TMBA: Tech MBA

BIG DATA & BUSINESS ANALYTICS

Twitter

Group 4:

Raúl Cárdenas

Luisa Toro

Thomas Werner

Aman Kumar Vikrant

Professor:

DAVID GORDO

IE Business School

Madrid, Spain

November 18, 2022

Problem Statement

Problem:
What makes a tweet viral? Is it possible to classify a tweet as viral or not based on its metadata? How can this information be leveraged to unlock new ways of monetization for users and increase the value proposition of Twitter Blue?

Background:
For the past couple of years, Twitter has been looking to rely less on ads and more on subscription services to generate revenue. With the introduction of Twitter Blue, the company is taking its first steps towards this goal by allowing users to pay for premium features and perks such as the ability to undo tweets, a reader mode for long threads, and bookmark features to manage saved content. However, more is needed to entice users to pay USD 4.99 monthly. In 2021, Twitter generated USD 5.08B in revenue, of which only USD 6.4M came from subscription services.

Relevance:
With current plans to revamp the Twitter Blue premium subscription by upping the price to $19.99 per month, Twitter needs to include additional features that add value to creators to increase the adoption of its subscription services. An option to do this could be to develop a more comprehensive suite of creator tools similar to what Youtube has done with Youtube Studio. These tools could range from monetization programs (like Super Chat and Super Stickers) to dashboards to help users create content more effectively and build audience engagement.

Objective:
This research aims to understand which components in a tweet's metadata are relevant to make a tweet go viral and classify them accordingly. The intent is to leverage these insights to create tools that increase the value of Twitter Blue (thus increasing user adoption) and help power users create more effective and engaging content for their audience.

Data Exploration

Context
The dataset in this project contains the metadata for 11,099 tweets extracted using Twitter's API. The tweets span from 2006 to 2018.

Content
The dataset contains a total of 31 columns. The features relevant to the analysis are:

- **retweet count:** total number of retweets (*3*)
- **favourite count:** total number of favorites (*5*)
- **created at:** date of creation of the tweet (*2018-07-31 13:34:40+00:00*)
- **text:** the content of the tweet (*RT @KWWLStormTrack7: We are more than a month …*)
- **user:** dictionary containing user-related metadata (*{'id': 145388018, 'id_str': '145388018', 'name…*)

- **metadata:** dictionary containing overall tweet metadata (*{'iso_language_code': 'en,' 'result_type': 're…*)

Key Considerations
Below are the steps and key considerations that were followed to prepare the dataset for modeling:

I. Target column - What makes a tweet viral?

In order to train a supervised classification model, a target column that determines whether a tweet is viral (*1=viral, 0=not viral*) is required. The criterion used to assess tweet virality is: *If "retweet count" is greater than the 75th percentile of "retweet count"* (in this case, 428.5 retweets), then a tweet is considered viral (*1*). Why? Although more complex criteria could have been used (e.g., days elapsed to reach x amount of retweets), the dataset does not contain the appropriate features to make such assumptions. In addition, going beyond 428 retweets complicates the modeling process since we don't have enough quality data on the high retweet count end of the spectrum. (See Exhibit 1.)

II. Feature engineering

In addition to the target column, new variables were created based on each tweet's metadata to enhance predictive modeling: (See Exhibit 2.)

List of features used for modeling:
- **tweet_length:** number of characters in tweet (*138*)
- **followers_count:** number of followers (*300*)
- **friends_count:** number of friends (145)
- **favourites_count:** total number of favorites (*5*)
- **verified:** account verification status (*True*)
- **language:** language used in tweet (*en*)
- **hashtags_count:** number of hashtags used in tweet (*2*)
- **words_count:** number of words in tweet (*75*)
- **month:** month in number format (*9*)
- **day_of_week:** day of the week in number format (*5*)
- **sentiment:** tweet sentiment analysis using dilbert NLP model (*positive, negative, neutral*)

IV. Fixing data imbalance

Since the threshold to consider whether a tweet is viral is above 428.5 retweets, and the retweet distribution is skewed to the right, there is a disproportionate ratio of samples in each class (*not viral: 8322, viral: 2777*). Therefore, to avoid biassing the model towards the dominant class during training, the data was resampled using imblearn's random oversampler after doing the train/test split (*not viral: 8322, viral:8322*).

Although this method led to overfitting during the modeling phase, the results were significantly better than with non-resampled data (see results section).

