Fake Review Detection

Austin Gilmore
Computer Science
Arizona State University
Tempe, United States
acgilmor@asu.edu

Hsiang-Yu Hsieh
Computer Science
Arizona State University
Tempe, United States
hhsieh14@asu.edu

Shiue-Fu Tzeng
Computer Science
Arizona State University
Tempe, United States
stzeng1@asu.edu

Ruthwik Reddy Ratna
Computer Science
Arizona State University
Tempe, United States
rratna@asu.edu

Sai Likhith Yadav Seera
Computer Science
Arizona State University
Tempe, United States
sseera@asu.edu

***Abstract*—As companies continue to grow their web presence and move away from the traditional brick-and-mortar approach for their business operations, the importance of online customer reviews in building a trustworthy reputation continues to grow as well. Due to the pivotal role that these reviews play in manipulating a potential customer's purchase decision, many companies have resorted to "opinion spamming." This illegal strategy involves hiring anonymous individuals to manually or programmatically inject fake reviews to purposely mislead customers into trusting their products or distrusting the products of their competitors. Though these artificial reviews can be convincing at face value, subtle tendencies in both their language and behavior can be utilized to detect and separate these fake reviews from the ones written by real customers. In this project, we use a multitude of linguistic and behavioral features and apply them to various machine-learning models in an attempt to automate fake review detection.**

***Index Terms*—opinion spam, fake review detection, machine-learning**

I. Introduction

In order to automate fake review detection, various machine learning and deep learning models are proven effective by training the models on various review datasets on websites such as Amazon and Yelp. In this project, we choose the Yelp dataset and train machine learning and deep learning models on various features generated from the dataset. This report states how we deal with imbalanced datasets such as the Yelp dataset, feature engineering methods, feature analysis using machine learning models, preprocessing details, hyperparameter tuning of the models, and experiment details. We also provide an analysis, a conclusion, and possible ways to improve the performance.

II. Problem Statement

Though review "judges" can be hired to manually read through and classify the authenticity of submitted reviews, this not only necessitates additional funding from a third-party group but humans have also been shown to be inherently biased in this classification. Thus, automated fake review detection is critical in combating opinion spammer groups' manipulative actions. Using a Yelp review dataset that contains filtering information to provide ground truth for our classification techniques, our project will experiment with various textual and behavioral features (n-grams, maximum content similarity, rating entropy, etc.) used in previous fake review detection studies. After applying these features to a range of machine learning models, we will assess the accuracy and effectiveness of our classification to determine which combination of features and models is best for predicting the authenticity of a company's review corpus.

III. Related Works

The most commonly used machine learning models for fake review detection are Support Vector Machine (SVM), Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR), and deep learning neural networks. Apart from Random Forest, other tree-type algorithms are also popular in this field of study, such as XG-Boost and Decision Tree. According to multiple results from different works of literature [1] [2] [3], SVM, RF, and LR performed very well. We also want to try deep learning models. The literature[1] shows multiple experiments using various models. The models mentioned in this paper are single classifiers, ensemble classifiers using boosting or bagging, and various deep learning models such as Long-Short-Term-Memory (LSTM), Convolutional Neural Network(CNN), and Feed-Forward Deep Learning (FFDL). From the results of the paper, FFDL performed the best.

According to the paper, the better the model performs, the more features it has and the more data it contains. The features proposed in the literature [2] give us a good starting point for feature engineering. There are Review-centric features and Reviewer-centric features. Both types of features contain textual features. Reviewer-centric features include sentiment evaluations, meta-data features, burst features, and temporal features.

Because the Yelp dataset is imbalanced, the paper [3] proposes different methods for dealing with imbalanced datasets. Oversampling, Undersampling, and Cost-sensitive Learning are the three main categories to address the problem caused by imbalanced labels. According to the literature [3], undersampling outperforms the other two categories.

By learning from the literature mentioned above and another paper[4] proposing how yelp might do with the review filtering methods. We have a clear beginning on what features we should try to retrieve, what models should be used as classifiers, and how to deal with imbalanced data such as the Yelp dataset.

