AuthentiFeel – Review Sentiment Analysis and Authenticity Detection System

Atyab Hakeem, Atreyo Das
Northeastern University
hakeem.at@northeastern.edu, das.at@northeastern.edu
https://github.com/hakeematyab/AuthentiFeel.git

Abstract

Sentiment analysis of product reviews is invaluable to both the customers and the businesses. Furthermore, determining the authenticity of the reviews post-GPT era is even more crucial. To solve this, we built AuthentiFeel, a sentiment analysis and authenticity detection system that not only determines the sentiment of a review but also predicts how likely the review is computer generated. AuthentiFeel employs well-established traditional machine learning algorithms like Logistic Regression, Support Vector Machine, and Naive Bayes with ensemble techniques to boost their individual performance and guard against overfitting. By opting for simpler algorithms as opposed to complex architectures, our models are not only fast at training and inference but are also less resource intensive. AuthentiFeel achieved an accuracy of about 87% for the sentiment analysis task and a precision of approximately 95% for the authenticity detection task. These metrics are comparable to our predecessors who utilized more sophisticated techniques. To further demonstrate the efficacy of our system, we scraped a few hundred reviews and evaluated the results.

1. Introduction

Analysis of product reviews is invaluable for the customers to be fully informed before making a purchase. Moreover, it is also vital for businesses to understand the performance of their products in the market. With the generative AI boom, genuine customer reviews are almost indistinguishable from computer generated. Determining the authenticity of these reviews thus becomes crucial to making sound and well-informed decisions.

To solve this, we propose a sentiment analysis system that determines whether a review is positive or negative and an authenticity detection to see how likely the review is computer generated. The input to our system would be the product review (text) and the output is the sentiment of the review (positive or negative) and its authenticity (computer generated or not.)

With sufficient reviews, these metrics would allow us to create a reasonable distribution of the general sentiment around the product and the degree to which we can trust this distribution to be genuine. To further demonstrate how the system would work in practice, and to measure the effectiveness of our system, we scraped a few hundred reviews and evaluated our findings.

2. Background

Since both above-described systems are classification problems, we experimented with a diverse range of classification algorithms. Naive Bayes, Logistic Regression, Decision Tree, Support Vector Machines, Ensemble Techniques, and Multilayer Perceptron were a few of the models we tested among many. In addition, for the classification tasks we also utilized algorithms like Decision Trees and Logistic Regression for feature selection and data interpretation.

The reason we opted for simpler traditional machine learning algorithms is threefold. Firstly, by opting for simpler algorithms as opposed to complex architectures, our models are not only fast at training and inference but are also less resource intensive. Secondly, simplicity of the model reduces the likelihood of overfitting, making AuthentiFeel reliable for real-world applications where the quality and genuineness of reviews are paramount. Finally, unlike many modern deep learning approaches, the models used in AuthentiFeel are highly interpretable, meaning you can easily understand and trace how and which input features affect predictions. This transparency is crucial for trust and reliability in review analysis.

3. Related Work

The inspiration for the project and its data comes from the paper published by Blitzer et al. (2007)[1] and Salminen et al (2022)[2]. Apart from collecting or generating the data, they have both used various algorithms to try to classify the reviews.