V. Encoding + Feature scaling

Finally, all features were scaled using sklearn's standard scaler to guarantee equal variance. It is essential to mention that this process was performed after the train/test split (with a pipeline) to ensure that the testing set remained untouched. In addition, all categorical features were encoded using one-hot encoding.

Method

In order to model and classify the data, a 80/20 train-test split was used to ensure enough test samples to verify the results. In addition, each model was trained twice, the first time without oversampling and the second with oversampling, to compare results (See Exhibit 4.). The following classification models were selected to test and validate our assumptions:

Random Forest Classifier

First, parameter tuning was performed using scikit's *GridSearchCV* to determine the optimal parameters for training. The classifier was trained using 300 estimators and a maximum tree depth of 10. Next, the model was evaluated using a confusion matrix and the performance metrics included in the classification report (recall, precision, f1-score). Finally, the relative importance of each feature was measured to understand what features actually matter to predict the target label.

K-Neighbors Classifier

Unlike random forest, the data was first normalized to improve model performance. Next, we optimized for k to find the ideal number of neighbors to use for training (See Exhibit 3.). Finally, the model was evaluated using a confusion matrix and the performance metrics in the model's classification report (f1-score, accuracy, recall).

Logistic Regression

Similar to the process with KNN, the data was first normalized using a pipeline. Then the model was trained and evaluated using a confusion matrix and the performance metrics in the model's classification report (f1-score, accuracy, recall). In addition, a test was conducted using the *class weight* parameter inside the Logistic Regression function (instead of the standard scaler) to normalize the data, but it did not yield good results.

Evaluation

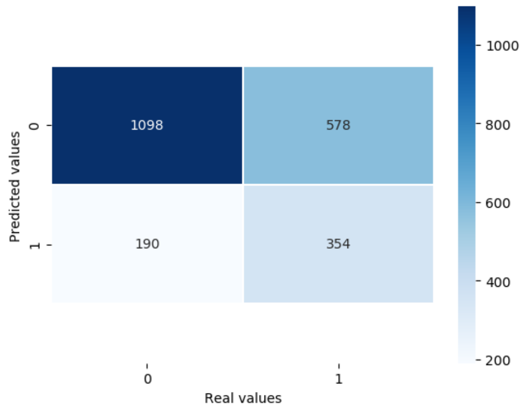
In order to evaluate, compare and select a model, the focus will be mainly on the f1-score for class 1 (viral). Why? As detailed in the results section below, the train/test scores and macro/micro averages are misleading since the models can easily predict if a tweet is not viral (most are not viral by nature). The real focus should be on how many true positives exist. Because of this, the f1-score is a good metric for unbalanced data since it combines precision and recall.

Results

Random Forest Classifier

Train Score: 0.8596
Test Score: 0.6540

	Precision	Recall	F1-Score	Support
0	0.85	0.66	0.74	1676
1	0.38	0.65	0.48	544
Accuracy			0.65	2220
Macro avg.	0.62	0.65	0.61	2220
Micro avg.	0.74	0.65	0.68	2220

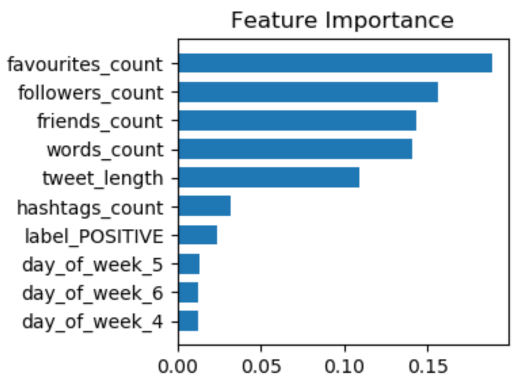


0=not viral, 1=viral - oversampled results

Initially, the model was trained without resampled data (See Exhibit 4.), and although it showed far fewer signs of overfitting (train: 0.80, test: 0.75), the model's f1 score was much worse (f1: 0.06 - only 18 correct predictions). After correcting the data imbalance (metrics shown above), the model's f1 score increased to 0.48 by accurately predicting 354 viral tweets. However, we can see clear signs of overfitting since the training score is ~20% higher than the test score. Moreover, we can see that our model's precision is poor (0.38), which means that we have a significant portion of false positives in the predictions.

As mentioned in the method section above, it is important to note that the test score of our model does not represent the whole picture. In this case, true negatives are helping boost the score since most tweets are not viral by nature and are easier to predict. In addition, while the random forest classifier got the highest f1-score out of all models, overfitting is too high to ignore. In this situation, collecting more data would be ideal for increasing the diversity in our classes (especially viral tweets) and improving performance during testing.

