Natural Language Processing
Binary Classification for r/DadJoke Subreddits.

Presented by:
Joey Navarro

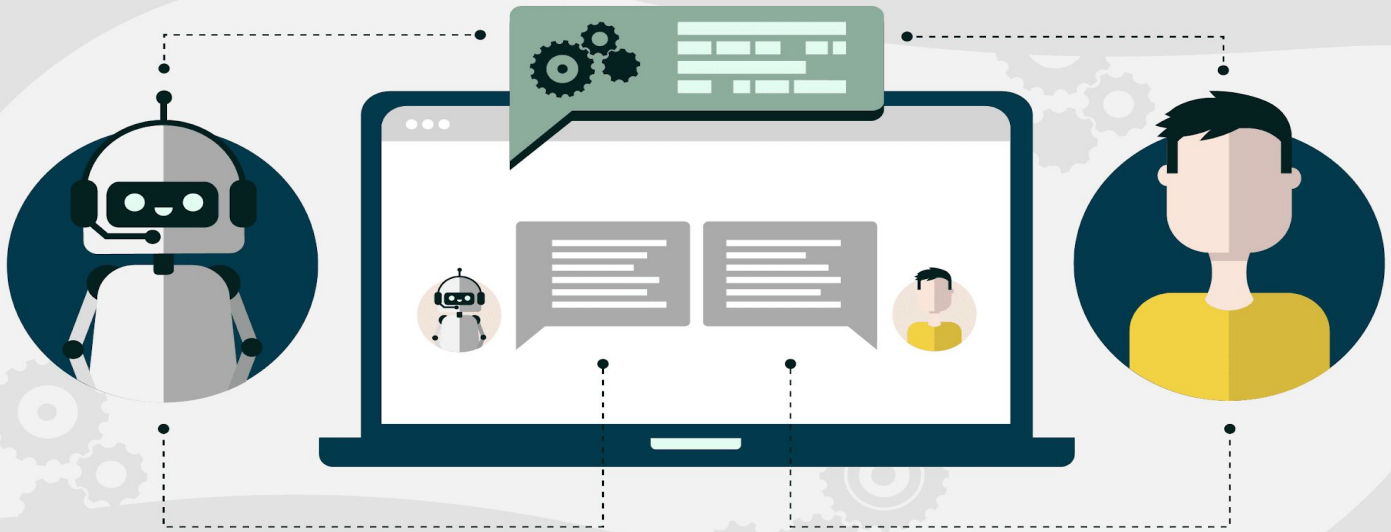


Image From: Natural Language Processing: A Short Introduction To Get You Started |

Contents

Photo by Atef Khaled from Pexels

The Situation

Oh no! My brother used my computer to get on reddit and now there are all a lot of subreddit pages open. I wonder if I could build a model that would predict a subreddit correctly if I gave it only the posts of all the subreddit pages he has open.

He has the following subreddits open:
r/80sRock, r/DadJokes, r/Electricity, r/history, r/philosophy, r/rant, r/showerthoughts.

Problem Statement

Which model in my Natural Language Processing toolbox would best classify a post of a specific subreddit topic from a group of subreddit posts, I scraped 1,000 subreddit posts from each topic. I chose to classify for r/DadJokes.

* My real brother is not pictured here.

My NLP Toolbox

Models

Linear Regression	K Nearest Neighbor
Extra Trees	Voting Classifier
Random Forest	--GridSearchCV

