Introduction

Our team is working with the Amazon Review Dataset to see if we can accurately predict the review of users based on the text. We focused on using NLP in our project to be able to make our predictions. We used Random Forest and AdaBoost as our models with different parameters on each. We further tuned our model for different statistics that we measured.

Background Knowledge

The dataset that we have chosen was the Amazon Review Dataset (2018). The link to the full page is here: https://nijianmo.github.io/amazon/index.html.

There are two parts to the dataset: review and metadata. The review part is what is normally shown as a review for a product on the Amazon store. The review had the reviewer's name, the id of the product, summary or the title of the review, review text, overall score (1-5), and other information. The metadata is about the product itself, it contains the id of the product which is mapped in the review dataset, the name of the product, small features, description, price, and other information.

The Amazon Dataset was not one large file but instead divided into the different categories that Amazon has in their store. These include Amazon Fashion, All Beauty, Books, Electronics, Pet Supplies, Tools and Home Improvement, and Video Games. There was also the complete dataset and the small subset. The complete dataset would be 34GB of data. The small subset is also known as the 5-core dataset because it only included products with at least 5 reviews. We chose the smaller dataset because the larger one required permission and it noted to use the smaller one if we are doing a class project. The dataset was not a CSV file but it was a JSON file which meant we had to convert it to a CSV in order for us to use the Pandas data frame.

Methodology

Choosing the Problem

The first issue we needed to solve was to decide what problem we wanted to solve using this dataset. Given the two datasets, we wrote down the problems that we could solve.

- Given a review, predict the overall score
- Predict the sales of a product
- How helpful is a review is (Helpful, not helpful)
- Predict what the type of product is (Sports, Video Games)
- Predict multiple categories of a product - Multi-Class Labels

Since the dataset was technically in two parts, the metadata and reviews, we decided it might be easier to just use one of them instead of attempting to combine both of them. We also wanted to use Natural Language Processing in our project so we shifted towards the review dataset rather than the metadata. We eventually picked the top choice of predicting the overall score because there was already a label and text mining can be done with it.

Parsing the text

To be able to use the text, we have to place it in a form that is usable for the model to train with. One option is the bag of words which is just treating the words as they are with no order to them. There are other NLP ways we can hold the data but we decided to use TF-IDF. TF means Term Frequency where it counts the number of times a word is used in each review. The IDF is inverse document frequency which normalizes the dataset since words like "the", "a", and "an" will be frequent but don't contribute to meaning. Putting it together, TF-IDF vectorization would need to compute the TF-IDF score for each of the words and place that information in the vector. Each review can be vectorized.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

A = "The car is driven on the road"; B = "The truck is driven on the highway" Image from freeCodeCamp - How to process textual data using TF-IDF in Python (https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/)

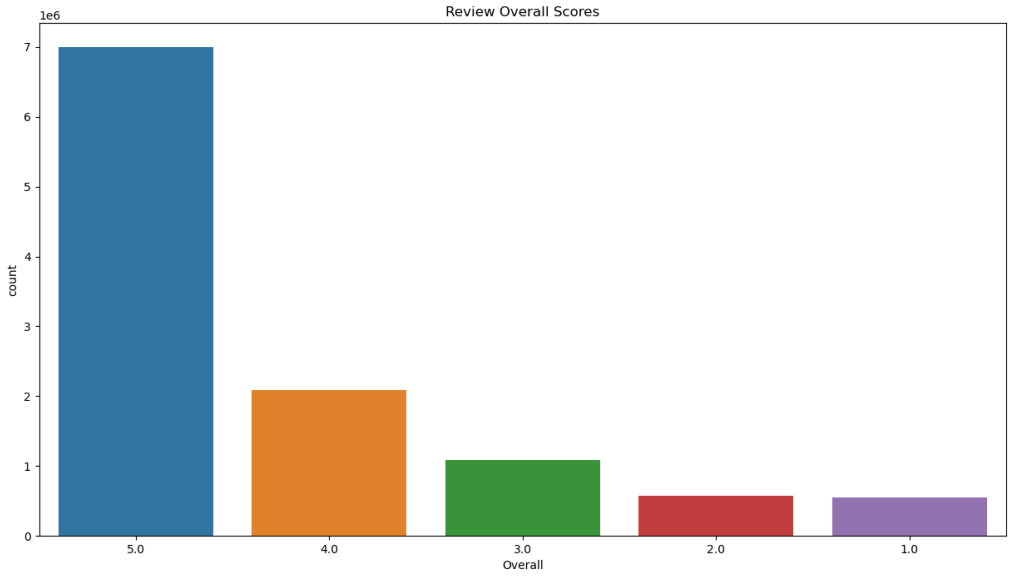
Once we decided on using the TF-IDF, we needed to clean the data. So for each review, we first needed to make all the words lower case. The reason we did this was that we don't want two words to be different just because one word is capitalized. The meaning is still the same. We then removed all punctuation and special characters and kept only the letters in each review. Then we removed the stopwords in the dataset. Stopwords are commonly used words such as

