Sentiment Analysis of Tweets about Tanishq Ad Fiasco

Osman Mamun

Data Scientist, Pacific Northwest National Laboratory

boycott
tanishq
jewelry
diwali
hindu
festival
cracker
bollywood
celebrate
laxmi
ad
arnab
people
firecracker
jewellery
amazon
justice
diya
support
buy
local
hai
want
brand
also
life
advice
culture
threat
pollution
stop
india
amp
religion
time
one
deepawali
anti
product
trending
bomb
muslim
say
day
gyan
campaign
make
giving
need
christmas
aishwarya
gold
dharma
business
teach
like
give
love
keep
burst
eid
bhi
ki
ban
vcs
monday
advertisement
tata
let
good
never
indian
nah
trend
buying
propaganda
shame
environment
dhanteras
customer
motivation
advise
know
family
year
go
kya
twitter
druggie
ek
making
r
even
back
light
delhi
gyaan
free
ye
u
get
hinduism
fire
new
tell
please
sentiment
secularism
animal
tradition

Motivation

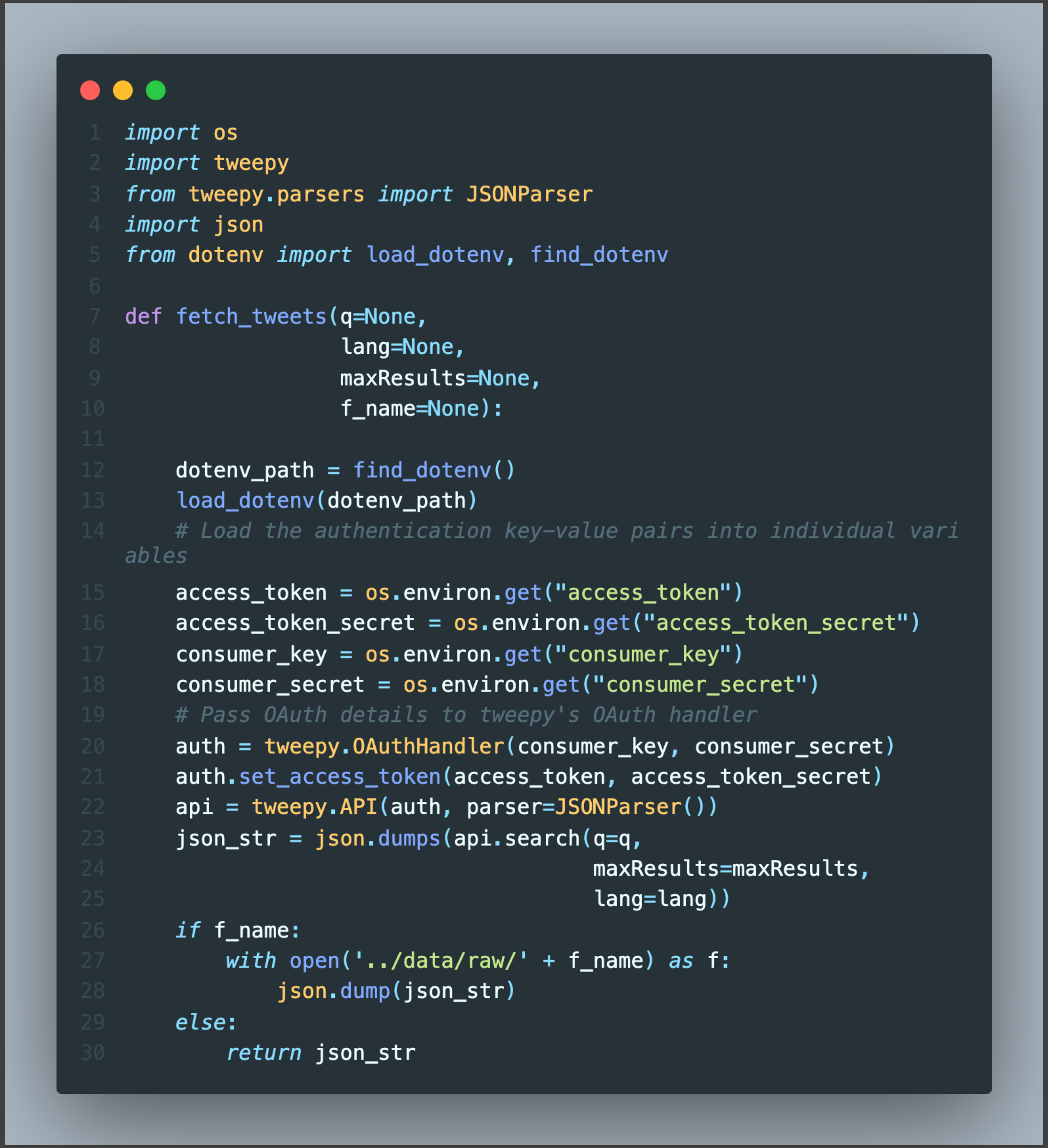
- During Diwali last year, Tanishq, an Indian Jewelry brand, made an ad featuring interfaith marriage which led to a widespread controversy throughout India. As a result, in initiative to boycott Tanishq was carried out on Twitter with the #BoycottTanishq slogan.
- It is essential for company to understand public sentiment to quickly correct their actions to avoid any negative consequences for the business.
- Here, I analyzed the twitter data related to this ad controversy to identify positive and negative twitter sentiment.

Approach

- Collect Twitter Data
- Data Cleaning and Visualization
- Manually Hand Label Data
- Analyze the Feasibility of Pretrained Sentiment Analyzer
- Unsupervised Learning Algorithm
- Supervised Learning Algorithm
- Weak Supervision Model with Pseudo Labeling
- Synthetic Minority Oversampling Technique (SMOTE)

Collect Twitter Data

- Tweepy python package is used to scrape Twitter data with Twitter API
- Sensitive information, i.e., access token, consumer key, secrets etc., are stored in .env file for easy retrieval by the code
- Data can be saved as json file or converted to a pandas dataframe to store selected information.



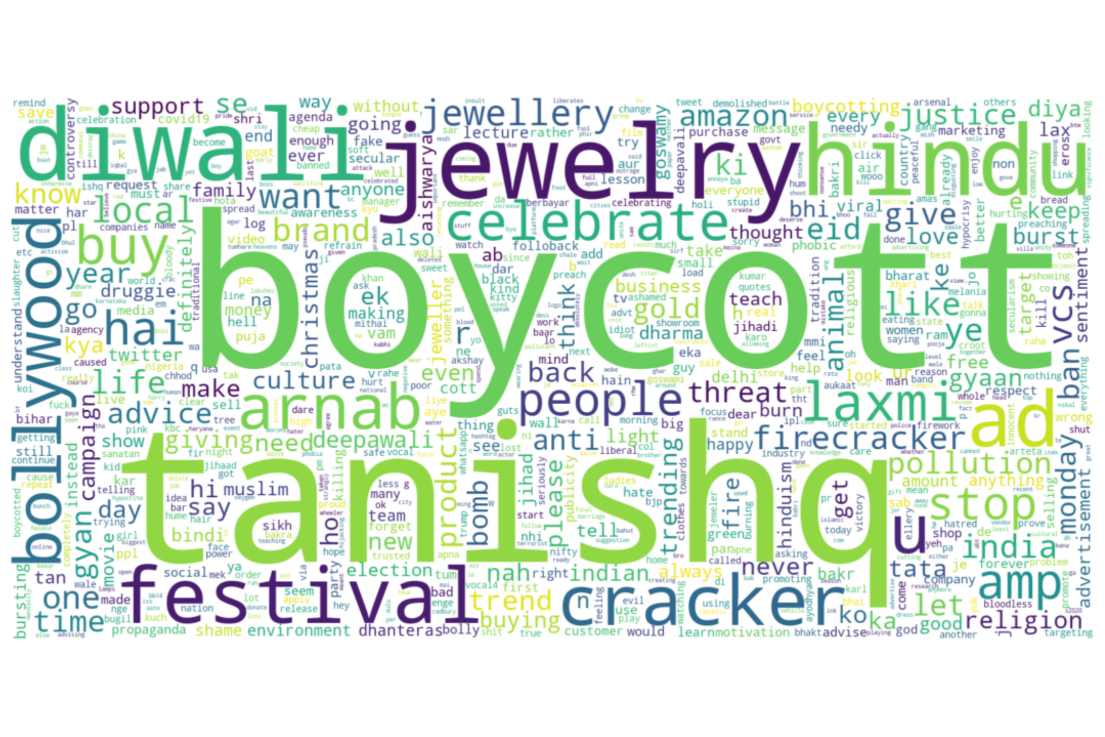
```python
import os
import tweepy
from tweepy.parsers import JSONParser
import json
from dotenv import load_dotenv, find_dotenv

def fetch_tweets(q=None,
                 lang=None,
                 maxResults=None,
                 f_name=None):

    dotenv_path = find_dotenv()
    load_dotenv(dotenv_path)
    # Load the authentication key-value pairs into individual variables
    access_token = os.environ.get("access_token")
    access_token_secret = os.environ.get("access_token_secret")
    consumer_key = os.environ.get("consumer_key")
    consumer_secret = os.environ.get("consumer_secret")
    # Pass OAuth details to tweepy's OAuth handler
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth, parser=JSONParser())
    json_str = json.dumps(api.search(q=q,
                                     maxResults=maxResults,
                                     lang=lang))
    if f_name:
        with open('../data/raw/' + f_name) as f:
            json.dump(json_str)
    else:
        return json_str
```

