HIGH SCHOOL EDITION
Journal of Student Research
Volume 11 Issue 1 (2022)

# Fake Review Detection Using Machine Learning

Aaryan Rustagi[1], Vajraang Padisetti[1], and Suresh Subramaniam[#]

[1]Irvington High School, USA
[#]Advisor

## ABSTRACT

E-commerce and online shopping is a booming industry, especially during the COVID-19 pandemic. Online shopping allows customers to browse and purchase products from home using just a phone or laptop. Customers often never end up seeing their product in real life before the purchase, and are instead dependent on photos and descriptions uploaded by the seller. Thus comes the need for customer reviews: an evaluation of a product by other buyers which can influence other shoppers' decisions to make the purchase. However, reviews can be tainted to provide a fake or unrealistic depiction of a product. Sellers can pay people or robots to leave fake reviews under competitors or their own stores to increase/decrease sales turnout. Such reviews can be harmful to the buyer or other sellers, often ending up with an unhappy customer. Using supervised datasets consisting of real and fake reviews, we can train a variety of machine learning and deep learning models to recognize attributes differentiating between the two types of reviews. Big e-commerce platforms such as Amazon, Yelp, and Tripadvisor are all common targets of fake reviews, and the implementation of fake review detection could create a more assuring shopping experience for customers. In this paper, we analyze and break down customer review data and attempt to build models that form conclusions from it.

## Introduction

E-commerce is a growing market, and with that comes millions of reviews flowing in every day for different products on different sites. With all of these customers and reviews, there isn't much regulation going into validating whether the review is true. The purpose of reviews for any product is to help new customers gauge if it's a good buy. Of course, if there are more good reviews than bad, a customer is more likely to purchase the product than if it was the opposite. According to Oberlo, 89%, or almost 9 out of 10 customers, read product reviews before making a purchase[1]. On top of that, 3 out of 4 people trust online reviews as much as personal recommendations. This shows that there is a lot of trust put into reviews, and it can be a make or break factor. However, reviews aren't as easy to rely on as that. Reviews can easily be faked, often done when companies pay people to write positive reviews even if they didn't purchase or like the product. The opposite is true too, in that companies can pay to have fake reviews written on their competitor's products. This is especially detrimental to smaller businesses that can easily be destroyed by a couple of fake reviews being pushed by competitors. The problem in these fake reviews is that customers are no longer able to rely on reviews, and are more likely to end up making a bad purchase or missing out on a good one.

There have been multiple instances of fake reviews being written for mainstream brands, often at the customer's expense. One example of a brand is Sunday Riley, a skin care company. According to an article by CNN, Sunday Riley employees had been instructed by their CEO to write fake reviews on Sephora regarding their products[2]. They were to create fake accounts and regularly write fake reviews for their products and dislike negative comments. With enough dislikes, negative reviews would end up being removed and instead replaced with loads of misleading positive reviews. The CEO went as far as recommending customers to use VPNs in order to prevent the reviews being traced back to them. This is just one example of how a company having more positive online reviews than negative isn't necessarily reliable, and instead only hurts customers.

Due to the major problem of fake reviews, many e-commerce companies have already taken action. Yelp, a platform for reading business reviews, has a method to filter through reviews, removing any they think may be fake. This is done by using data regarding the reviewer and attributes of the actual review. While Amazon doesn't detect fake reviews, it confirms that the reviewer bought the product before leaving the reviews. While this does add a layer of validity, it isn't foolproof as the product is either free or returned after the fake review is written. Another website, Fakespot, gives a grade for any product correlating to how real the reviews seem. This is helpful, but it doesn't specify if specific reviews are real or fake which is often what customers want to know.

It is clearly difficult for humans to differentiate between real and fake reviews, mainly because they seem to be very similar. This is where machine learning comes into play. By converting reviews to a series of numbers, machine learning models can analyze the numbers for trends that humans are unable to detect. With supervised learning, models are able to detect patterns commonly found in fake reviews that aren't present in real reviews. Classification models like logistic regression, random forest, and naive Bayes all classify reviews as likely real or likely fake with the help of training data it's given. While these models aren't perfect, they are certainly better than humans and will only get better with the help of more data and better algorithms. The purpose of this paper is to test and fine-tune these models on review data, and ultimately reach a point where they can outperform humans, making them useful in real-life application.