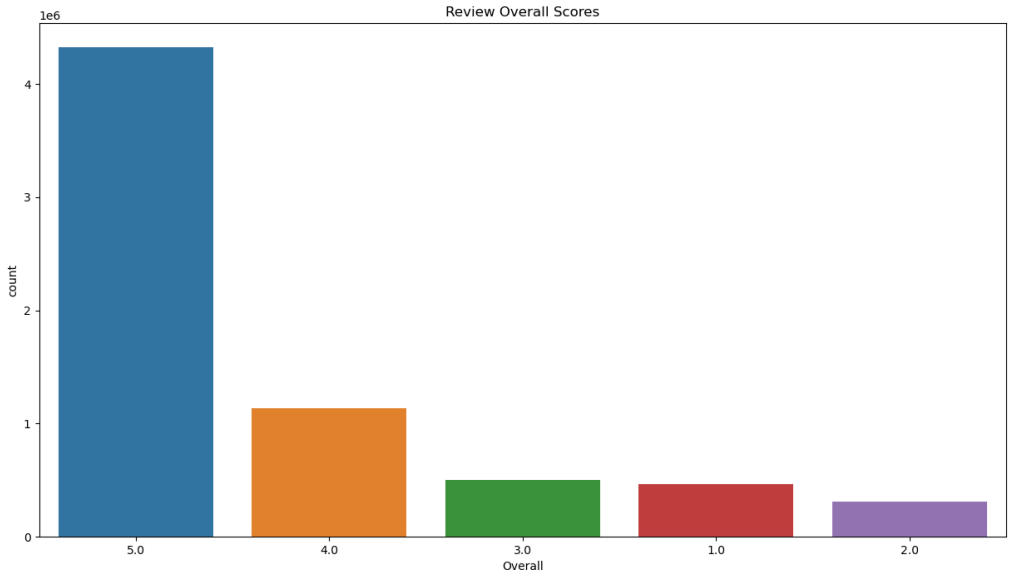
“The”, “a”, “have” and so on. These words don’t have much meaning so we removed them. Finally we used lemmatization. Lemmatization essentially combines words to a root word. So words such as “playing”, “played” “play” get all shortened to “play”. This is so that there isn’t separate rows for very similar words.

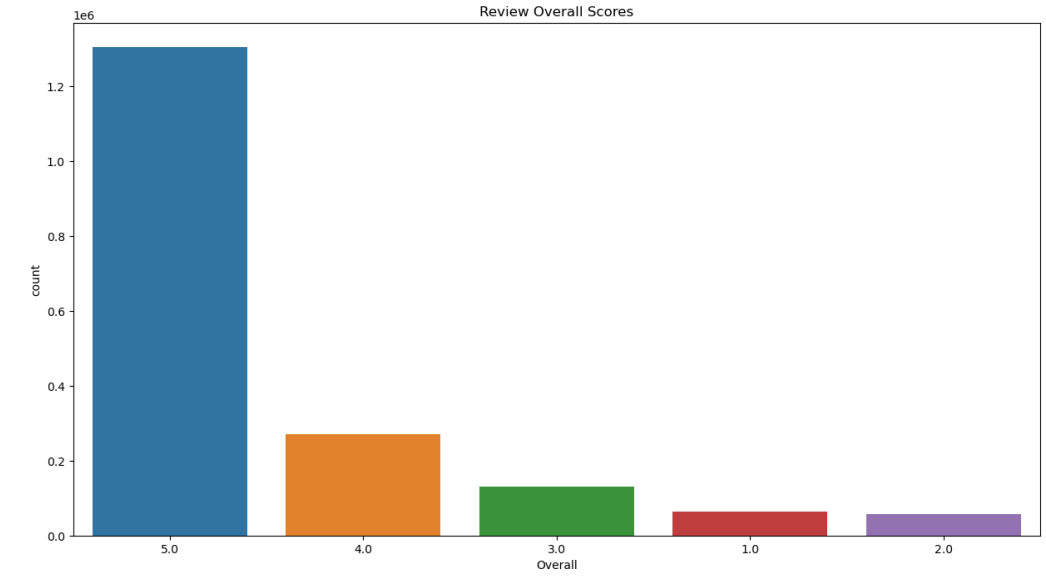
Data Analysis

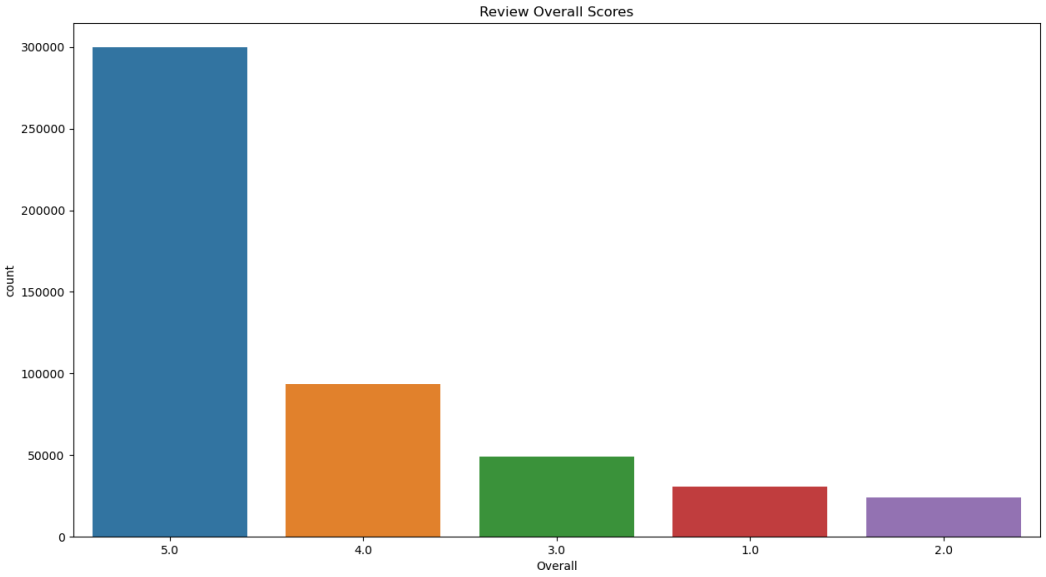
We wanted to look at some of the most interesting features to see which of them could affect the overall score the most. The features we looked at were overall score, verified, votes, and review text length.

