Module 3 Summary

October 13, 2019

Paul Raymond
February 2019 cohort
Brandon Lewis

0.1 Project focus

My primary focus for my Module 3 Project will be executing various classification methods on a set of data from the New York Times. Additionally, I will integrate Time Series and NLP analysis into the workflow.

I've chosen a dataset provided by The New York Times that includes data on articles and comments written during Jan-April 2017 and Jan-April 2018.

I plan to measure whether features - numerical, categorical, text - can be used to accurately predict which comments are selected by NYTimes editors.

0.2 Exploratory Data Analysis

I'll be working with the following files made available by the New York Times on Kaggle: **Articles and Comments submitted duirng January - April 2017 and 2018** After combining and processing the datasets, here's a breakdown on the Articles and Comments data:

Breakdown by EditorsSelection

Numerical features of focus

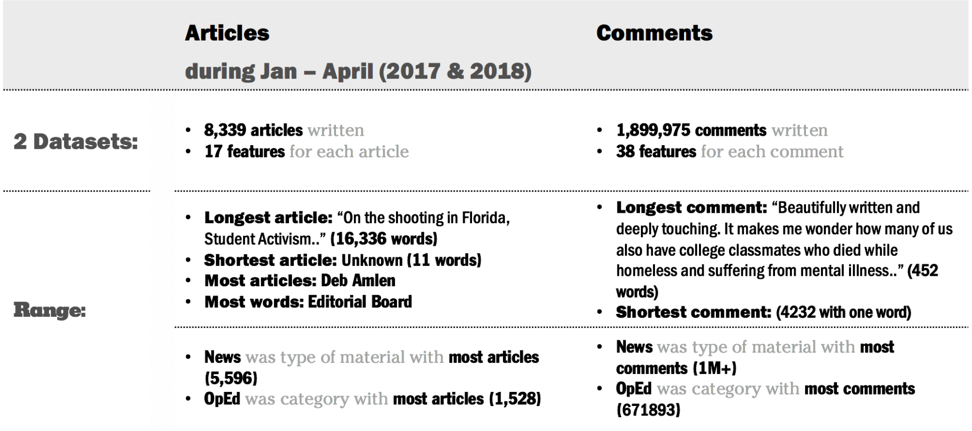
There isn't consistent dilinearation between selected and nonselected comments with respect to numerical features. The word count distribution for nonselected comments is skewed right whereas that for selected comments is more normally distribution.

The violin plots confirm this as well.

Data breakdown

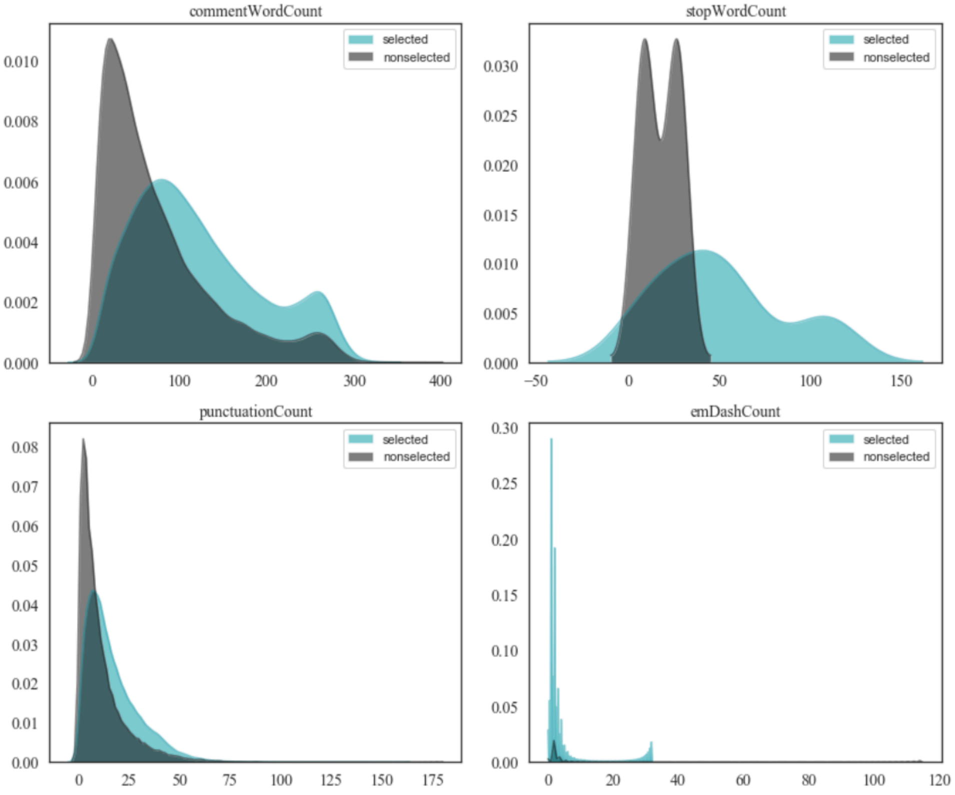
Type of material (Total articles & comments)

Some data and meta-comments

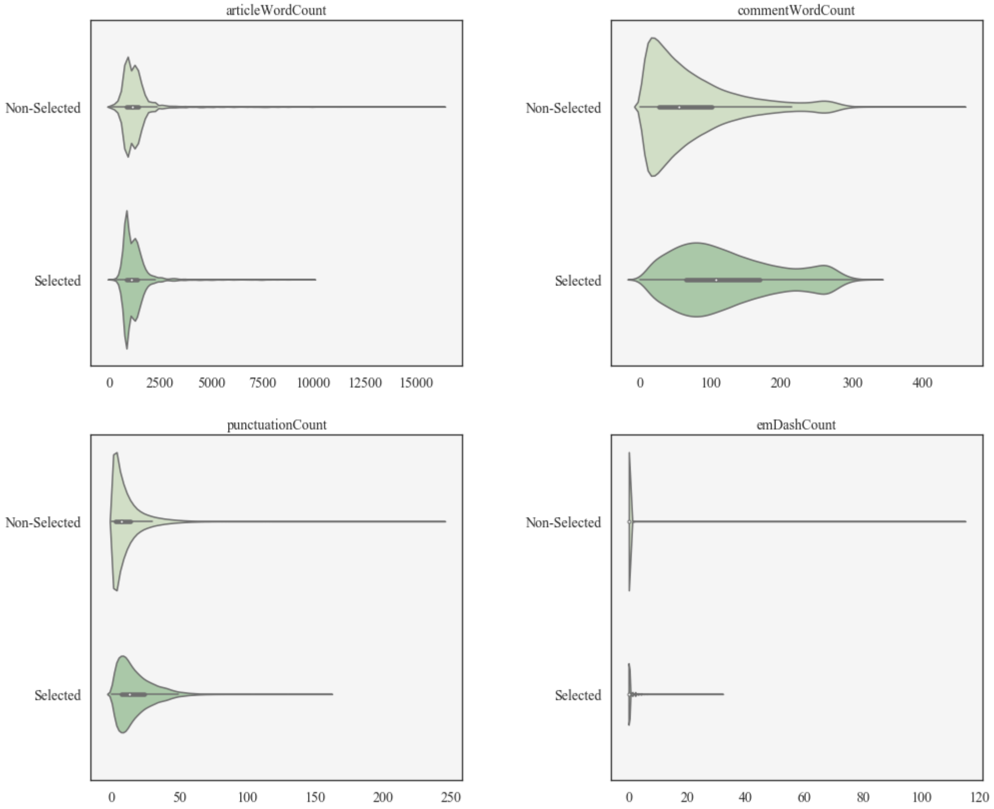


	Articles **during Jan – April (2017 & 2018)**	**Comments**
2 Datasets:	• **8,339 articles** written • **17 features** for each article	• **1,899,975 comments** written • **38 features** for each comment
Range:	• **Longest article:** “On the shooting in Florida, Student Activism..” **(16,336 words)** • **Shortest article:** Unknown **(11 words)** • **Most articles:** Deb Amlen • **Most words:** Editorial Board	• **Longest comment:** “Beautifully written and deeply touching. It makes me wonder how many of us also have college classmates who died while homeless and suffering from mental illness..” **(452 words)** • **Shortest comment:** (4232 with one word)
	• **News** was type of material with **most articles (5,596)** • **OpEd** was category with **most articles (1,528)**	• **News** was type of material with **most comments (1M+)** • **OpEd** was category with **most comments (671893)**

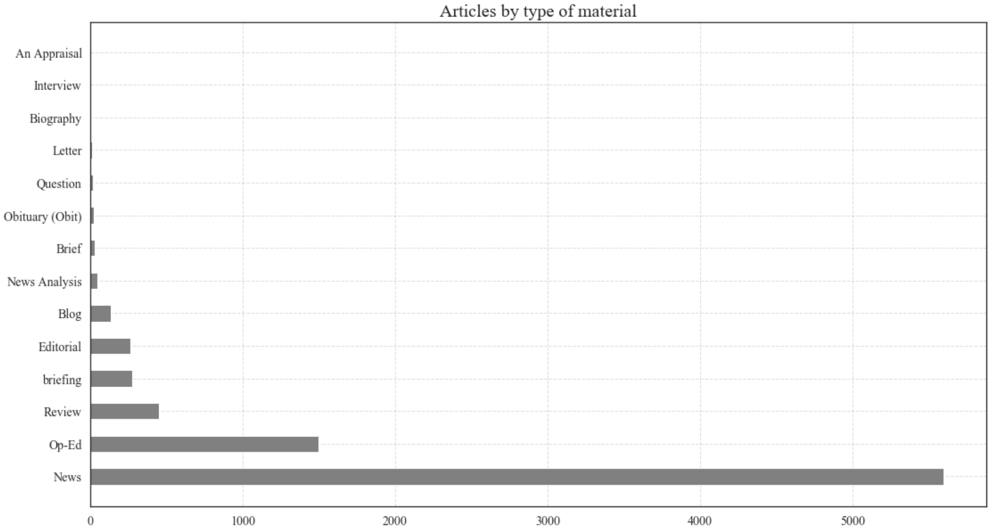
title



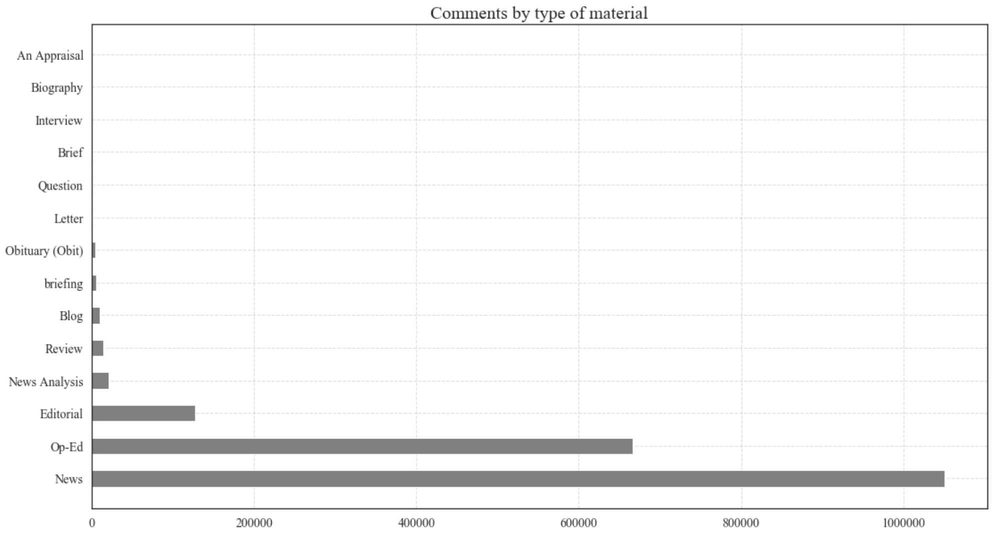
title


articleWordCount
commentWordCount
punctuationCount
emDashCount
Non-Selected
Selected
0
2500
5000
7500
10000
12500
15000
0
100
200
300
400
0
50
100
150
200
250
0
20
40
60
80
100
120