Finding a good dataset for review detection was a challenge, as supervised learning requires pre-labeled data. This means we need a dataset consisting of data that is for certain real and fake. This means that scraping reviews off e-commerce sites like Yelp and Amazon wouldn't work as we don't know if the reviews are real or fake. The dataset we ended up using was one developed by Ott et. al[3]. The dataset consists of real and fake reviews, each with positive and negative sentiment. Of them, we are using 320 real positive, 320 real negative, 320 fake positive, and 320 fake negative reviews. The reviews are regarding 20 different Chicago hotels. The real reviews were scraped from sites like Trip Advisor, while the fake reviews were made using Amazon Mechanical Turks, a platform where one can outsource any work. In this case, they were to write fake reviews about the 20 different Chicago hotels. By hiring people to specifically write fake reviews, we get a realistic look at how fake reviews are made. The benefits of this dataset is that it is fully labeled and has reviews of both, positive and negative sentiment. It is still lacking in the fact it is missing data regarding the information about the reviewer, and doesn't have many reviews. Using this dataset, we train our classification models to determine review authenticity.

## Methods

### TF-IDF

Natural language processing involves converting text to something a machine learning model can understand, numbers. This can be done using a vectorizer, which creates a vector or matrix that can be input into the model. The two main types of vectorizers are a count vectorizer and the TF-IDF vectorizer. The count vectorizer works by counting the number of occurrences of each word in a document, and ends up with a matrix representing word frequency. The problem with a count vectorizer is that it gives too much weight to common words such as 'a', 'and', and 'of'. Another downside is that it doesn't take into account the rest of the words in a document or review. For example, if a review is very short and uses a specific word, that word might have more of a meaning than if it was in a review with many more words. This is where the TF-IDF (Term Frequency Inverse Document Frequency) vectorizer comes into play. By comparing the number of times a word appears in a review with the number of reviews containing a word, TF-IDF can measure the originality and importance of a word.

### N-grams

Natural language processing often involves the use of n-grams. This is a method of splitting the text in documents, or in this case each review, into tiny parts. The obvious way to do this would be by splitting the review into individual words. However, another approach could be to split it into groups of words instead of single words. This might be useful when there are specific phrases that have more value than individual words. N-grams are groups of n words, and the main examples are unigrams, bigrams, and trigrams. We tried using all three of these approaches when vectorizing our review data, trying to find what performed the best.

## Ensemble Techniques

Ensemble techniques are methods that essentially combine multiple models to get an output. One of these techniques is bagging, and the main example of this is random forest. Random forest is a model that splits the training data into multiple mini-datasets, each with only a portion of the rows and features of the original training data. This process is called bootstrapping. The next step is training multiple decision trees with the bootstrapped datasets. Finally all of these decision trees are applied to the test data, with the majority vote being the final result. These work far better than simple decision trees, because decision trees tend to be very specific to training data and are overfit. This means it could work well for training data, but underperform on the actual test data. However by taking the majority vote of multiple decision trees on multiple partitions of the data, we can get a more accurate model with lower variance. The only parameter that needed tuning is 'n_estimators' which determines how many decision trees the random forest uses. Additionally, another machine learning algorithm called Adaboost was used as well. Adaboost forms multiple decision trees with only only nodes and two branches called stumps. These stumps then decide the importance of each feature and give them each a unique value which can also be negative. While random forest is a democracy where each output has a value of one, Adaboost assigns weights to the outputs of the trees and then makes a decision on the final output.

## Support Vector Machines

Support vector machines are a high dimension model that plots the data points and creates a hyperplane to separate the points into classes in classification problems. The goal is to find the best hyperplane that can classify the most points correctly, and that is done by maximizing the margin between the hyperplane and support vectors. The advantage of support vector machines is it works better for smaller datasets with less noise, and that is helpful in our case. The parameter to tune for this model is 'C', which determines how many errors should be allowed to prevent overfitting of the model.

## Logistic Regression

Logistic regression is a binary classification technique that aims to convert a linear model representing the data points into a sigmoid curve that goes from range 0 to 1. This is great because now every point can have an output on the function which is closer to 0 meaning fake review or closer to 1 meaning a real review.

## Multinomial Naïve Bayes

Multinomial naive bayes is a binary classification algorithm that uses bayes theorem to calculate the probability of an event to be true. It works especially well with text classification. Using the term frequency values of each n-gram, it estimates the likelihood of a review being true or false.