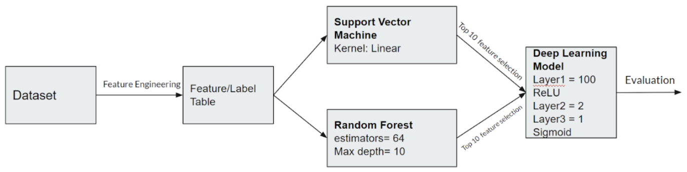

Fig. 1. System Architecture Flow Chart.

The architecture of our implemented fake review detection system can be viewed in Figure 1 above. To begin this detection process, the full Yelp dataset containing the labeled set of reviews was parsed through a series of scripts utilizing the Pandas python library. Leveraging the attributes contained in the dataset (review content, rating, date, etc.), statistical calculations were conducted to produce a set of features for each review that provided additional insight into the behavioral, textual, and temporal tendencies of the provided set of Yelp reviewers. Aside from the Pandas library, the NLTK library was also utilized in this step to tokenize the review content for various textual features.

From this feature engineering process, we ultimately generated 22 total features, which will be described in detail in the section below. These features were then consolidated into one table to be used in the model training/classification process, during which we utilized two well-known classification algorithms: support vector machines and random forest.

For the parameters of the random forest, we selected 64 estimators and 10 depths. For SVM, we used a linear kernel to train the model and classify each review in the dataset as either fake or real. After computing various metrics concerning the effectiveness of both of these algorithms, they were additionally utilized to compute the relative importance of each of the 22 features in their classification decision. After extracting the top 10 features from this ranked importance list from each trained model, these selected features were then fed separately into a feed-forward neural network deep learning model to provide a means of comparison.

The deep learning model we designed contains three layers. The first layer contains 100 neurons. The second layer contains 2 neurons, and the last layer (the output layer) contains 1 neuron. Between the first and second layers, the tensor has to pass through an activation function called the Rectified Linear Unit (ReLU). After the output layer, an activation function, Sigmoid, is used to bound the output value between 0 and 1. A review is classified as fake (label = 1) if the output value of the deep learning model is greater than 0.5; otherwise, the review is classified as real (label = 0). The optimizer we used is ADAM.

After training this deep learning model with the full set of 22 features, the top 10 features from the Random Forest model, and the top 10 features from the Support Vector Machine model, the accuracy, precision, and recall of each of the classification results were compared. From this comparison, we were able to determine the most effective classification algorithm for identifying fake reviews as well as the overall effectiveness of the features we chose to analyze.

IV. Dataset

A. Description

We use the Yelp dataset for our work. Rayana and Akoglu [5] used the Yelp dataset for their analysis, which included 608,598 reviews of 5,044 restaurants and hotels from 260,277 reviewers in New York State. This dataset consists of product and user information, the timestamp of the review, ratings, and a plain text review, which allows us to analyze our model. The authors have classified the reviews as genuine and fake with the help of the Yelp anti-fraud detector for the Yelp dataset, which produces accurate results. The reviews are divided into two classes: the spammer class (with fake reviews) and the benign class (with no fake reviews). By using this dataset, we believe that it allows us to retrieve both textual and behavioral features to feed into our classifiers.

We decide to do undersampling on the data because the dataset is unbalanced, with 528132 real reviews and 80466 fake reviews. The reason for choosing to use undersampling to solve the unbalanced problem is because, based on the results in the paper [3], undersampling is more beneficial to the model's training.

B. Feature Generation

Selecting the right set of initial features to compute from the dataset is pivotal to the effectiveness of both the model training and prediction processes. Because much prior research on this selection step has been done with high accuracy rates, we chose to integrate and combine the features from two separate research papers in the hope that the increased number of features will increase the probability of higher prediction accuracy and effectiveness.