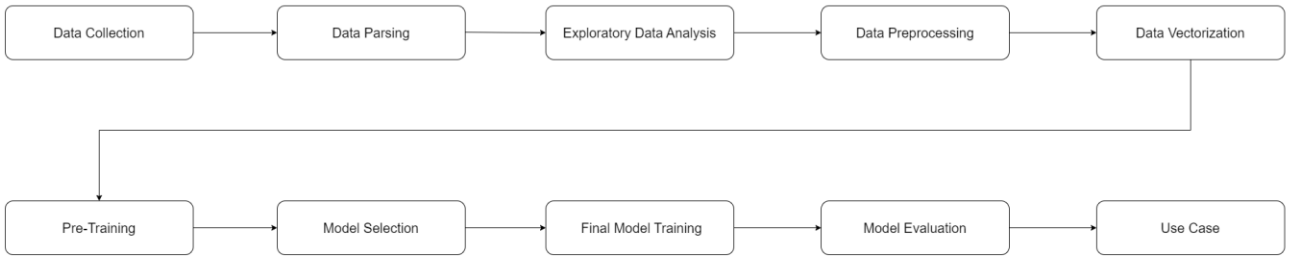

Figure 1: Project Workflow

Blitzer et al. (2007)[1] collects reviews from four different kinds of products from Amazon: Books, DVD, Kitchen Appliances and Electronics. They further classified those with ratings greater than 3 as positive and less than 3 as negative. Each of the categories had a thousand reviews for each sentiment, giving 8000 reviews for the entire dataset. While some of the categories also contained 'unlabeled' data with random ratings, they were not used for model building or evaluation as they were not consistently available for all the categories and might create bias in the model.

In the paper, they used the Structural Correspondence Learning (SCL) algorithm, where the model chooses appropriate 'pivot features' and then it models correlations between the pivot features and all the other features. They used it to find the sentiment of the reviews in each of the categories and then applied that model to reviews in other categories to see how it performs. The accuracy of the model depended on the category the model was trained in and the accuracy ranged from 80.4% to 87.7%. On the other hand, our model trained the data without any segregation of the categories. This lead to a more generic model that can classify the review, regardless of the category of product.

Salminen et al. (2022)[2] used two different language models to generate fake reviews. They decided to use OpenAI's GPT-2 model since it outperformed ULMFit (Universal Language Model Fine-tuning) in all parameters like training time, perplexity and validation loss. Using the language model, they created 20,000 fake reviews and then took an equal number of reviews from the Amazon dataset to create a dataset with 40,000 reviews.

Keeping SVM (with Naïve Bayes features) and the OpenAI fake model detection as baseline, they fine-tuned a RoBERTa model, which was based on Google's BERT (Bidirectional Encoder Representation from Transformers) model and named it 'fakeRoBERTa'.

This model gave a precision of about 97% outperforming both baseline models.

4. Project Description

4.1 Project Workflow

The workflow of our project is described in *Figure 1*. We started with the collection of product review data for both the tasks, sentiment analysis and authenticity detection. This was followed by the parsing of the collected data to the required format most suitable for downstream tasks. We then explored the data and performed basic text preprocessing. The processed data was then vectorized. Once we had our data ready, we employed a wide range of classification algorithms from Logistic Regression to Multi-Layer Perceptron and gauged their metrics. The best performing models were then tuned for optimized hyperparameters. We then evaluated the final models created for each of the tasks based on their respective metrics of evaluation. Finally, we employed the models we trained for a use case.

4.2 Data Collection & Parsing

We used two datasets primarily for each of our tasks. For sentiment analysis we employed [1] and for authenticity detection we utilized [2]. For sentiment analysis, the data was in XML format. As such, it had to be parsed to make it more suitable for the downstream tasks. The data consists of about 8,000 product reviews and following are its attributes:

- Unique ID: Identifier for each review (Integer).
- ASIN: Amazon Standard Identification Number for the product (String).
- Product Name: Name of the product being reviewed (String).

- Product Type: Category or type of product (String).
- Helpful Votes: Number of users who found the review helpful (Integer).
- Rating: Star rating given to the product by the reviewer, typically from 1 to 5 (Integer).
- Review Title: Title of the review provided by the reviewer (String).
- Review Date: Date when the review was posted (Date).
- Reviewer: Name or ID of the reviewer (String).
- Reviewer Location: Geographical location of the reviewer, if available (String).
- Review Text: Full text of the review (String).
- Sentiment: Categorical sentiment classification (1 for 'Positive', 0 for 'Negative') (Integer).

