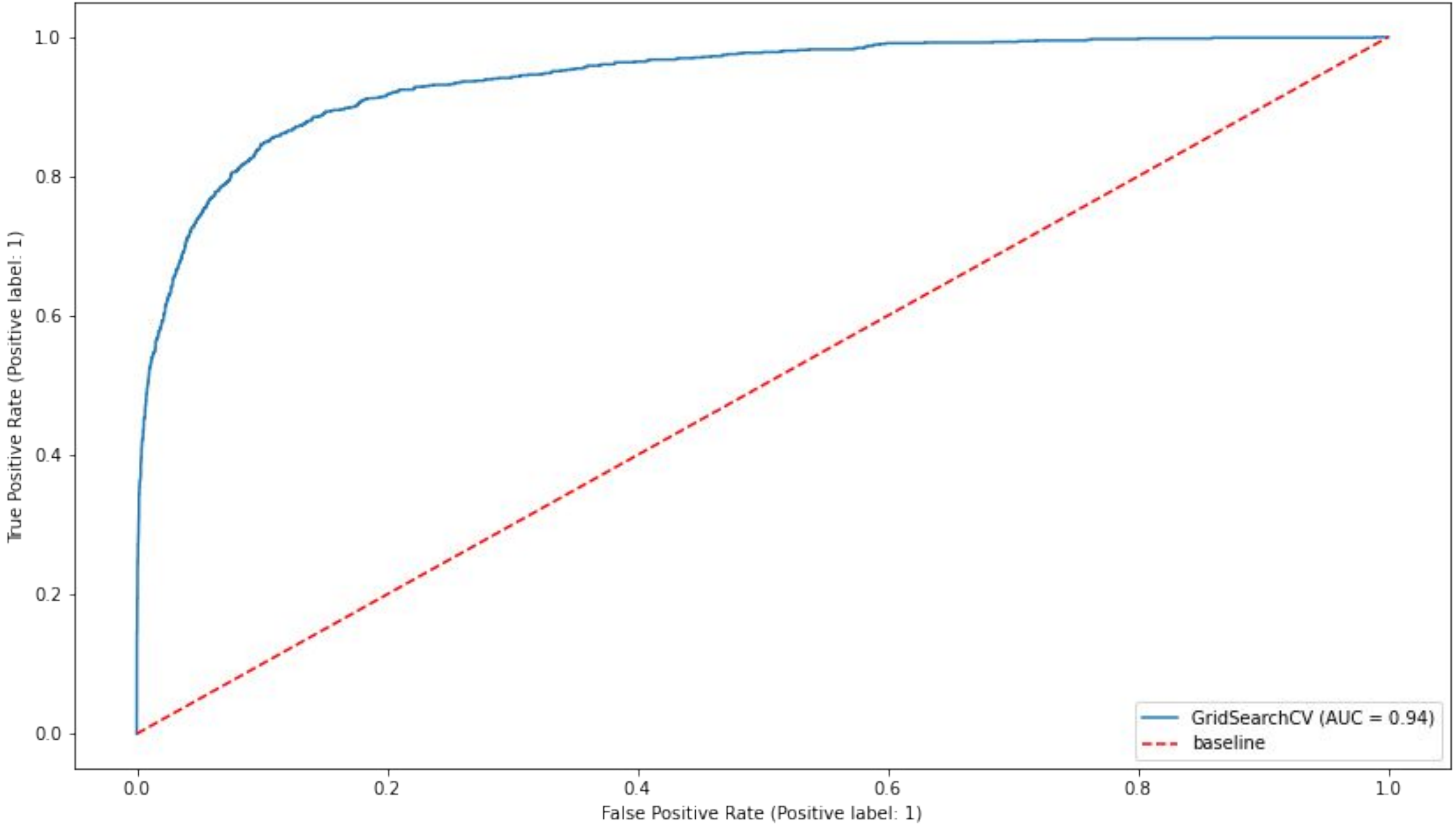
Support Vector Machine

Encoding	Accuracy	Recall	Precision	F1	AUC
Baseline	87%	-	-	-	0.5
CountVectorizer	93.3%	54%	88%	67%	0.94
TF-IDF	93.4%	56%	87%	67%	0.94

Support Vector Machine

Encoding	Accuracy	Recall	Precision	F1	AUC
Baseline	87%	-	-	-	0.5
CountVectorizer	93.3%	54%	88%	67%	0.94
TF-IDF	93.4%	56%	87%	67%	0.94

SVM with TF-IDF



Word2Vec (via Gensim)

- Word2Vec: ML model takes words and assigns them to vectors.
- The “closer” two words are in meaning, the closer the assigned vectors are too (closeness is measured by the cosine of their angle).
- Because vectors can be added/subtracted, we can now do arithmetic on words!

“King” - “Man” + “Woman” = “Queen”

“OnionNet” (W.I.P.)

1. Use GloVe (Stanford’s pre-trained Word2Vec model) to map each document to an array of vectors.
2. Flatten array into a single long vector. Now each document is a vector, the meaning of the words are encoded, and ordering of words is preserved!
3. Use these long vectors as inputs to a Neural Network (1 hidden layer, 2000 hidden units).

“OnionNet” (W.I.P.)

Results:

- Trained for 10 epochs; Accuracy 90.7% (worse than Naive Bayes and Logistic Regression!). Only Marginally better than baseline :(
- Currently unable to handle people names (ignoring A LOT of info!)
- Might perform better with a larger vocabulary (currently using GloVe with 6 billion tokens, maybe try 840 billion tokens model).
- Might perform better if it had millions of training examples.

Scenario 2: Balanced Classes

Balanced Classes

- Alternatively, we can balance the classes by Under Sampling.
- This has the added benefit of reducing the data down to a more manageable amount (20,000 examples instead of 80,000)
- Faster training time and more feasible to prototype models with.

SVM + Ensemble Methods

SVM + Ensemble Methods (with TF-IDF)

Model	Accuracy	Recall	Precision	F1	AUC
Baseline	50%	-	-	-	0.5
SVM (rbf kernel)	76%	93%	70%	80%	0.88
Bagged Trees	72%	63%	77%	69%	0.80
Random Forest	79%	73%	83%	78%	0.89
AdaBoost	72%	72%	72%	72%	0.78
Gradient Boosting	78%	75%	80%	77%	0.87
OnionNet	79%	78%	80%	79%	-

SVM + Ensemble Methods (with TF-IDF)

Model	Accuracy	Recall	Precision	F1	AUC
Baseline	50%	-	-	-	0.5
SVM (rbf kernel)	76%	93%	70%	80%	0.88
Bagged Trees	72%	63%	77%	69%	0.80
Random Forest	79%	73%	83%	78%	0.89
AdaBoost	72%	72%	72%	72%	0.78
Gradient Boosting	78%	75%	80%	77%	0.87
OnionNet	79%	78%	80%	79%	-

Final Results:

Situation	Best Model	Training Set	Validation Set	Final Test Set
Imbalanced	**Linear SVM**	99.29%	93.4%	93.85%
Balanced	**Random Forest**	91%	79%	85%

Conclusions and Recommendations

- Logistic Regression and Linear SVM with TF-IDF did ok.
  - Probably biased towards buzzwords and current news.
  - Evidence that a good model can be built with enough research.
- Sentiment Analysis NOT effective in classifying satire/humor/non-serious headlines.
- Invest in cloud computing / big data frameworks to train ensemble methods.
- Can work if we find a good way to encode document structure (word2vec just the start).