“Plot”

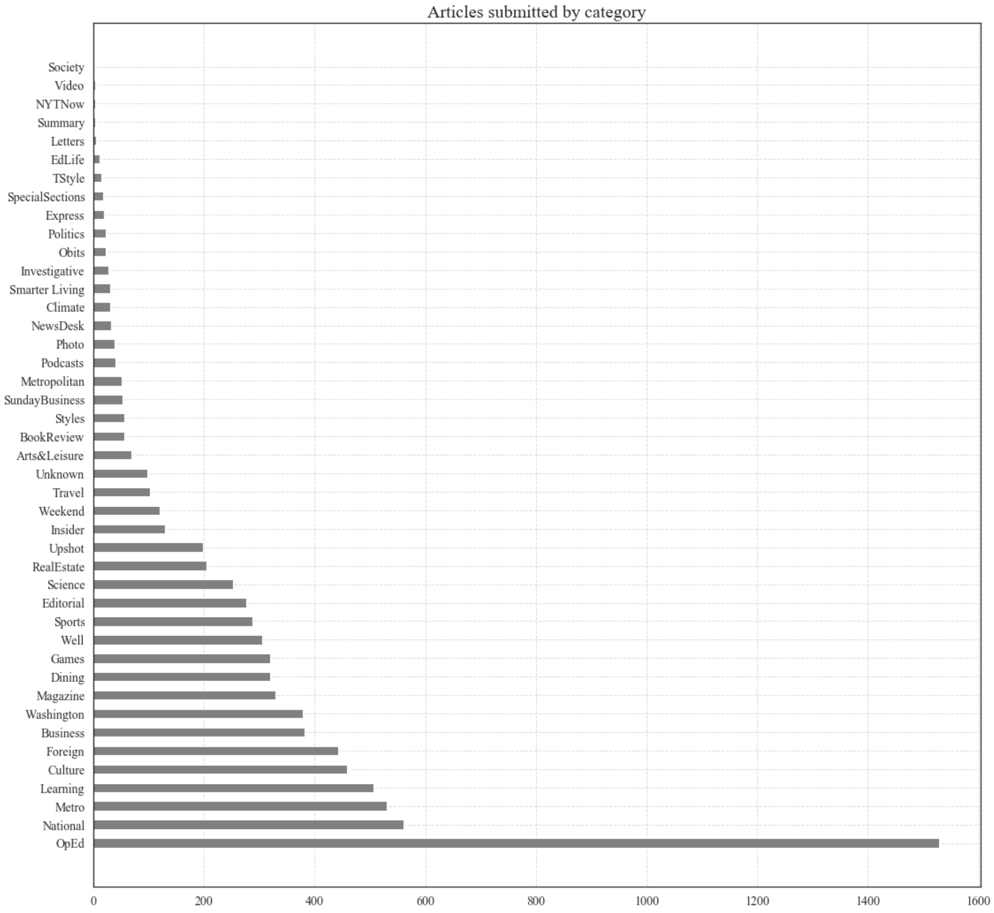

Articles by type of material
An Appraisal
Interview
Biography
Letter
Question
Obituary (Obit)
Brief
News Analysis
Blog
Editorial
briefing
Review
Op-Ed
News
0
1000
2000
3000
4000
5000

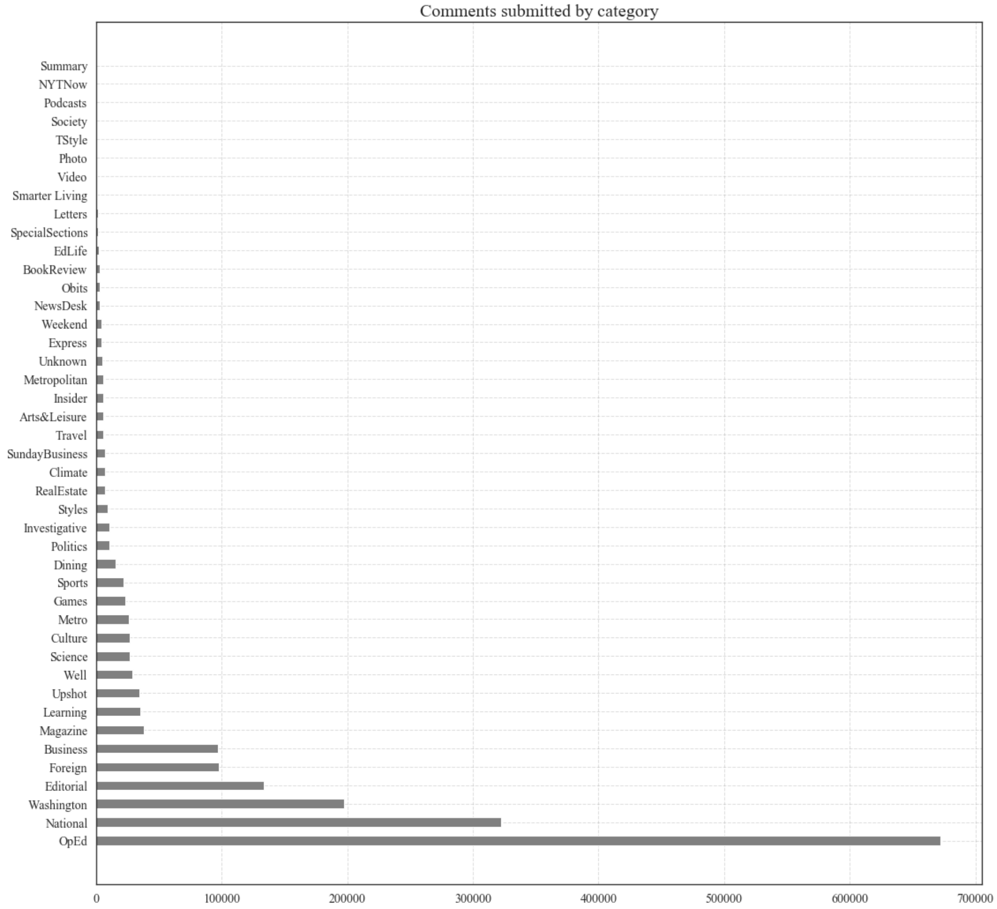
title



There appears to be more comments submitted per Op-Eds than ratio for any other main category — what I would expect.

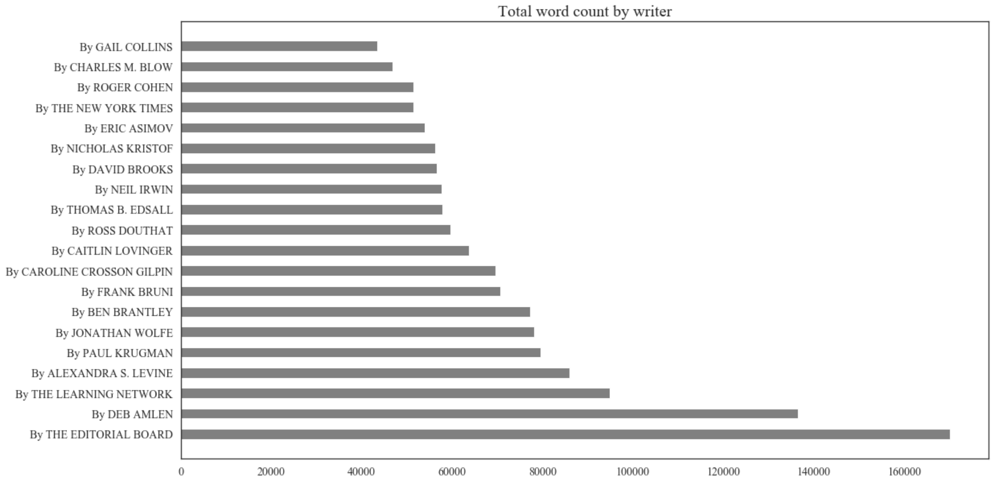
Category (Total articles & comments)

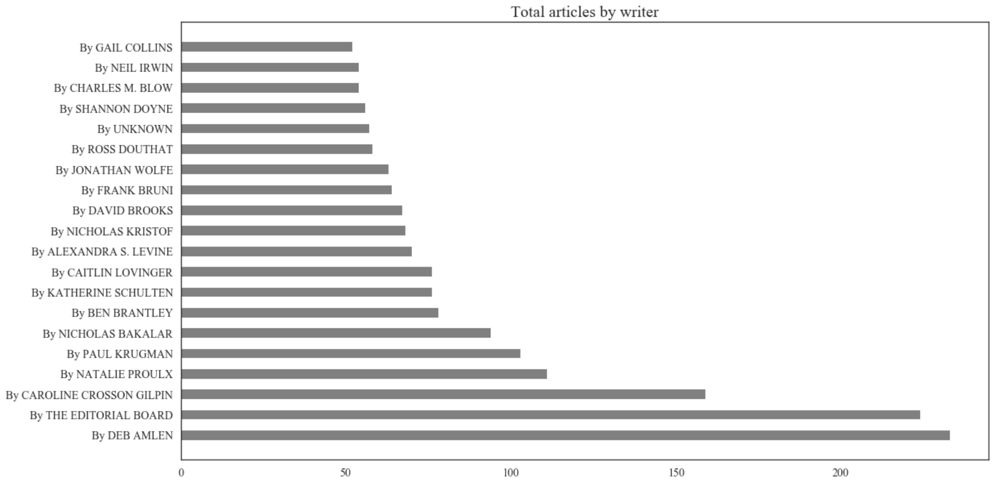




There appears to be more comments submitted per Op-Eds than ratio for any other main category — what I would expect.