## Pipeline

Our code currently has multiple steps. First it needs to vectorize the data using TF-IDF, and then apply the models. Everytime we change the data, or want to try our model on new data, we have to run each step individually in a slow and unorganized manner. This is where pipelines come in. Pipelines help combine the multiple parts of code into one function which can now be applied to new data or changed data. But more importantly, it is essential for parameter tuning of our models and vectorizer.

Grid search is a method of parameter tuning for our multiple models. By inputting our pipeline that we have created, as well as the parameters we want to tune for our vectorizer and models, we can figure out which parameters work best to ensure the highest accuracy. The parameters we are testing out are the different n-grams, and specific parameters of our models mentioned above.

### Deep Learning

We also approached this problem using a completely different method called deep learning. Deep learning is a branch of machine learning that uses a complex structure modeled around the principle of a human brain called neural networks. The advantage of neural networks is that it creates independent features, whereas the features must be provided before hand in machine learning. Neural networks do three main things: take in data as input, train themselves to understand the data, and output useful predictions. A neural network consists of 3 different layers. The first layers are called the input layer and hidden layers. This is where the weighted sum of the inputs is calculated, the bias is added, the result is fed to an activation function, and specific neurons are activated. Neural networks also learn from its predicted outputs. After forwards propagating and reaching an output, the function back propagates to verify it's result. Since the biases are set randomly, the output goes through a loss function which quantifies the deviation from the expected output. An optimizer then ties together the loss function and model parameters by updating the network so that the values better fit the model. An epoch is one passing of the data through the neural network. Too many epochs can lead to overfitting in the data, so a stop function is needed to stop the epochs just as the accuracy starts to drop. Our neural network uses NLP(Natural Language Processing) from Tensorflow. In order to prevent overfitting, each review was limited to the 100 most recurrent words in each review, and the 1000 most recurrent words were allowed. In order to make sure all lists were of equal and comparable length, we padded our data by adding 0's at the end.

## Results

The parameter tested for our random forest model was 'n_estimators' which represents the number of decision tree base models built within the forest. We tested values of 100, 500, 1000, and 1500. The random forest model and vectorizer with the default parameters, 100 for n_estimators and unigrams, resulted in a score of 71.09%. After using gridsearch for parameter tuning, the best parameters found were 1500 for n_estimators and unigrams for the vectorizer. The score from the tuned model came out to be 83.59%, an around 12.5% better performance than the default parameters. Figure 1 shows the review accuracy distribution for real and fake reviews.

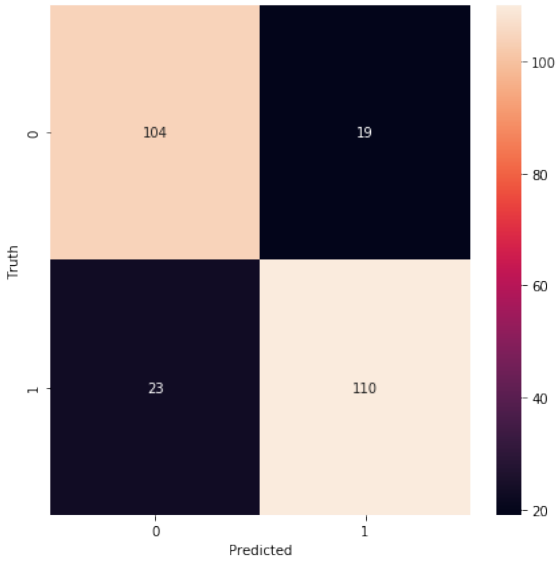


Figure 1. Of the 256 reviews in the training data, 104 fake reviews and 110 real reviews were correctly classified with the random forest model. 23 real reviews and 19 fake reviews were incorrectly classified.

The parameter tested for our support vector machines model was 'C' which represents how many misclassified points are acceptable in order for a more generalized model. We tested values of 50, 100, 250, 500, and 1000. The support vector machines model and vectorizer with the default parameters, 1.0 for n_estimators and unigrams, resulted in a score of 48.44%. After using gridsearch for parameter tuning, the best parameters found were 250 for C and unigrams for the vectorizer. The score from the tuned model came out to be 86.33%, an around 37.89% better performance than the default parameters. Figure 2 shows the review accuracy distribution for real and fake reviews.

