Prediction of Whether the New York Times Comments Picked by Editors

Yu Tian
MS Data Science, University of San Francisco

Introduction

Data

	approveDate	articleID	articleWordCount	commentBody	commentID	commentSequence	commentTitle
0	1483455908	58691a5795d0e039260788b9	1324.0	For all you Americans out there --- still rejo...	20969730.0	20969730.0	
1	1483455656	58691a5795d0e039260788b9	1324.0	Obamas policies may prove to be the least of t...	20969325.0	20969325.0	
2	1483455655	58691a5795d0e039260788b9	1324.0	Democrats are comprised of malcontents who gen...	20969855.0	20969855.0	

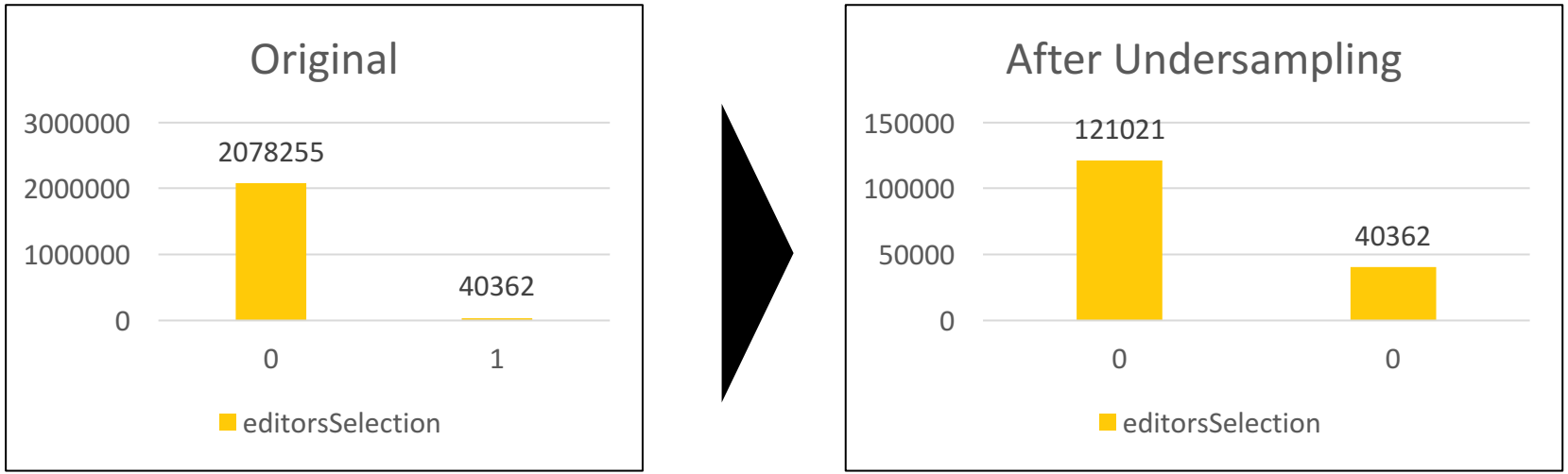
The data contains information about the comments made on the articles published in New York Times in Jan-May 2017 and Jan-April 2018.

- 2 million comments
- 34 features

	articleID	commentBody	editorsSelection
0	58691a5795d0e039260788b9	For all you Americans out there --- still rejo...	0
1	58691a5795d0e039260788b9	Obamas policies may prove to be the least of t...	0
2	58691a5795d0e039260788b9	Democrats are comprised of malcontents who gen...	0
3	58691a5795d0e039260788b9	The picture in this article is the face of con...	0
4	58691a5795d0e039260788b9	Elections have consequences.	0
		X	Y

EDA & Preprocessing

The data is very unbalanced. To deal with that situation, undersampling the data that have majority samples. Select all positive data (comments that indeed were editor's pick), and set the number of negative data (not editor's pick) to be around 3 folds.