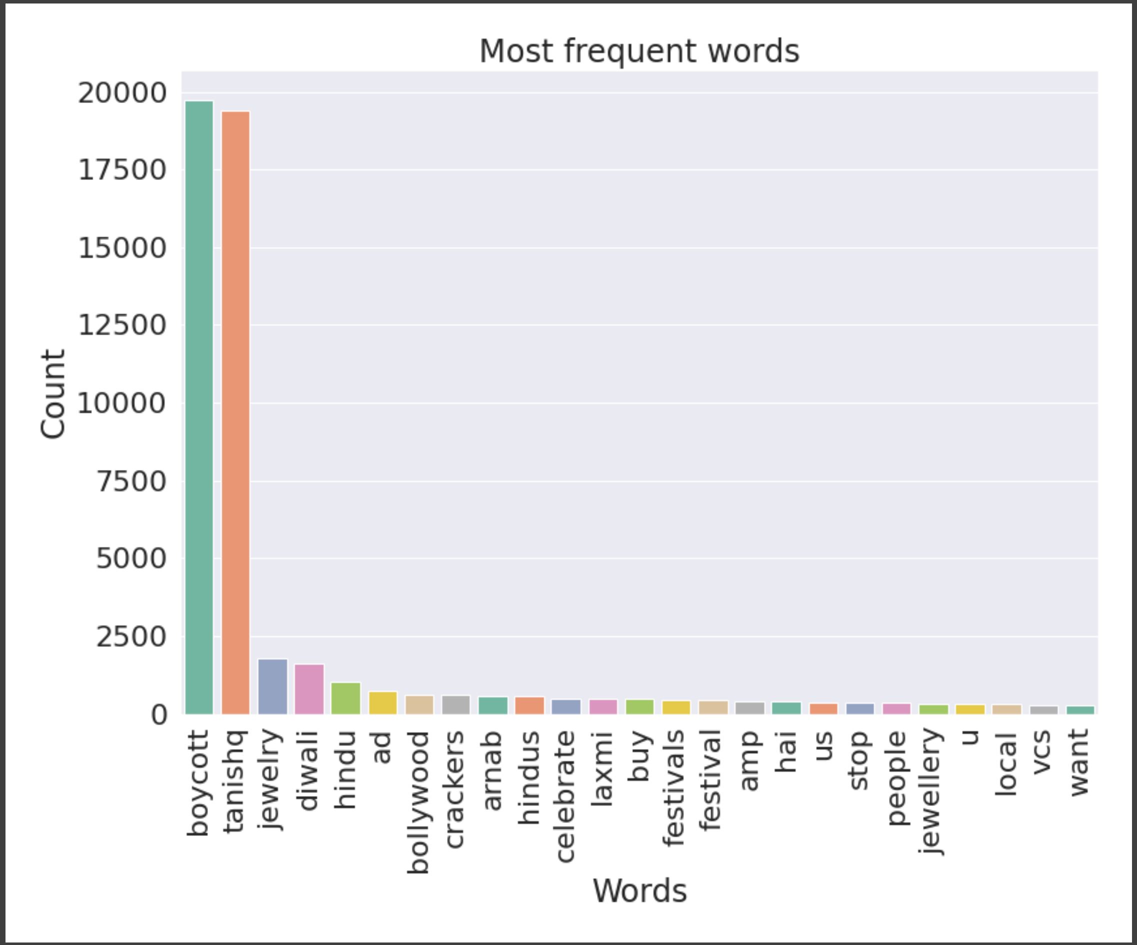
Data Cleaning and Visualization

- Hashtags (#), URL, emoji, mentions, and numbers were removed using tweet preprocessing python package.
- All the texts were converted to lowercase and any special characters were removed.
- WordNetLemmatizer and TweetTokenizer from nltk package were used to lemmatize and tokenize the sentences.