Vectorizers

Count Vectorizer
TF-IDF Vectorizer

PRAW API Reddit Scraper

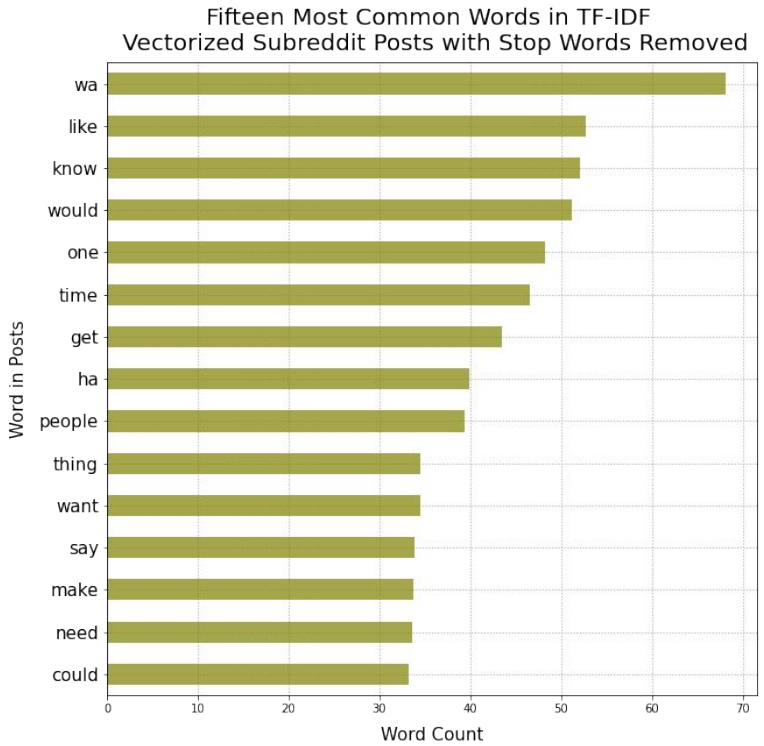
What I'll be Looking For

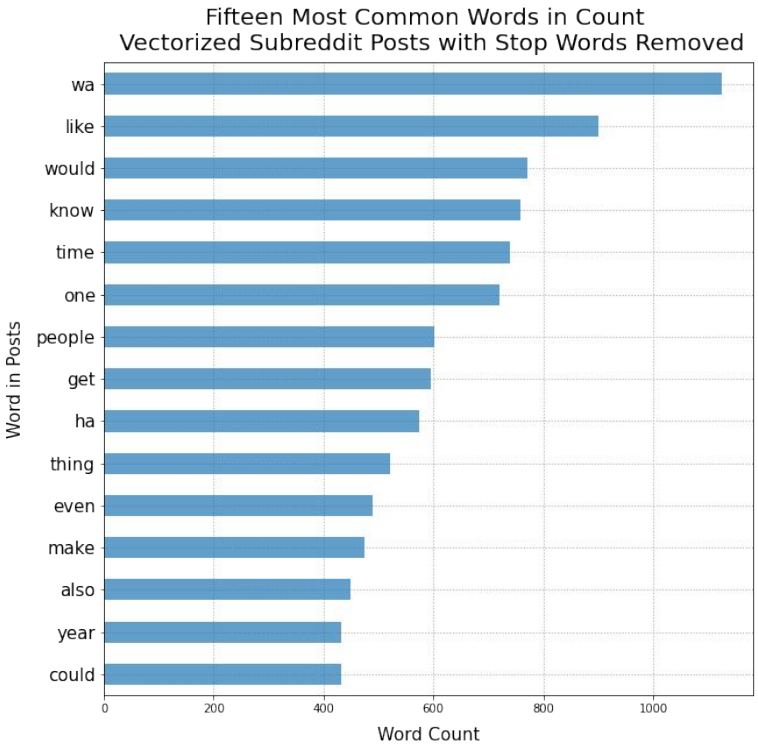
Classification Matrix

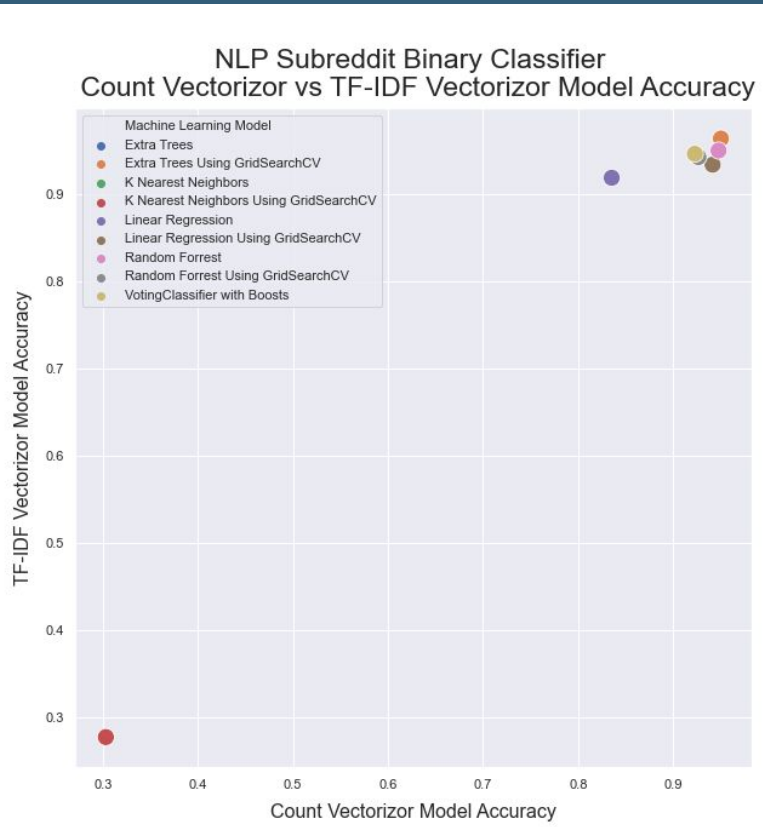
	Actual Yes	Actual No
Predicted Yes	True Positive (TP)	False Positive (FP)
Predicted No	False Negative (FN)	True Negative (TN)

