REDDIT NLP

PRESENTATION BY ROWAN SCHAEFER

TL;DR

The goal of this analysis is to classify text posts as either belonging to one subreddit or another, using natural language processing.

Success will be evaluated using the model's accuracy score in correctly classifying the posts, with a second goal of trying many models to obtain the highest accuracy score possible.

RESEARCH QUESTIONS

>

```
lass 'pandas.core.frame.DataFrame'>
t64Index: 3396 entries, 0 to 98
ta columns (total 12 columns):
   Column                  Non-Null Cou
-  ------                  ------------
   author                  3396 non-nul
   created_utc             3396 non-nul
   id                      3396 non-nul
   is_created_from_ads_ui  1098 non-nul
   num_comments            3396 non-nul
   score                   3396 non-nul
   selftext                3388 non-nul
   subreddit               3396 non-nul
   title                   3396 non-nul
   total_awards_received   3396 non-nul
   upvote_ratio            2398 non-nul
   url                     3396 non-nul
/pes: float64(1), int64(4), object(7)
nory usage: 344.9+ KB
```

```
.stdfs=[]
or key, val in range_ge
   i = query('learnsql'
   print('pulled 100 ro
   listdfs.append(i)
   time.sleep(10)
```

<

- What preprocessing steps lead to highest accuracy in classifying subreddits?
- Which classification models and hyperparameters lead to highest accuracy and precision in classifying subreddits?

SUBREDDITS

r/learnpython and r/learnsql are mostly filled with questions and resources for learners.

r/ learnpython

- 613,000 subscribers
- 3,396 rows of data

r/ learnsql

- 17,700 subscribers
- 3,407 rows of data
- Had to pull from API differently to avoid class imbalance.