We wanted to see the overall score distribution for four different categories. We chose Clothing, Shoes, and Jewelry, Electronics, Toys and Games, and Video Games. Here are the results from those categories.


Review Overall Scores
1e6
7
6
5
4
3
2
1
0
count
5.0
4.0
3.0
2.0
1.0
Overall

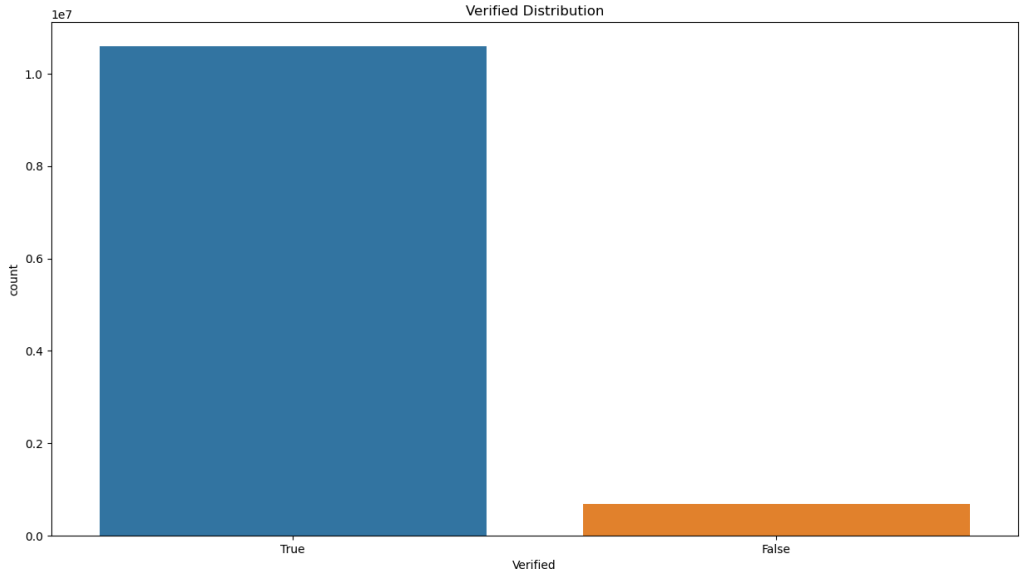

Review Overall Scores
1e6
4
3
2
1
0
count
5.0
4.0
3.0
1.0
2.0
Overall





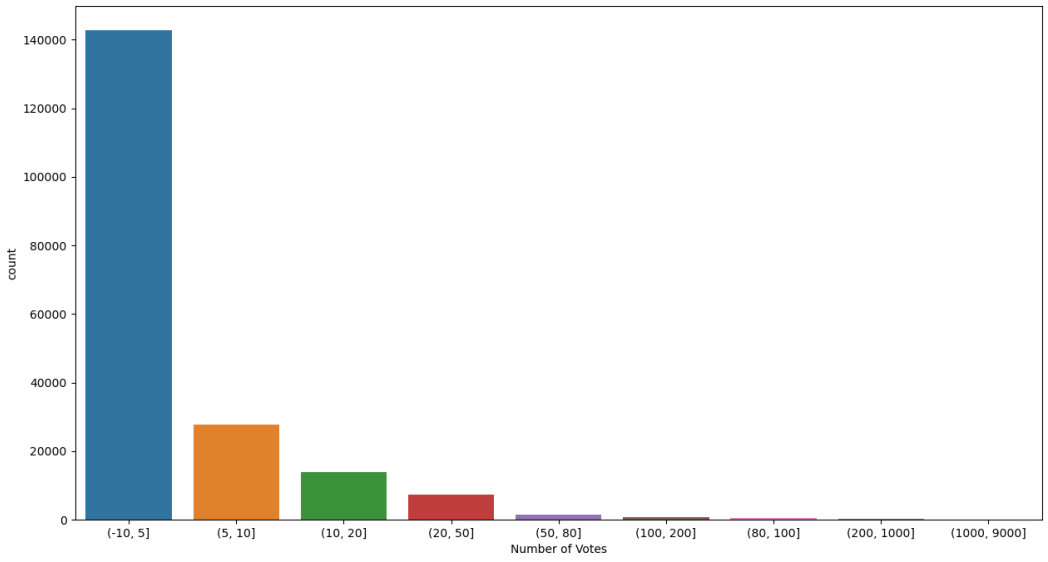
From all of the graphs, we can notice that 5 star reviews take up most of the dataset. This was to be expected because most of the time people generally give 5 stars to things that they like. Also noting that 2-star is the least popular which also makes sense because if people generally don't like a product then they would just do 1 star. We want to see if our model will do better at predicting the labels other than 5.