In the ”Feature Analysis” paper written by Julien Fontanarava, Gabriella Pasi, and Marco Viviani, they split up their features into two main categories: review-centric and reviewer-centric [2]. Review-centric features refer to merely the textual content and metadata contained in each review, whereas reviewer-centric features look to examine the behavior of the user who made the review through various techniques. While they utilized 27 total features in their research, we excluded various semantic and computation-heavy features due to our team's limited computational resources. These excluded features include unigrams and bigrams computed from the word or phrase used within the textual content of the reviews, as well as average and maximum content similarity that examine the cosine similarity between the set of a particular user's reviews. Thus, we ultimately integrated 17 of their features into our feature set.

Additionally, in the ”What Yelp Fake Review Filter Might Be Doing?” paper written by Mukherjee, Venkataraman, Liu, and Glance, they chose a more limited set of 5 features pertaining only to the behavior of the reviewer and combined them with unigrams and bigrams to train their models and make classification predictions [4]. Due to the limitations mentioned above, we excluded unigrams, bigrams, and maximum content similarity and ultimately integrated four of their features into our feature set. The full description of each feature is described below.

C. *Review-Centric Features*

- **Ratio of capital letters** [2]: The number of words containing capital letters in comparison to the total number of words in the review.
- **Ratio of capital words** [2]: The number of words containing ONLY capital letters with respect to the total number of words in the review.
- **Ratio of-first person pronouns** [2]: The number of first-person pronoun words with respect to the total number of words in the review.
- **Ratio of 'exclamation' sentences** [2]: The number of exclamation sentences (ending in '!') with respect to the total number of sentences in the review.
- **Rating** [2]: The numerical rating (1-5) given by the reviewer.
- **Rating deviation** [2]: The deviation of the rating given by the reviewer from the business's overall average rating.
- **Singleton** [2]: Binary classification (1 or 0) indicating if the review was the only one written by its reviewer on the date of the review.
- **Density** [2]: The number of reviews received for the business on the day of the current review.
- **Mean Rating Deviation** [2]: The deviation of the average rating given to the considered business on the considered date from the overall average rating of the business.
- **Deviation from the Local Mean** [2]: The deviation of the rating given in the review from the average rating of the business on the considered date.

D. *Review-Centric Features*

- **Rating variance** [2]: The squared deviation of the rating given in the review from the overall average rating of the business.
- **Rating entropy** [2]: The entropy of rating distribution of a considered user's reviews.
- **Average deviation from entity's average** [2]: The average deviation of a considered user's rating from the average rating of the considered business.
- **Activity time** [2]: The number of days between the first and last review of the considered user.
- **Maximum rating per day** [2]: The maximum rating given by the user on a considered date.
- **Date entropy** [2]: The number of days between the current review and the next consecutive review of the considered user.
- **Date variance** [2]: The squared deviation of the date of the considered review from the average review date of the user.
- **Maximum number of reviews** [4]: The maximum number of reviews written by the considered user on a single day.
- **Percentage of positive reviews** [4]: The number of positive ratings (4-5 stars) with respect to the total number of ratings provided by the user.
- **Percentage of negative reviews**: The number of negative ratings (1-2 stars) with respect to the total number of ratings provided by the user. (This feature was not included in either paper, but we included it as an additional metric).
- **Review length** [4]: The average review length (in words) of all reviews written by the user.
- **Reviewer deviation** [4]: The average rating deviation across all user's reviews.

E. *Under Sampling*

We tried three different kinds of ways to do undersampling. We tried the NearMiss undersampling method with versions 1 and 3. Version 1 extracts majority-class examples with a minimum average distance from the three closest minority-class examples. Version 3 extracts majority class examples with a minimum distance to each minority class example. Both version 1 and version 3 are sampling a specific group of data, which is not random enough and might lead the model to learn a skew distribution of the data.

From the definition of version 3, we can see that the distance between the two classes in the feature space can be very small and lead to difficulty in training since we are extracting the majority of examples from the nearest point to a minority class. Version 1 improves on Version 3 by sampling majority examples from three reference points of minority examples. Figure 1 shows the 2-D projection of data using version 1 undersampling and principal component analysis (PCA). Figure 2 shows that the two classes are far too easy to distinguish, and this type of data distribution will render the model insufficiently robust when tested on a private test set.