Classification Metrics

Accuracy	Specificity
Sensitivity	Precision
Misclassification Rate	


Fifteen Most Common Words in TF-IDF
Vectorized Subreddit Posts with Stop Words Removed
wa
like
know
would
one
time
get
ha
people
thing
want
say
make
need
could
Word in Posts
0
10
20
30
40
50
60
70
Word Count


Fifteen Most Common Words in Count
Vectorized Subreddit Posts with Stop Words Removed
wa
like
would
know
time
one
people
get
ha
thing
even
make
also
year
could
Word in Posts
0
200
400
600
800
1000
Word Count

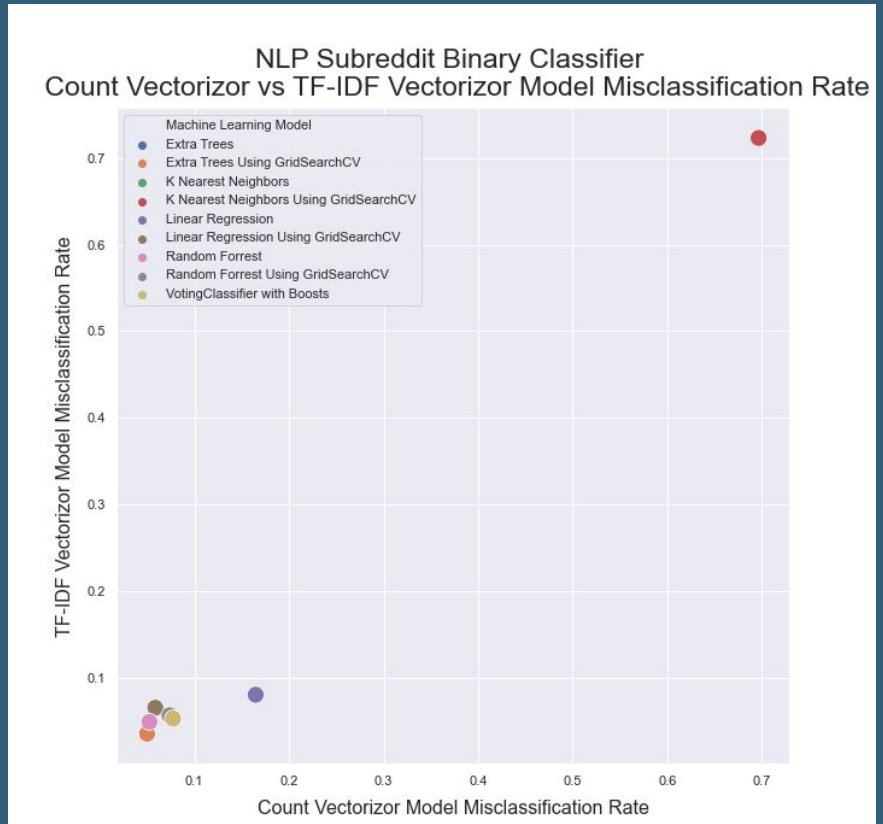


Accuracy

$$\frac{TP+TN}{TP+TN+FP+FN}$$

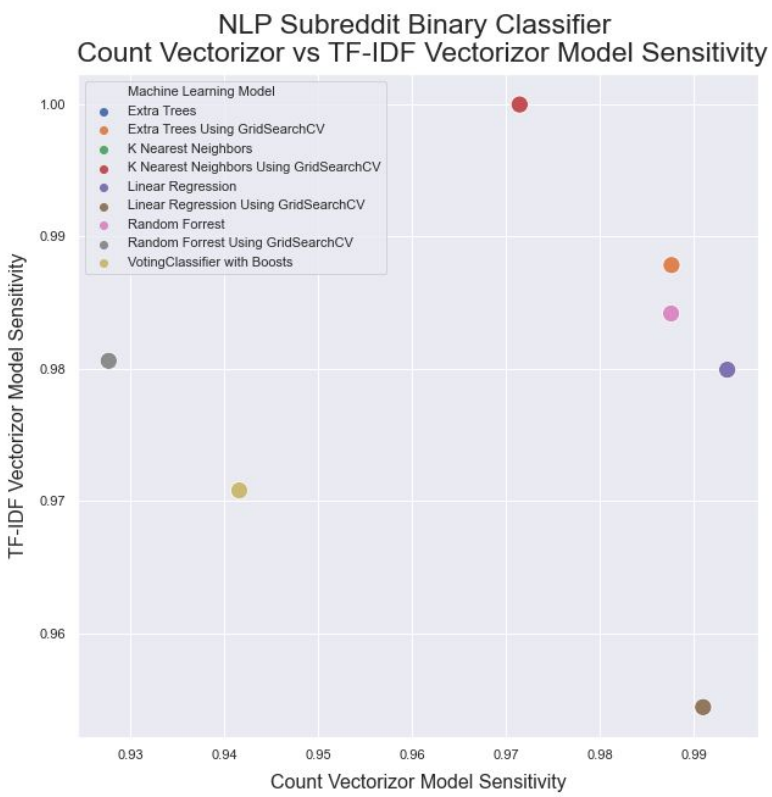
Model	TF-IDF	CVEC
Linear Regression	91.9%	83.5%
Linear Regression Using GridSearchCV	93.4%	94.1%
