NLP & APIs

Using Machine Learning
to classify reddit posts

Problem Statement:

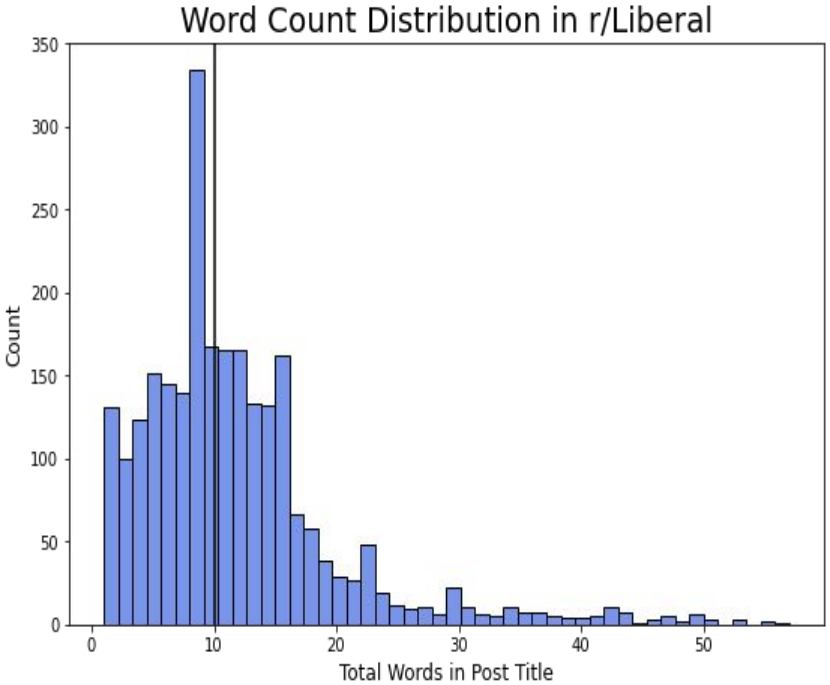
When given a dataframe of posts from 2 different subreddits, can Python use NLP to classify and differentiate between them?

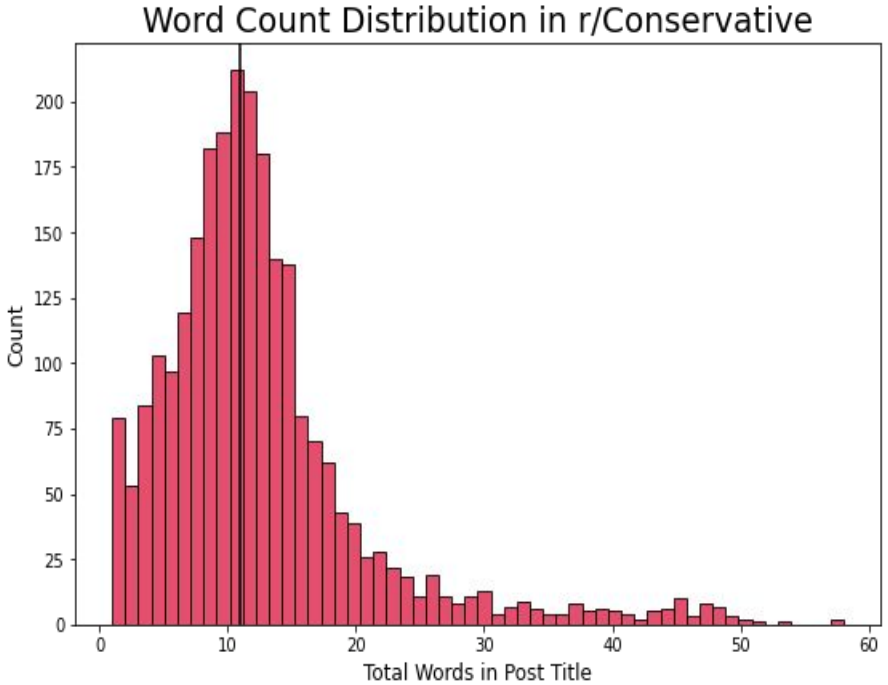
Data Used:

2500 posts taken from r/liberal

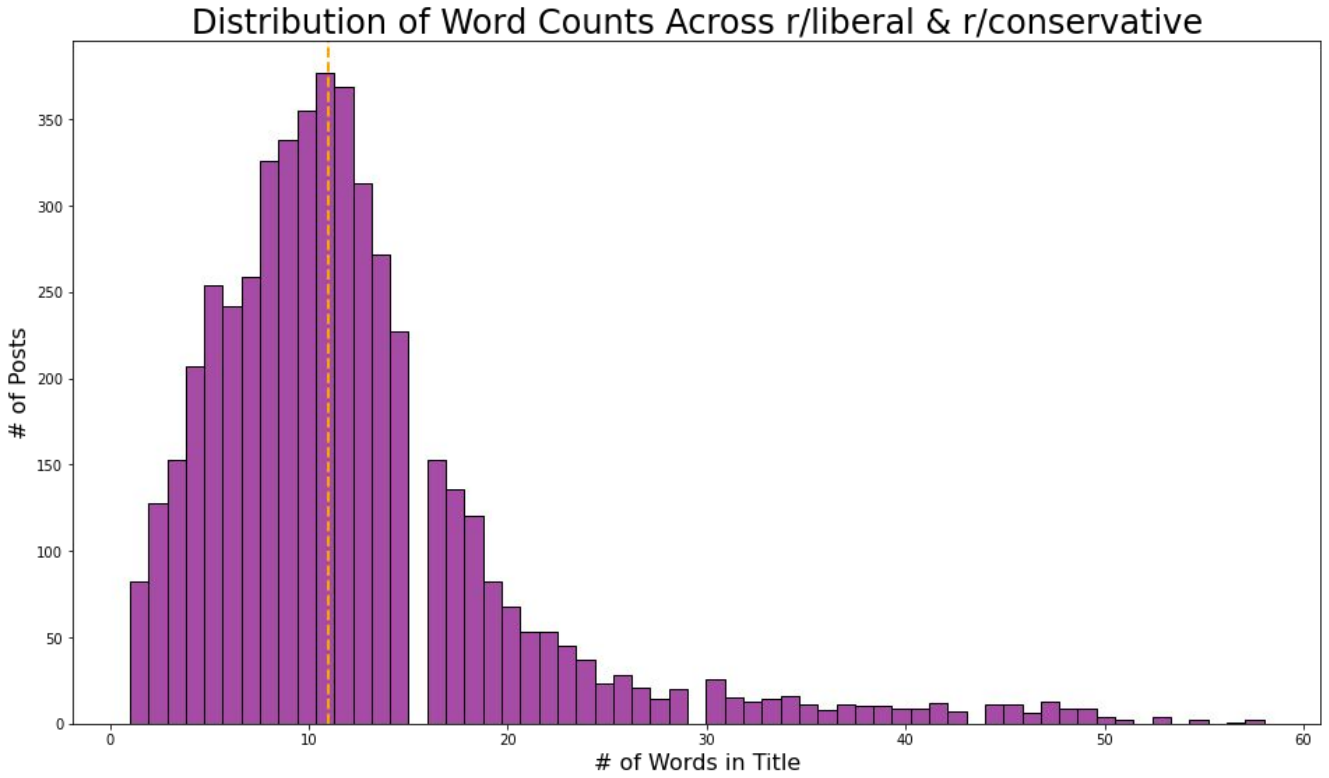
2500 posts taken from r/conservative

Word Counts:


Word Count Distribution in r/Liberal
Count
Total Words in Post Title
0
50
100
150
200
250
300
350
0
10
20
30
40
50


Word Count Distribution in r/Conservative
Count
Total Words in Post Title
0
25
50
75
100
125
150
175
200
0
10
20
30
40
50
60

Word Counts:


Distribution of Word Counts Across r/liberal & r/conservative
# of Posts
# of Words in Title
0
50
100
150
200
250
300
350
0
10
20
30
40
50
60