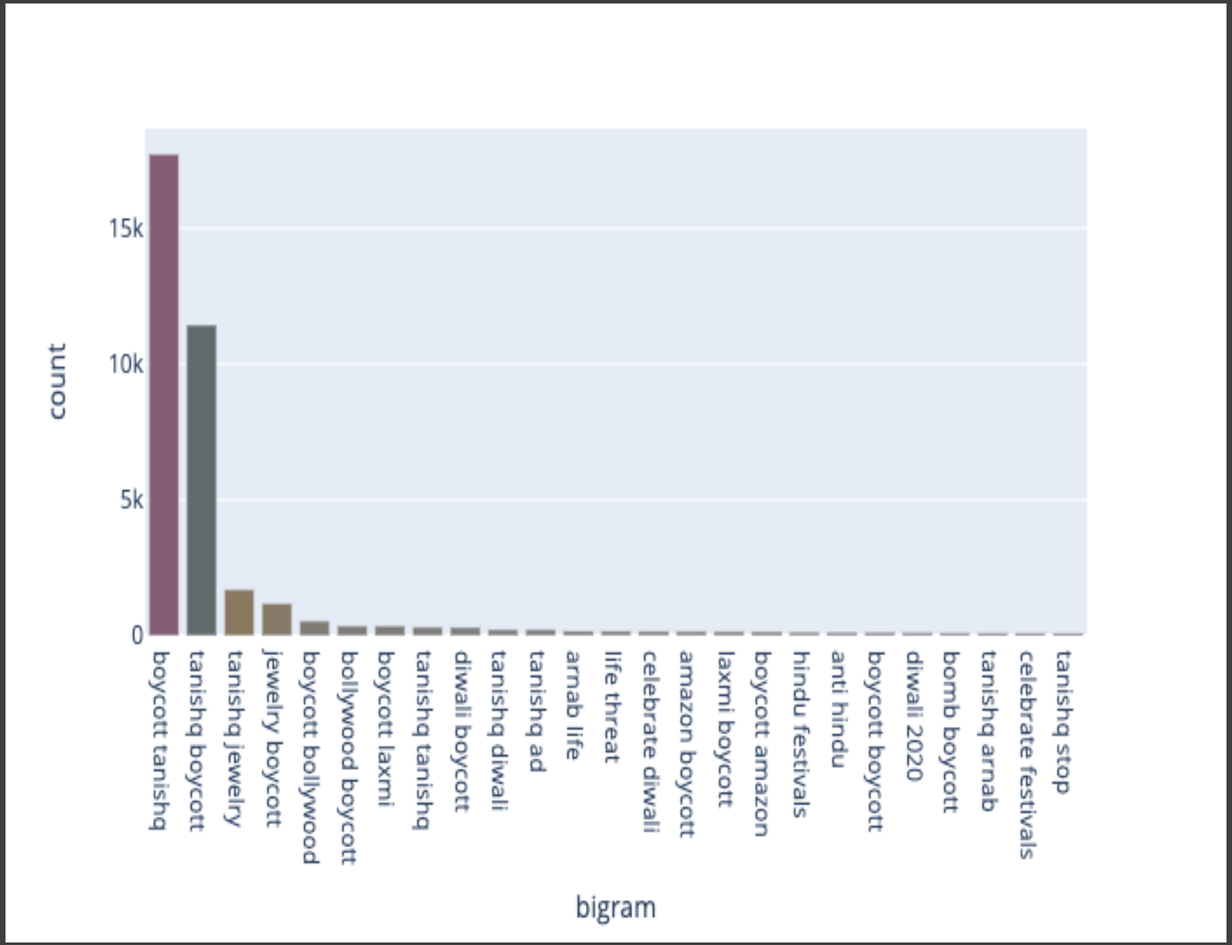


Data Cleaning and Visualization

Word Frequency



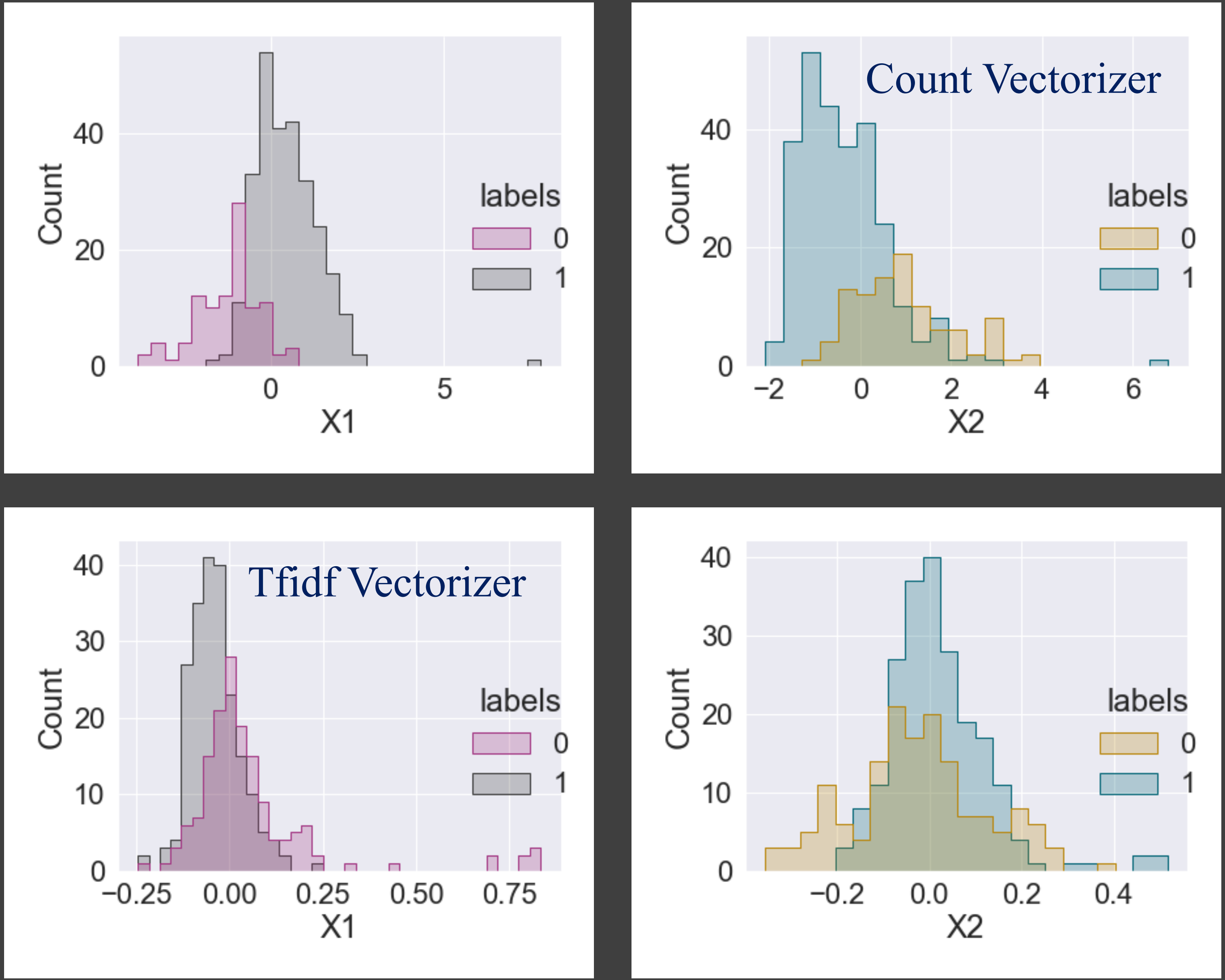
Bigram Frequency



Manually Hand Label Data

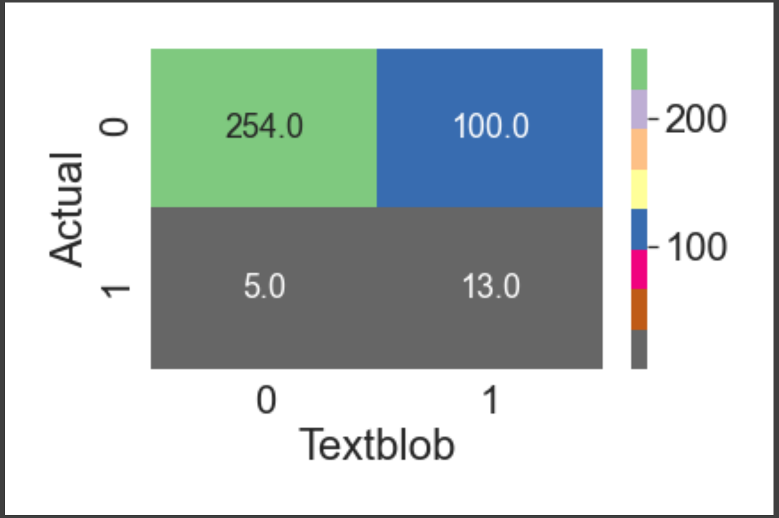
- In order to quantify the performance of different approaches, I manually hand label about 1000 data. It is of note that only about 300 labels were either positive or negative.
- Due to the ambiguity and sarcasmic nature of tweets, it was very difficult to label tweets as either positive or negative, even by a human observer.
- For better modeling for business application, it is recommended to manually hand label about 2-3K more data. However, in the interest of time efficiency, we refrained from further hand labeling the data.