Vectorization

Using TFIDF for words and character n-grams and combine them using FeatureUnion.

	articleID	commentBody	editorsSelection
0	58691a5795d0e039260788b9	For all you Americans out there --- still rejo...	0
1	58691a5795d0e039260788b9	Obamas policies may prove to be the least of t...	0
2	58691a5795d0e039260788b9	Democrats are comprised of malcontents who gen...	0
3	58691a5795d0e039260788b9	The picture in this article is the face of con...	0
4	58691a5795d0e039260788b9	Elections have consequences.	0
		X	Y

```
vectorizer = FeatureUnion([
    ('word_tfidf', TfidfVectorizer(
    analyzer='word',
    token_pattern=r'\w{1,}',
    ngram_range=(1, 2),
    max_features=600,
    )),

    ('char_tfidf', TfidfVectorizer(
    analyzer='char',
    ngram_range=(2, 4),
    max_features=600,
    ))
])
start_vect = time()
vectorizer.fit(commentBody)
train_text = vectorizer.transform(train_text)
test_text = vectorizer.transform(test_text)
```

Training

```
clf_logistic = Pipeline([
    ('lsa', TruncatedSVD(n_components=1000, random_state=0)),
    ('logistic', LogisticRegression(C=150))
])
```

Use Latent Semantic Analysis to perform dimensionality reduction on the TF-IDF vectors and then train the Logistic regression to make predictions.

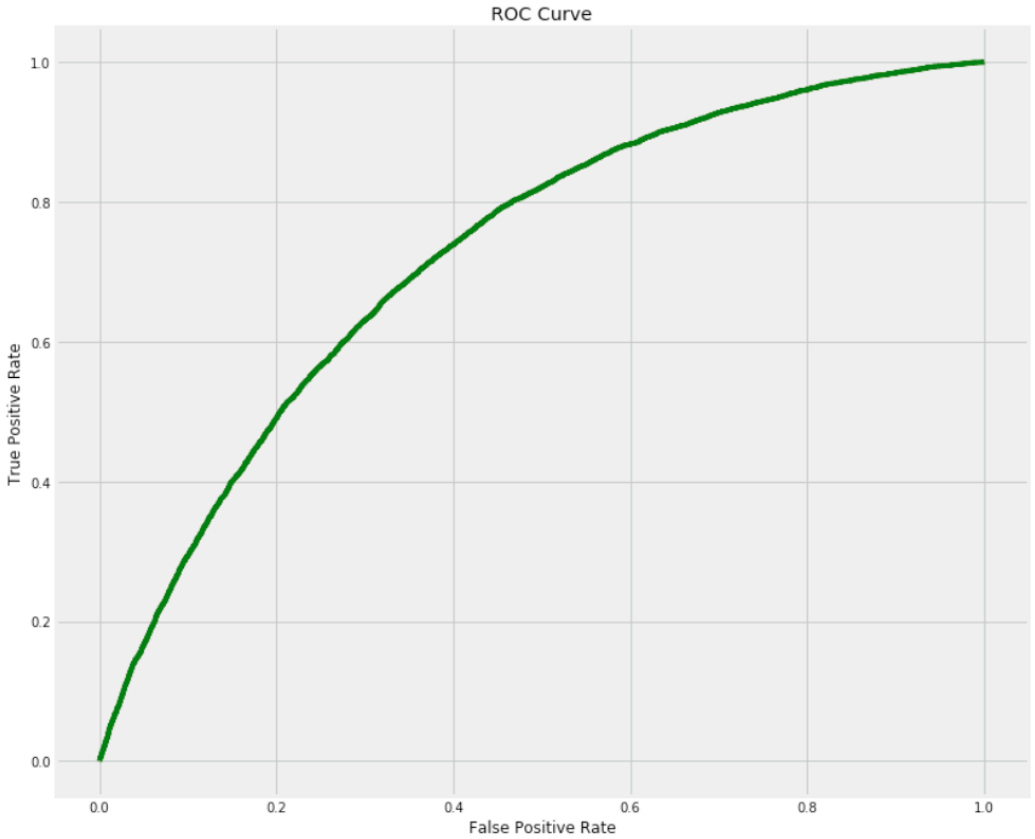
Tune hyper-parameters by grid searching

```
param_grid = [
    {'logistic__C': [150, 200]}
]
```

```
gkf = GroupKFold(n_splits=3).split(train_text, train_target, groups=train_groups)
scorer = make_scorer(recall_score)

grid_search = GridSearchCV(clf_logistic, param_grid=param_grid, cv=gkf, scoring=scorer)
grid_search.fit(train_text, train_target)

print("Best parameters found:")
print(grid_search.best_params_)
print("Best score:")
print(grid_search.best_score_)
```

```
Best parameters found:
{'logistic__C': 200}

Best score:
0.14194786219966077

ROC AUC Score:  0.7267
logloss:  0.5013
```

ROC Curve



The prediction of the selection of comments as editor's picks has a fine ROC score of 0.72.

Future improvement can be tried by using word embedding with RNN.

Thanks!