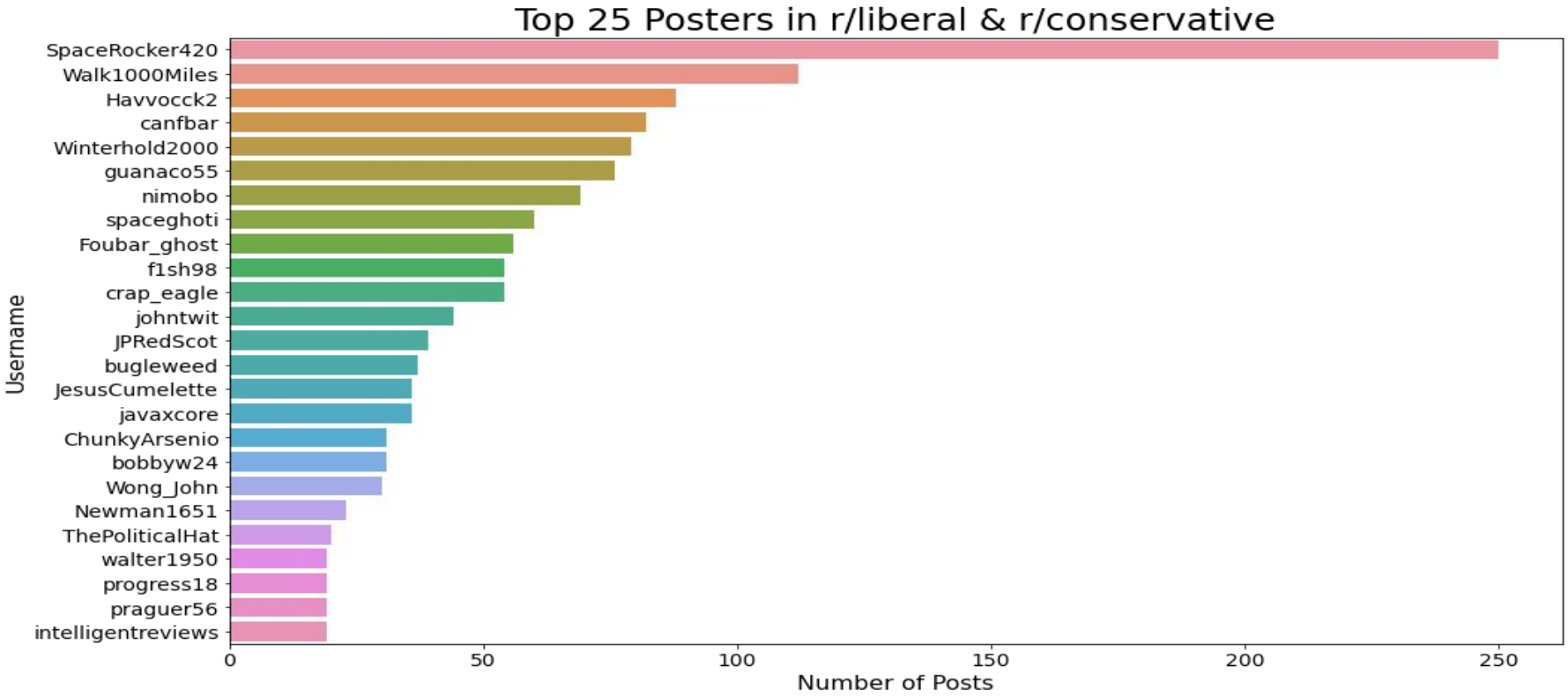
Repost Alert!

- Aggregated the number of reposts in both subreddits:
  - 94 total duplicated posts when extras were dropped
  - 61 came from r/conservative, 33 from r/liberal
  - Returns about 65% to 35% from one to another

Unique Users:

- Total unique users from both subreddits:
  - 2,216 unique users
  - 44.32% of total user base from collected posts is unique
- 834 Unique Users in r/conservative
- 1,398 Unique Users in r/liberal
- 17 Users who post in both r/liberal and r/conservative

User Posts:


Top 25 Posters in r/liberal & r/conservative
SpaceRocker420
Walk1000Miles
Havvocck2
canfbar
Winterhold2000
guanaco55
nimobo
spaceghoti
Foubar_ghost
f1sh98
crap_eagle
johntwit
JPRedScot
bugleweed
JesusCumelette
javaxcore
ChunkyArsenio
bobbyw24
Wong_John
Newman1651
ThePoliticalHat
walter1950
progress18
praguer56
intelligentreviews
Username
0
50
100
150
200
250
Number of Posts

