NYU

# Spring 2022 DS GA 1015 Capstone Project: Amazon Reviews Ranking System

By Ilias Arvanitakis and Alexandre Vives

**Professor Andy Halterman**

**Due 12/22/2021**

**Research question**

The ever-increasing popularity of e-commerce and online shopping, highlights the need to discern the quality of the reviews posted and as a result help the consumer choose the right product based on high quality review data. With a vast number of available reviews, it is of primary importance to classify which ones are useful and which ones are not. Many e-commerce websites are providing helpful and unhelpful review insights. However, this is not efficient, because new reviews require time to be classified as helpful or unhelpful. Focusing on Amazon, we notice that poor quality reviews in many instances take the top spot in several products. This project aims to address this issue by using text analytics that will be able to classify a review as helpful or unhelpful even before the users start voting on its helpfulness. That way we will be able to pre filter the posted reviews and provide users with a better shopping experience.

Most Amazon products have an attached reviews section where users that bought the item are able to express their opinion about the product and/or assign a number of stars from one to five. These reviews are then used by other users interested in the product to get a sense of the true value associated with it and compensate for not being able to physically see it like they otherwise would in an actual store. The problem that arises with Amazon's approach is the chance for a new review to be seen by the community without any condition.

Instead of giving a recent review a free pass to the top reviews section, our group proposes designing a machine learning model to rate the usefulness across the recent reviews and give that free pass to the one with the highest chance to become useful. This way, poor quality reviews will be more unlikely to be shown at the top of the reviews section and mislead future clients into buying a product that might increase their distrust towards buying Amazon products.

The way that we are going to approach this problem is by treating the helpfulness of the review as the dependent variable and the rest of the variables that are provided through the dataset as the independent ones. Feature engineering will be utilized in order to expand the capabilities of our dataset. We will employ multiple machine learning algorithms such as logistic regression, random forest, decision trees and others and eventually select the algorithm with the highest accuracy. We will choose the appropriate accuracy indicator later in this paper.

One major question that arose when we started working with the data was if it would be possible to predict helpfulness without utilizing the text of the review. This, of course, would have made things easier, since fitting a model with text is computationally very intensive and takes a lot of time. However, we noticed that without using the text helpful and unhelpful reviews have a lot of overlap. Thus, taking advantage of the text in the reviews is a one-way road. Preprocessing and feature engineering is required in this case in order to convert our text to a form that can be used for machine learning.

**Previous approaches**

Throughout time there have been many publications both online and in scientific journals where researchers are trying to address this issue of the quality for online reviews. Some of the noteworthy approaches use probabilistic models and machine learning for classification purposes. Sipos, Ghosh et al., 2014, argue that the more the votes a review receives, the more miss ranked it is. Based on that assumption they create a probabilistic model to classify amazon reviews. Following their observation, we decided to classify as helpful only those reviews that have a high ratio of helpfulness (number of helpful votes divided by total votes). Rodak, Xiao et al. utilize naive bayes and SVM for their classification problem. Their SVM approach reaches around 70% accuracy, which is a good baseline comparison for our model. Zhang, Wei et al., 2014 propose a classification method based on the helpfulness distribution that is observed across reviews with similar characteristics (product, rating of product, time, location etc). They evaluate their approach using confidence intervals. Finally, Hjalmarsson, 2021 uses linear regression and convolutional neural network on tf-idf and vord2Vec datasets, trying to obtain the lowest MSE value. The convolutional neural network approach on the tf-idf dataset had the lowest MSE. Based on that we decided that the tf-idf would be a good method to use on our text dataset. What is extremely interesting in this area, is that there are many ways that the data can be utilized and there is also a wide variety of algorithms that can be used. A very essential element in our opinion that was missing during our peer review, was the recall rate. When classifying reviews, the most important target is to predict with a higher precision the helpful reviews. Thus, our analysis is based on the recall rate and the F1 score as the main factor of model choice. The second element that was missing from our peers, was the utilization of many different machine learning algorithms and their comparison. As a result, what differentiates our approach is that we will use the Logistic Regression, Random Forest Classifier, AdaBoostClsssifier and a Decision Tree Classifier and we will evaluate their efficiency based on the recall and F1 rate.