As our objective was to analyze the sentiment of the reviews, we primarily utilized only the 'Review Text' attribute for the downstream tasks. Other attributes like 'Helpful Votes' could potentially be employed to add to the genuineness of the review.

For authenticity detection, the data was available in a comma separated values format and so parsing wasn't required. The data has about 40,000 instances with the following attributes:

- Product Category: Category or type of product (String).
- Review: Full text of the review (String).
- Label: Categorical authenticity classification ('CG' for computer generated, 'OR' for human generated) (String).

4.3 Exploratory Data Analysis

Both the datasets had an approximately equal distribution of classes as seen in *Figure 2 and Figure 3*. It was also found that the reviews mostly ranged from 0 to 1000 characters. Most of the reviews had around 100 characters. Very few reviews had more than 1500 characters. This makes sense as very few people would take the time and effort to write a review in detail. We also looked at the most common words in each of the datasets as seen in *Figure 4 and Figure 5*.

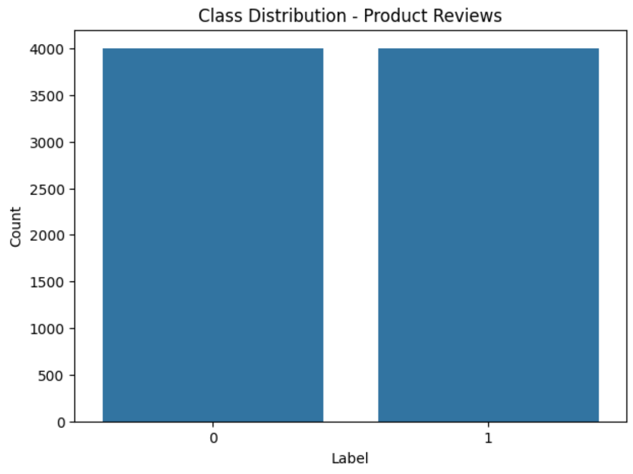

Figure 2: Distribution of classes in the Sentiment Analysis Dataset.

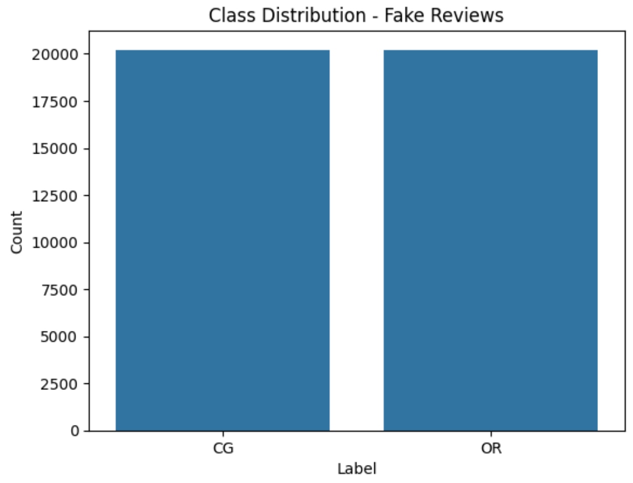

Figure 3: Distribution of classes in the Authenticity Detection Dataset.