Top Words:

r/liberal

trump	560
biden	258
election	145
capitol	110
liberal	88
president	75
republicans	71
vote	68
people	68
joe	67

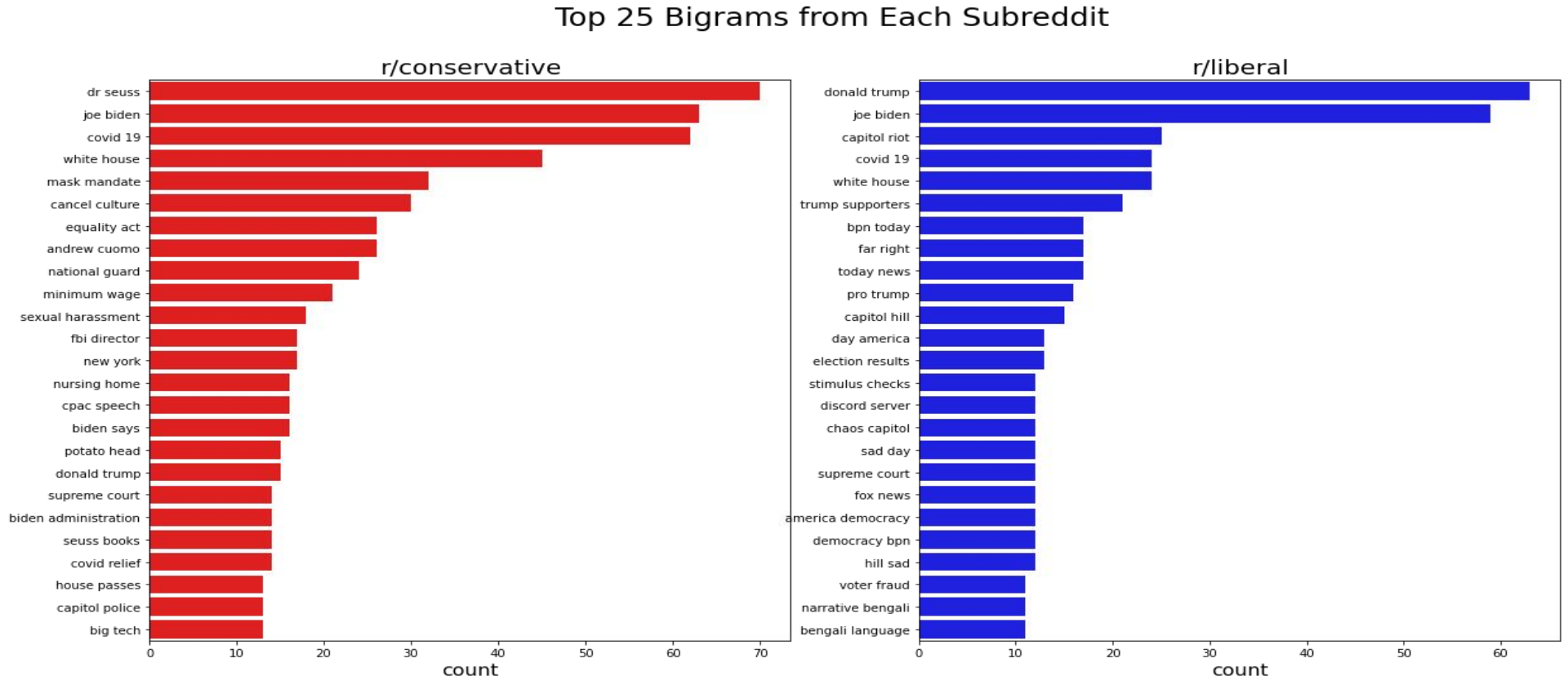
r/conservative

biden	375
trump	182
covid	157
cuomo	113
new	107
house	107
says	102
white	100
dr	86
texas	86