**Why text analysis techniques are appropriate for this research project**

Our main goal in this project is to predict the usefulness of reviews. Initially it is worth searching for alternative ways to evaluate this, without utilizing the textual data. For this we used the t-SNE function from scikit learn, which reduces the dimension of the dataset. Excluding the textual data and reducing the rest of the dataset in 2 dimensions we see that the 2 classes overlap with each other. That means that defining the helpfulness as a function of the other variables excluding the text is not informative at all. That makes sense since the helpfulness of the review is based on the text that the reviewer writes. Further to that, helpfulness can be defined as the combination of words that are informative about the quality of the product. Creating a tf-idf we convert the text to word features and the value of each feature is the frequency of that word inside the review. After training the model and estimating the weights of the factors, we will be able to classify new reviews as helpful or not. Other features that we are going to use that stem directly from the text is the length of each review. After doing some peer analysis we concluded that the length of a review is an important factor in defining the helpfulness of a review. All the above will not be possible to do without the use of the text in our dataset, thus text analysis techniques are very important for this project. The usefulness of a review can be barely captured by these other features; therefore, it was crucial for us to use the text of each review to assess its usefulness.

**About the data**

To grasp how users classify reviews as helpful or unhelpful, we are going to use a publicly available dataset from Amazon which we obtained from Kaggle. In particular we will use a subset of Amazon reviews containing 576,908 reviews from January 1st, 2006, to December 31st, 2011. For each review we have 7 features: ProductId, Ushered, HelpfulnessNumerator which is the total number of helpful votes for every review, HelpfulnessDenominator which is the total number of votes per review, Score which is the star rating associated to that review, Time, and the review text. The pros of this dataset is that it is used by many sources, and we can be sure about its validity and accuracy. Also, it provides us with all the necessary data we need for our analysis. A major strength of this dataset is the provision of the HelpfulnessDenominator, which will be very informative on the factors that make a review unhelpful.

Our model will classify a review as helpful if the ratio of HelpfulnessNumerator/HelpfulnessDenominator is greater than 0.70. Considering that Amazon stopped using the unhelpful choice for its reviews, future research will be harder by not being able to access this information. Thus, using this dataset, we will be able to factor in the unhelpful votes and create a more accurate model. Moving to the cons and limitations of the dataset, a major question that arises given the current lack of access in the unhelpful votes, is whether the model will be able to perform well in the long run. In our opinion, since we train the model using the tf-idf on the helpfulness ratio, which is very strict, we believe that the model will remain useful for out of sample classification, even without the choice of the unhelpful votes.

**Justification of the text analysis method**

As previously mentioned, the aim of this project is to train a model using reviews' text and some associated review features to predict the helpfulness (helpful, or not helpful) of recent and therefore unrated reviews. We are going to perform this analysis by using four major machine learning algorithms: Logistic Regression, Random Forest Classifier, AdaBoostClsssifier and a Decision Tree Classifier. Before running the algorithms, certain preprocessing steps took place.

Initially, reviews with less than five votes (up or down) were removed from the dataset. This was done to favor items with more votes. For example, reviews with only one vote as helpful, would be classified as 100% helpful, when in reality we don't have enough information. Further to this we decided to do smoothing, in order to further reduce the effect of reviews with very few votes. Part of the preprocessing stage is to create a column containing the helpfulness ratio. Our approach was simple and only included the addition of 5 in the HelpfulnessDenominator, which was part of the calculation of the helpfulness ratio.

The aftermath of the preprocessing stage included some further feature engineering. Taking advantage of the UserId, we created a new feature that includes the total number of reviews that the given id had made. So instead of using the UserId as a feature, we will use the number of reviews of a certain user to determine the helpfulness of the review. Additionally, we decided to drop the UserId after this step. Looking at the previous approaches we got the idea to use the length of the review as a feature, with the main argument being that from a theoretical point of view that longer reviews should be more informative and thus more helpful. Finally, we dropped the time column, the product id and both the HelpfulnessNumerator and HelpfulnessDenominator.