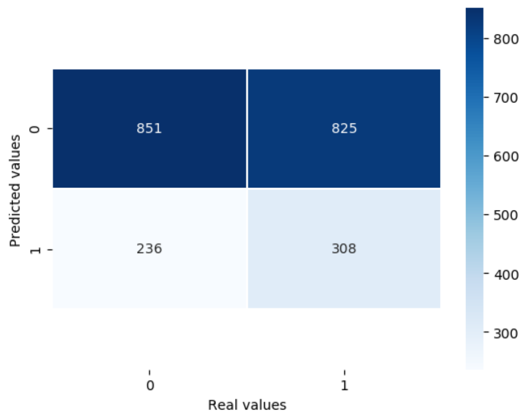
Finally, an added benefit of this classifier is that we can use the Gini impurity of each split to evaluate which features are more important. As shown in the graph below, it seems that the *language* and *month* features are not as relevant as we thought.



K-Neighbors Classifier

Train Score: 0.5868
Test Score: 0.5220

	Precision	Recall	F1-Score	Support
0	0.78	0.51	0.62	1676
1	0.27	0.57	0.37	544
Accuracy			0.52	2220
Macro avg.	0.53	0.54	0.49	2220
Micro avg.	0.66	0.52	0.56	2220



0=not viral, 1=viral - oversampled results

Similar to what occurred with our random forest classifier, KNN showed high train/test scores during the first pass without resampled data (train: 0.74, test: 0.75). However, the model made 0 true positive predictions, yielding an f1-score of 0.0 for class 1 (viral tweets). After rebalancing the data (example above), we can see that the f1-score increased to 0.37. However, the recall score is now ~10% lower than the random forest, which already had many false positives. In this case, KNN yielded the lowest f1-score out of the three models, making it the worst-performing option.

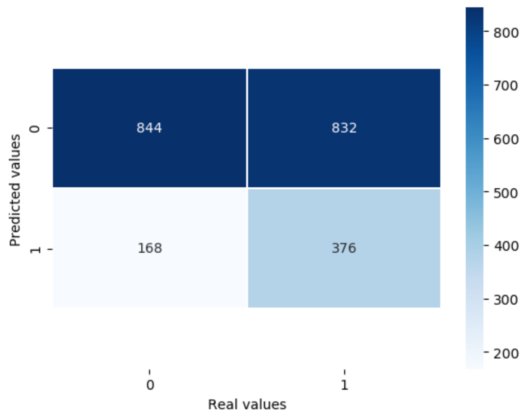
Finally, to understand how the classifier performs with different k values, we modeled and graphed all test scores between k=1 and k=200. The results show that the score stabilizes at k=~25 without significant improvement after that point. (See Exhibit 3.)

(See Exhibit 4 for unsampled vs. resampled results)

Logistic Regression

Train Score: 0.5941
Test Score: 0.5495

	Precision	Recall	F1-Score	Support
0	0.83	0.50	0.63	1676
1	0.31	0.69	0.43	544
Accuracy			0.55	2220
Macro avg.	0.57	0.60	0.53	2220
Micro avg.	0.71	0.55	0.58	2220



0=not viral, 1=viral - oversampled results

Following the same process, we first trained our logistic regression model with the unbalanced dataset (See Exhibit 4.). The results, as expected, show poor f1 and recall scores (0.02 and 0.01, respectively) with solely six accurate predictions for class 1 (tweet is viral). However, the model improved significantly after resampling the dataset (see results above). For example, even though the train/test scores are low compared to random forests, there is significantly less overfitting. In addition, the f1-score of 0.43 is our second best (376 correct predictions) compared to an f1 of 0.48 with random forests.

Until now, random forest and logistic regression seem like promising options for our task, given that we can continue to reduce overfitting and improve f1-scores with enhanced data.

Conclusions

Model Selection

Based on the results and the objective established at the beginning of the paper, we would select a Random Forest for our tweet classification task. The rationale behind this decision is as follows:

1. Although the model shows clear signs of overfitting, it performed the best when looking at the individual f1-score and precision for class 1 (viral tweet). The overfitting issue is addressed in the recommendation section below.
2. Precision is an essential metric for our experiment since we want to avoid classifying tweets that are not viral as viral (false positives).
3. Highest accuracy. Best overall performance when classifying tweets.
4. Recall for class 0 (non-viral tweet) is the highest. This means we are classifying most of our non-viral tweets correctly.
5. Ability to understand feature importance by using Gini impurities.