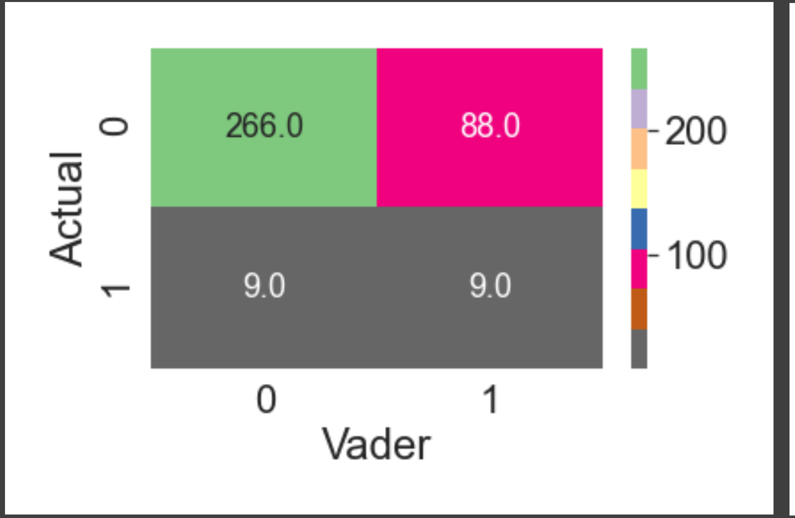
Why is It a Challenging Problem

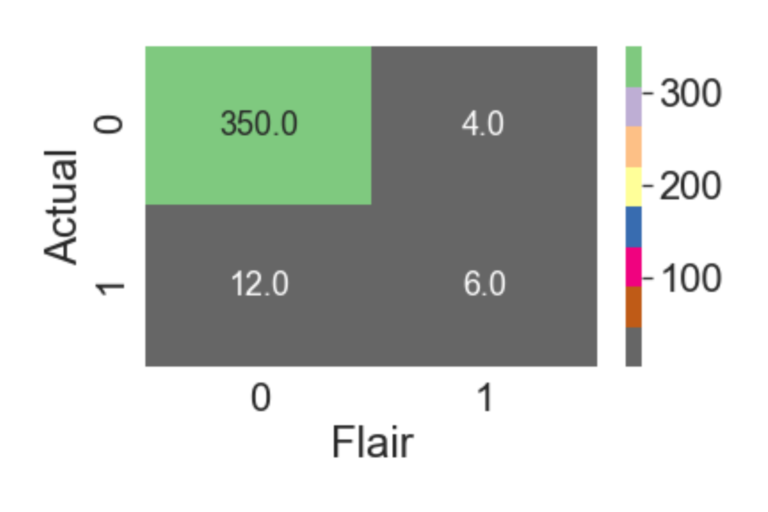


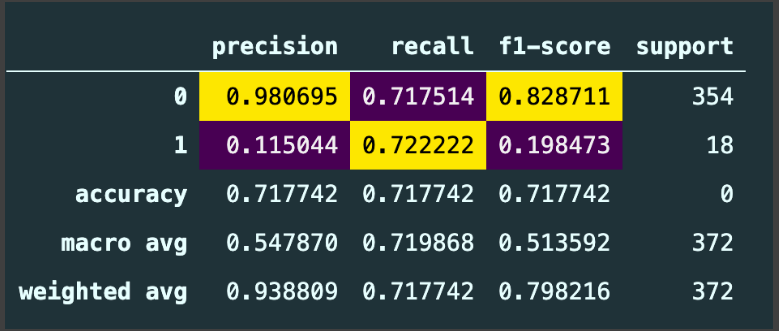
Significance feature space overlapping with both the count vectorizer and tfidf vectorizer approach.

Analyze the Feasibility of Pretrained Sentiment Analyzer

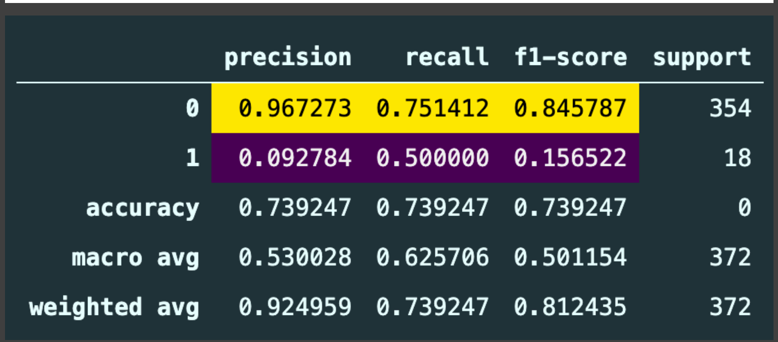




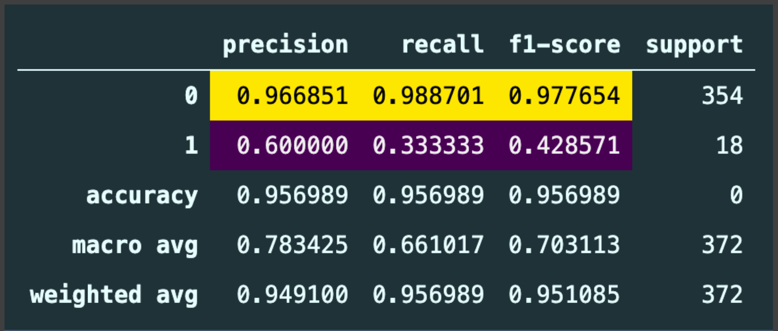




	precision	recall	f1-score	support
0	0.980695	0.717514	0.828711	354
1	0.115044	0.722222	0.198473	18
accuracy	0.717742	0.717742	0.717742	0
macro avg	0.547870	0.719868	0.513592	372
weighted avg	0.938809	0.717742	0.798216	372



	precision	recall	f1-score	support
0	0.967273	0.751412	0.845787	354
1	0.092784	0.500000	0.156522	18
accuracy	0.739247	0.739247	0.739247	0
macro avg	0.530028	0.625706	0.501154	372
weighted avg	0.924959	0.739247	0.812435	372



	precision	recall	f1-score	support
0	0.966851	0.988701	0.977654	354
1	0.600000	0.333333	0.428571	18
accuracy	0.956989	0.956989	0.956989	0
macro avg	0.783425	0.661017	0.703113	372
weighted avg	0.949100	0.956989	0.951085	372