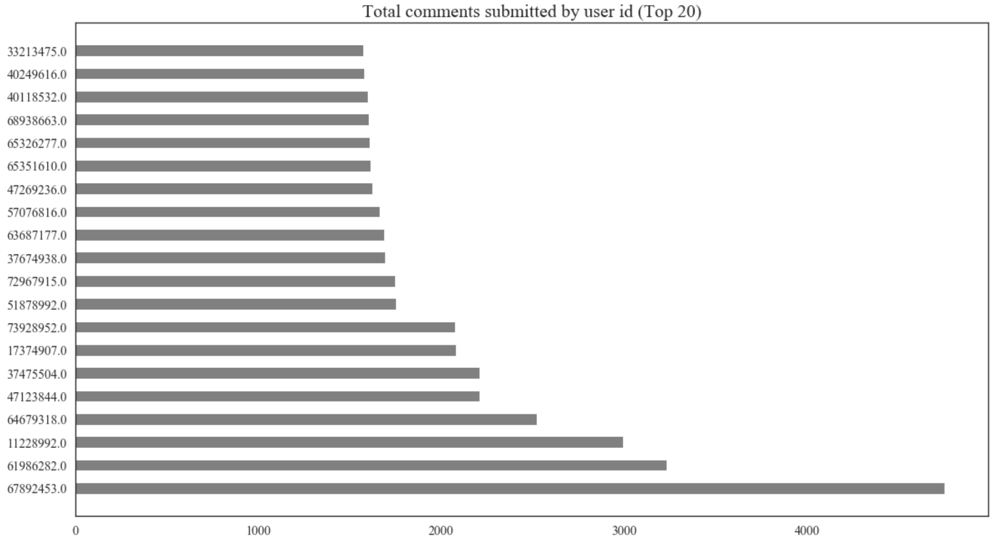
Word and article count

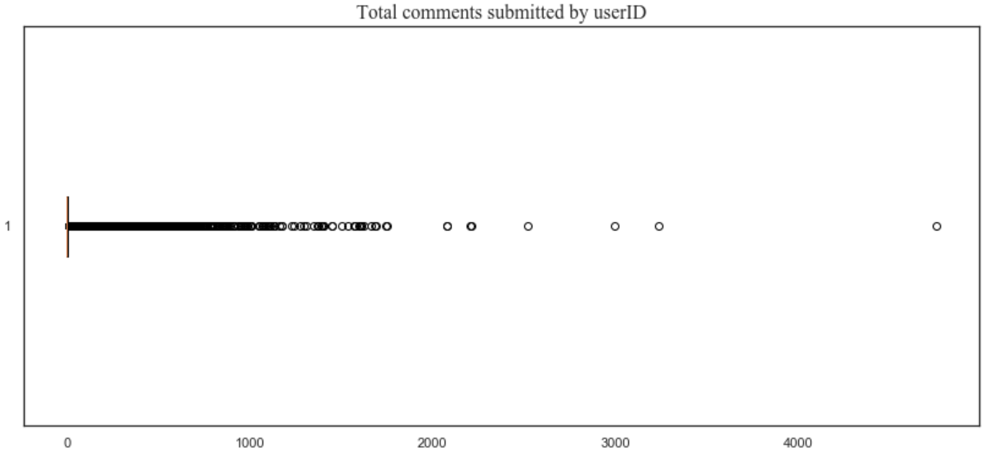


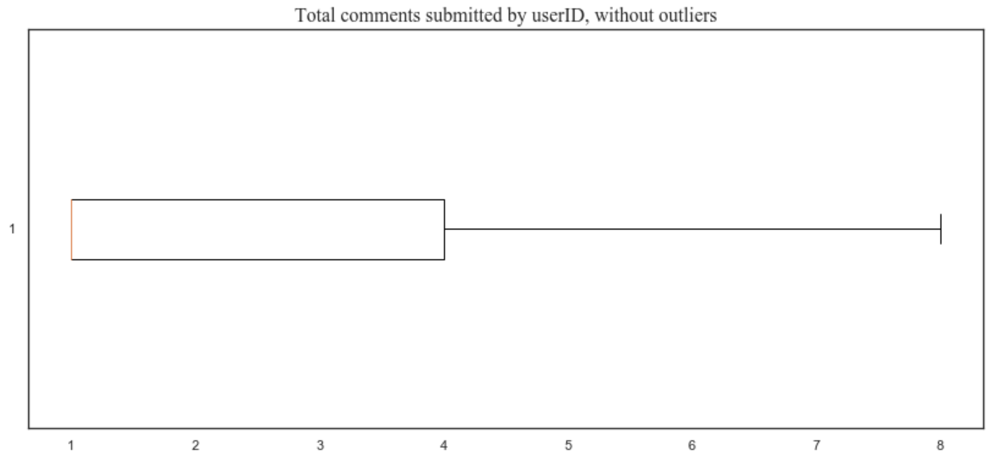


```
In [ ]:
```

Comments distribution





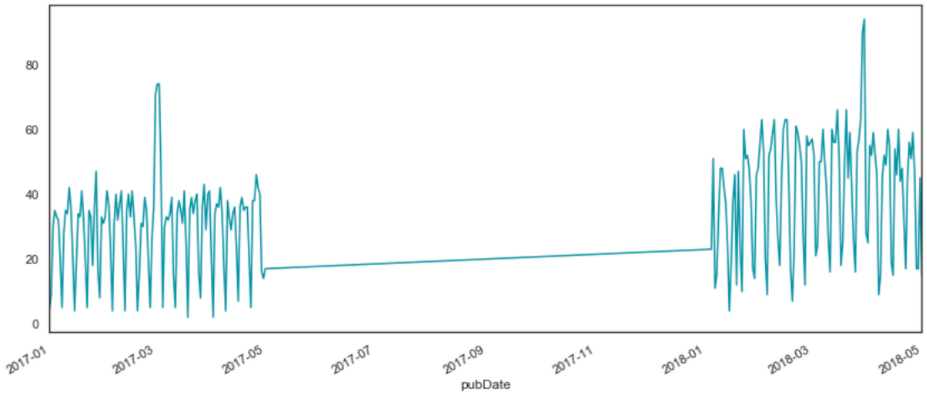


As evidenced above, there are significant outliers, yet the average reader has submitted closer to 1 comment.

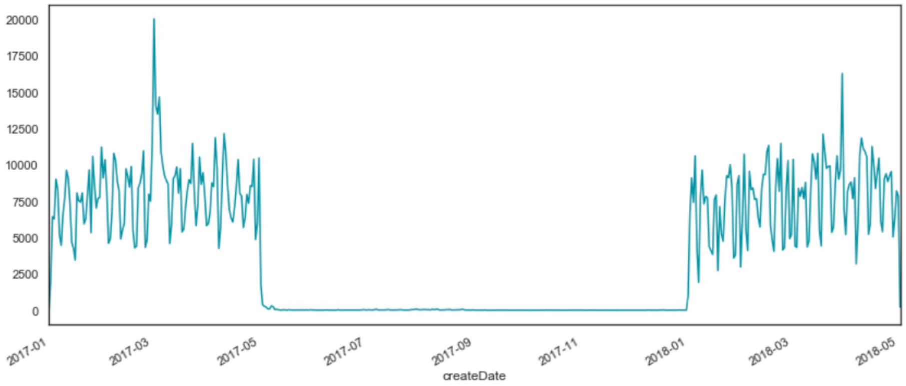
Next, I'll explore the changes in our data over the two periods: Jan - May 2017 and Jan - May 2018.

0.3 Time-series analysis

Submissions in Jan - April (2017 and 2018)

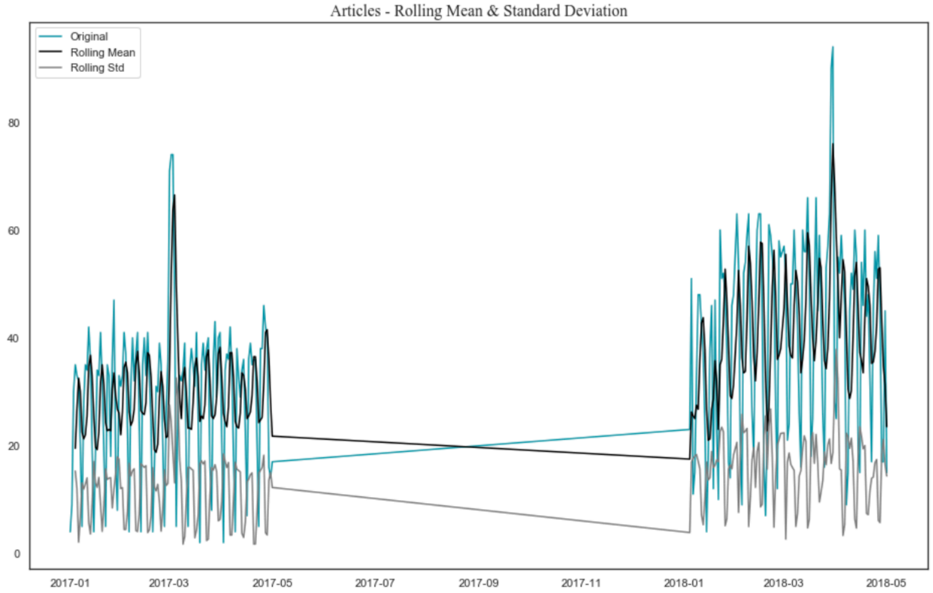


Visually, there appears to be an increase in articles from the period in 2017 to that in 2018.