Having created the above features, we proceeded to the preprocessing of the text of the review. We removed many unnecessary words such as the URLs, numbers and stop words, and also, we converted everything to lowercase, removed stemming and page breaks. These steps were very necessary in order to reduce the size of the text and exclude non-informative words. With this part done, eventually we will have to convert the

text into a machine-readable format. The method of our choice is the Tf-idf, which essentially converts all words into features and uses the frequency of each word as the value. When creating the Tf-idf we deemed as important to use the 5000 most used words when at the same time dropping words with a frequency of less than 0.01. Since online reviews contain a lot of typos because people write fast and do not watch out for their spelling, dropping outliers would be helpful for the accuracy of our model.

The last part of feature engineering included the implementation of the Kmean model, where our goal was to cluster our data and create extra features based on these clusters. For the clustering we used all the available features, implemented the KMeans algorithm and plotted the inertia vs the various values of the number of clusters. Observing the plot and using the Elbow method with inertia we defined that we should have a total of 4 clusters. Essentially 4 clusters can be interpreted as a group of objects with similar characteristics. The way that this can be useful in our analysis, is by controlling within our model for the characteristics of the cluster. Ultimately, we aim to eliminate the problem of the possible correlation between our features and the error term and as a result get better and more unbiased estimates. The clusters require one hot encoding in order to convert them into binary variables.

Being done with feature engineering, partitioning the dataset into training and testing subsets follows. By imposing the constraint that only reviews with a 70% helpfulness ratio will be considered as helpful, we create an unbalanced sample. When deciding the cut-off to divide unhelpful from helpful reviews it was important to consider that the aim of this model is to maximize the confidence that the chosen review amongst the recent reviews is truly a helpful review. l.  Dealing with an unbalanced sample always poses a challenge. With our final dataset containing 53,770 rows and 9,262 helpful and 44,508 helpful and unhelpful reviews respectively, we defined a test set with 30% of the total reviews. The test set contains around 2,806 helpful reviews and 13,325 unhelpful reviews. The training set on the other hand required some further modifications. Since we mostly care to classify the helpful reviews, we decided to drop 15000 unhelpful reviews. That left us with 6,456 helpful and 16,183 unhelpful reviews. That way we partially solved the unbalanced dataset problem. To completely solve this issue, eventually we unsampled the helpful reviews and we ended up with a balanced sample containing 16,183 reviews in each category.

Finally, we noticed the need to normalize some of our features. Essentially, the score of the review, the text length, and the number of reviews of each user followed very different patterns in their scale. Thus, normalizing them using the StandardScaler method, would eliminate scaling issues that might affect our predictions.

After the preprocessing stage, feature engineering, Tf-idf and the creation of the training and testing subsamples, the dataset is eventually ready for the machine learning algorithms. The machine learning algorithms we chose were the LogisticRegression, the RandomForestClassifier, the AdaBoostClassifier and the DecisionTreeClassifier. With our main goal being classification, these are the most used algorithms for these problems. The reason that we didn't use the SVM is because we have a benchmark from our previous approaches review. Essentially, using these algorithms we will get an output for the accuracy of the training and the testing dataset. However, what we mostly care about is the amount of correct helpful reviews that we predict. In other words, we care about the recall. Initially we run each algorithm on its own to determine which one has the highest recall rate and then for that algorithm we do grid search in order to tune our hyperparameters.

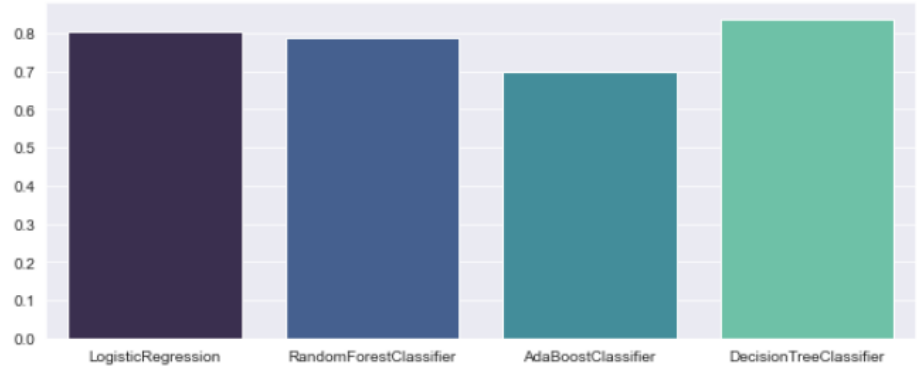
**Results**

Once the data was ready, four models were trained: Logistic Regression, Random Forest Classifier, AdaBoostClsssifier and a Decision Tree Classifier.