Total dataframe

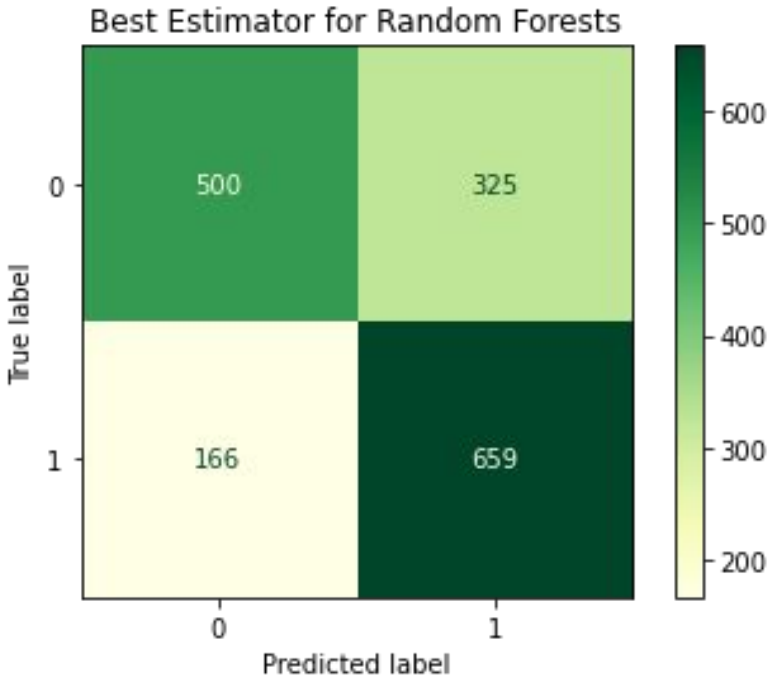
trump	742
biden	633
covid	210
election	199
capitol	180
new	166
says	164
white	154
house	151
joe	146

Bipartisan Bigrams:



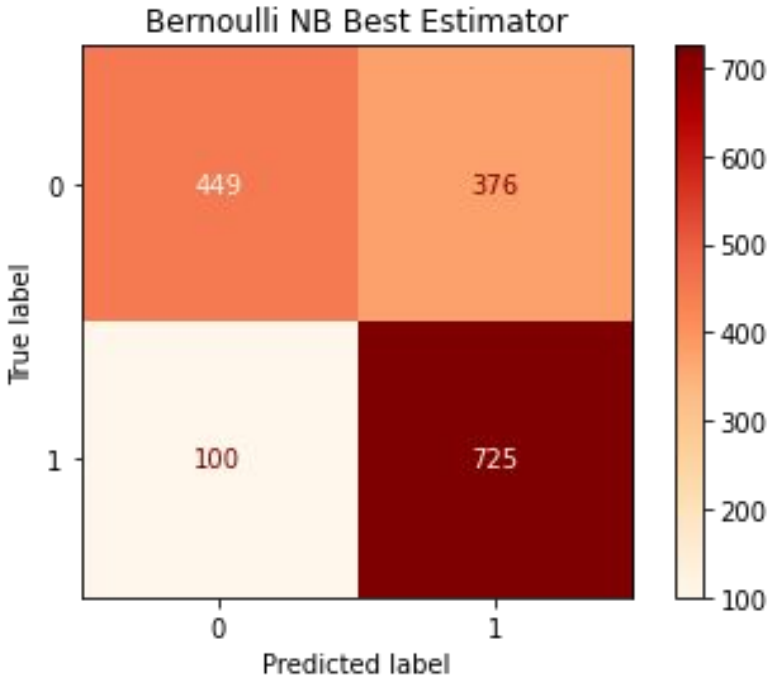
Modeling: Random Forest

- Scoring:
  - Training: **99.76% accuracy**
  - Testing: **70.24% accuracy**
- Steps:
  - Count Vectorizer
  - Random Forest
- Best Parameters:
  - N-gram range : (1,1)
  - Stop words: English
  - # Estimators: 200