THE PROCESS

find data

>

cleaning/EDA

>

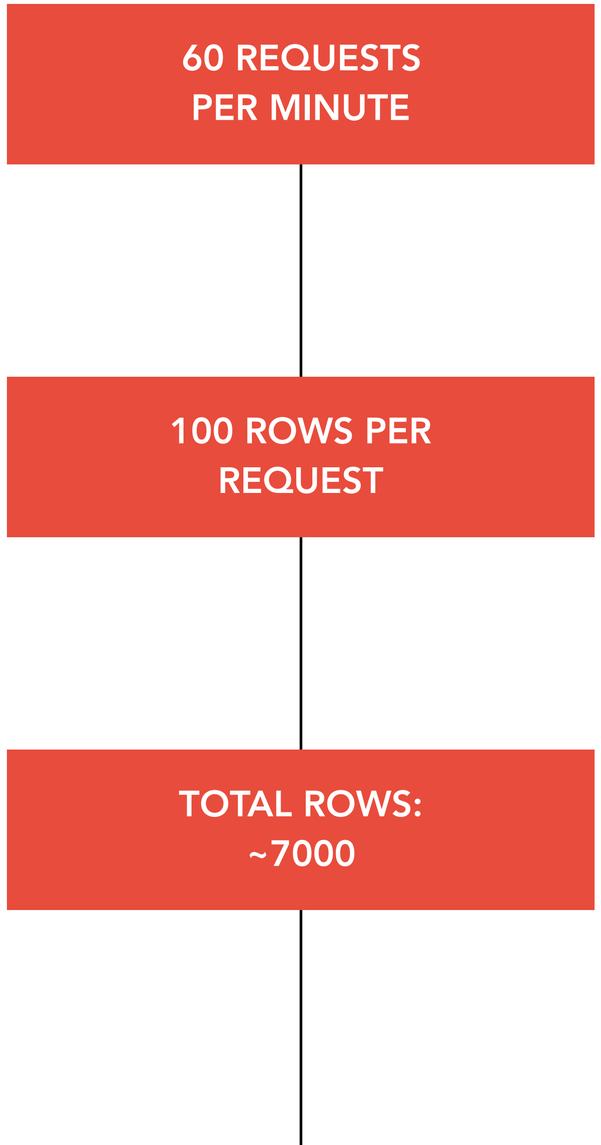
preprocessing

>

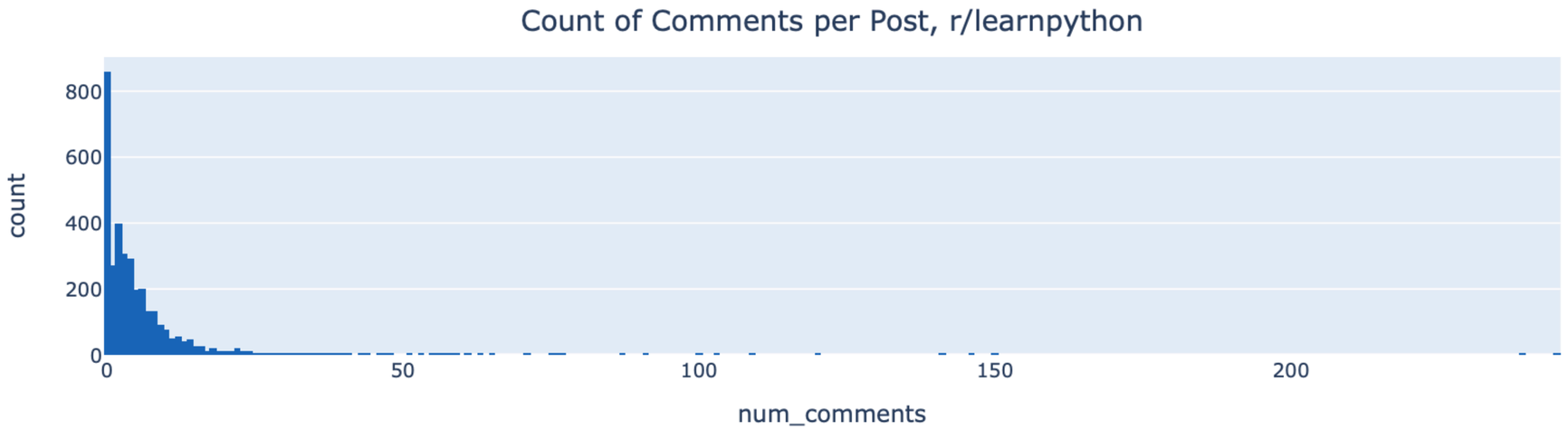
modeling

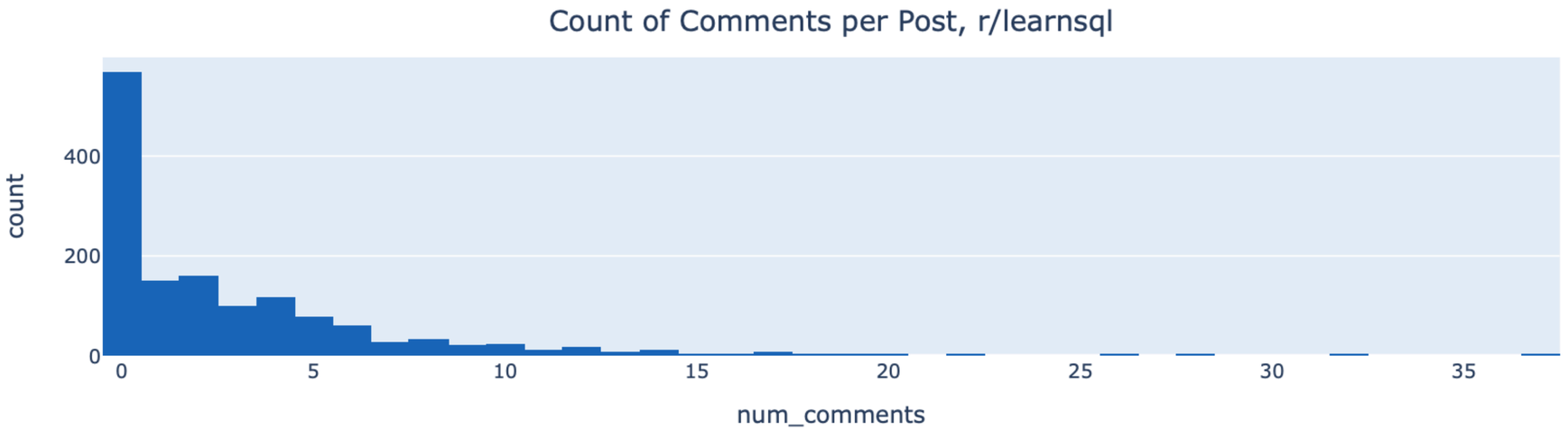
GETTING DATA

To get data, I used the requests library with reddit's API, **PushshiftAPI**. In order to automate pulls, I wrote a function that looped through "before" and "after" criteria for the date of the reddit posts, and set a sleep timer in between so I didn't overwhelm the API.