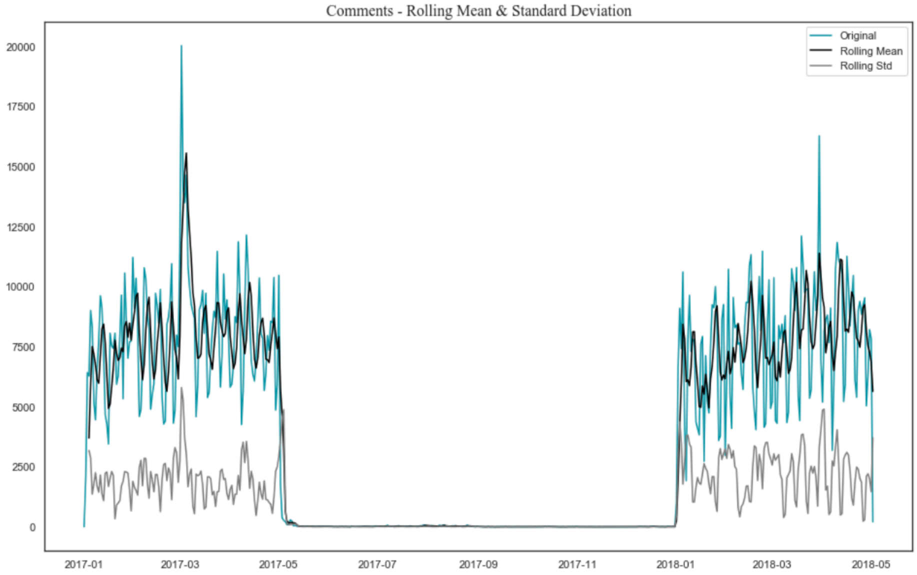


However, an increase in comments during the similar period as Articles isn't apparent from the plot of submitted comments over the two year period.

Adjusted trends in 2017 and 2018

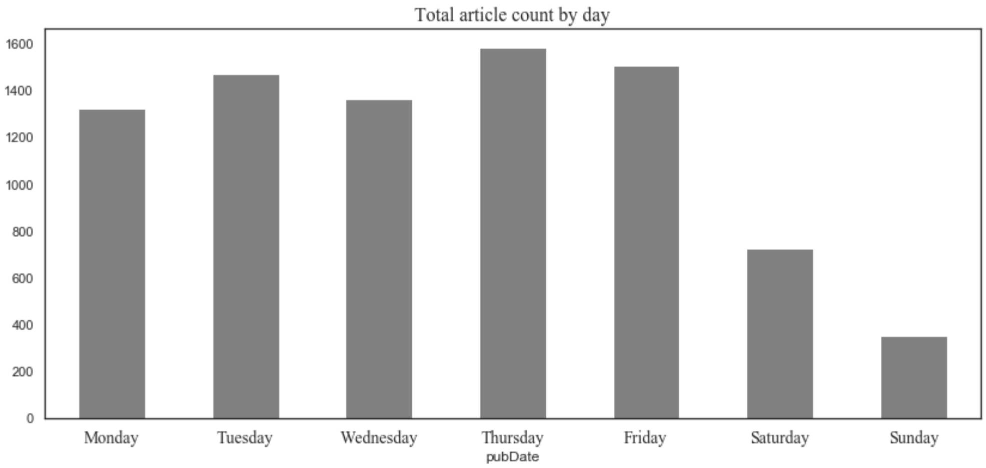


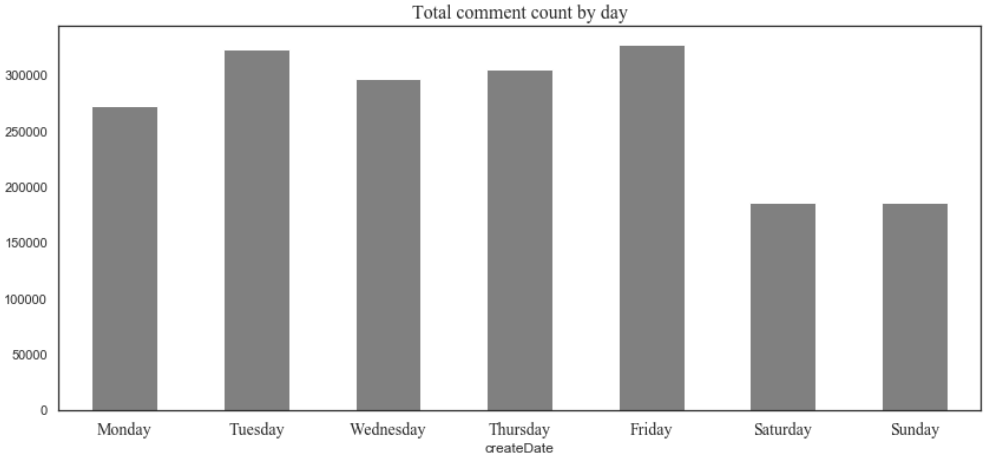
From the plot, it's apparent that the mean article count has increased and that there isn't considerable increase in variation of article counts between the two periods.



Upon further inspection, comments display consistent trends between both periods. These results are confirmed with a Dickey-Fuller test.

Daily totals





The total articles and comments submitted by day display similar distributions. As expected, there is a drop-off in articles over the weekend, with a similar decrease in submitted comments. This is consistent with the weekly seasonality seen in the full time-series plot above.

0.4 Classification

I create classification models with various approaches and through two different lenses: 1.predictive performance of non-text features and 2. predictive performance of text-features. In addition to finding the best model, I'm curious to compare the performance of both approaches.

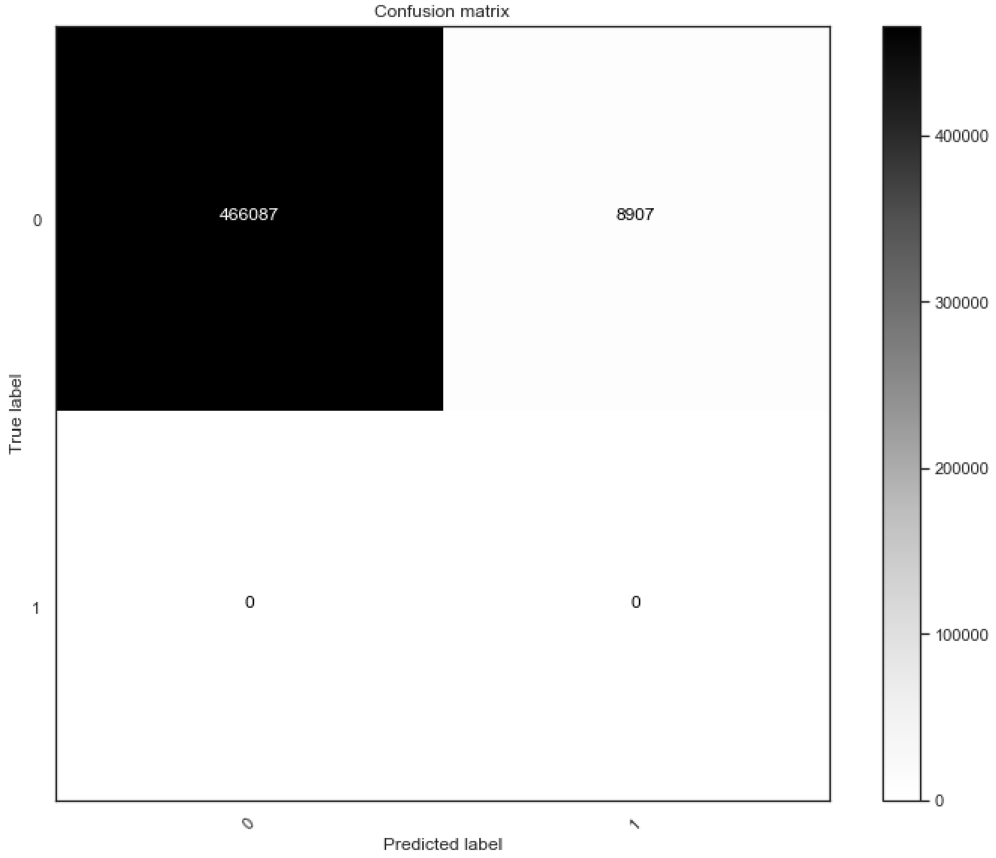
The classification model workflow is as follows: 1. **Create two datasets** - one with original columns and one that includes original and calculated columns (updated dataset) 2. **Classification models for non-text features** A. Complete baseline model (Logistic Regression) B. Complete SMOTE model (Logistic Regression) C. Complete Under-sampled model (Logistic Regression) D. Complete pipeline models (Logistic, SVM, RandomForest) E. Complete model with gridsearch 3. **Classification models for text features** A. Complete baseline model (Logistic Regression) B. Complete SMOTE model (Logistic Regression) C. Complete Under-sampled model (Logistic Regression) D. Complete pipeline models (Logistic, SVM, RandomForest) E. Complete model with gridsearch

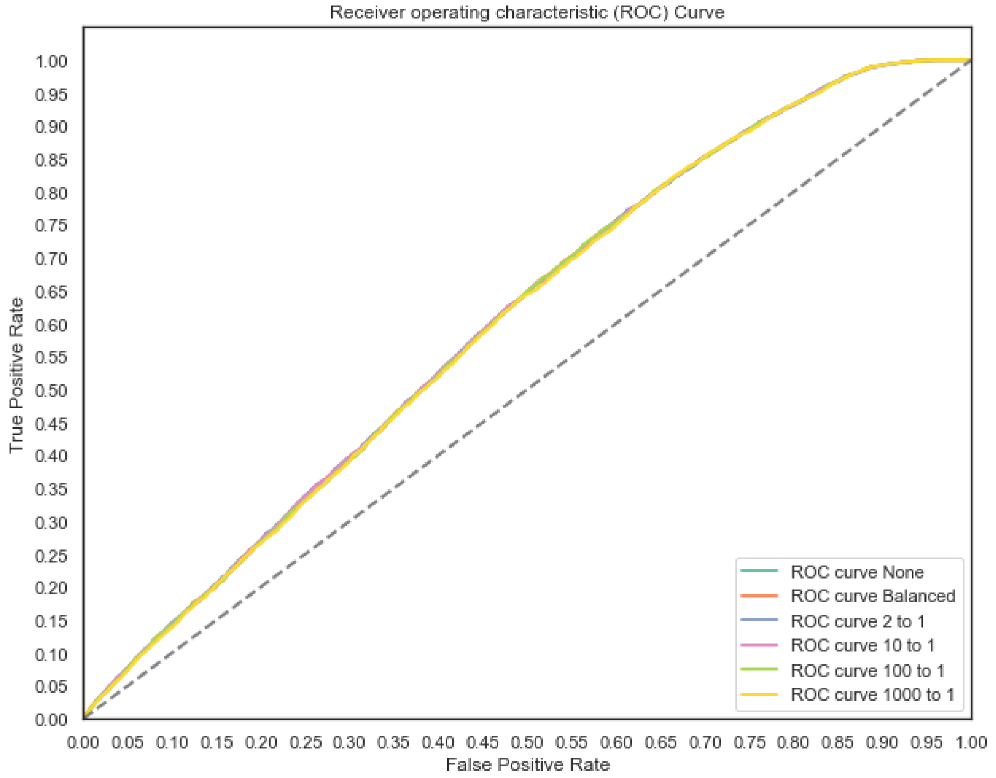
Classification models with solely text feature