We also checked the other features but to save space we will only show one graph since they are all very similar. We checked the Verified feature. This feature determines if the review is verified by someone.

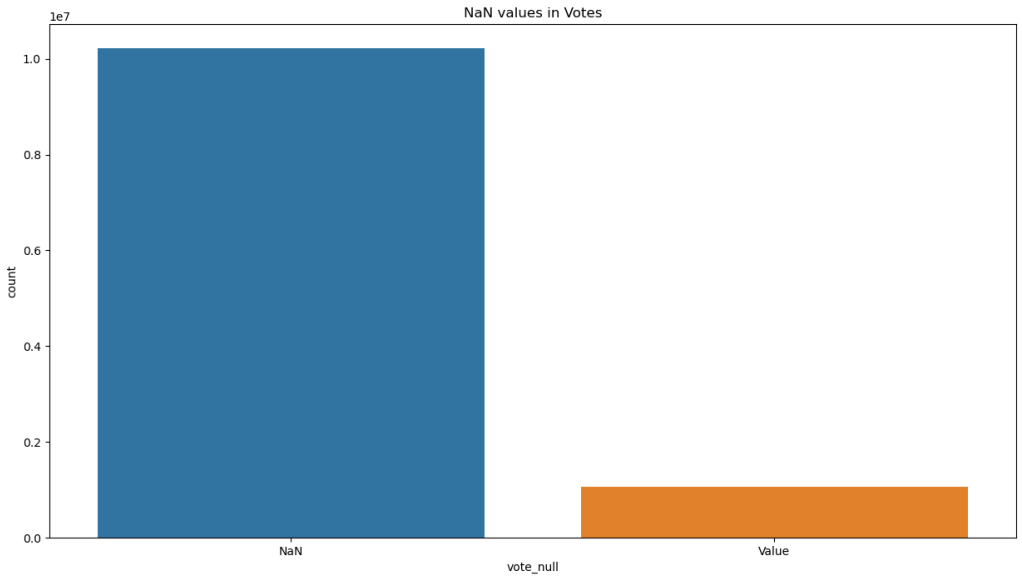


From this we determined that since most of the data is already True, we don't believe removing the False Verified reviews would affect the data.

We then looked at the vote count. We were thinking that we should only use reliable reviews that have a lot of votes.

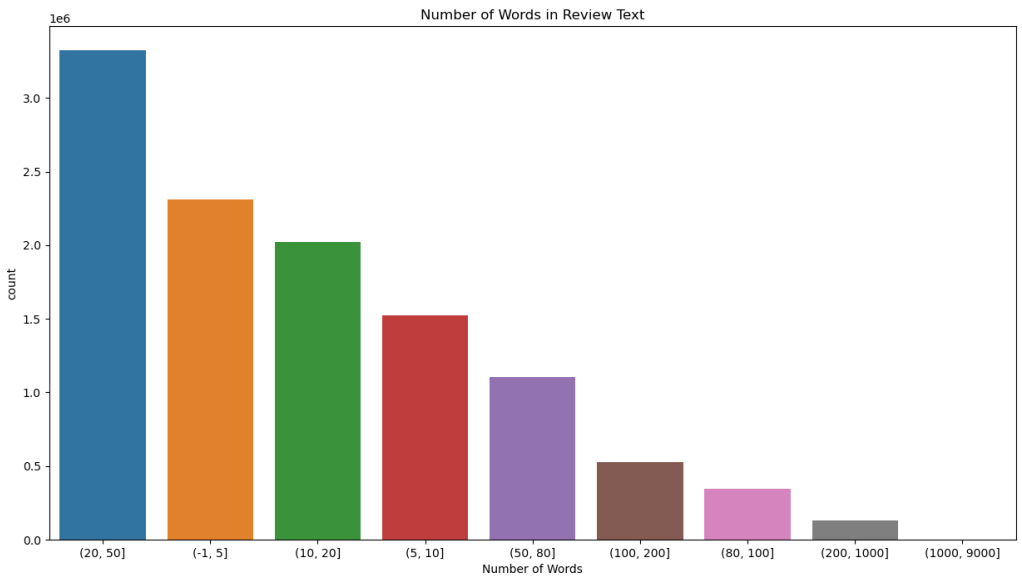


But by looking at this graph, we notice that most of the reviews don't even include many votes. Also another note, is that if there are 0 votes then there would be NaN in its place so we plotted those too.



Obviously most of the data don't even have votes so that might be useless.

We then decided to look more closely at the reviewText. Since that would be the input for our models. We chose to plot the distribution of review length.



Here we see a much more interesting spread of data. We noticed that a large number of reviews had 5 words or less. But the largest bin was reviews with 20-50 words. We wanted to use

this information to see if overall score could be predicted with a small number of words and the rest is just noise or a larger word set is needed so to extract hidden features.

Choosing Models

At the starting of our project we choose three primary models to perform the classification on our database which were as follows:

1. Random Forest Algorithm
2. AdaBoost Classifier
3. K-Nearest Neighbours

We aimed to perform classification with each model and try to achieve the best result on each one of them and further distinguish between them the best working and suitable algorithm. We moved ahead to the ace cluster with the models and experienced that the resources provided by the cluster used to die out while running the KNN algorithm. We tried our best to work on the solution but couldn't find a way to reduce the time complexity of the classifier. We moved ahead with the AdaBoost Classifier, with the Decision Tree Classifier as the base classifier and Random Forest Algorithm. The results are discussed ahead.