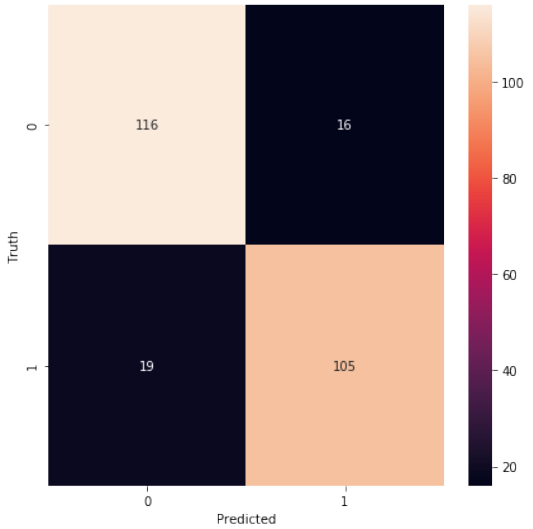


Figure 2. Of the 256 reviews in the training data, 116 fake reviews and 105 real reviews were correctly classified with the support vector machines model. 19 real reviews and 16 fake reviews were incorrectly classified.

The parameter tested for our logistic regression model was 'C' which represents an inverse of regularization strength. We tested values of 0.001, 0.01, 0.1, 1, 10, and 100. The logistic regression model and vectorizer with the default parameters, 1.0 for C and unigrams, resulted in a score of 85.16%. After using gridsearch for parameter tuning, the best parameters found were 10 for C and unigrams for the vectorizer. The score from the tuned model came out to be

85.55%, an around 0.39% better performance than the default parameters. Figure 3 shows the review accuracy distribution for real and fake reviews.

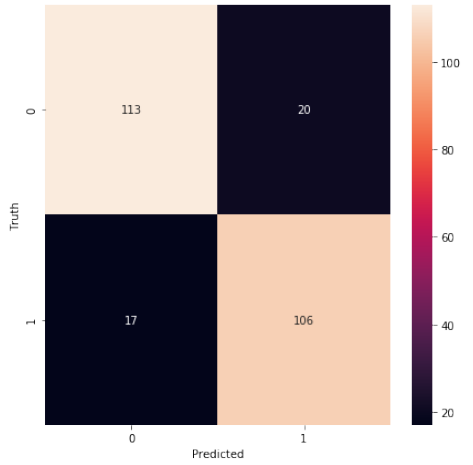


Figure 3. Of the 256 reviews in the training data, 113 fake reviews and 105 real reviews were correctly classified with the logistic regression model. 17 real reviews and 20 fake reviews were incorrectly classified.

No parameters were tuned for the multinomial naive bayes model. The multinomial naive bayes model and vectorizer with the default parameters, unigrams, resulted in a score of 84.38%. Figure 4 shows the review accuracy distribution for real and fake reviews.

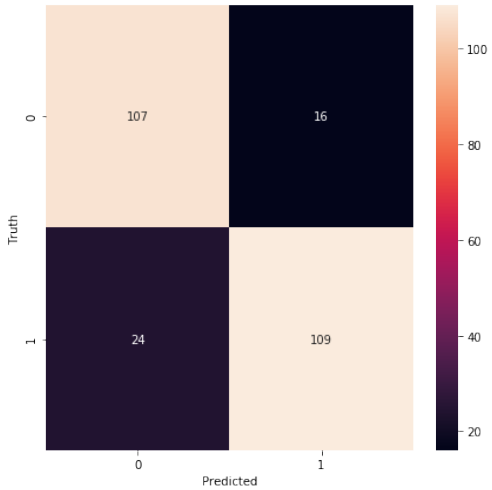


Figure 4. Of the 256 reviews in the training data, 107 fake reviews and 109 real reviews were correctly classified with the multinomial naive bayes model. 24 real reviews and 16 fake reviews were incorrectly classified.

The parameters of the Adaboost model were set to a learning rate of 1, and 100 n-estimators. The Adaboost model had an accuracy of 83.6%. Figure 5 shows the review accuracy distribution for real and fake reviews.

