Fake Review Detection:
An Ensemble Approach to Building a Classification Model

Team Artificial Cilantro Soojin Kim(sk5291): Member responsible for uploading submissions
Sujeong Cha(sjc433), Eric He(eh1885), Hyun Jung(hj1399), Young Jin Park(yjp228)

1. Introduction

Online reviews have increasingly gained popularity and become an integral part of customers' purchasing decisions. According to the Local Consumer Review Survey, 82% of customers regularly read online reviews and find information about local business, and 76% of people surveyed responded that they trust online reviews as much as the personal recommendations[1]. As a result, detecting fake reviews has gained much importance for many online venues in order to provide the customers more credibility and to protect the venue's reputational risks. This project aims to compare different classification models that detect fake reviews. The best score was obtained by XGBoost model, combined with engineered features, Doc2Vec features from Gensim library, and TF-IDF probability estimate feature, with the average validation and test precision score reaching 0.446 and 0.440 respectively.

2. Approach

Our approach to this problem was to utilize both review texts and its metadata to extract meaningful features to train our model. After rounds of preliminary analyses on the metadata, we identified and engineered metadata features to build our baseline model using XGBoost. Next, we performed review text analysis using multinomial Naive Bayes to capture sentiment and text patterns from the contents. We also applied Doc2Vec, converting each review into numerical vector representation using the word vectors.

3. Experiments

3.1. Datasets

The fake review detection dataset was downloaded from CodaLab, which utilized the review data from Yelp.com with feature variables including ex_id (review id), user_id, prod_id (restaurant id), rating (on a 1-5 scale), date, review text and the label. The entire dataset contains 358,957 instances, with train/validation/test set split of 70%/10%/20%. We have observed imbalanced classes (roughly 12% of reviews were fake) in the label but decided not to use advanced sampling techniques such as SMOTE or ADA-SYN as we deemed those techniques may not markedly improve the performance given that we still have adequate amount of instances for the minority class. The class imbalance issue is further addressed in the discussion section (Section 4.1).

3.2. Baseline Model

As a baseline approach, we decided to use the boosting tree model using XGBoost with metadata only. We came up with a few hypotheses for feature engineering and validated some of them, which were later used to build features for our final model.

3.2.1. Feature Engineering

One of the hypotheses we had was that if a user has written a fake review, then naturally all of the reviews written by that user would be fake. To validate our assumption, we studied the relationship between the number of users, their review counts, and the proportion of the fake reviews (Figure 1). The top 2 rows in the heatmap correspond to users who have written at least 1 spam review, with the top row corresponding to users who have written nothing but spam reviews. The bottom row of the heatmap shows the number of users who have 0% of their reviews tagged as fake. In general, there were more users whose reviews were all fake as opposed to those who wrote "partially" fake reviews, supporting our hypothesis. In order to capture this trend in the dataset, we have added features "user_review_count" and "has spammed before".
Other features added to detect any anomalies or patterns from the fake review data are "review_word_count" and "review_letter_count", as we have noticed that fake reviews were more likely to have lower word count compared to the legitimate reviews. Moreover, to capture the sentiment, we incorporated "review_capitalization_count" and "review_punctuation_count". Also, we suspected that some of the restaurants may draw more fake reviews compared to other restaurants. In order to capture our findings and other hypotheses, the final list of the features we extracted from the metadata is the following: *User review count, Has spammed before, Restaurant review count, Days since first user review, Days since first restaurant review, Review word count, Review letter count, Review capitalization count, Review punctuation count.*