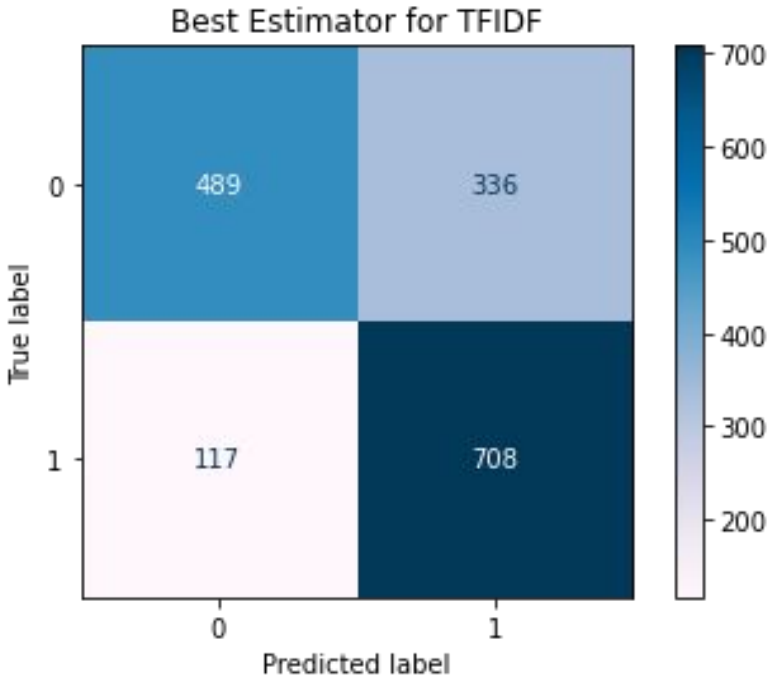
Model: Bernoulli NB w/ Count Vectorizer

- Scoring:
  - Training: **83.40% accuracy**
  - Testing: **71.15% accuracy**
- Steps:
  - Count Vectorizer
  - Bernoulli Naive Bayes
- Best Parameters:
  - Min DF: 2
  - Max DF: 95%
  - Max Features: 4000
  - N-Gram Range: (1,2)



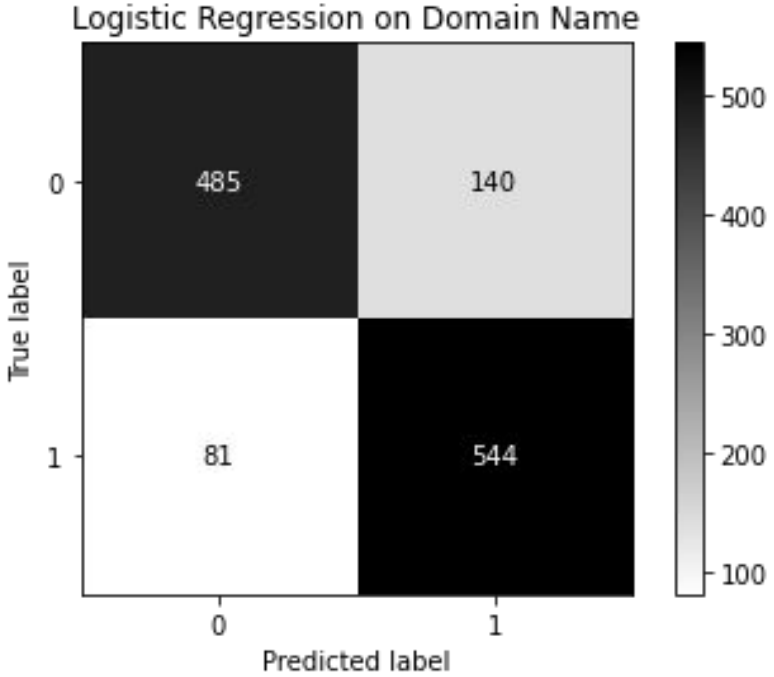
Bernoulli NB w/ TFIDF

- Best Model
- Scoring:
  - Training: **88.14% accuracy**
  - Testing: **72.55% accuracy**
- Steps:
  - TFIDF
  - Bernoulli Naive Bayes
- Best Parameters:
  - Max Features: 6500
  - Stop words: None
  - N-Gram Range: (1,1)



Model: Logistic Regression on Domain Name

- For Funsies
- Modeled on Domain name rather than text
- Scoring:
  - Training: **87.28% accuracy**
  - Testing: **82.48% accuracy**
- Best Testing Score of all Models



Conclusions & Recommendations:

- More difficult to predict than anticipated
- Try more parameter tuning and other models
- Polynomial Features