After experimenting with these two methods, we decide to sample real reviews at random until the number of real reviews equals the number of fake reviews. We think that if the random sampling method is random enough, then th model we trained should be more robust.

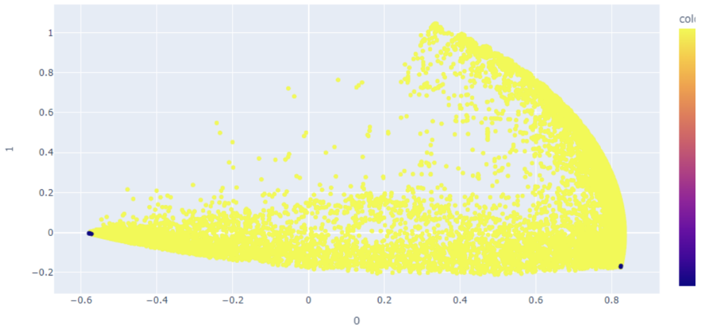

Fig. 2. PCA Analysis Graph.

F. Normalization

The features we generated may not be on the same scale, and different scales of different features will also affect the robustness of the model since the model might rely on some extremely large or extremely small value features for the prediction. To overcome this issue, we normalize the features between 0 and 1.

V. Dataset

In this project, we trained SVM, Random Forest, and Deep Learning models with various features. In this section, we will present the experiments and results. Our experiment shows that by using the generated features, Random Forest performs best on accuracy and SVM performs best on recall rate. Bot RF and SVM take in all the features we generated. Th deep learning model, which takes in all features, perform the worst on all three metrics. The deep learning mode trained on features selected by RF performs best on precisior The metrics we used to evaluate the models are accuracy precision, and recall. We would like to score as high a possible on these three metrics.

A. Random Forest

The way we fine-tuned the Random Forest model was by trying different combinations of parameters. Estimators in Random Forest forest mean the number of trees in this model. Max_Depth denotes the maximum depth of a tree. We tried the combination of n_estimators = [1, 2, 4, 8, 16, 32, 64, 100, 200] and max_depths = [1, 2, 3, 4, 8, 9, 10, 16, 32]. Then we observed the performance of different parameter combinations. As we can see from Figure 3, when estimators reach 64, the model has the best performance and no overfitting issue. Therefore, we selected 64 estimators and 10 depths as our model parameters.

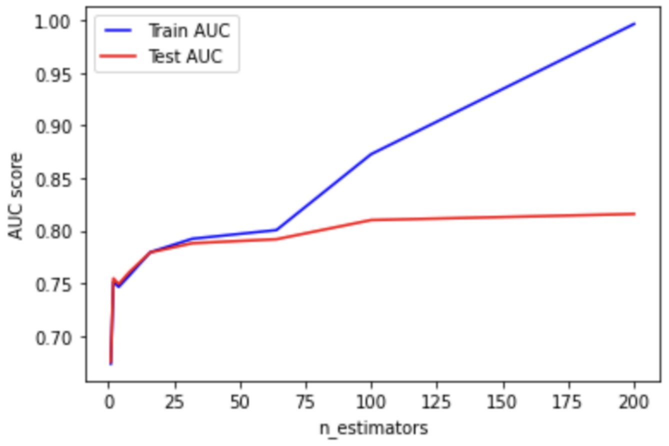

Fig. 3. Comparison of Different Combinations of Parameters for Random Forest.

Finally, we calculated the importance of our training features. After the feature engineering process, the number of review-centric and reviewer-centric features is equal. We chose the top ten most important features and used them to train deep learning models. According to Figure 4, our top ten important features are activity time, rating entropy, date variance, review length, rating variance, reviewer deviation, mean rating deviation, date entropy, the maximum number of reviews, and deviation from the local mean. Furthermore, three reviewer-centric features rank among the top five on this list: activity time, date variance, and rating entropy. Therefore, we can conclude that features that are reviewer-centric are more important than features that are review-centric

.