This decision is based on the assumption that we will continue to improve our dataset to mitigate overfitting. If this is not possible, a second good option would be to use logistic regression.

Value Add

As stated at the beginning of the report, our objective is to understand what components are relevant to whether a tweet goes viral or not and to be able to classify it accordingly. Even if our models could have performed better, this is a first step in the right direction. We have learned which features related to a tweet's content, user profile metrics, and DateTime are relevant to the prediction. For example, this data can be leveraged to develop tools that help users better understand the components they should focus on to improve their engagement.

Finally, understanding virality could be an additional lever for the company to enhance engagement. Knowing what affects user behavior and how users interact with the content they create is essential to improve user experience and the platform.

Recommendations

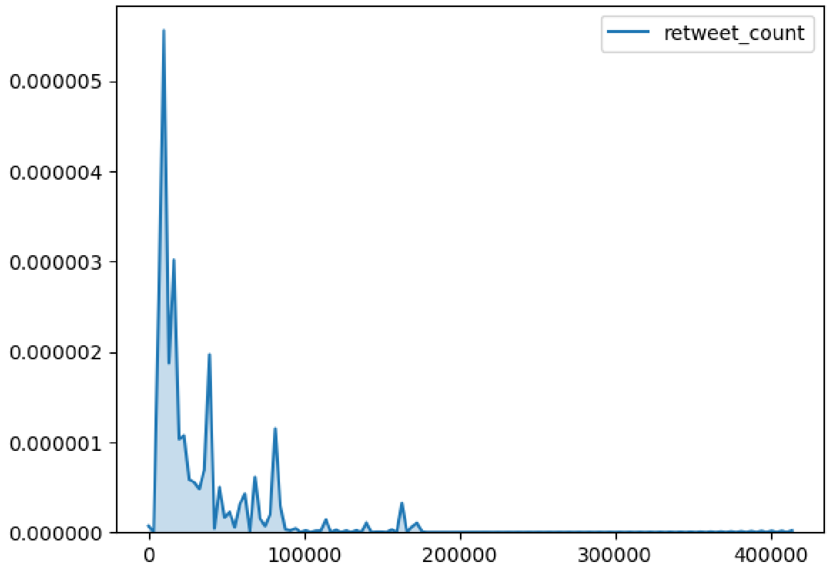
Based on what we learned, the following is a list of suggestions that might help improve model performance in future iterations:

1. Improve the definition of what makes a tweet viral. This could be complemented with DateTime data (currently unavailable in this dataset) to understand how long each tweet takes to reach its total retweet count.
2. More ensemble methods could be used to produce one optimal predictive model.

3. Collect more data to increase the diversity of both classes (especially viral tweets) to reduce overfitting.
4. Adding features that contain geolocation could also be beneficial to improve model accuracy.
5. More complex NLP models can be used to improve every tweet’s sentiment and context analysis.
6. Additional resampling methods could be tested to correct data imbalance.