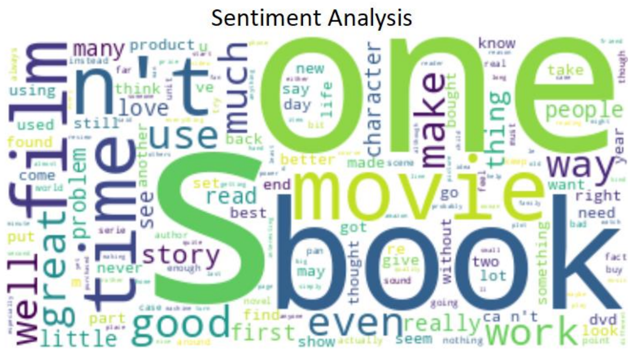

Figure 4: Most Common Words - Sentiment Analysis Dataset

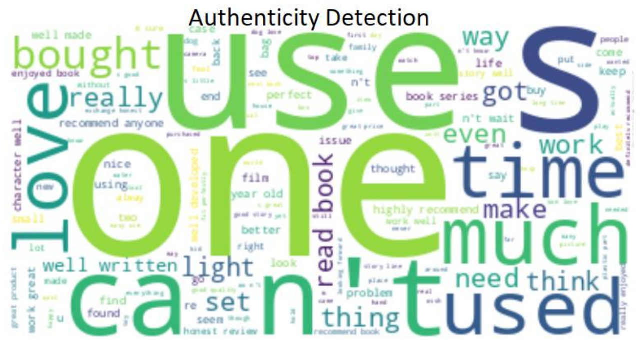

Figure 5: Most Common Words - Authenticity Detection Dataset

4.4 Preprocessing & Vectorization

Basic text preprocessing was performed on the reviews. This included tokenization, lowercasing, and lemmatization. Stop-words were retained to prevent the removal of sentiment indicative words. For vectorization, 'Term Frequency, Inverse Document Frequency' (TF-IDF) provided the greatest boost in performance. Hence, this vectorization was chosen.

5. Empirical Results

5.1 Preprocessing and Vectorization

Prior to training the algorithms, the data was first preprocessed by normalization, tokenization, and lemmatization. While removing stop words is also an important part of preprocessing, we noticed that it caused a decrease in performance, possibly because it removes sentiment indicative words.

Then the data was vectorized using TF-IDF as opposed to Count Vectorizer for its inherent advantages. It was also extremely useful to tune the hyperparameters of the chosen vectorizer to maximize the performance. A grid search was run to test out the various hyperparameters such as the maximum and minimum document frequency threshold, the maximum number of features to be considered, whether sublinear term frequency scaling should be applied, etc. Finally, the following hyperparameters were selected:

- Maximum Document Frequency: 0.95
- Minimum Document Frequency: 2
- N-Gram Range: (1,3)
- Sublinear Term Frequency: True

Their change in performance can be clearly seen in *Figure 4* where the tuned TF-IDF provides much better results than when it is on its default hyperparameters.

5.2 Sentiment Analysis

After preprocessing and vectorization, the dataset was classified using several algorithms. Most of them were first run on their default hyperparameters so that their base performance can be noted down. The models that had higher accuracy and lower training time were favored for further hyperparameter tuning. It can be noticed from *Table 1* that Logistic Regression, Multinomial Naïve Bayes and Linear SVC had the best performance and were the fastest.

A random search was conducted on Logistic Regression to try out different hyperparameters like the value of the strength of regularization, the solver used, whether l1 or l2 penalty should be used and the maximum number of iterations to be run. An accuracy of 86.2% was obtained when the following hyperparameters were changed:

- Solver: Newton's method with a Conjugate Gradient
- Max Iterations: 200
- Regularization: 100

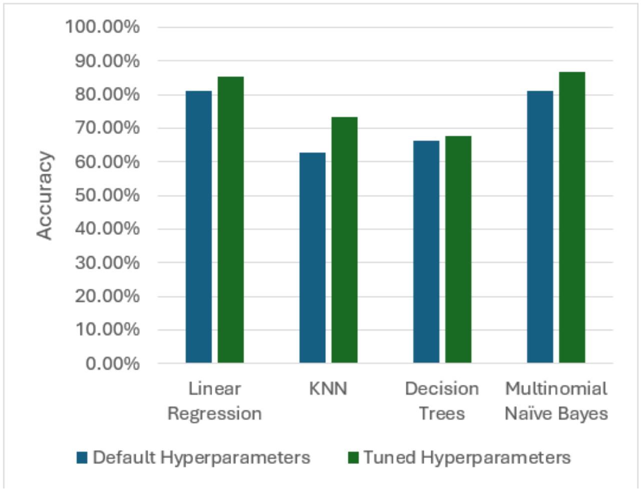

Figure 6: Bar chart denoting the change in test set accuracy on different classifiers on the Sentiment Analysis Dataset when the vectorization hyperparameters are unchanged or tuned to the dataset.