Tuning our Models

We focused on tuning our models in as many ways as possible to get a variety of outputs and check the impact of every combination of hyperparameters on our classifier. Some of the tunings we did are discussed below:

Changing parameters for the models

For the two algorithms, we tried to change different hyperparameters to see the output and impact on the results. In Adaboost Classifier, we used Decision Tree Classifier as our base classifier. Considering both classifiers, while performing the classification we changed the number of trees between 10, 50, and 100 and changed the maxdepth between 1, 20, 50, and 100. The **depth of a node** is the number of edges from that node to the tree's root node. As such, **the depth of the whole tree** would be the depth of its deepest leaf node. The number of trees means the number of trees present in the classifier.

Evaluating with word count datasets

After performing and evaluating the data analysis on various features of the data, we concluded that review text and summary were the only valid feature for predicting the ratings from 1 to 5.

- If the user was verified or not and the style of the text used had no correlation with the ratings and hence these two columns were not used to predict rating.
- A vote is a good option but after data analysis, we discovered that maximum values were null and hence would not have been effective.
- So, finally, due to the good distribution of the graph for reviewText, we decided to go ahead with that along with combining it will a summary because it gave good information only in a few number of words.
- A good chunk of the review text data was between 0 to 5 words. So we created a separate dataset to perform classification on that data in an attempt to reduce noise from the data and hence trying to get better accuracy.
- Using trial and error we also tried to evaluate the reviews with 20-50 number of words to understand more about the dataset and its behavior.

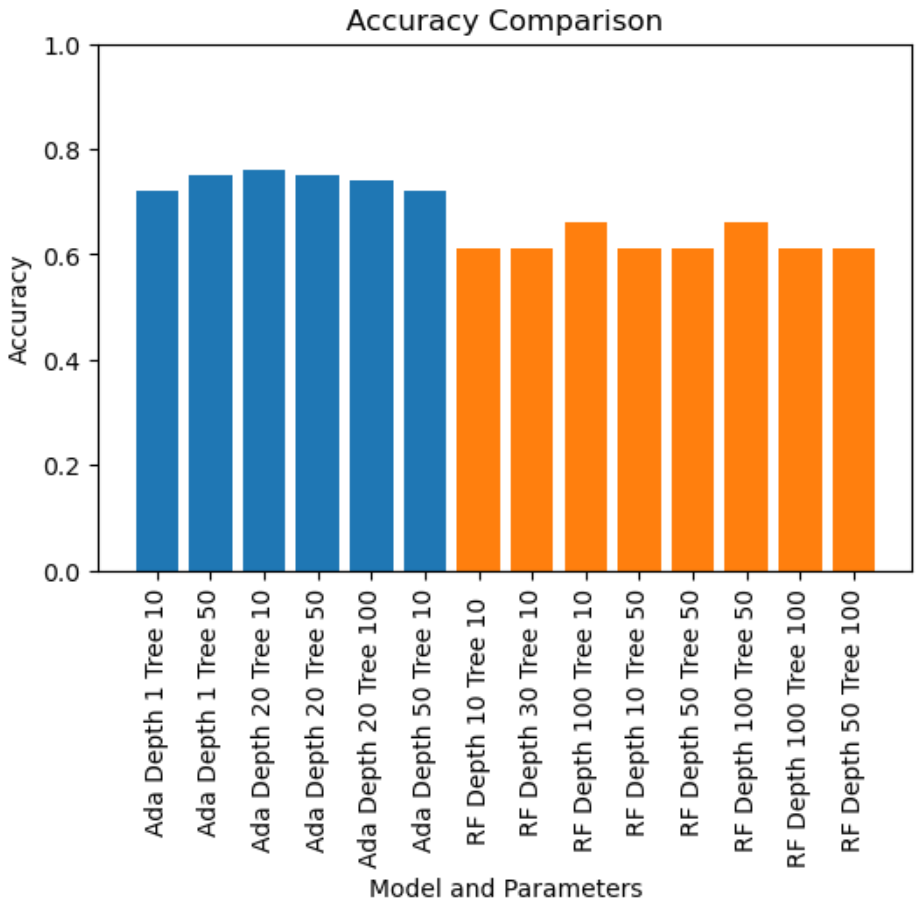
Challenges

- The majority of our time was devoted to solving the “Time Limit Exceeded” error. No matter which classifier we used, no matter how much we downsampled the data (to reduce data size), we always got the “Time Limit Exceeded” error. At first we did not think that there was anything wrong with our code because it did not give us any runtime errors. But upon further examining the code, we found that our code was taking the most time trying to clean/pre-process the ReviewText. More specifically when trying to remove stopwords. We then found the mistake in our code. Each word was being searched in **stopwords.words** (i.e. searching the word in stopwords for all the languages) however it should only search it in **stopwords.words(“english”)**. After making the required changes, our code significantly sped up and we were able to get the required results without ever getting the “Time Limit Exceeded” error.