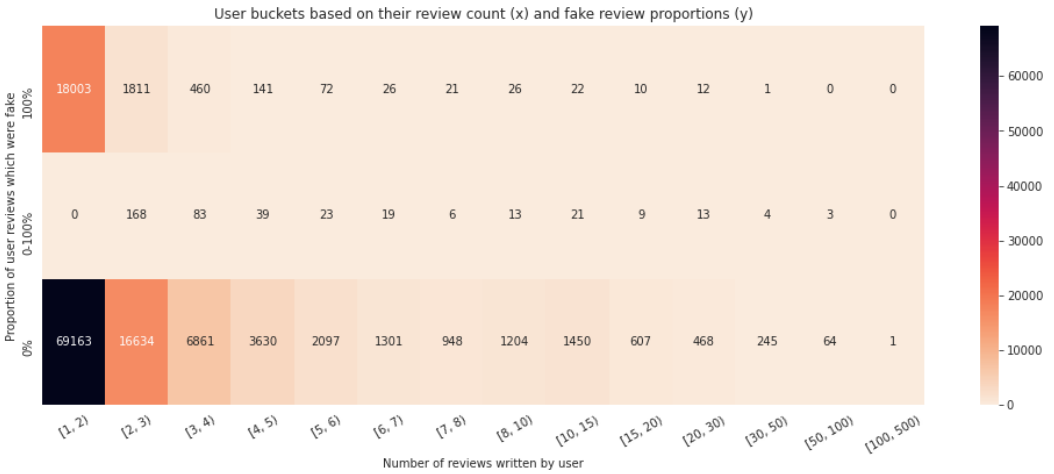

Figure 1. Heatmap showing the number of users based on the review count and fake review proportion

3.2.2 Model Selection

Using the aforementioned features (metadata only), we trained three models: Boosting Tree with/without positive weight adjustment and Logistic Regression. Among the three models, Boosting Tree without positive weight adjustment outperformed the other two in both training and validation set with average precision score of 0.672 and 0.401 respectively (Figure 2).

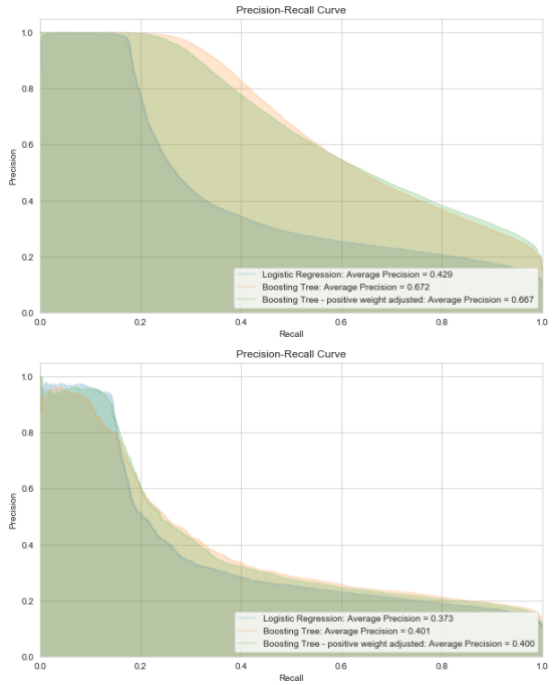

Figure 2. Average Precision Score of Boosting Tree and Logistic Regression (Top: Training Set, Bottom: Validation Set)

3.2.3. Text Mining

After running our base model, we realized that simply extracting word counts or counts of capitalization/punctuations does not suffice the text sentiment we wanted to observe. Thus, we decided to perform relatively quick text mining to further improve our Boosting tree model. We first transformed the review text with the normal CountVectorizer and fed into the Naive Bayes model and was able to obtain the average precision of 0.204 on training data (without the metadata features) and 0.184 on validation. After processing the review text through TfidfVectorizer, we observed a huge performance gain over the one with CountVectorizer with average precision of 0.232 on training data and 0.203 on validation. We also explored using the Complement NB as suggested by sklearn for imbalanced data sets but did not achieve any performance gain over the standard Multinomial NB model[2].

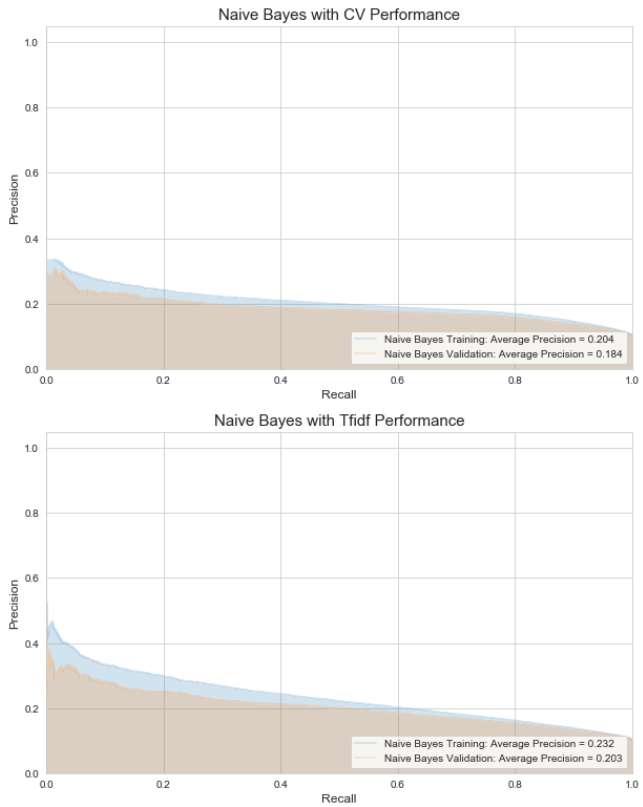

Figure 3. Average Precision Score of Naive Bayes (Top: CountVectorizer, Bottom: TF-IDF)