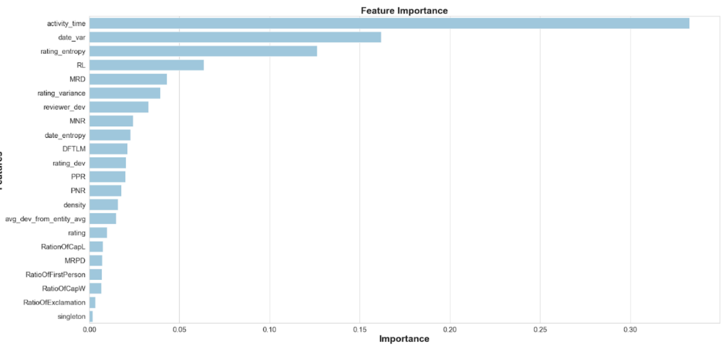

Fig. 4. Random Forest Feature Importance Graph.

B. Support Vector Machine

Using the full set of 22 features for each of the reviews in the training set, a support vector machine model was trained using a default linear kernel. Within this automated training process, the support vector machine algorithm computed an optimal hyperplane in the 22-dimensional space that classified each of the reviews as either 1 (indicating a fake review) or 0 (indicating an authentic review).

After the training process, the scikit-learn python library, from which we created the SVM model, allowed us to access the coefficient weights for each of the 22 considered features. These weights represent the vector coordinates for each of the features that are orthogonal to the computed hyperplane, with the direction of the vector indicating the predicted class. Thus, the greater the absolute size of the coefficient, the more effective the feature had on the model's classification decision. The top 10 features with the highest absolute coefficients can be seen in Figure 5. According to this figure, the top 10 features of the SVM model were: activity time, rating entropy, date entropy, singleton, review length, mean rating deviation, density, the maximum number of reviews, and date variance.

Similar to the Random Forest model, these top features are predominantly reviewer-centric. Only 3 review features (singleton, mean rating deviation, and density) are included, none of which rank within the top 4 most important features. This indicates that the behavior of the reviewer, from the perspective of the support vector machine, is more indicative of the review's authenticity than the content of the review itself.

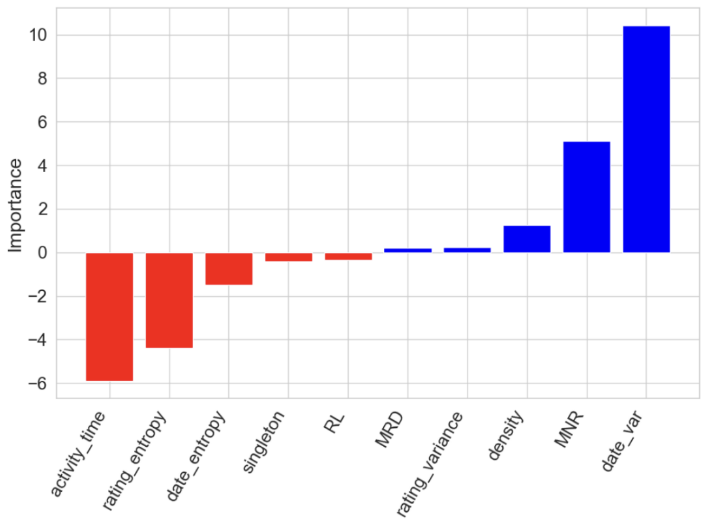

Fig. 5. Support Vector Machine Feature Importance.

C. Deep Learning Method

The experiments we tried are using different features and seeing how the model performs. Table 1 shows the evaluation metrics for various models using various features. We tried to feed all the features we generated to the deep learning model, and we also tried with features selected from the SVM model and the Random forest. Moreover, we selected the union and interaction of selected features from SVM and Random Forest and trained the deep learning model.