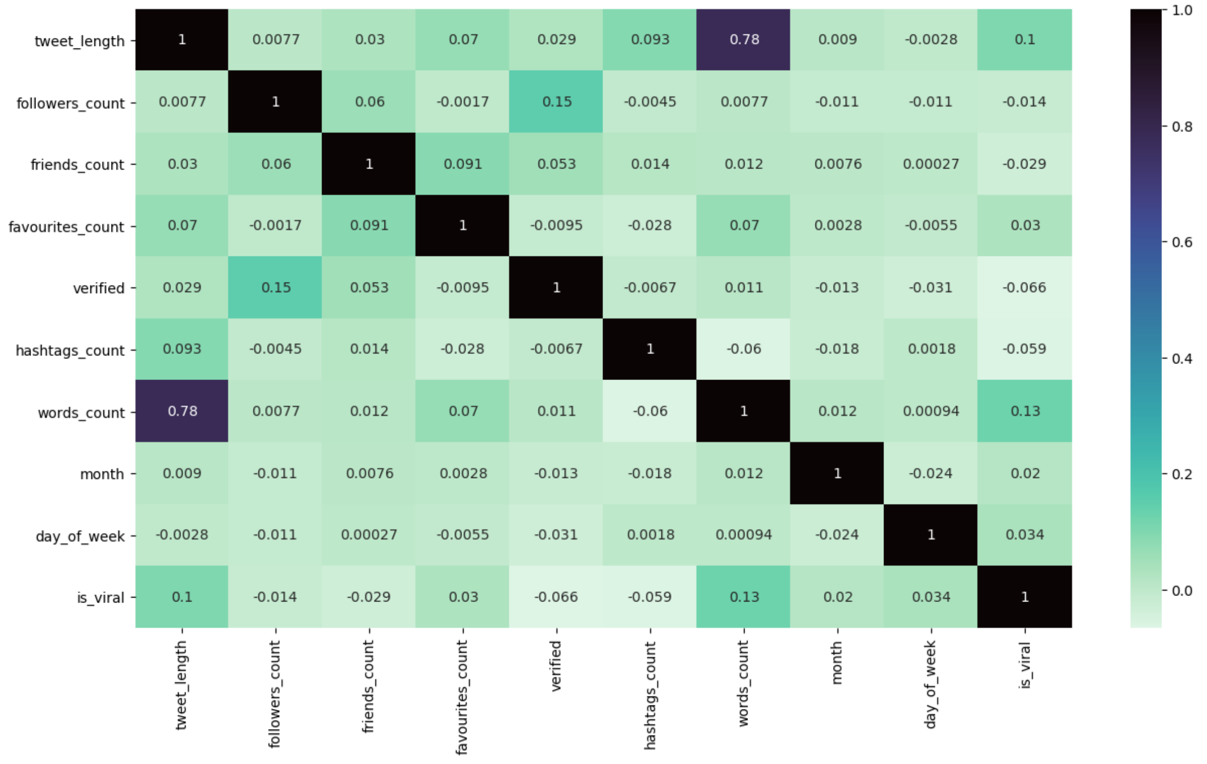
References

Exhibit 1 Distribution of retweet counts



```
count     11099.000000
mean       2777.956392
std       12180.169923
min           0.000000
25%           0.000000
50%          13.000000
75%         428.500000
max      413719.000000
Name: retweet_count, dtype: float64
```

Exhibit 2 Correlation matrix (before one-hot encoding)



	tweet_length	followers_count	friends_count	favourites_count	verified	hashtags_count	words_count	month	day_of_week	is_viral
tweet_length	1	0.0077	0.03	0.07	0.029	0.093	0.78	0.009	-0.0028	0.1
followers_count	0.0077	1	0.06	-0.0017	0.15	-0.0045	0.0077	-0.011	-0.011	-0.014
friends_count	0.03	0.06	1	0.091	0.053	0.014	0.012	0.0076	0.00027	-0.029
favourites_count	0.07	-0.0017	0.091	1	-0.0095	-0.028	0.07	0.0028	-0.0055	0.03
verified	0.029	0.15	0.053	-0.0095	1	-0.0067	0.011	-0.013	-0.031	-0.066
hashtags_count	0.093	-0.0045	0.014	-0.028	-0.0067	1	-0.06	-0.018	0.0018	-0.059
words_count	0.78	0.0077	0.012	0.07	0.011	-0.06	1	0.012	0.00094	0.13
month	0.009	-0.011	0.0076	0.0028	-0.013	-0.018	0.012	1	-0.024	0.02
day_of_week	-0.0028	-0.011	0.00027	-0.0055	-0.031	0.0018	0.00094	-0.024	1	0.034
is_viral	0.1	-0.014	-0.029	0.03	-0.066	-0.059	0.13	0.02	0.034	1

Exhibit 3 K-Neighbors - Model Score vs. K (Train)