By observing highest and lowest TF-IDF scores, we noticed that the words with the highest scores were more specific words related to certain cuisines or ambience or specific sentiments (e.g. “belly”, “complement”, “remarkable”, “patio”, “farm”, “5pm”) where the words with the lowest scores were extremely generic words such as (e.g. “food”, “good”, “like”, “delicious”). The result resonates with one of our hypotheses (and the feature review word count) where the reviews with the generic words only tend to be short and the reviews with the specific words were longer, which requires the users to put more efforts into writing reviews.

3.3. Results

For the final model, we decided to add the NB probability estimates from our text mining as a feature and word-vectors generated by Doc2Vec from gensim library. For the final model, we used the boosting tree from XGBClassifier with the following hyper parameters, which yielded the test average precision score of **0.440** as shown in Figure 4.

```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.300000012, max_delta_step=0, max_depth=4,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=42, reg_alpha=5,
              reg_lambda=30, scale_pos_weight=8.716642782447035, subsample=1,
              tree_method=None, validate_parameters=False, verbosity=None)
```

Figure 4. Final Model Parameter Configurations
(max_depth=4, reg_alpha=5, reg_lambda=30)

A big gap between initial training and validation performance difference was observed when the default parameter setting was used. Hyperparameters were adjusted to prevent overfitting on the training data. The results from running different classification models are listed below.

Model	Features	Dev Avg Precision	Test Avg Precision
Naive Bayes	Preliminary (w/o review contents)	0.132	-
Logistic Regression	Preliminary (w/o review contents)	0.373	-
XGBoost (Baseline Model)	Preliminary (w/o review contents)	0.401	-
XGBoost w/ Positive Weight Adjusted	Preliminary (w/o review contents)	0.400	-
Naive Bayes with CV	‘Review’	0.184	-
Naive Bayes with Tf-idf	‘Review’	0.203	-
Complement Naive Bayes with Tf-idf	‘Review’	0.203	-
XGBoost w/ Positive Weight Adjusted	‘Review’ (Doc2Vec)	0.168	-
XGBoost w/ Positive Weight Adjusted	Preliminary + NB-TF-IDF + Doc2Vec	**0.446**	**0.440**

Table 1. Evaluation Results from Running Various Classification Models

3.4. Error Analysis

Upon analyzing the sources of errors of our model, we learned that Yelp’s filter system weighs behavioural features more than linguistic features. We found several differently labeled but identical reviews which denotes that Yelp’s filtering system places more weight on behavioral features than linguistic features (Figure 5, 6). This suggests possible improvement in our model performance by incorporating more behavioral features, which was not available in the given dataset. Possible future work may involve adding features such as user trust factor based on friendship measures between users.

	ex_id	user_id	prod_id	rating	label	date	review
166588	238237	8995	625	2.0	0	2014-11-03	Appetizers were good. Had scallop carpaccio wh...
166523	238150	4186	625	3.0	1	2014-04-16	Appetizers were good. Had scallop carpaccio wh...

Figure 5. Table that shows identical review contents with different labels

4. Discussion

4.1. Evaluation of Findings

Class Imbalance Issue In order to address the class imbalance issue, we balanced the positive and negative weights using scale_pos_weight parameter from XGBClassifier. Also, we experimented with a Complement Naive Bayes model to combat this issue. However, the performance did not significantly improve, but rather slightly worsened or remained the same. This could be due to the overfitting as the class imbalance (genuine: 88% vs. fake: 12%) is not very severe and there are still sufficient data points for the minority class.