Evaluation of Models

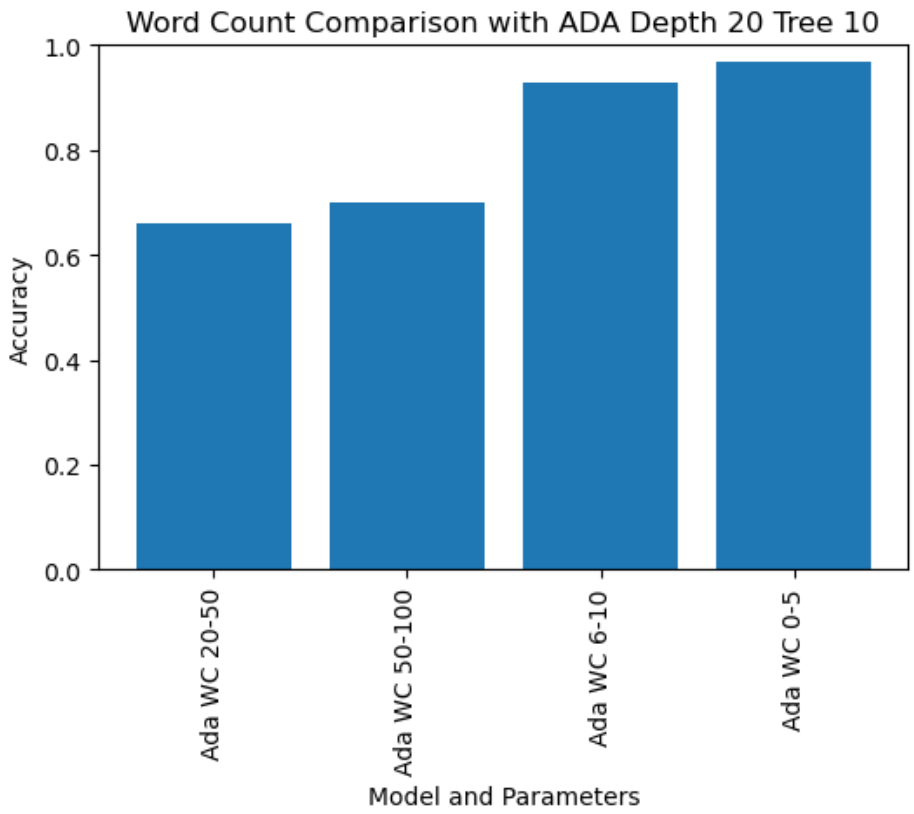
From the models mentioned above, we could only get the results from AdaBoost and RandomForest classifiers. K-Nearest-Neighbor gave us a “Time Limit Exceeded” error which is understandable since in KNN there are a lot of computations that need to be done. Also when we vectorize the ReviewText data, the number of features is significantly increased and the more features there are the more complex the KNN algorithm will get.

We ran the **AdaBoost** classifier, keeping the base classifier as the **Decision Tree Classifier** while changing the max depth and number of trees. We also ran **Random Forest Classifier** while changing max depth and number of trees. The results are shown below



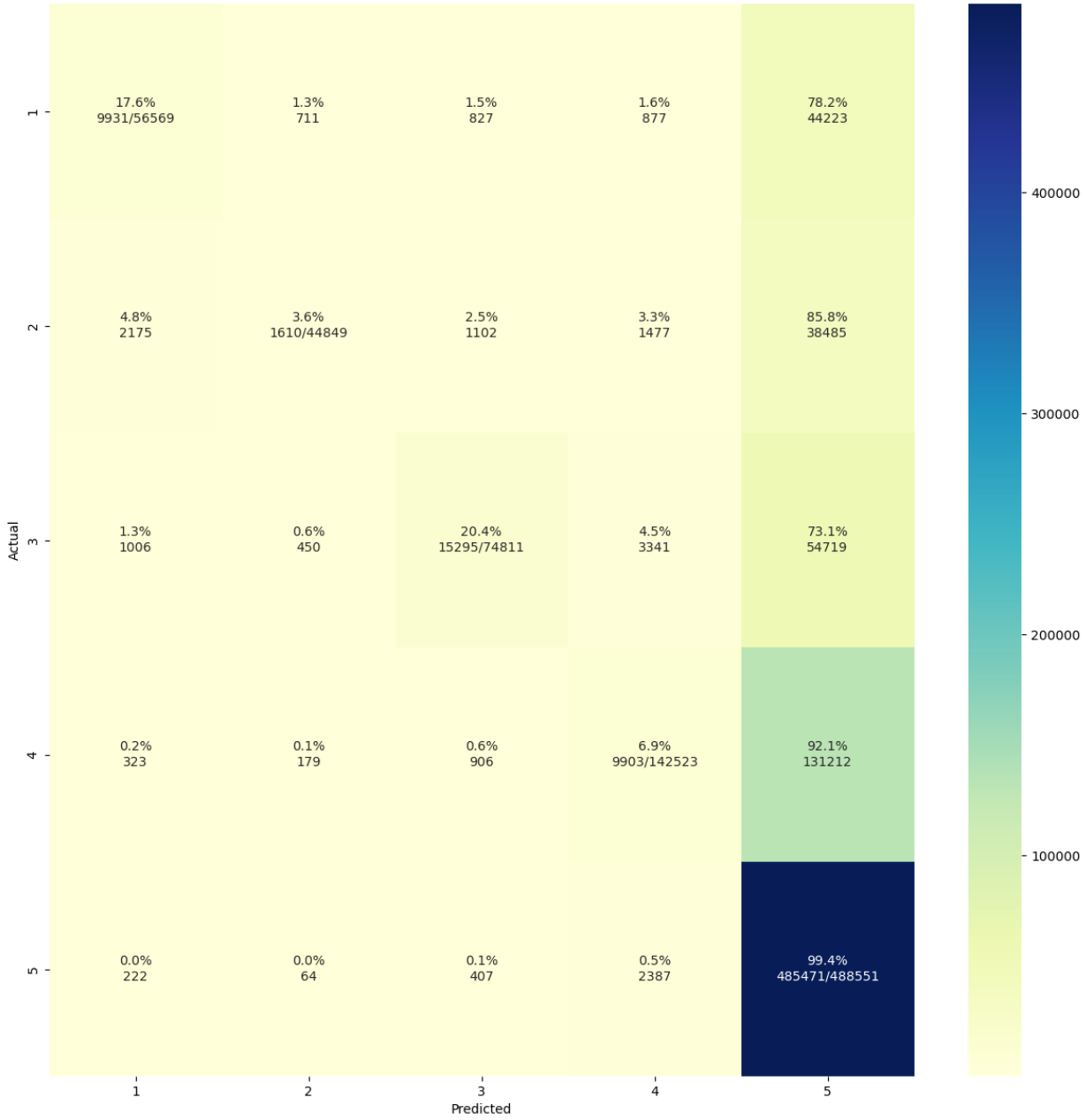
From the above models and the combination of parameters, we came to see that the **AdaBoost classifier (maxdepth-20 and number of trees-10)** had the highest accuracy of **76%**.