We also split the data set into training and validation, where the validation set contains 20% of the training data. The way we store the model is that we train the model with 100 epochs, and once the validation loss is the lowest of all previous iterations, we replace the previously stored model with the current model.

TABLE I
METRIC COMPARISONS BY MODEL

Model Name	Metrics		
	Accuracy	***Precision***	***Recall***
SVM	75.11%	69.68%	88.32%
RF	79.12%	74.72%	87.61%
DL (All)	70.65%	69.67%	72.86%
DL (SVM)	76.87%	72.82%	85.57%
DL (RF)	78.65%	74.97%	85.85%
DL (Union)	78.26%	74.35%	86.12%
DL (Intersect)	76.67%	72.26%	86.39%

VI. VISUALIZATION

We tried to find out the reason for the prediction of the model by plotting the top 2 features, which are "data variance" and "activity time," with red dots representing real reviews and blue dots representing fake reviews that are classified by the deep learning model.

From the ground truth graph, we can see that even when looking at the top two important variables, the two classes are not linearly separable, and the model needs to find a non-linear boundary on more dimensions (features). The model prediction of using all features is overkill for the real reviews, as we can see from Figure 6. The overkill here means that a lot of real reviews are considered fake. The prediction of models trained with selected features, although mitigating the overkill problem mentioned previously, shows that the model failed to capture some fake reviews.

VII. TEAM CONTRIBUTIONS

- **Austin Gilmore**: He was the point of contact for progress in the project since the beginning and was responsible for researching the project, analyzing the dataset, and coming up with new ideas on how to perform the project. He helped in the feature engineering of the review-centric features, implemented the support vector machine model, and visualized the support vector machine feature selection. He also worked on reorganizing the project code in the GitHub repository. He also helped in the documentation of the project proposal report and

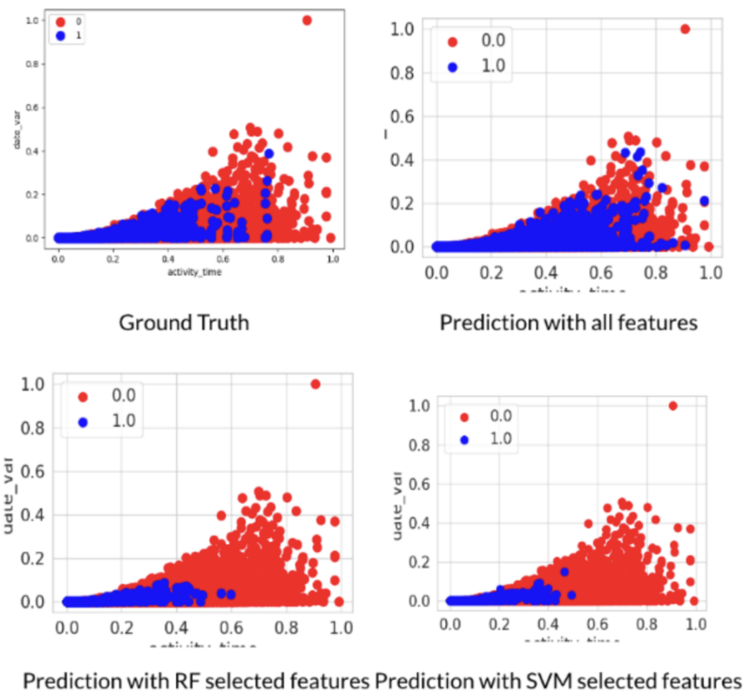

Fig. 6. Prediction with RF selected features vs. SVM selected features.

presentation and played a crucial role in documenting the final project report and presentation.

- **Hsiang-Yu Hsieh**: He was actively involved in going through the research papers to analyze and understand the project in its entirety; he was also involved in analyzing the datasets for the project. He managed to document and prepare a design of different steps that needed to be followed to achieve the project. He assisted with feature extraction and was focused on reviewing centric features; he also worked on a deep learning model and implemented it using PyTorch.He was also a major contributor to the documentation of the final report and final presentation.