Feature Importance of TF-IDF As expected, utilizing TF-IDF vectorizer performs better than using Count vectorizer for Naive Bayes model. In general, term frequency transformation is effective for MNB since MNB is not very effective to model text data and empirical term frequency distributions have a heavier tail, which makes it look like a power-law distribution instead of multinomial distribution[3]. Thus, using term frequency transformation helps transform it to a multinomial-like distribution. However, CountVectorizer forces the frequent words to have high weights in the model, which leads to the common words like “a”, “the”, “and” to have higher weights than the topic bearing words. This would likely skew the model. TF-IDF performs better because it balances term frequency with inverse document frequency, which assigns rarer words with higher scores, thus effectively preventing common words from dominating in classification.

Utility of Doc2Vec It is surprising that Doc2Vec didn’t significantly improve the performance (Figure 6). Doc2Vec was applied to calculate similarity measures between documents by adding a document vector, which holds a

numeric representation of a document. When we compared the final XGBoost model with Doc2Vec to the same model without Doc2Vec, the performance difference was only 0.001. We speculate that this trivial difference is due to the TF-IDF prediction feature that may have already captured the similarity pattern from the review text by assigning high weights to important words and thus implicitly memorizing the concept of a document.

XGBoost over MNB The XGBoost model combined with the engineered features, NB-TFIDF prediction feature and Doc2Vec feature displays the best performance amongst other classification models we experimented. We chose XGBoost as our final model instead of MNB because the XGBoost model with the precision of 0.401 outperforms the MNB model with the precision of 0.132 when trained with preliminary features only (Table 1). Also, XGBoost is proven to improve speed and push the limits of classification performance, which was also proven by our experiments on different classification models[4].

	feature_name	importance
11	spammed	0.677544
1	user_review_count	0.159512
12	pred_nb	0.035692
0	rating	0.014118
7	capitalization_count	0.010217
3	restaurant_review_count	0.006812
8	letter_count	0.005507
10	days_since_first_restaurant	0.005346
4	restaurant_average_rating	0.004840
2	user_average_rating	0.004660
6	wordcount	0.004540
9	days_since_first_user	0.003344
20	d2v_7	0.003044
17	d2v_4	0.002994
36	d2v_23	0.002598

Figure 6. Feature Importance Table Ordered by Importance. Doc2Vec features, such as d2v_27 and d2v_7, don't carry much importance with the maximum importance of 0.008374, while the features representing reviewers' behaviors such as the indicator feature that tells whether a user has spammed in the past or not ("spammed") and the number of reviews written by a user ("user_review_count") carry the highest importance values of 0.68 and 0.16 respectively. TF-IDF feature(pred_nb) is displayed as the third most important feature carrying a feature importance measure of 0.036.

4.2. Conclusion & Possible Next Steps

In this project, a total of 9 different variations of classification models were reviewed for detecting fake Yelp reviews. Our experiments show that the XGBoost model displays the highest prediction score when combined with engineered features, Doc2Vector and the NB-prediction feature while logistic regression with engineered features only shows the lowest score.

As far as the training time is concerned, vectrozing words using Doc2Vector took the longest with the minimal improvement in performance. Though we included this feature in our final model, we suggest removal of this feature for the larger dataset.

It's shown that the features representing reviewers' behaviors carry heavy weights compared to linguistic features, which shows the limitation of Yelp's filtering system. There are possibilities of improvement in performance by adopting other features such as utilizing reviews left by verified customers or tip count data. Tips are generally shorter than regular reviews and designed to convey quick suggestions or insights about the restaurants. Studies have shown that about 90% of spammers leave 0 tip, which suggests tip count data is possibly a valuable behavioral feature data for the future studies[5].

5. Bibliography

[1] Murphy, Rosie (2019). "Local Consumer Review Survey." *BrightLocal*. www.brightlocal.com/research/local-consumer-review-survey/.

[2] "1.9. Naive Bayes." *Scikit-Learn*, scikit-learn.org/stable/modules/naive_bayes.html.

[3] Rennie, J., Shih, L., Teevan, J. & Karger, D. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers. *Proceedings of ICML '03.*

[4] Brownlee, Jason. (2016). "A Gentle Introduction to XGBoost for Applied Machine Learning." *Machine Learning Mastery*. www.machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[5] Mukherjee, A., Venkataraman, V., Liu, B., & Glance, N. (2013). "Fake Review Detection: Classification and Analysis of Real and Pseudo Reviews." *University of Illinois at Chicago.* https://www.semanticscholar.org/paper/Fake-Review-Detection-%3A-Classification-and-Analysis-Mukherjee-Venkataraman/4c521025566e6afceb9adcf27105cd33e4022fb6#citing-papers