Baseline models

Logistic regression

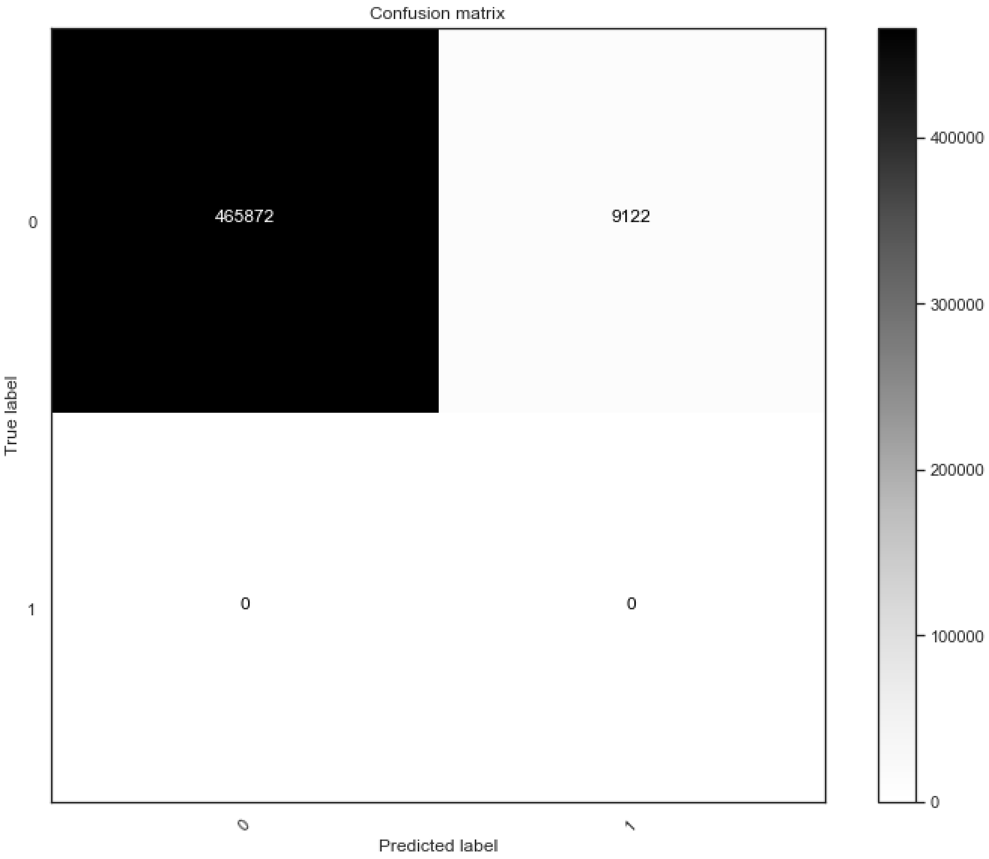
Original dataset

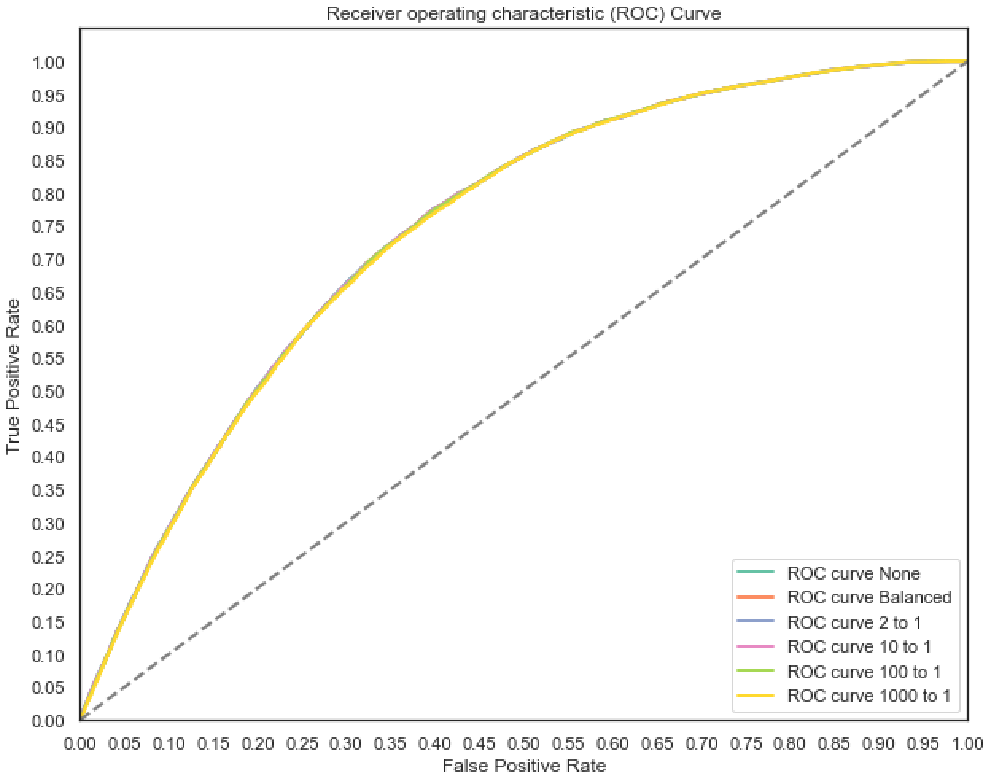




As evident in the plots above, the model is affected by the class imbalance present in the target feature. As a result, ...