- **Shiue-Fu Tzeng**: He did the research and referred to related research papers for a deeper understanding of the project and was able to demonstrate his understanding and findings with the team members. He was also focused on feature engineering, worked on review-centric features, and was able to extract the relevant features. He implemented the random forest model and trained it. He also helped everyone on the team with any help needed and was actively involved in the documentation work for the initial report, final report, and presentation.

- **Ruthwik Reddy Ratna**: He is responsible for research in the project on how to proceed at different stages and was involved in deciding and proceeding with the dataset. He extracted features from reviewer-centric features, with a focus on rating features and assistance with textual features. In rating features, he managed to extract the features from the total number of reviews, average deviation from entities' averages, rating entropy, and rating variance. He also worked on and contributed to the achievement of one of the textual features, the word-number average. He helped in preparing and documenting the final report and presentation.

- **Sai Likhith Yadav Seera**: He is responsible for the exploration of the dataset and performing feature extraction of reviewer-centric features, focusing mainly on temporal and textual features. In temporal features, he managed to retrieve features like maximum rating per day, date variance, and date entropy. In textual features, he managed to retrieve one of the features, which is the word-number average. He also contributed to documenting the proposal document, final presentation, and final report.

VIII. Conclusion

According to our experiments, Random Forest performs the best in general regarding the three metrics. Unlike the linear kernel SVM, where the model is a linear classifier, a random forest classifier depends on splitting the range of each attribute (feature) multiple times, which can add some nonlinearity to the model. Since we already know that this dataset is not easily classified by linear models (see Figure 4), it is reasonable to assume that the Random forest performs better than the linear SVM.

Our deep learning model contains two activation functions, which make the model flexible enough to draw a nonlinear boundary between two classes. We were expecting to get a better result from the deep learning model, but the deep learning model didn't perform better than the traditional machine learning models by training on the features we generated. One reason for not getting better performance on deep learning models is that we didn't find the best hyperparameters of the model, such as layer counts, neuron accounts, activation functions, and an optimizer. Another reason is that the generated features are not good enough for the model to distinguish between the two classes. To improve our model's performance, we think that we should generate more features from the reviews themselves. We were planning to generate sentiment features labeling whether the review is positive or negative by using a pre-trained DistilBERT model [6]. We also attempted to generate unigrams and bigrams as features after removing unnecessary words from the reviews, as well as calculate review similarity for each user. However, the above features are difficult to generate using just our laptop due to the enormous size of the dataset. Due to hardware limitations, we have to skip these features.

The intersection between the two sets of 10 features selected by the two machine learning models (SVM and Random Forest) contains seven features: activity time, rating entropy, date variance, review length, mean rating deviation, date entropy, and the maximum number of reviews. Due to the agreement of the two algorithms on these features, we can conclude that these 7 features derived from the original dataset are the most indicative measures of a review's authenticity (from our set of 22 features). Additionally, since only one (the mean rating deviation) is review-centric, we can also conclude that the user's behavior in comparison to other reviewers is more important than the review content itself for this detection process. This illustrates the difficulty of detecting fake reviews, for reviews must be considered within the context of a large body of reviews to make an effective decision about their authenticity.

References

[1] Gregorius Satia Budhi, Raymond Chiong1, Ilung Pranata1, Zhongyi Hu. Using Machine Learning to Predict the Sentiment of Online Reviews: A New Framework for Comparative Analysis.

[2] Julien Fontanarava, Gabriella Pasi, Marco Viviani. Feature Analysis for Fake Review Detection through Supervised Classification.

[3] Gary M. Weiss, Kate McCarthy, and Bibi Zabar. Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs?

[4] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, Natalie Glance. What Yelp Fake Review Filter Might Be Doing?

[5] Collective Opinion Spam Detection: Bridging Review Networks and Metadata. Shebuti Rayana, Leman Akoglu, ACM SIGKDD, Sydney, Australia, August 10-13, 2015.

[6] Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.