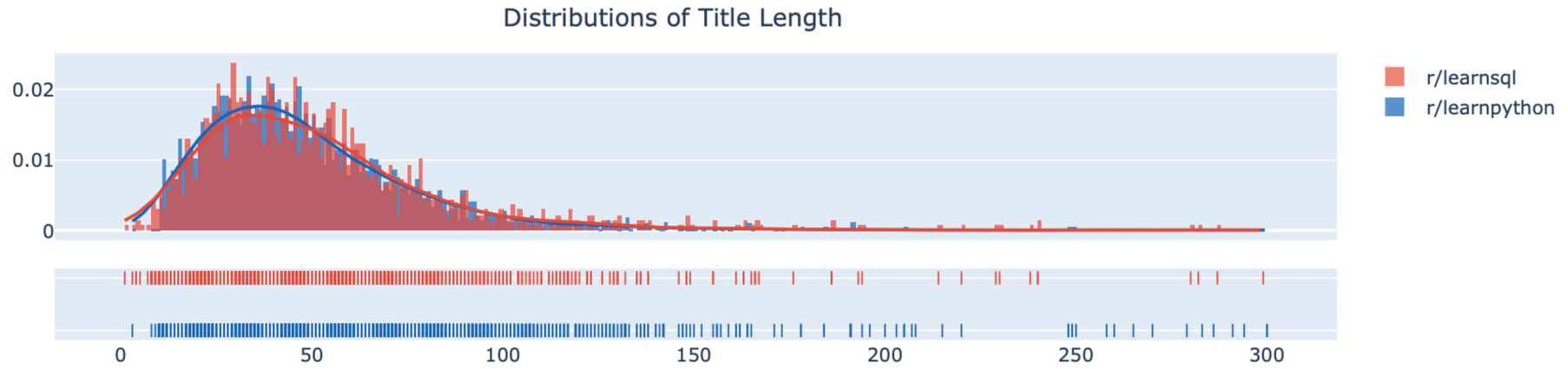


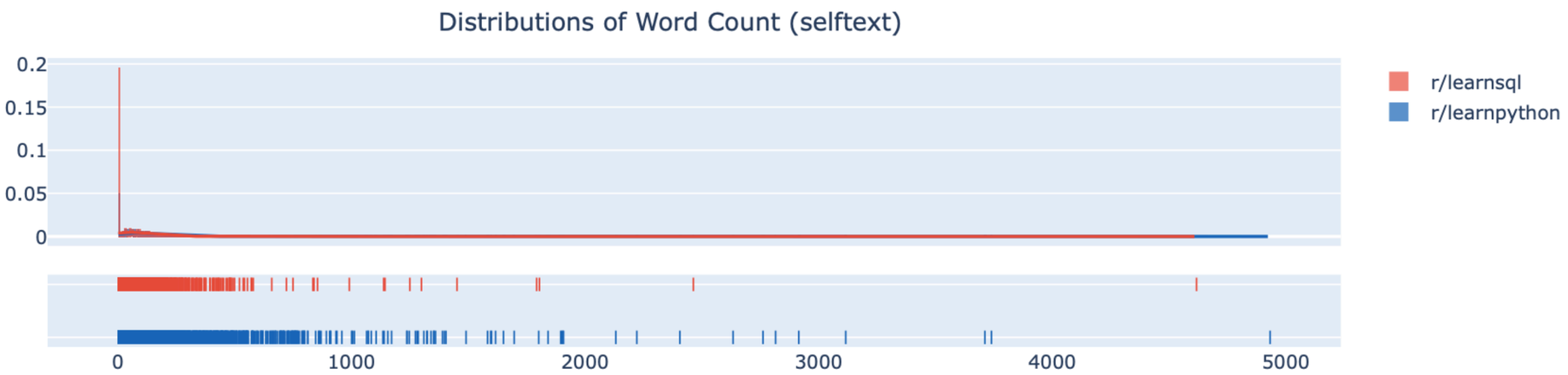
EXPLORATORY ANALYSIS





EXPLORATORY ANALYSIS

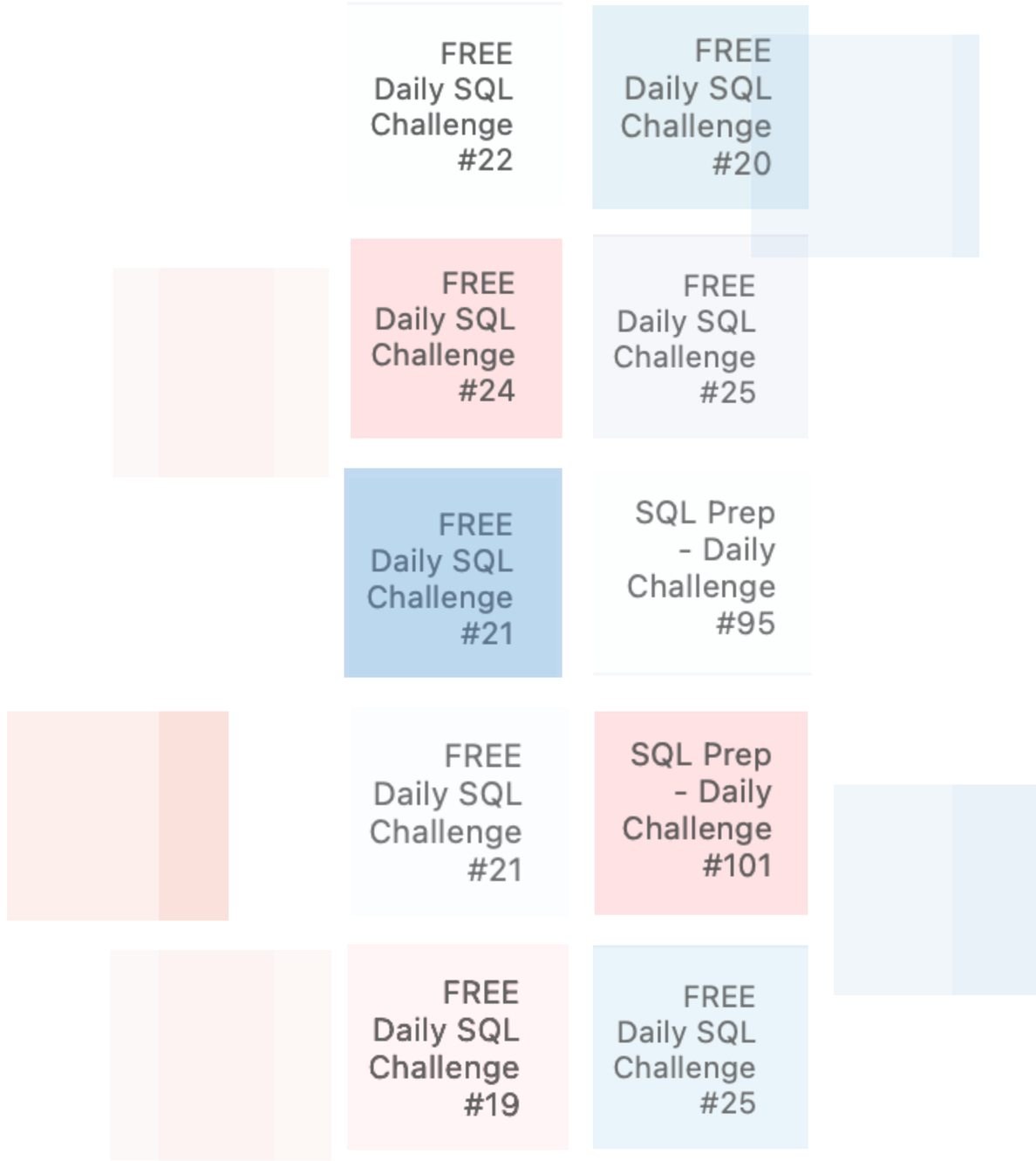


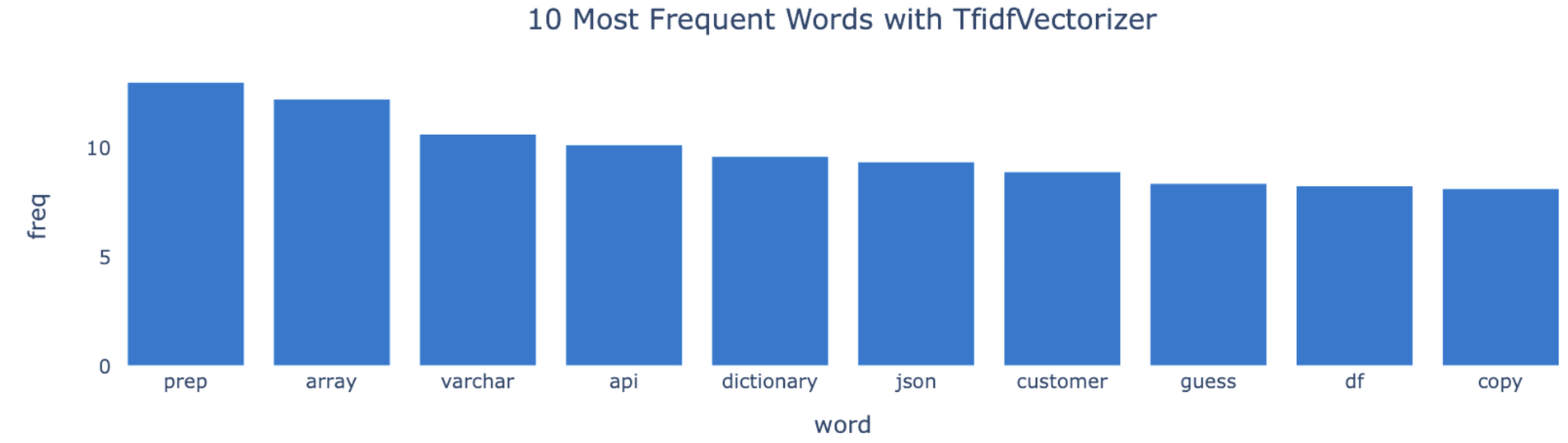