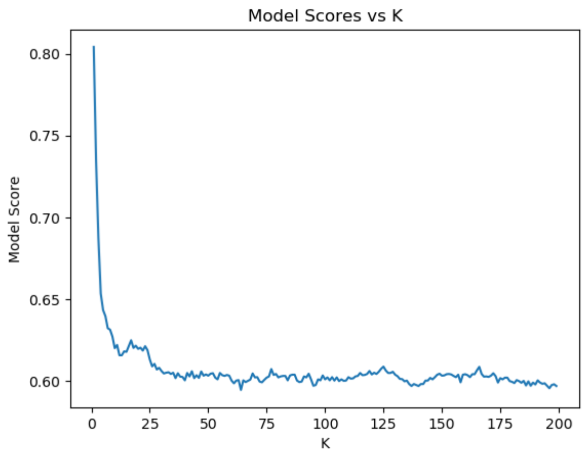
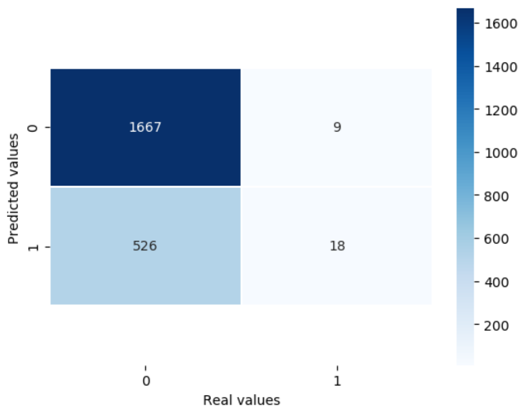
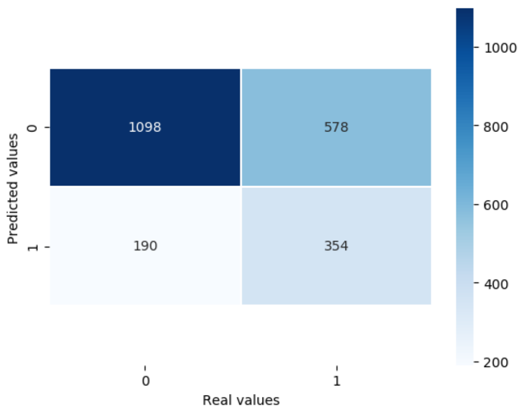


Exhibit 4 Unsampled vs. Resampled Dataset Scores

Random Forest

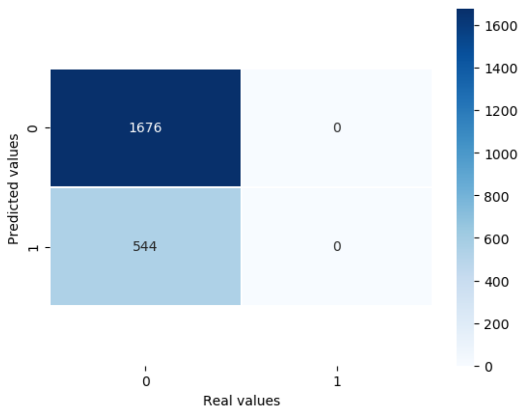


Unsampled Dataset

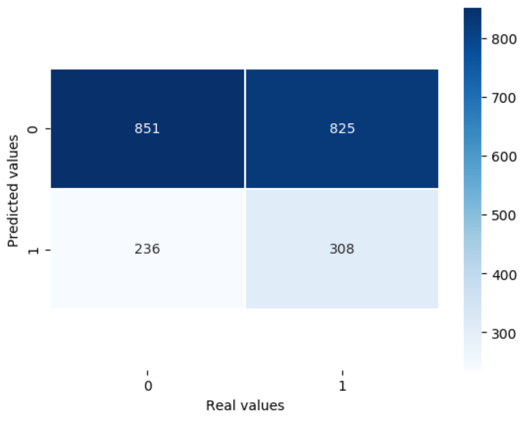


Resampled Dataset

KNN

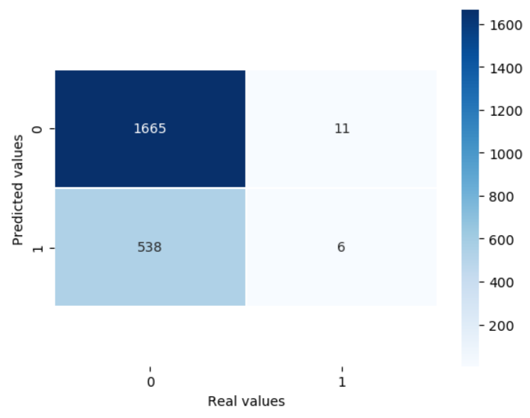


Unsampled Dataset

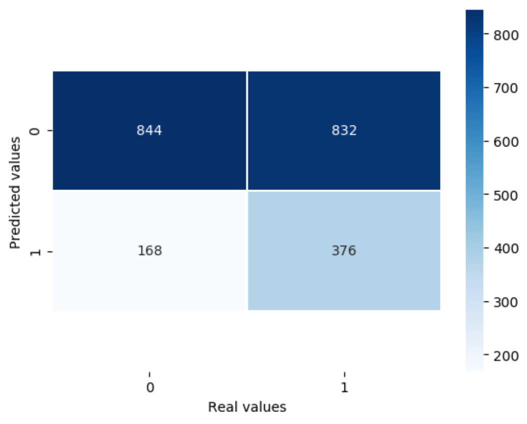


Resampled Dataset

Logistic Regression



Unsampled Dataset



Resampled Dataset