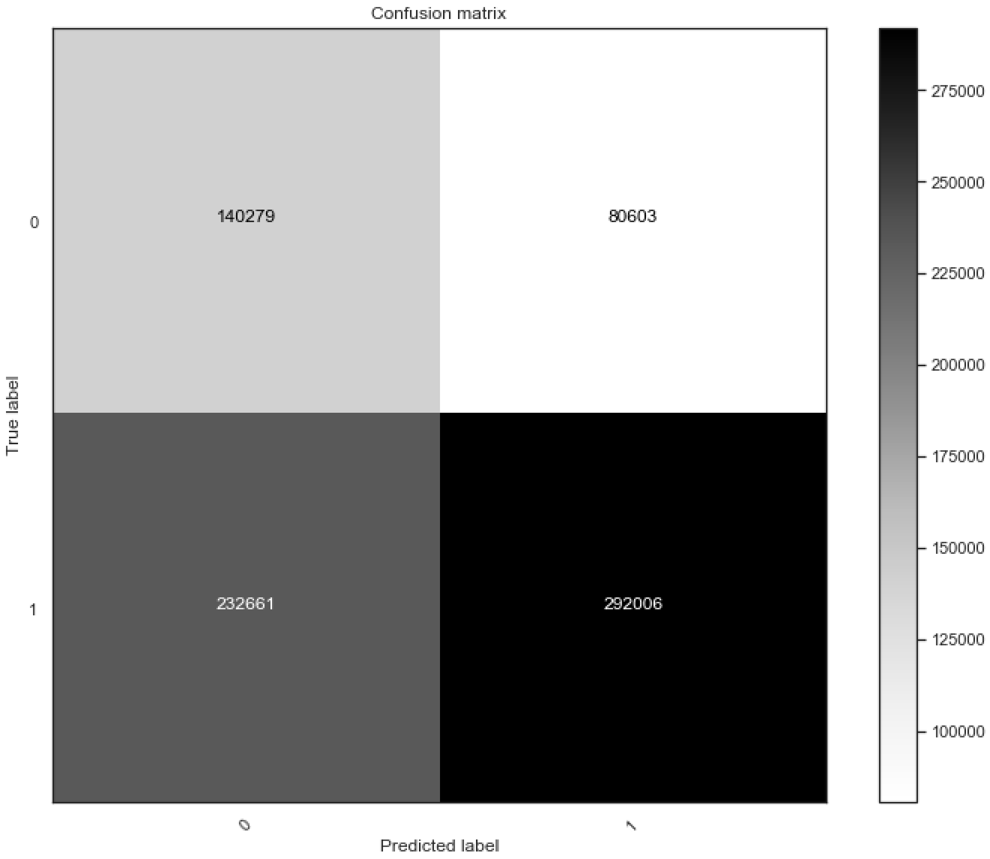
Updated dataset

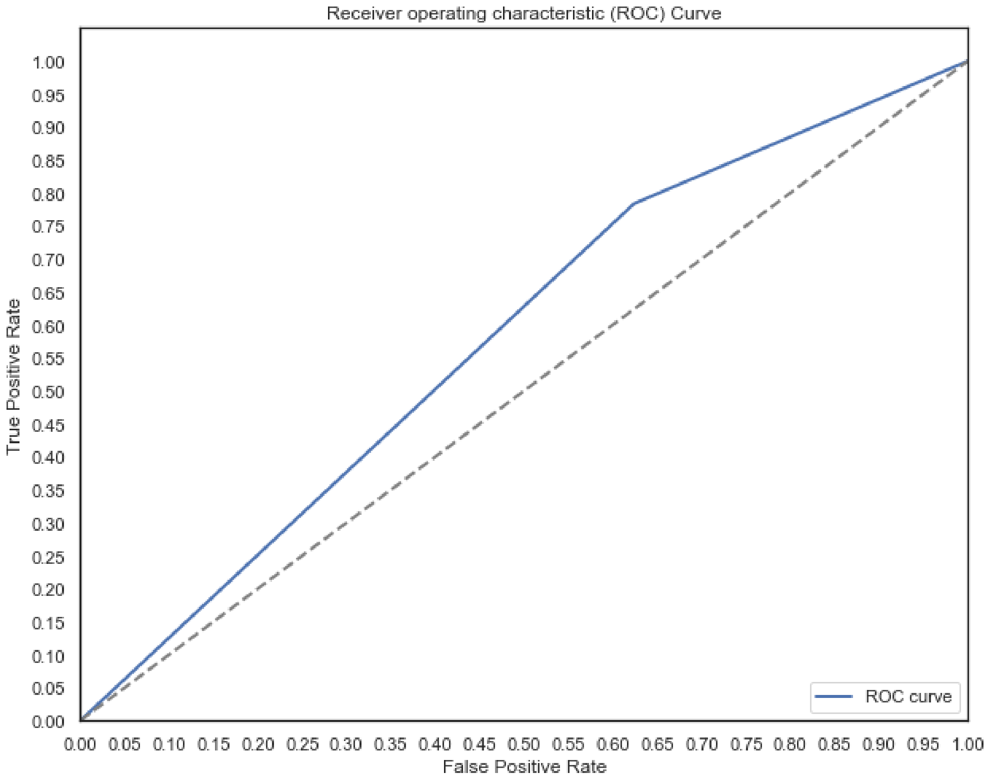




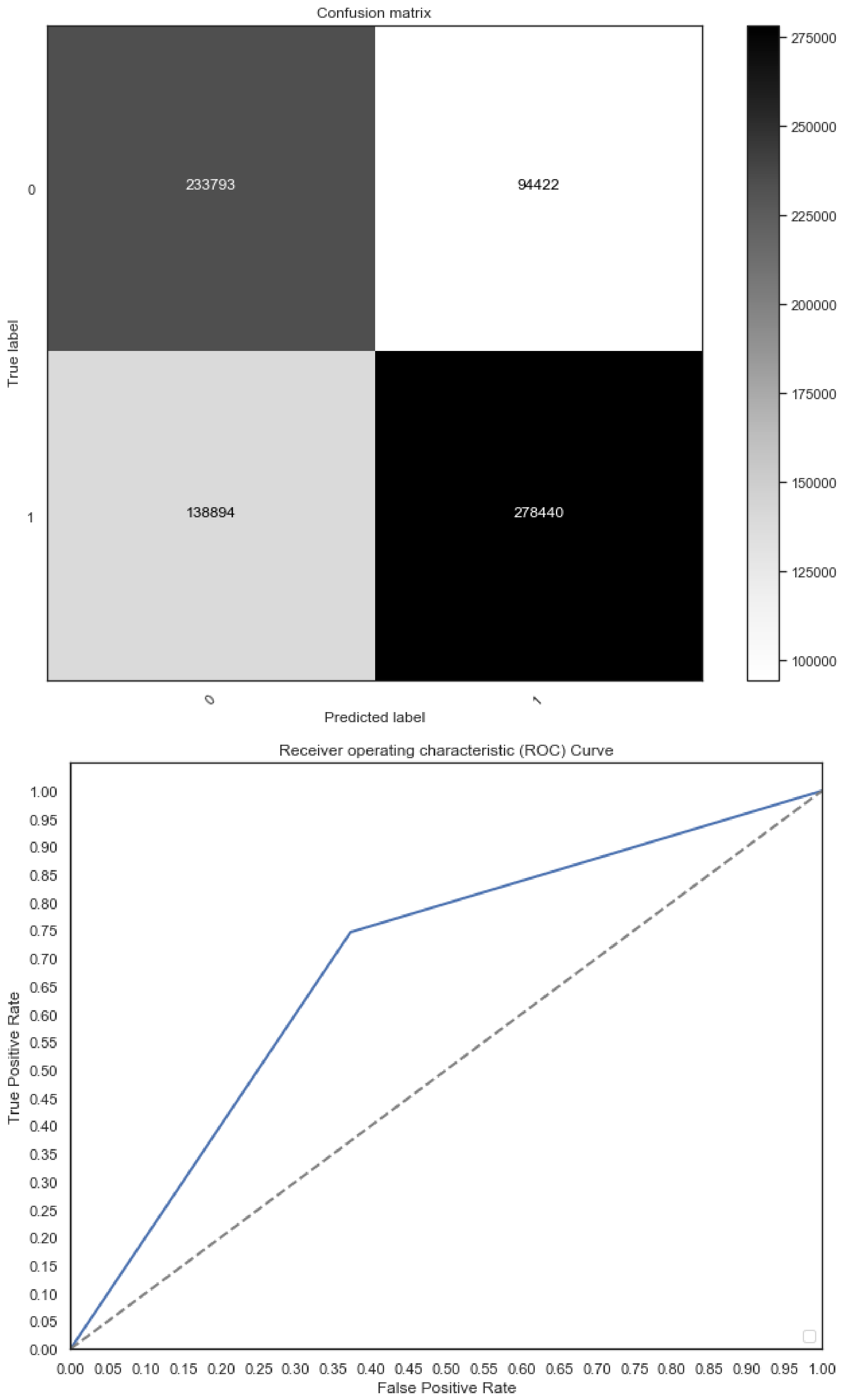
The class imbalance issue is again apparent, distorting the performance results. To address this, I'll rebalance the samples with both a SMOTE and under-sampling approach, and complete the models again.

SMOTE models with Logistic regression


Confusion matrix
True label
0
1
140279
80603
232661
292006
Predicted label
0
1
275000
250000
225000
200000
175000
150000
125000
100000

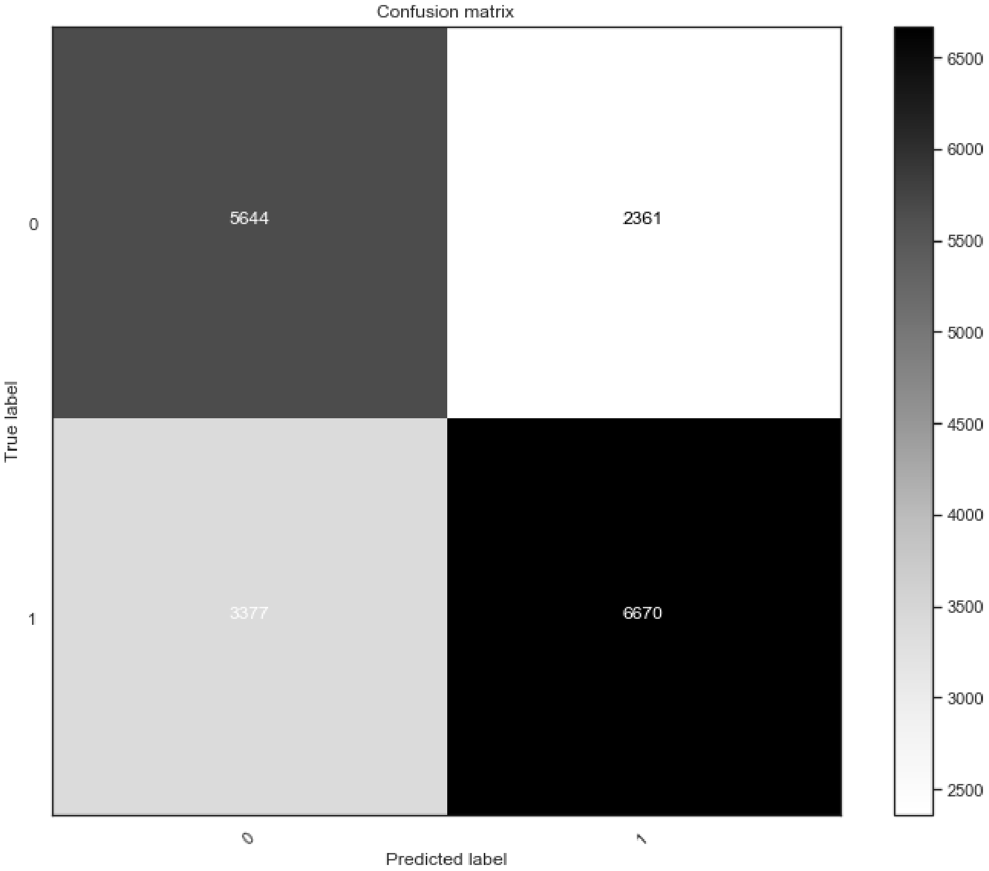

Receiver operating characteristic (ROC) Curve
True Positive Rate
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
False Positive Rate
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
ROC curve

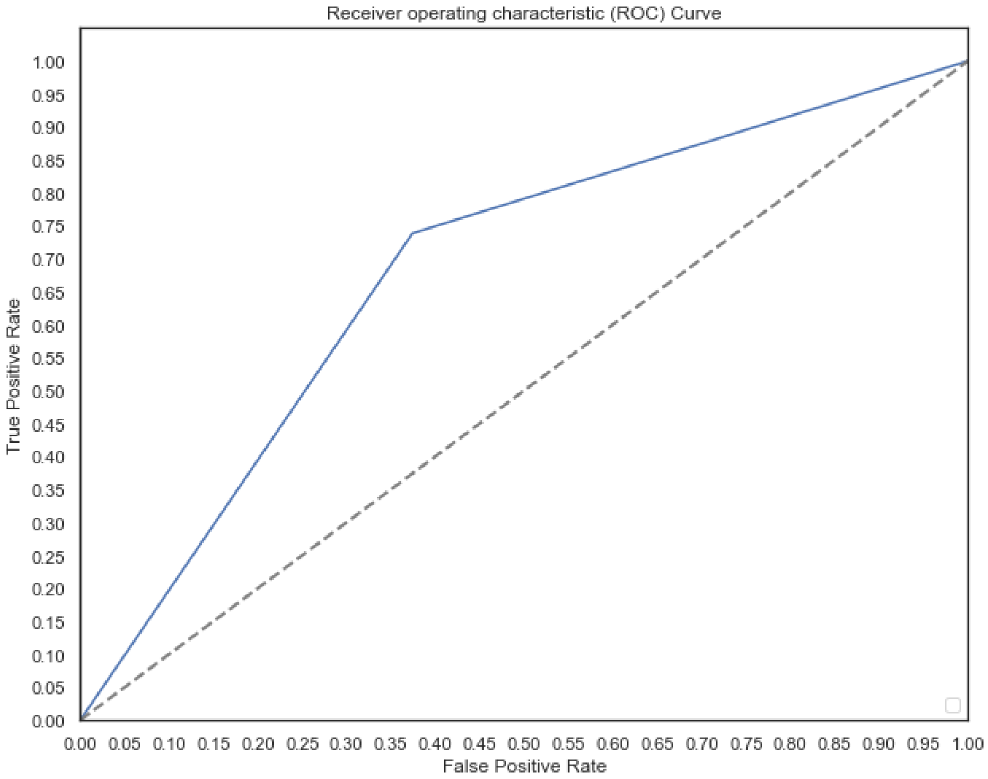
```
In [ ]:
```



Under-sampled models with Logistic regression

For this baseline logistic model, the udpated dataset has better performance. I'll move forward with these complete set of features as I modify the model moving forward.