Similarly, a grid search was performed on Linear SVC to test out similar hyperparameters and received an accuracy of 86.45% when C was set as 20.

Furthermore, we decided to create an ensemble of the best models to improve accuracy and reduce overfitting. We achieved our best accuracy of 87.05% when we used an ensemble of the following classifiers:

- Logistic Regression
- Multinomial Naïve Bayes
- Linear Supper Vector Classifier (Regularization: 20)

Algorithm	Test Set Accuracy	Training Time
Logistic Regression	85.35%	4.204s
KNN	73.35%	0.230s
Decision Tree	67.60%	18.751s
Multinomial Naïve Bayes	86.80%	0.184s
AdaBoost (Number of Estimators=200)	78.50%	197.486s
Gradient Boosting	76.50%	136.948s
Random Forest (Number of Estimators=200)	81.35%	82.536s
Extra Trees (Number of Estimators=200)	83.45%	158.146s
XGBoost	80.45%	83.05s
Linear SVC	86.25%	0.916s
SVC (Kernal)	85.40%	66.748s
Multi-Layer Perceptron	86.40%	1095.863s

Table 1 1: The test set accuracy and training time of different classifier algorithms when run on the Sentiment Analysis dataset.

5.3 Authenticity Detection

Similar procedure was followed for authenticity detection task. Logistic Regression, Multinomial Naïve Bayes and Linear SVC once again performed well in a comparatively low amount of time.

After tuning the hyperparameters of the Logistic Regression algorithm, the accuracy increased from 93.11% to 94.85% with the following hyperparameters:

- Solver: Stochastic Average Gradient Augmented
- Maximum Iterations: 500
- Regularization: 100

Here, we saw an improvement in MultinomialNB when we changed the value of alpha. Setting the value as 0.01, gave an accuracy of about 93.31%. Linear SVC proved to be the best model after hyperparameter tuning, giving an accuracy of about 95.18% and a precision of 95.59% after setting the values of the hyperparameters as:

- Max Iterations: 100
- Regularization: 100

We also tried to use an ensemble of all three tuned algorithms. While it slightly increased the precision to 95.78%, it also reduced the accuracy to 95.07%. Even though the difference isn't large enough to select either as the better model, we still choose Linear Support Vector Classifier as our final model because of its significantly lower training and inference time.

Algorithm	Test Set Accuracy	Training Time
Logistic Regression	93.11%	13.525s
KNN	69.36%	0.333s
Decision Tree	78.01%	129.476s
Multinomial Naïve Bayes	90.34%	0.265s
AdaBoost (Number of Estimators=200)	87.55%	1042.317s
Gradient Boosting	84.18%	704.782s
Random Forest (Number of Estimators=200)	90.79%	798.405s
Extra Trees (Number of Estimators=200)	91.93%	1333.435s
Linear SVC	94.93%	1.028s

Table 2: The test set accuracy and training time of different classifier algorithms when run on the Authenticity Detection dataset.

5.4 Word Importance

For model interpretability, we used the built models to obtain the word importance. These terms correspond to the words most indicative of the sentiment of the review. The word clouds in *Figure 7* and *Figure 8* represent this visually.