ROC AUC: 0.72

ROC AUC: 0.63

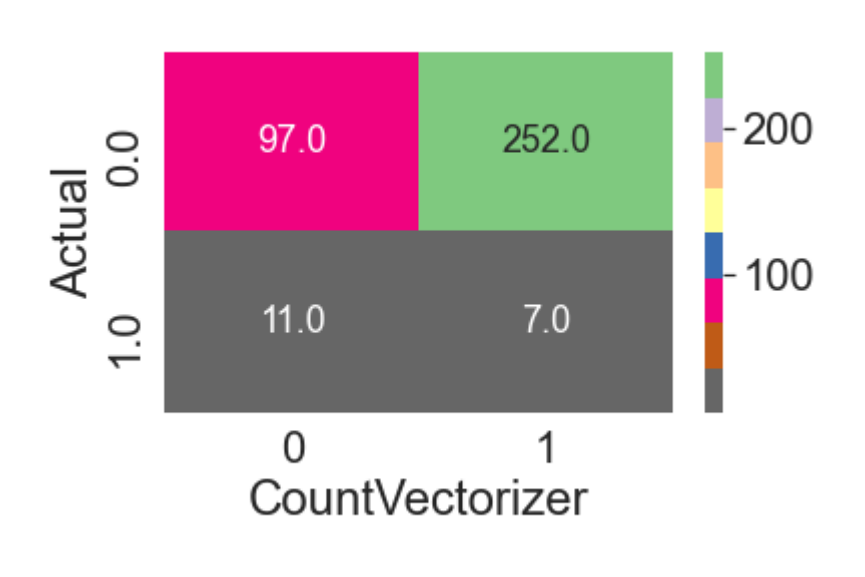
ROC AUC: 0.66

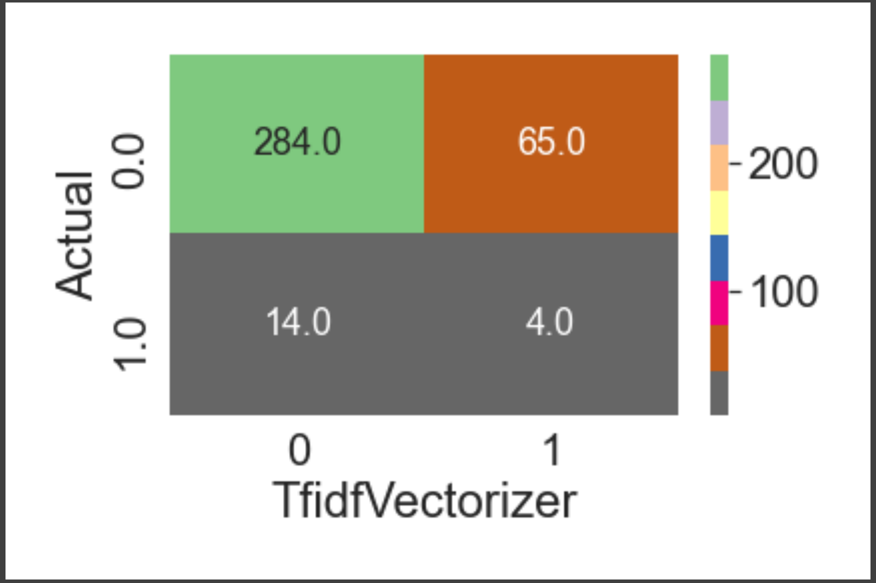
Textblob

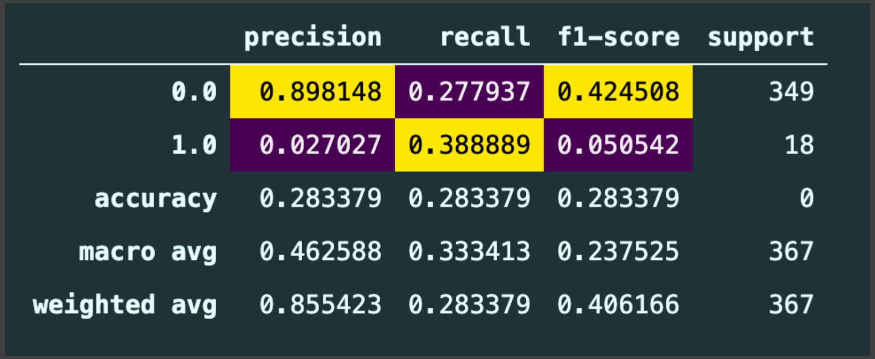
Vader

Flair

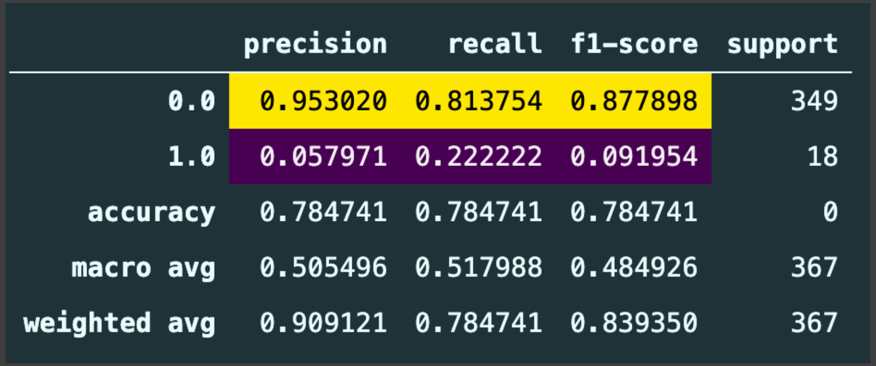
Unsupervised Learning Algorithm







	precision	recall	f1-score	support
0.0	0.898148	0.277937	0.424508	349
1.0	0.027027	0.388889	0.050542	18
accuracy	0.283379	0.283379	0.283379	0
macro avg	0.462588	0.333413	0.237525	367
weighted avg	0.855423	0.283379	0.406166	367



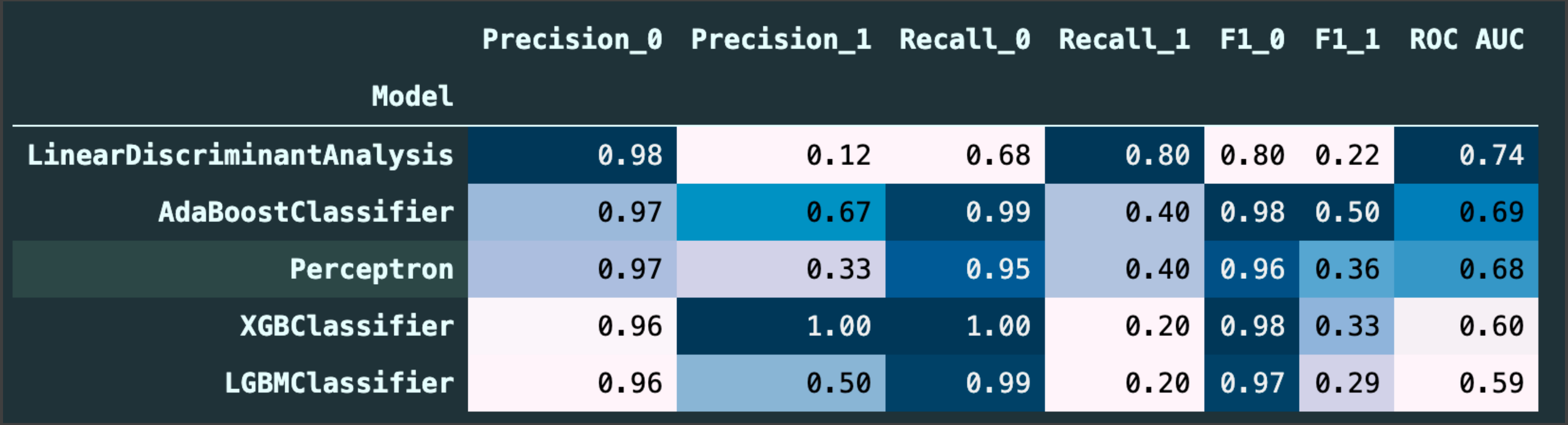
	precision	recall	f1-score	support
0.0	0.953020	0.813754	0.877898	349
1.0	0.057971	0.222222	0.091954	18
accuracy	0.784741	0.784741	0.784741	0
macro avg	0.505496	0.517988	0.484926	367
weighted avg	0.909121	0.784741	0.839350	367

ROC AUC: 0.33

ROC AUC: 0.51

Pretrained Model Performance are inarguably better than the unsupervised learning algorithm.

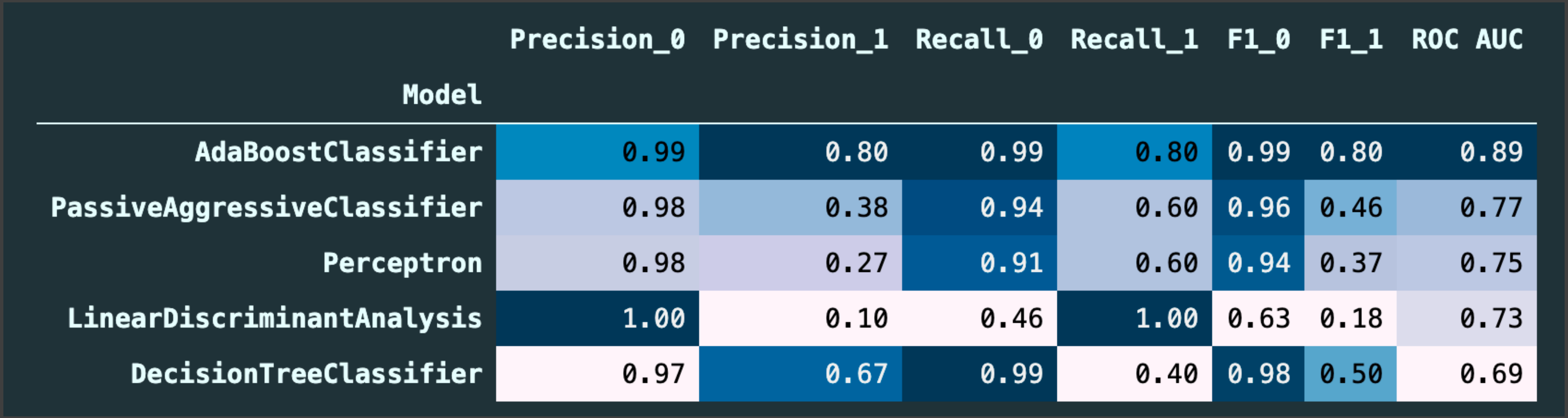
Supervised Learning Algorithm (Count Vectorizer)