We then chose this classifier and these parameters to further train the dataset while tuning it according to the word count of the ReviewText. This simply means that we filtered the data according to the number of words in the ReviewText. The results are shown below.

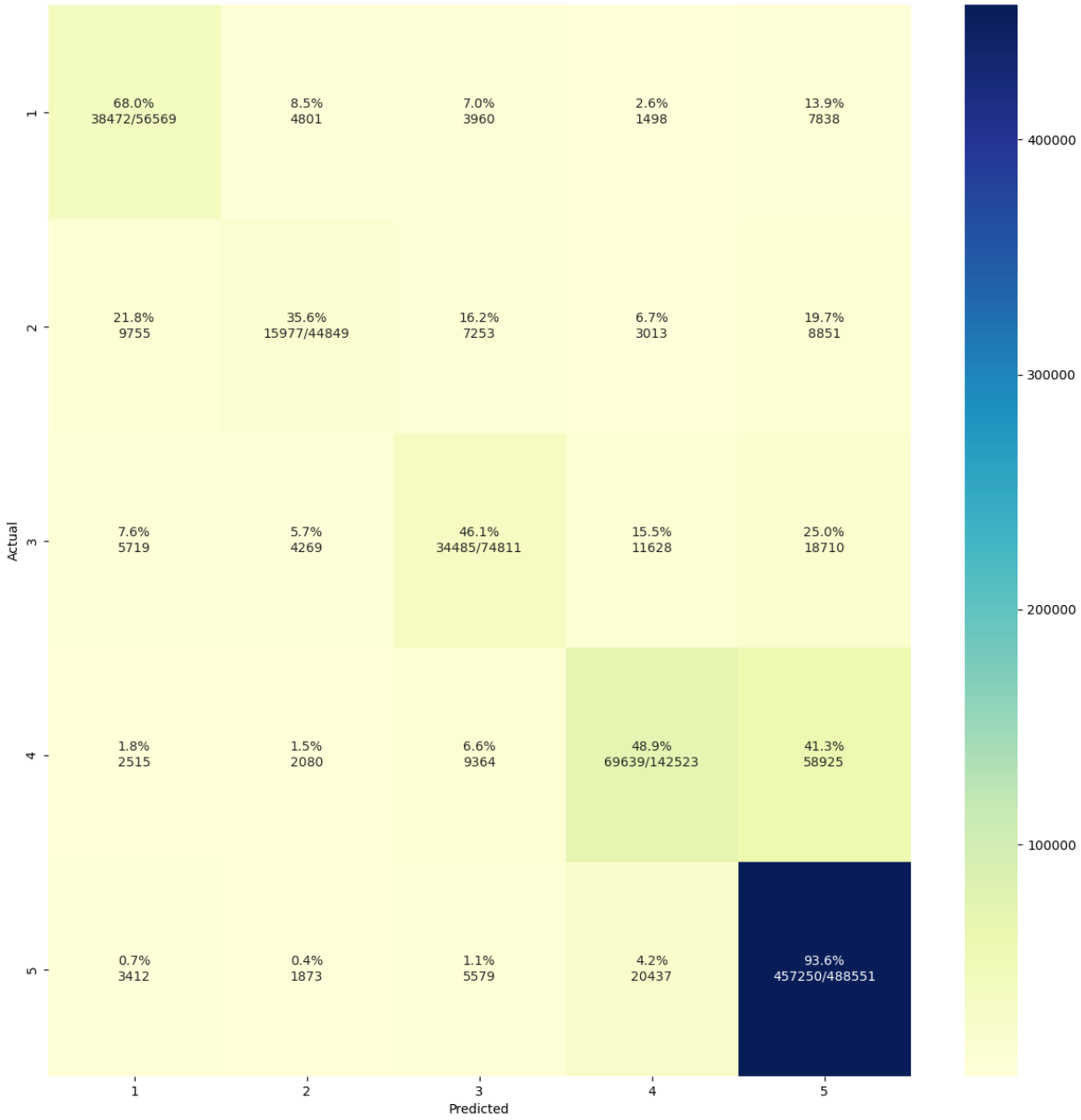


From the above ReviewText tuning, we came to see that the when we applied **AdaBoost classifier** on **ReviewText with a word count between 1-5** had the highest accuracy of **97%**

Next we plotted the confusion matrix of AdaBoost Classifier as well as RandomForest Classifier. Below are their respective figures.