For sentiment analysis, from *Figure 7* we can see that the model is highly interpretable. Words like 'excellent', 'love' and 'great' have high importance implying that these words are largely indicative of the sentiment of the review. However, it is not as apparent in the case of Authenticity Detection as we can see in *Figure 8*. This is most likely because humans are very bad at detecting fake reviews as found in a study by Sun et al. (2013)[3]

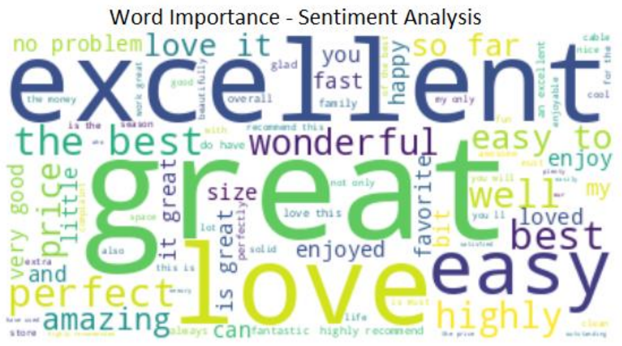

Figure 7: Word Cloud of the words which have a higher impact on determining the sentiment of the review.

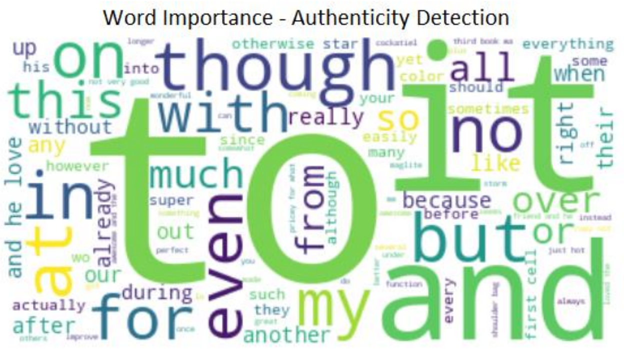

Figure 8: Word Cloud of the words which have a higher impact on determining whether the review is artificially generated.

5.5 Use Case

After building and evaluating the built models, we used them to make inferences about the reviews on Samsung Galaxy S22 Ultra. We scraped about 180 reviews and ran it through our sentiment analysis model. We also manually labeled these reviews for evaluation. The model accurately classified the reiews 86.90% of times. We also saw both the predicted and actual sentiment was leaning towards the positive side. We also ran the Authenticity Detection model which predicted that more than 97% of the reviews were human generated.

This tells us that the reviews tend to mostly have a positive opinion about the phone and can be largely trusted to be written by actual people. Using this information, the customers can choose to further investigate the product before deciding to make a purchase. The seller on the other hand can make decisions about stocking the product or make improvements to make the sentiment distribution more positive.

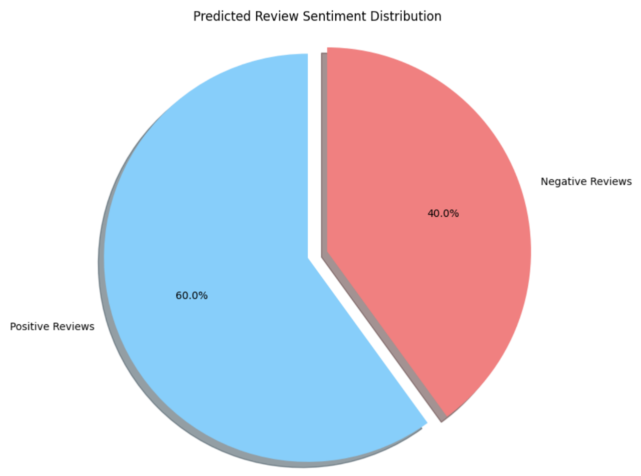

Figure 9: Pie chart signifying the proportion of positive reviews compared to the number of negative reviews as predicted by the Sentiment Analysis model.