Model	Precision_0	Precision_1	Recall_0	Recall_1	F1_0	F1_1	ROC AUC
LinearDiscriminantAnalysis	0.98	0.12	0.68	0.80	0.80	0.22	0.74
AdaBoostClassifier	0.97	0.67	0.99	0.40	0.98	0.50	0.69
Perceptron	0.97	0.33	0.95	0.40	0.96	0.36	0.68
XGBClassifier	0.96	1.00	1.00	0.20	0.98	0.33	0.60
LGBMClassifier	0.96	0.50	0.99	0.20	0.97	0.29	0.59

- Very high precision and recall for negative sentiment with most of the classifier
- ROC AUC is also reasonably high
- Low precision and recall for positive sentiment class; however, XGBoost has high precision and Linear Discriminant Analysis has high recall (need proper cross-validation)

Supervised Learning Algorithm (Tfidf Vectorizer)

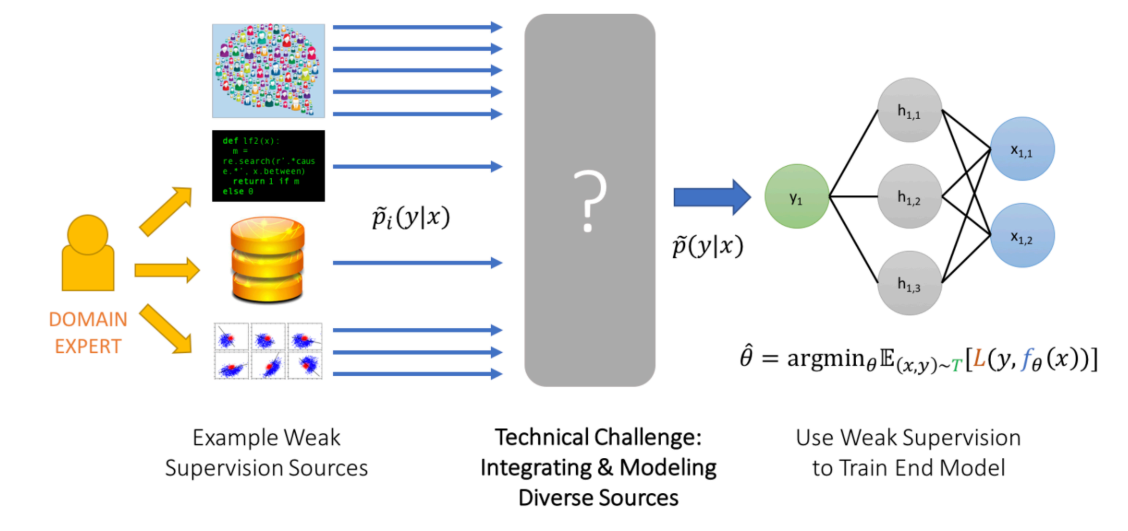


Model	Precision_0	Precision_1	Recall_0	Recall_1	F1_0	F1_1	ROC AUC
AdaBoostClassifier	0.99	0.80	0.99	0.80	0.99	0.80	0.89
PassiveAggressiveClassifier	0.98	0.38	0.94	0.60	0.96	0.46	0.77
Perceptron	0.98	0.27	0.91	0.60	0.94	0.37	0.75
LinearDiscriminantAnalysis	1.00	0.10	0.46	1.00	0.63	0.18	0.73
DecisionTreeClassifier	0.97	0.67	0.99	0.40	0.98	0.50	0.69

- Overall better performance than the count vectorizer
- ROC AUC is also very impressive across the board
- Low precision and recall for positive sentiment class (high precision in AdaBoost and high recall in AdaBoost and Linear Discriminant).

Weak Supervised Model with Pseudo Labeling

- Snorkel python package was used which is a probabilistic approach to infer the data label given approximate labels
- Requires expensive MCMC sampling
- Once the generative model is trained and labels are generated, a discriminator based on traditional ML algorithm can be trained to decouple expensive generator model



```
def lf2(x):
  m =
re.search(r'.*caus
e.*', x.between)
  return 1 if m
else 0
```

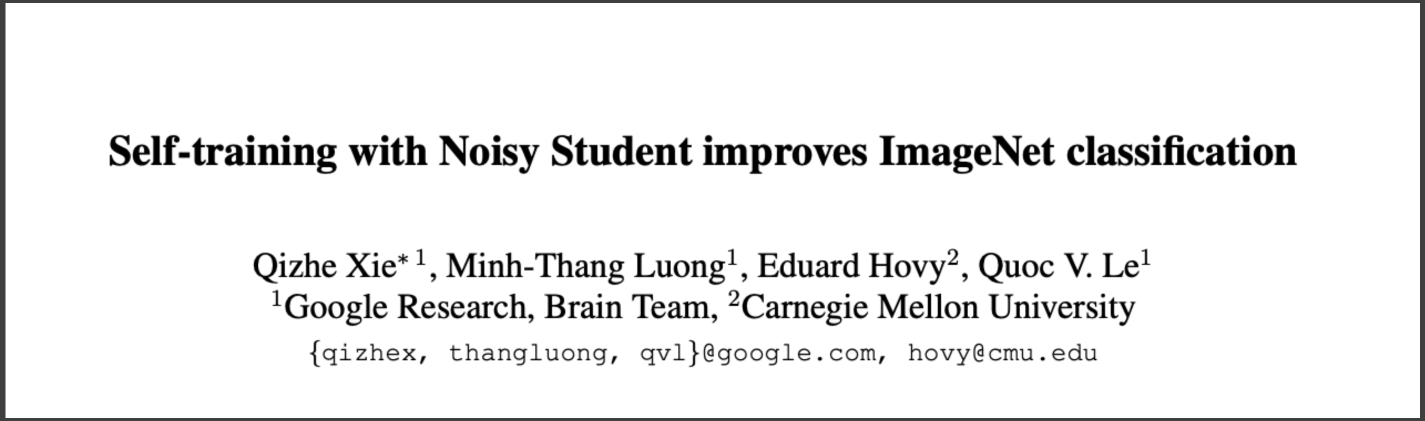
DOMAIN EXPERT
$\tilde{p}_i(y|x)$
?
$\tilde{p}(y|x)$
y_1
$h_{1,1}$
$h_{1,2}$
$h_{1,3}$
$x_{1,1}$
$x_{1,2}$
$\hat{\theta} = \text{argmin}_\theta \mathbb{E}_{(x,y)\sim T}[L(y, f_\theta(x))]$
Example Weak Supervision Sources
Technical Challenge: Integrating & Modeling Diverse Sources
Use Weak Supervision to Train End Model

Image credit: https://www.snorkel.org/blog/weak-supervision

Labeling Function

- Label based on presence and absence of certain keyword or phrases in the tweet
- Learned labels from the previously discussed supervised model.

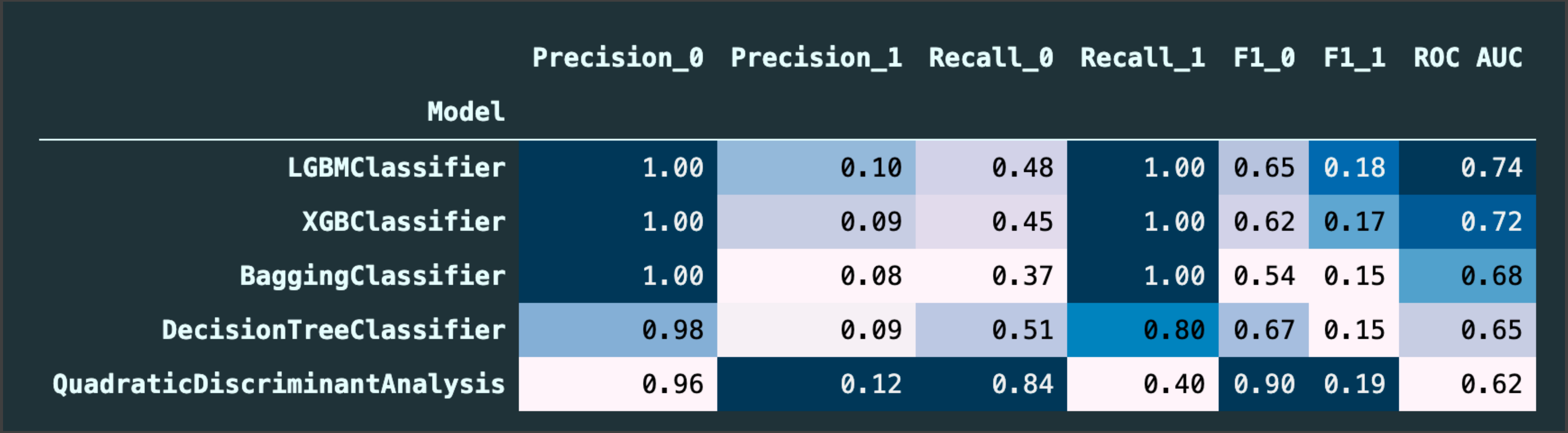
The rational for using supervised model learned labels is that a team from Google and CMU reported significant improvement of performance with this approach