K Nearest Neighbor	27.7%	30.3%
K Nearest Neighbor Using GridSearchCV	27.7%	30.3%
Random Forest	95%	94.8%
Random Forest Using GridSearchCV	94.3%	92.6%
Extra Trees	96.4%	95%
Extra Trees Using GridSearchCV	96.4%	95%
Voting Classifier with Boost	94.6%	92.3%

Misclassification Rate



$$\frac{FP + FN}{TP+TN+FP+FN}$$

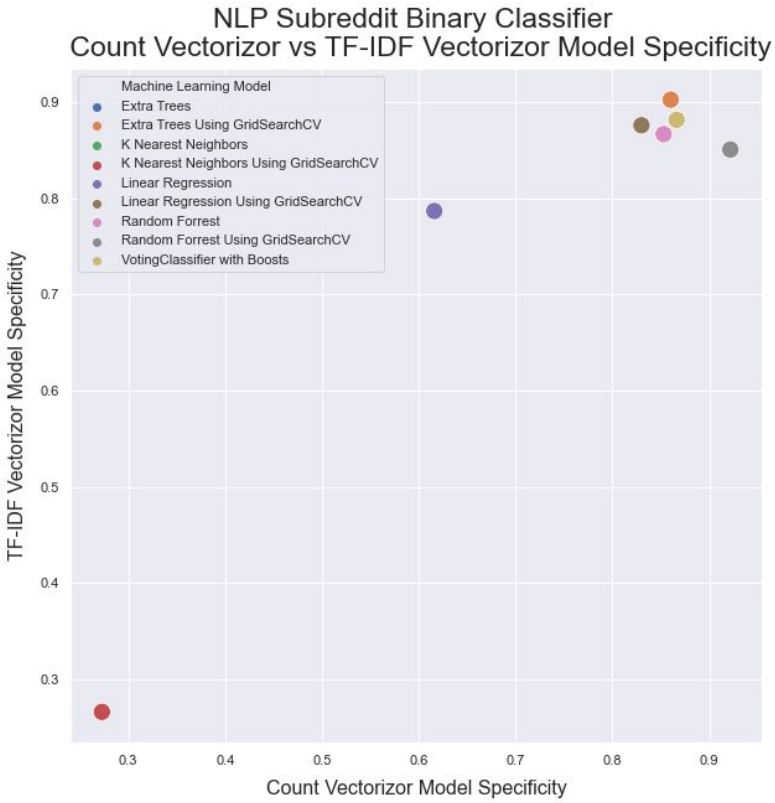
Model	TF-IDF	CVEC
Linear Regression	8.1%	16.5%
Linear Regression Using GridSearchCV	6.6%	5.8%
K Nearest Neighbor	72.2%	69.7%
K Nearest Neighbor Using GridSearchCV	72.2%	69.7%
Random Forest	5%	5.2%
Random Forest Using GridSearchCV	5.7%	7.4%
Extra Trees	3.6%	5%
Extra Trees Using GridSearchCV	3.6%	5%
Voting Classifier with Boost	5.3%	7.7%