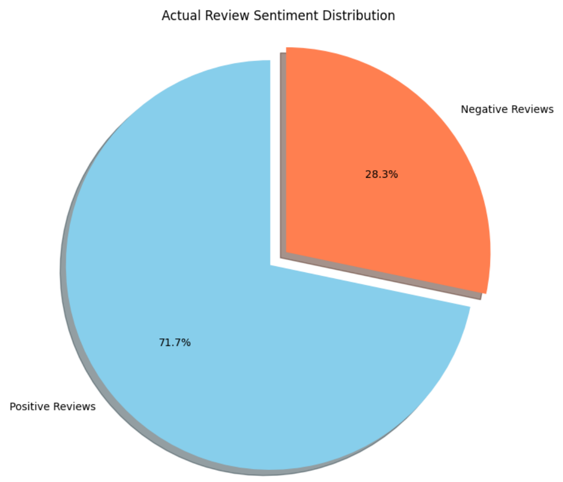

Figure 10: Pie chart signifying the proportion of actual positive reviews compared to the number of actual negative reviews confirmed by manual labeling.

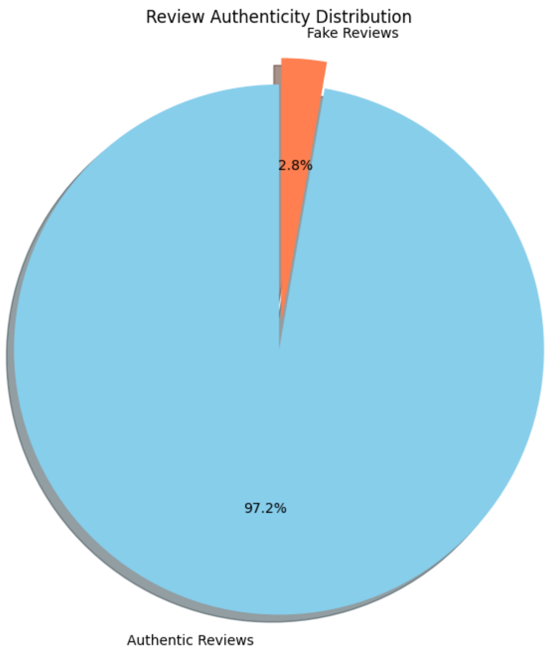

Figure 11: Pie chart signifying the proportion of human-generated reviews compared to the number of computer-generated fake reviews as predicted by the Authenticity Detection model.

Conclusion

In the world of online shopping, reviews have become one of the most important ways to judge a product. This brings up the importance of using machine learning to predict the general sentiment of the review and to understand whether it is written by a human or generated by a bot. In this project, we have successfully developed two models that have been able to solve this problem with high accuracy. We have also applied this model to external reviews to see its use in a real-world scenario.

We have seen the importance of vectorization where sometimes it can be more beneficial than tuning the hyperparameters of individual algorithms. The importance of stop words also cannot be understated as it aided in increasing accuracy in the algorithms. We also saw that simple models, like Logistic Regression, often outperformed many complex algorithms like Kernal SVC and tended to be much more interpretable as well.

If we could spend more time on this project, we would have liked to employ word embeddings, like word2vec, and deep learning algorithms to comparatively analyze the performance of our models to more sophisticated state-of-the-art models.

We would like to advise future DS 5220 students to choose a project you are passionate about or find interesting. It will motivate you to work harder on it and delve deeper into your findings. If you are working in a group, it helps to have frequent communication with your teammate(s) since they could help you out if you are stuck somewhere. It is also important to pace yourself and not overwhelm yourself with too much work at the same time as it can erode the quality of your work. Most importantly, you should have fun while doing your project as it is the best way to learn from the work that you are doing.

References

[1] John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

[2] Salminen, Joni, et al. "Creating and detecting fake reviews of online products." Journal of Retailing and Consumer Services 64 (2022): 102771.

[3] Sun, Huan, Alex Morales, and Xifeng Yan. "Synthetic review spamming and defense." Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013.

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. https://scikit-learn.org

[5] Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Inc. https://www.nltk.org/

[6] Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95. https://matplotlib.org/

[7] Waskom, M. L. (2021). Seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://seaborn.pydata.org/

[8] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362. https://numpy.org/

[9] McKinney, W. (2010). Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference (pp. 51-56). https://pandas.pydata.org/