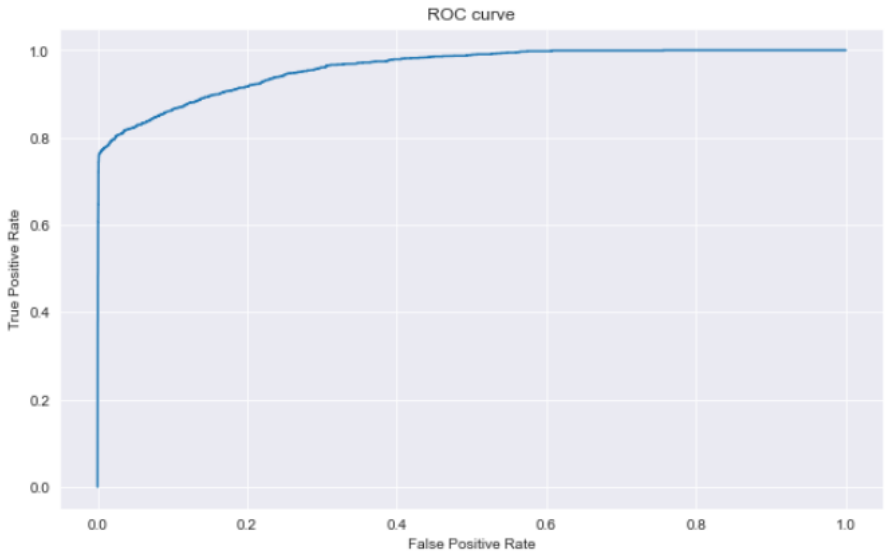
When analyzing the model performances, the focus was put on the recall of each model along with its F-1 score because we mostly value the ability of the model to correctly identify helpful reviews. As a reminder, recall is the proportion of actual positives (helpful reviews) that were identified correctly.

After training all the models, with the default hyperparameters, we found that the Random Forest classifier was the most efficient one. It had a recall rate of 78.5% and an F1 score of 84.7%, followed by the Decision Tree classifier which had a higher recall at 84.5% and an F1 score of 72%. With our main goal being to maximize the recall rate, we decided to choose the Decision Tree as our best model.

```
LogisticRegression has a recall of 80.43% and an F1-score of 62.59%
RandomForestClassifier has a recall of 78.78% and an F1-score of 87.56%
AdaBoostClassifier has a recall of 69.74% and an F1-score of 43.61%
DecisionTreeClassifier has a recall of 83.68% and an F1-score of 82.82%
```




The best models were the Random Forest Classifier and the Decision Tree Classifier. The latter was chosen, therefore our team performed hyperparameter tuning using several values for max_depth and testing different configurations of class_weights (where the helpful class was weighted higher).
Eventually, the best model obtained achieves an F1-score of 84.36% and a recall of 78.93%. Its ROC curve is the following:



**Conclusion Remarks**

Extending the work done by several other data scientists, we decided to follow a comprehensive approach including several machine learning algorithms and their evaluation based on the Recall and F1 scores. After preprocessing the data, cleaning them and doing feature engineering, we ran four machine learning algorithms, out of which the best were the Random Forest Classifier and the Decision Tree Classifier. With a higher overall F1 score of 84.36% after hyperparameter tuning, the Decision Tree Classifier was chosen as the best algorithm to predict the helpfulness of a review. Interestingly, some words that play a significant role in making a review helpful are the following: favorite, better, price, not, get, great, love, recommend, product, price, best and buy. One would expect that helpful reviews use words that are both descriptive about the product and comparative against other products. The words above have both these characteristics. They can be used both for comparison against other products and for descriptions. With this naive robustness check we can verify that our algorithm is working as expected and we would highly recommend more use of it in the future for making predictions. Further work can be done in the better tuning of our model using more and bigger datasets. As we use bigger and more diverse datasets from various sources, we will end up getting more accurate and unbiased results.