Sensitivity

$$\frac{TP}{TP+FN}$$

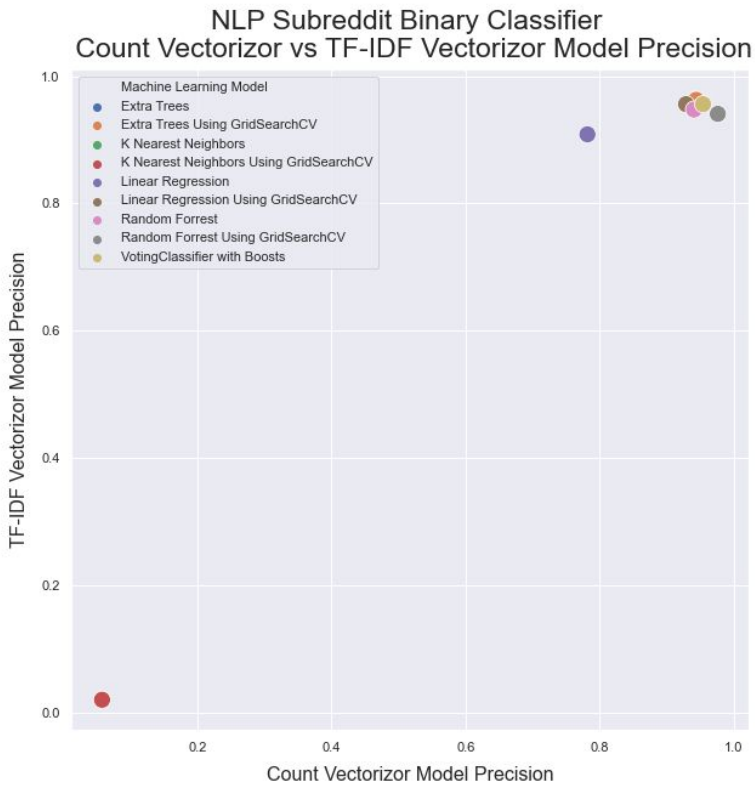
Model	TF-IDF	CVEC
Linear Regression	98%	99.4%
Linear Regression Using GridSearchCV	95.4%	99.1%
K Nearest Neighbor	100%	97.1%
K Nearest Neighbor Using GridSearchCV	100%	97.1%
Random Forest	98.4%	98.8%
Random Forest Using GridSearchCV	98.1%	92.7%
Extra Trees	98.8%	98.8%
Extra Trees Using GridSearchCV	98.8%	98.8%
Voting Classifier with Boost	97.1%	94.2%



Specificity

$$\frac{TN}{TN+FP}$$

Model	TF-IDF	CVEC
Linear Regression	78.7%	61.6%
Linear Regression Using GridSearchCV	87.6%	83%
K Nearest Neighbor	26.6%	27.3%
K Nearest Neighbor Using GridSearchCV	26.6%	27.3%
Random Forest	86.6%	85.3%
Random Forest Using GridSearchCV	85%	92.2%
Extra Trees	90%	86%
Extra Trees Using GridSearchCV	90%	86%
Voting Classifier with Boost	88.1%	86.6%



Precision

$$\frac{TP}{TP+FP}$$

Model	TF-IDF	CVEC
Linear Regression	90.9%	78.2%
Linear Regression Using GridSearchCV	95.6%	92.9%
K Nearest Neighbor	2.0%	5.8%
K Nearest Neighbor Using GridSearchCV	2.0%	5.8%
Random Forest	94.8%	94.1%
Random Forest Using GridSearchCV	94.1%	97.6%
Extra Trees	96.3%	94.4%
Extra Trees Using GridSearchCV	96.3%	94.4%
Voting Classifier with Boost	95.6%	95.4%

Conclusion / Q&A

- The K Nearest Neighbors model performed terribly by the likes of an approximate 60% difference in accuracy and specificity. It was the best at misclassifying in both vectorized dataframes. K Nearest Neighbors might not have performed well due to it being a non-parametric model.

- I believe the Random Forest and Extra Trees using GridSearchCV models to have performed the best in regards to classification metrics and would strongly consider using those models first for binary classification problems.

- Of the vectorizers, I received the best accuracy rates, lowest misclassification rates, highest specificities, best precisions, and best sensitivity metrics, with the exception of K Nearest Neighbors on all classification metrics, using TF-IDF, however it does come with some caveats as it took longer to work as this vectorizor is computationally heavy and also creates quite a large datafile.