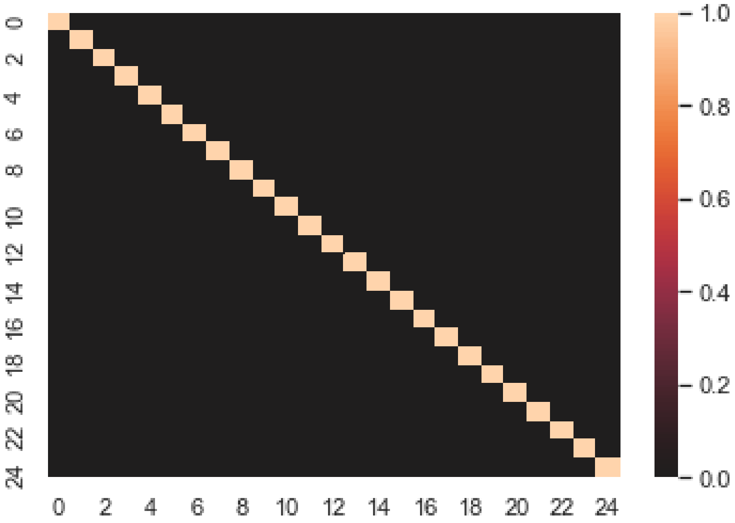




Pipeline classification models

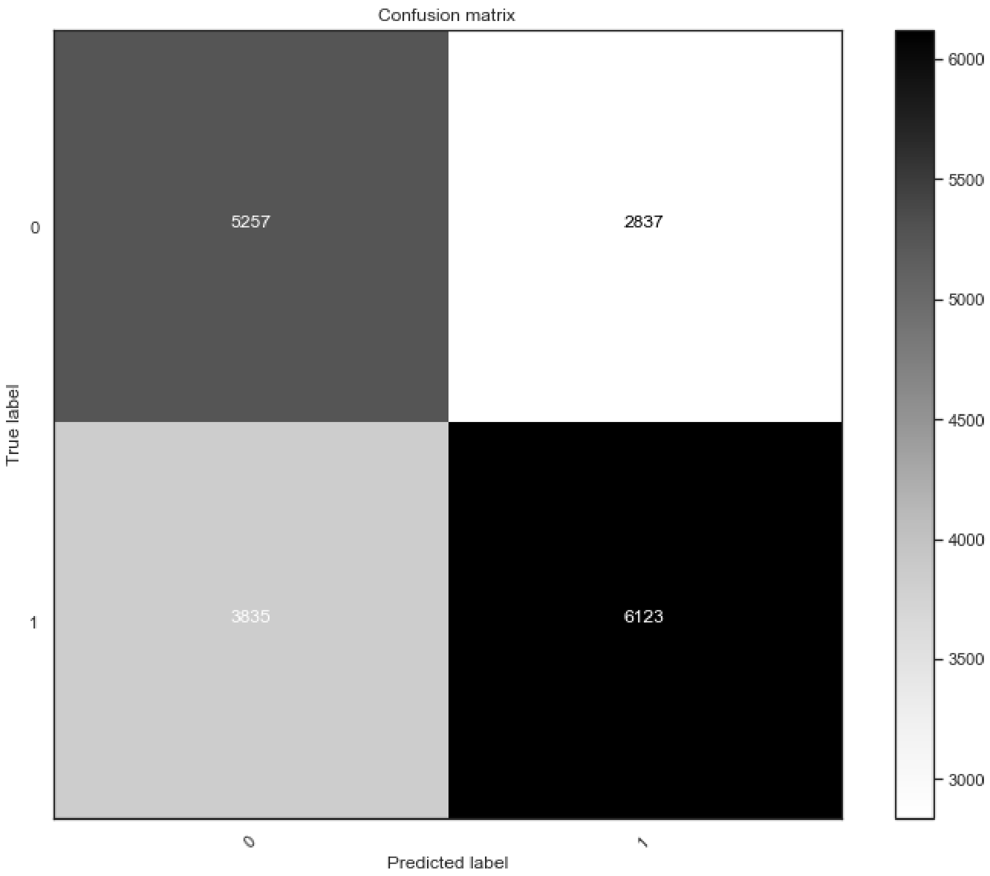
Principle Component Analysis

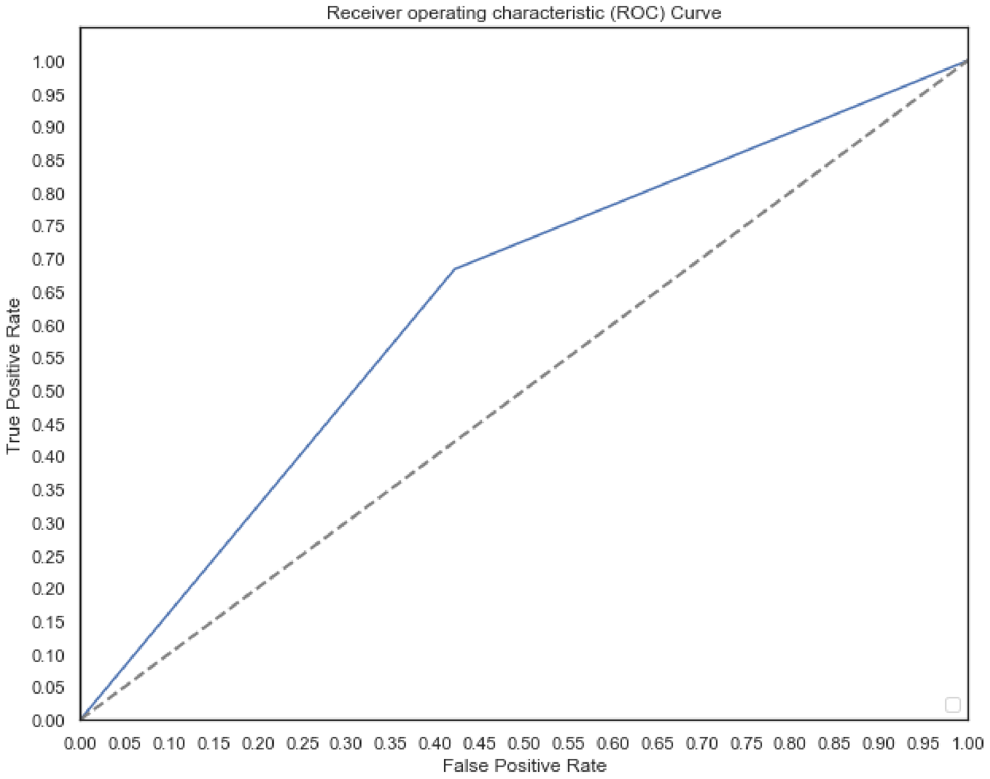
For this final model, to start, I implemented pricinple component analysis to reduce the number of dimensions as a precaution against issues arising from too many features. I kept 25 features in my model since it still enabled a model that could explain over 90 percent of the varioation. As you can see below, the colinearity among the remaining features is insignificant.



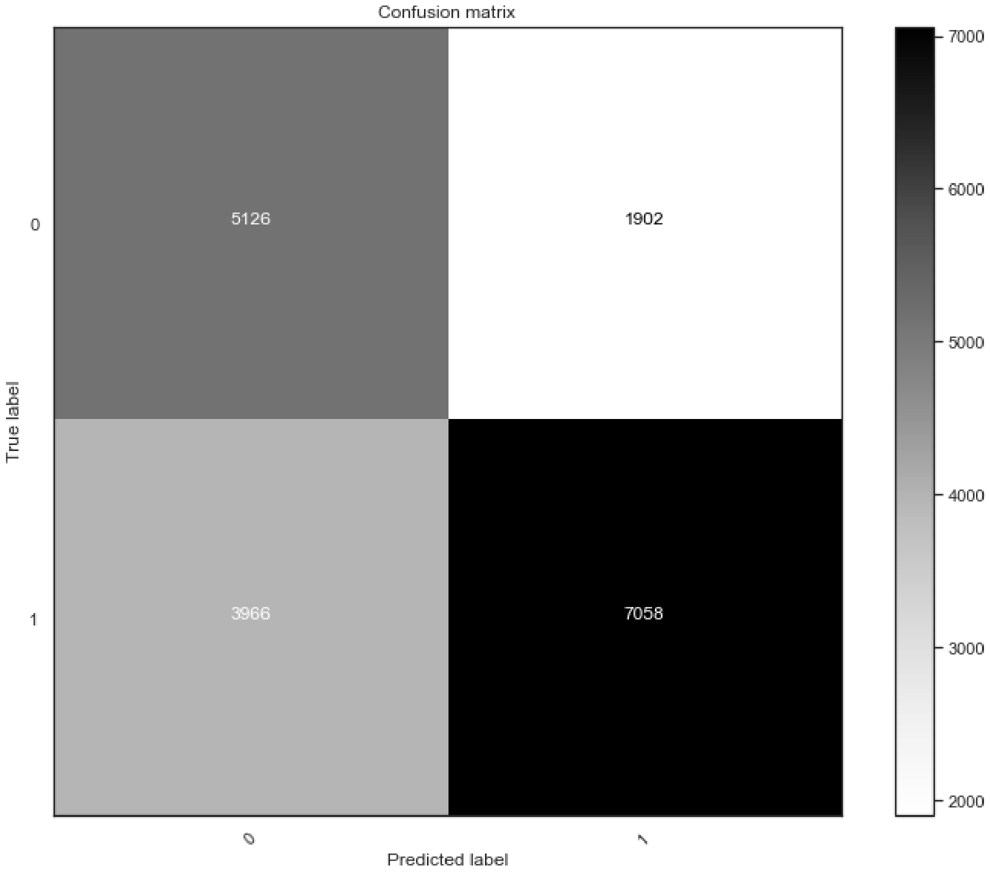
Pipelines

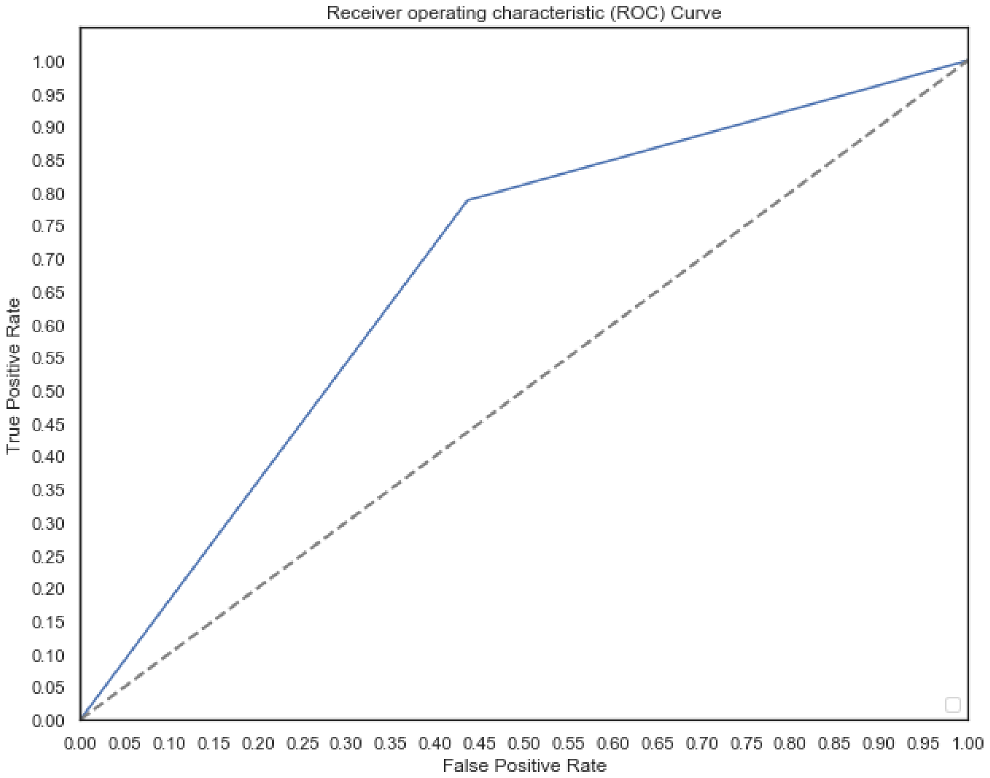
```
In [ ]:
```


Confusion matrix
0
1
True label
5257
2837
3835
6123
0
1
Predicted label
6000
5500
5000
4500
4000
3500
3000