Self-training with Noisy Student improves ImageNet classification

Qizhe Xie* [1], Minh-Thang Luong[1], Eduard Hovy[2], Quoc V. Le[1]
[1]Google Research, Brain Team, [2]Carnegie Mellon University
`{qizhex, thangluong, qvl}@google.com, hovy@cmu.edu`

Weak Supervised Model with Pseudo Labeling



Model	Precision_0	Precision_1	Recall_0	Recall_1	F1_0	F1_1	ROC AUC
LGBMClassifier	1.00	0.10	0.48	1.00	0.65	0.18	0.74
XGBClassifier	1.00	0.09	0.45	1.00	0.62	0.17	0.72
BaggingClassifier	1.00	0.08	0.37	1.00	0.54	0.15	0.68
DecisionTreeClassifier	0.98	0.09	0.51	0.80	0.67	0.15	0.65
QuadraticDiscriminantAnalysis	0.96	0.12	0.84	0.40	0.90	0.19	0.62

- With weak supervision, precision for negative sentiment and recall for positive sentiment is very high, but recall for negative and precision for positive is very low
- The algorithm is having trouble navigating a wide overlapping feature space

Synthetic Minority Oversampling Technique (SMOTE)

- Based on the nearest neighbor and efficient interpolation, synthetic minority data samples are generated
- In contrast, ENN (edited nearest neighbor) artificially removes samples of majority class, resulting in oversampling.

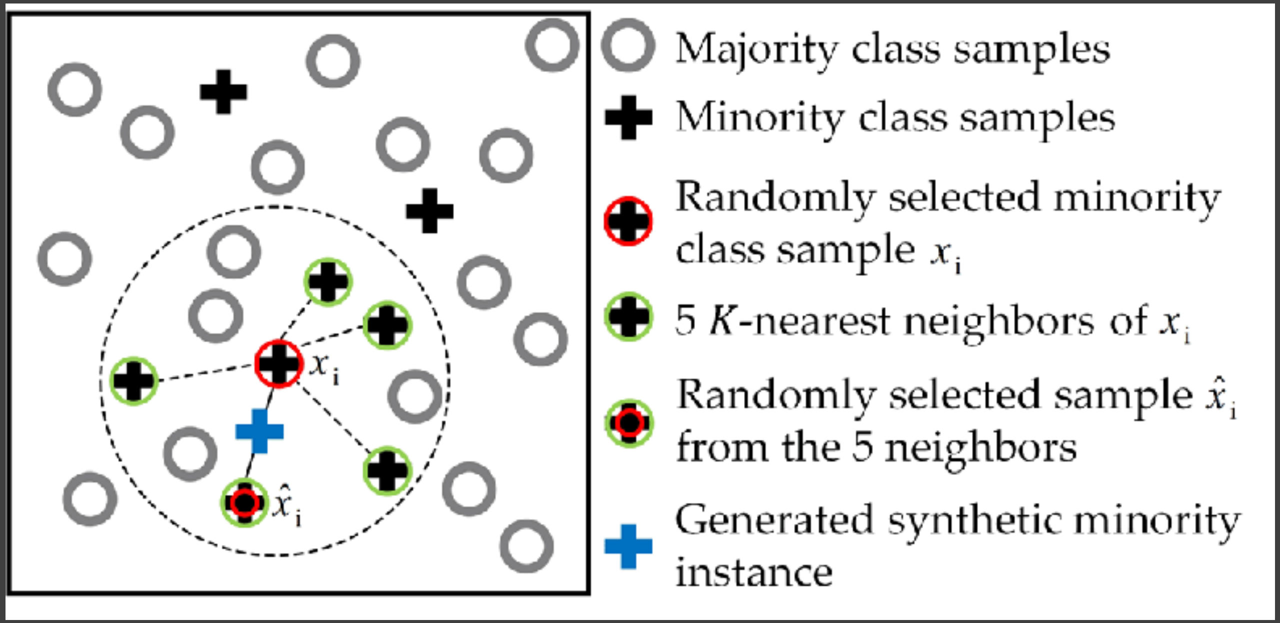
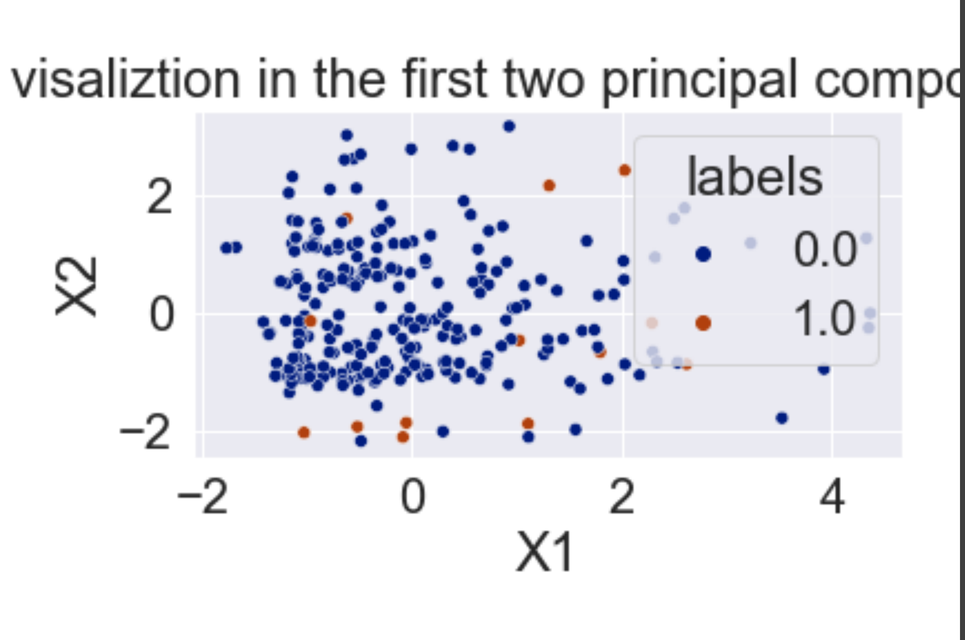
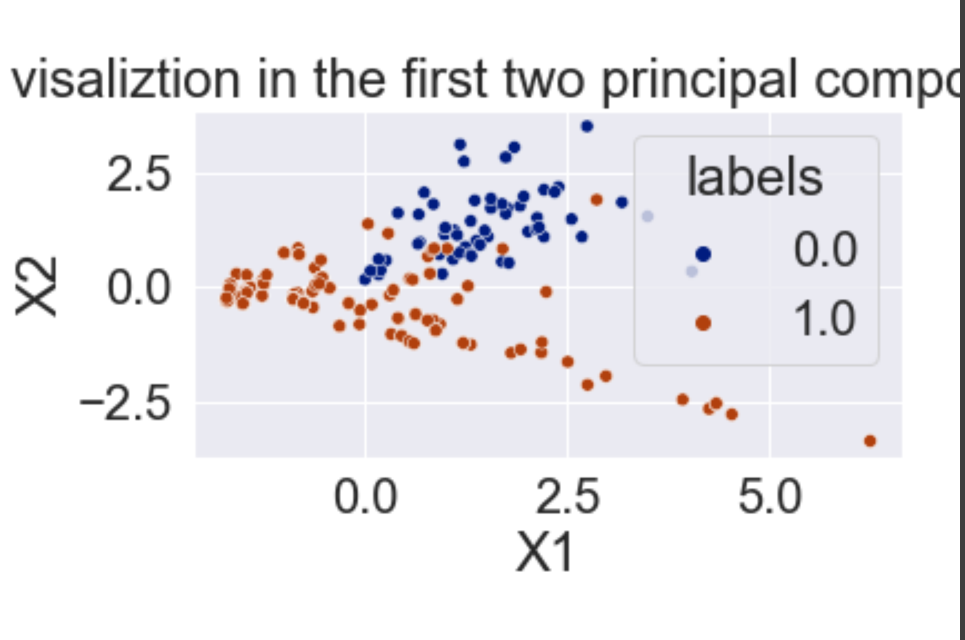

x_i
$\hat{x}_i$
○ Majority class samples
✚ Minority class samples
Randomly selected minority class sample x_i
5 K-nearest neighbors of x_i
Randomly selected sample $\hat{x}_i$ from the 5 neighbors
Generated synthetic minority instance