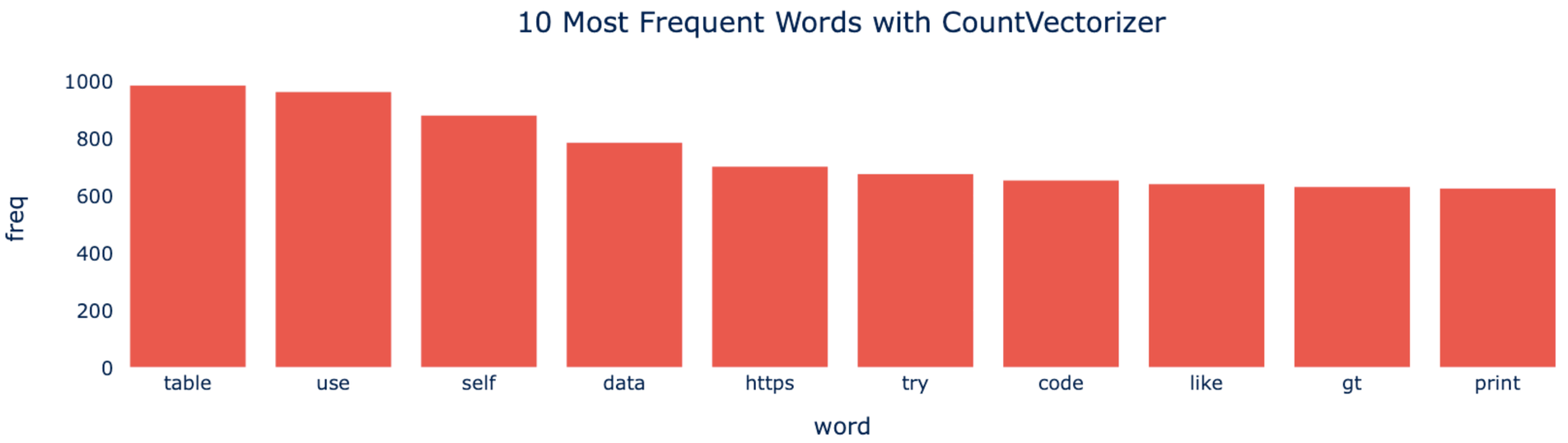
PREPROCESSING

AND SOME DATA CLEANING

- Removed 170 posts from user Andre380
- Removed [‘removed’] selftext
- Removed numbers
- Removed giveaway words: SQL, postgre, python, snowflake, etc
- Models performed best with no bigrams




10 Most Frequent Words with TfidfVectorizer
freq
10
5
0
prep
array
varchar
api
dictionary
json
customer
guess
df
copy
word