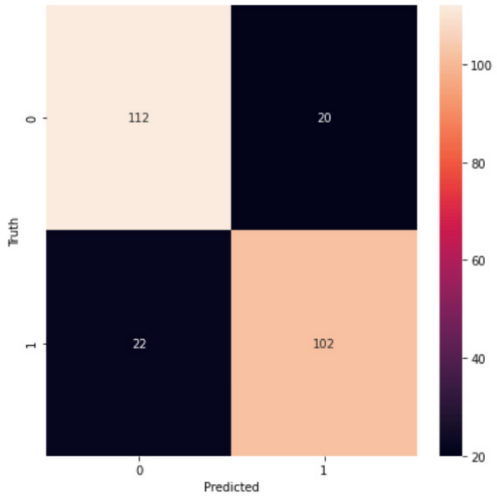


Figure 5. Of the 256 reviews in the training data, 112 fake reviews and 102 real reviews were correctly classified with the Adaboost model. 22 real reviews and 20 fake reviews were incorrectly classified.

The deep learning method produced an accuracy of around 50% at first, but through training itself through many epochs, this model reached an accuracy of 87%. Epochs determine how many times the data passes through a neural network, and comes back. Since the bias adjusts after every epoch after the loss is quantified, the model learns from itself and the validation accuracy typically increases.

## Discussion

Upon tuning the parameters, deep learning had the highest accuracy with 87%. Deep learning might have performed the best due to the fact that features are independently created in contrast to machine learning where the features need to be determined beforehand, and deep learning ability to learn from it outputs unlike machine learning that doesn't take it's output into account. Deep learning's ability to independently create features also has drawbacks compared to machine learning in that deep learning needs a lot of training data. Another problem was knowing how many epochs to use. After trying several optimizations, a conclusion was made to implement early stopping which stops the epochs as the validation accuracy significantly drops.

One thing we noticed about all our models is that they worked best with unigrams data instead of bigram or trigram. This could be because in reviews there are individual words that are more valuable to the output than phrases or groups of words. Oftentimes the bigrams performed a little worse than unigrams, however trigrams always performed the worst.

The biggest limitation to our research was the dataset used, as it was only about hotel reviews in Chicago. For that reason, we are unable to test our models on typical product reviews that one would find on amazon. Another problem with the dataset was the number of reviews it contained. For a more effective model, having more reviews would help train the model more as well as give us a better idea of how it performs on training data. Our research focused on the review text data, but using reviewer information such as how many reviews written and the date the profile was made can help us make better conclusions about the validity of reviews. This data is missing in this dataset, but using a scraper can be obtained from sites like yelp.

When continuing the project in the future, we plan on gathering more review data on a variety of different topics and sectors. We plan on incorporating more data such as reviewer information for better performance. As for the model training, we hope to get to try out more advanced models that are specific to text classification. In particular, tuning and improving the deep learning model would be the priority as it appears to have performed the best out of all methods tested.

## Conclusion

Detecting fake reviews is an important task that can be useful for online shoppers as well as businesses. In this paper we used a supervised dataset containing real and fake reviews regarding Chicago hotels to train machine learning and deep learning models. The first steps consisted of transforming the review data into matrices using a TFIDF vectorizer. The models tested included random forest, support vector machines, logistic regression, and multinomial naive bayes. All these methods are especially useful for text classification. Our initial test produced results in the 40-80 percent range. However, upon tuning the models, accuracy scores increased to the 80-90 percent range. Additionally, our deep learning model was able to achieve the highest score of 87% after tuning. Other observations were that training models on unigrams instead of bigrams or trigrams worked the best. Future plans include utilizing new datasets consisting of more information such as reviewer data that could be scraped from review platforms such as yelp. New models will also be tested, using the same process of parameter tuning, and we hope to see better results than what we have obtained so far.

## Acknowledgements

We would like to conclude by thanking ASDRP for hosting the research and our advisor, Mr. Subramaniam, for guiding us.

## References

1. Lin, Ying. “10 Online Review Statistics You Need to Know in 2021.” Oberlo, Oberlo, 17 Dec. 2020, www.oberlo.com/blog/online-review-statistics.

2. Elassar, Alaa. “Skin Care Brand Sunday Riley Wrote Fake Sephora Reviews for Almost Two Years, FTC Says.” CNN, Cable News Network, 23 Oct. 2019, www.cnn.com/2019/10/22/us/sunday-riley-fake-reviews-trnd/index.html.

3. “Deceptive Opinion Spam Corpus Myle Ott.” Myle Ott, myleott.com/op-spam.html.