Image credit: https://www.mdpi.com/2072-4292/11/7/846

Synthetic Minority Oversampling Technique (SMOTE)

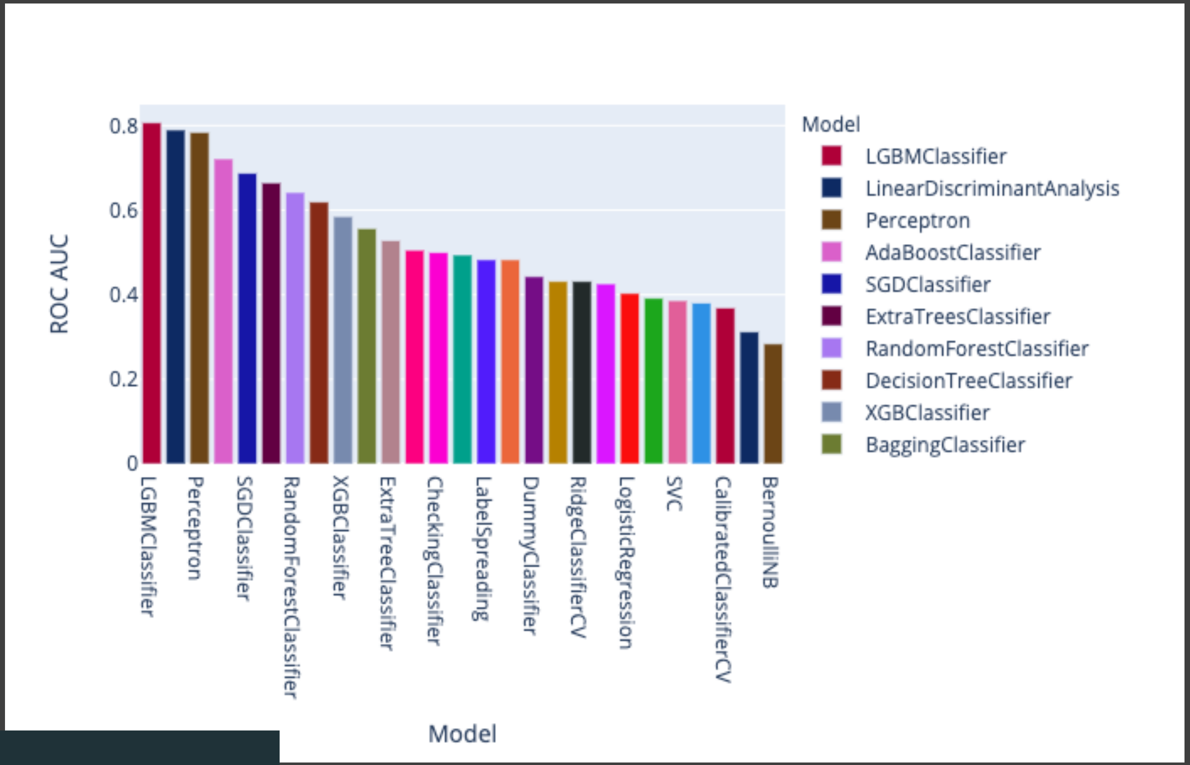


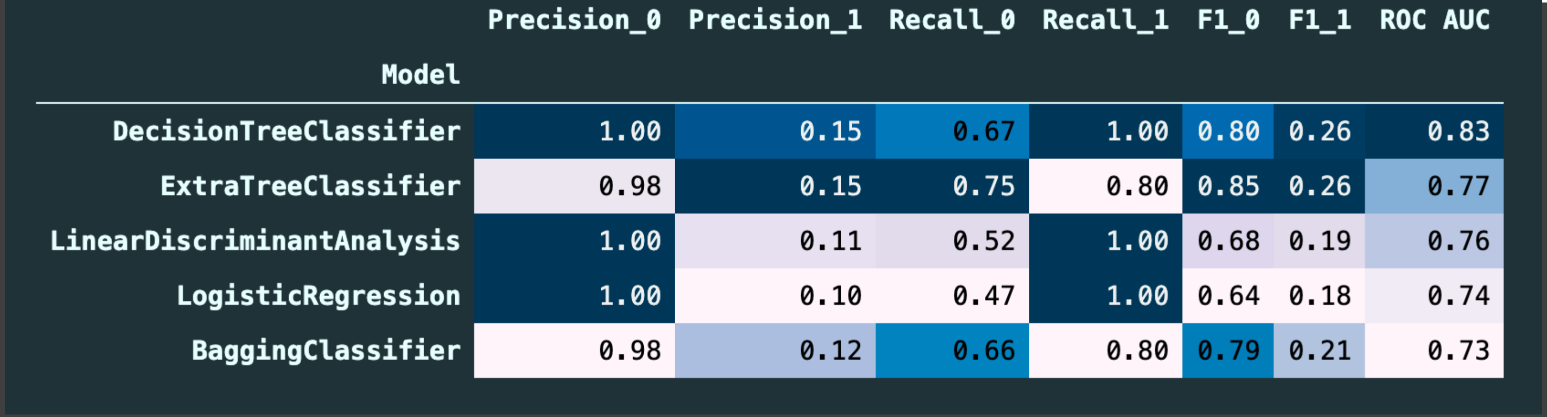


With SMOTEENN, two well defined cluster has been formed by over sampling the minority class and under sampling the majority class in the low PCA region

Synthetic Minority Oversampling Technique (SMOTE)

The performance is not much better than the supervised learning on the labelled dataset only. Next, we will try to perform hyperparameter optimization to improve the performance of SMOTEENN.





Model	Precision_0	Precision_1	Recall_0	Recall_1	F1_0	F1_1	ROC AUC
DecisionTreeClassifier	1.00	0.15	0.67	1.00	0.80	0.26	0.83
ExtraTreeClassifier	0.98	0.15	0.75	0.80	0.85	0.26	0.77
LinearDiscriminantAnalysis	1.00	0.11	0.52	1.00	0.68	0.19	0.76
LogisticRegression	1.00	0.10	0.47	1.00	0.64	0.18	0.74
BaggingClassifier	0.98	0.12	0.66	0.80	0.79	0.21	0.73

Hyperparameter Optimization

- AdaBoostClassifier (GridSearchCV): learning_rate, n_estimators , base_estimator
- DecisionTreeClassifier (GridSearchCV): max_depth, min_samples_split, max_features
- XGBClassifier (Bayesian optimization): n_estimators, max_depth, reg_alpha, reg_lambda, eta, gamma, subsample, colsample_bytree

Hyperparameter optimization didn’t improve performance much. It seems like, I need to invest more time into tuning the supervised learning hyperparameter to improve the performance

Conclusion

- Out of all the approach, Supervised learning with just the labelled data performed better than other approaches
- For real data analytics, investing more time into labeling data will be crucial to make more reliable model

Acknowledgement: Prasad Seemakurthi (Data Scientist, Navi Ltd.)

Thank You!