10 Most Frequent Words with CountVectorizer
freq
1000
800
600
400
200
0
table
use
self
data
https
try
code
like
gt
print
word

BEST MODELS

1 > 2 > 3

Random Forest

Accuracy: **91.5%**

Precision: **91.5%**

Multinomial Bayes

Accuracy: **90.4%**

Precision: **93%**

Logistic Regression

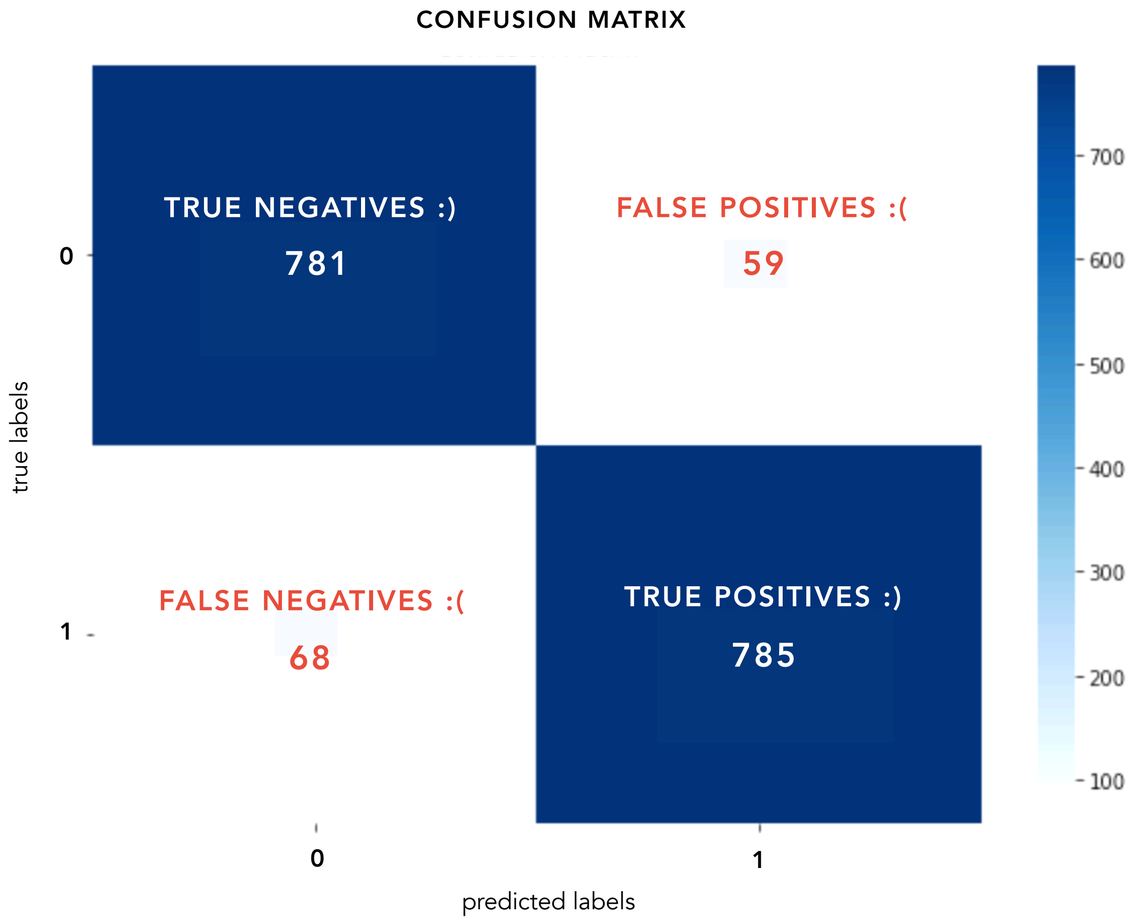
Accuracy: **90.5%**

Precision: **90.9%**

ENSEMBLE MODEL

Accuracy score: 92.5% / Precision score: 93%

ENSEMBLE MODEL



SUGGESTIONS

For someone else looking to solve a similar classification problem, I'd recommend the following:

- Try using boosting algorithms on highest-performing models
- Make sure one user isn't carrying the whole subreddit.
- Consider finding two subreddits that are more "evenly matched", or be careful about balancing classes.

	ಠ_ಠ
5475	1

THANK YOU