Confusion Matrix of AdaBoost Classifier



Confusion Matrix of RandomForest Classifier

Confusion Matrix of AdaBoost Classifier when word count of ReviewText is 1-5

Findings

From the above figure when comparing Adaboost and Randomforest we can clearly see that, in general, **Adaboost performs better than RandomForest**.

This can be reasoned with the following:-

1) RandomForest samples training data based on the bagging technique. This means it randomly samples data with replacement. Whereas AdaBoost assigns weights to the data and the data with a higher weight is selected to be sampled. The data samples are assigned higher weights based on miss-classification and are more likely to be selected for sampling in the next turn.
2) In RandomForest, each decision tree is made independently of other trees. The ordering in which decision trees are created is not important at all. However, in AdaBoost, the ordering in which decision trees are created is important. The errors made in the first decision tree influence how the second decision tree is made.
3) In RandomForest, a decision made by each tree carries equal weight. In other words, each decision tree has equal say or weight in the final decision. Whereas in AdaBoost, some decision trees may have a higher say or weight in the final decision than the others.

The above-mentioned points are the reason that Adaboost has performed better than RandomForest.

Next, we applied Adaboost Classifier to the dataset while tuning the dataset based on the word count of ReviewText. We observed that when keeping the **word count between 1-5 and 6-10** we achieved a very high accuracy of **97% and 92%** respectively. Whereas when we increase the word count the accuracy drops to 70% then to 66%.

This can be reasoned by saying that a higher word count will have more noise in the data when compared to a lower word count and noise in the data will lead to lower accuracy. From these observations, we can conclude that we do not need a higher word count for ReviewText to accurately predict the rating score. We can consider only just a few words from ReviewText of 50+ words and accurately predict the rating score.

Next, we looked at the confusion matrix of Adaboost and RandomForest. We observed that Class 5 predictions for both the classifiers were really good however for other classes (1,2,3,4) it did not perform well. We can confirm this with the classification report of recall, precision and f1 score of the classifiers.

	precision	recall	f1-score	support
1	0.73	0.17	0.28	56569
2	0.77	0.12	0.21	44849
3	0.75	0.18	0.29	74811
4	0.62	0.11	0.19	142523
5	0.65	0.99	0.79	488551
accuracy			0.66	807303

Classification Report of RandomForest

	precision	recall	f1-score	support
1	0.64	0.67	0.66	56569
2	0.54	0.36	0.43	44849
3	0.56	0.46	0.51	74811
4	0.66	0.49	0.56	142523
5	0.83	0.94	0.88	488551
accuracy			0.76	807303

Classification Report of AdaBoost

From the above Classification reports, we can see that the F1 score of class-5 is significantly higher than the F1 score of all other classes and since the support of class-5 is significantly large we get a higher overall accuracy whereas in truth the classifiers are only good at predicting class-5 scores and poor at predicting the other class' scores.

Next, we looked at the confusion matrix of AdaBoost Classifier when applied to the dataset of tuned ReviewText having a word count between 1 and 5. We observed that when limiting ReviewText our classifier performs really great in all classes and not just in class-5. This can be confirmed by observing the F1 scores of all the classes in the classification report.

	precision	recall	f1-score	support
1	0.92	0.93	0.93	3627
2	0.92	0.90	0.91	2376
3	0.93	0.87	0.90	4560
4	0.95	0.88	0.91	14113
5	0.97	0.99	0.98	76384
accuracy			0.97	101060

Classification Report of AdaBoost Classifier when word count of ReviewText is 1-5

We can see that the F1 scores of all classes are above 0.9 which is really good. This means that the classifier can accurately predict all the classes. Whereas in the previous classification reports, we could see that those classifiers were only good at predicting class-5.

Who did What

While working on the project Yash did the planning of meetings and we all used WhatsApp for communications. Yash and Rushabh worked on converting the data to csv and sample datasets. Yash worked on the data analysis section. Isha worked on the parsing of the text. Shashwat worked on creating the scripts for the models. Rushabh, Isha, and Shashwat worked on running the initial models and getting the results. We all worked on deciding what needs to be done to tune our models and how we should break our data up. Rushabh and Isha worked on running the final models for the final data distributions. Shashwat worked on plotting the final accuracies.

Conclusion

On the chosen Amazon reviews dataset, we performed thorough data analysis, data cleaning, and model selection to predict scores from the user review text and summary. We used the data analysis findings to assess utilizing the word count as well as altering the parameters of the models to acquire the optimum outcome. As a conclusion, we were able to acquire a F1 score of more than 0.9 for all classes, making it the greatest outcome of all techniques used because it effectively predicts all the ratings.