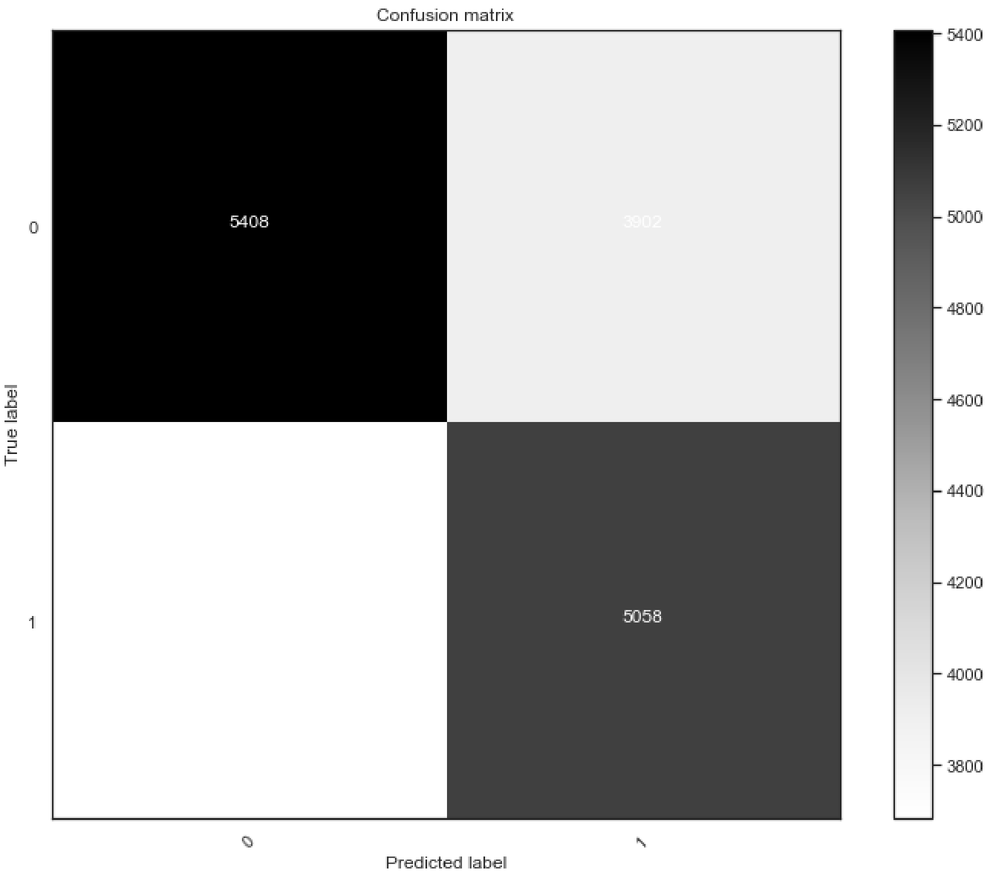

Receiver operating characteristic (ROC) Curve
True Positive Rate
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95 1.00
False Positive Rate

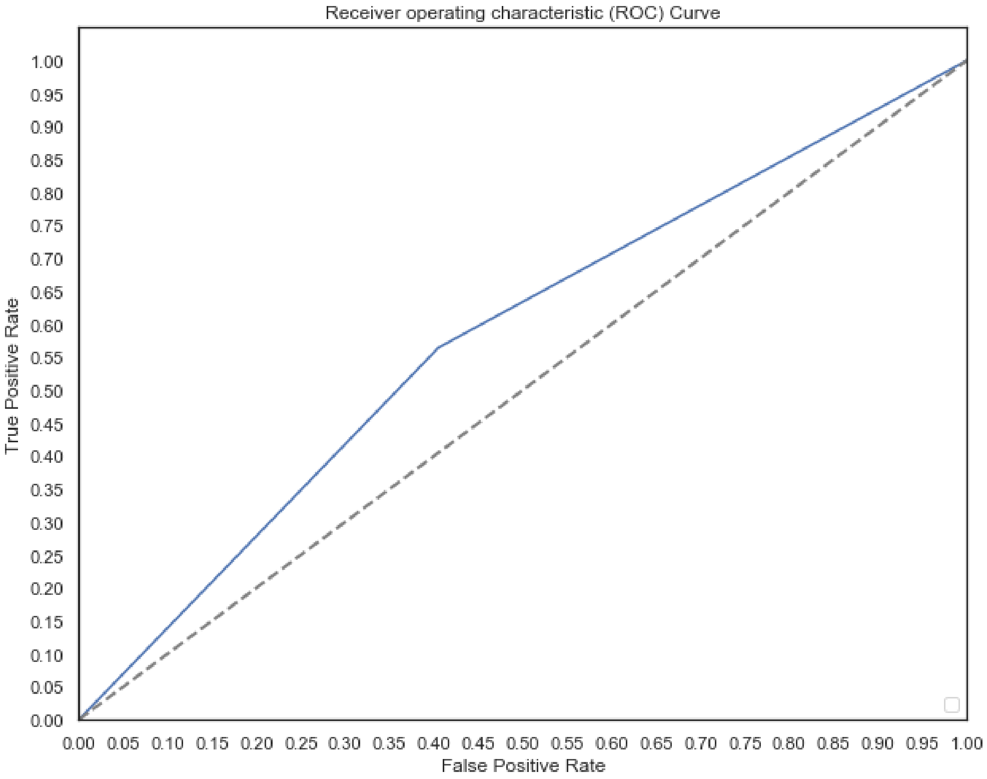
```
In [ ]:
```


Confusion matrix
0
1
True label
5126
1902
3966
7058
0
1
Predicted label
7000
6000
5000
4000
3000
2000


Receiver operating characteristic (ROC) Curve
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
True Positive Rate
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95 1.00
False Positive Rate

```
In [ ]:
```


Confusion matrix
0
1
5408
3902
5058
True label
0
1
Predicted label
5400
5200
5000
4800
4600
4400
4200
4000
3800


Receiver operating characteristic (ROC) Curve
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
True Positive Rate
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
False Positive Rate

```
In [ ]:
```

Gridsearch

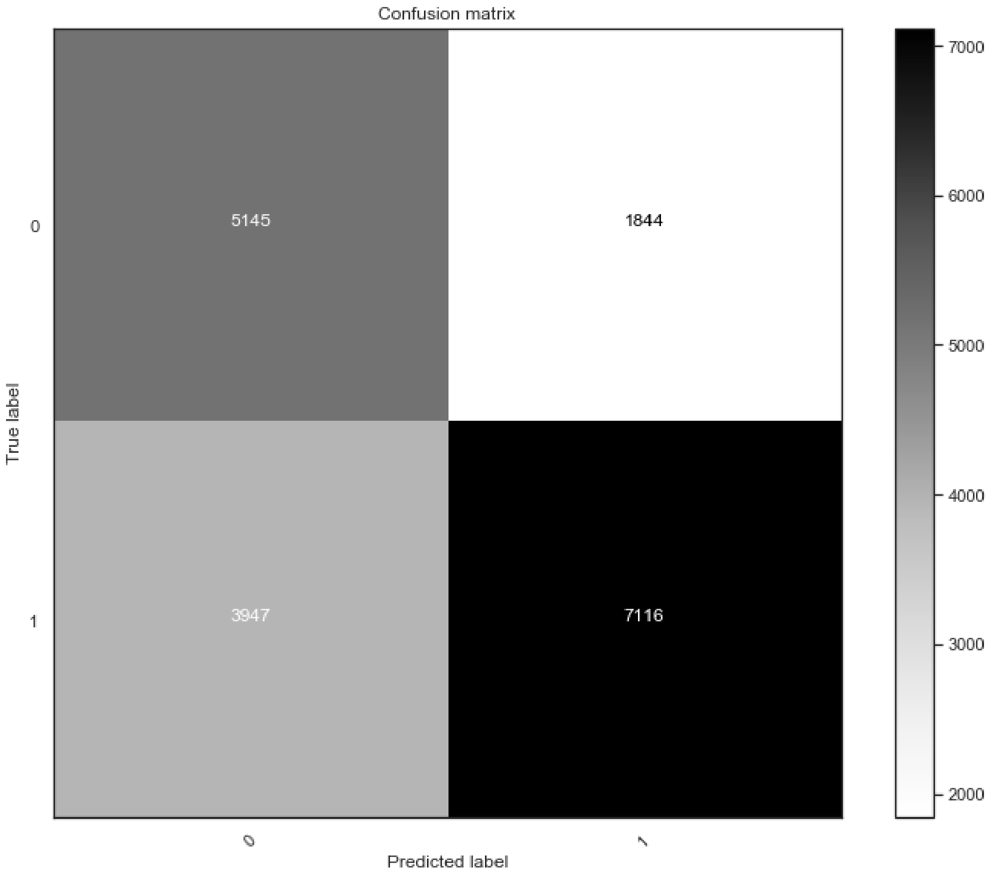
Given my decision to use an udnersmpampled model, I next used grid search to find the right parameters to use for the SVM model. The gridsearch results indicated that the best parameters were:

```
In [ ]:
```

```
Best accuracy: 0.681

Best params:
 {'clf__C': 10, 'clf__gamma': 0.01, 'clf__kernel': 'rbf'}
```

```
In [ ]:
```

```
              precision    recall  f1-score   support

           0       0.74      0.57      0.64      9092
           1       0.64      0.79      0.71      8960

    accuracy                           0.68     18052
   macro avg       0.69      0.68      0.68     18052
weighted avg       0.69      0.68      0.68     18052
```


Confusion matrix
True label
Predicted label