Sentiment Analysis on Amazon Reviews

IST 664

Dale Deans, Patrick Furlong, Monica Fam

**Introduction:** For our final project, we decided to use a dataset containing Amazon reviews to create our own sentiment analysis model and explore the dynamics of word features and model parameters in an effort to make an accurate and precise classification tool. We also incorporated other methods of classification such as lexicons to observe how those models handled the reviews data and compared their results to that of our own created model.

**Data:** For our data, we went with a Kaggle dataset containing 4914 Amazon product reviews for a SanDisk hard drive. The source for the data is below:

<https://www.kaggle.com/datasets/tarkkaanko/amazon>

The dataset contained the following attributes:

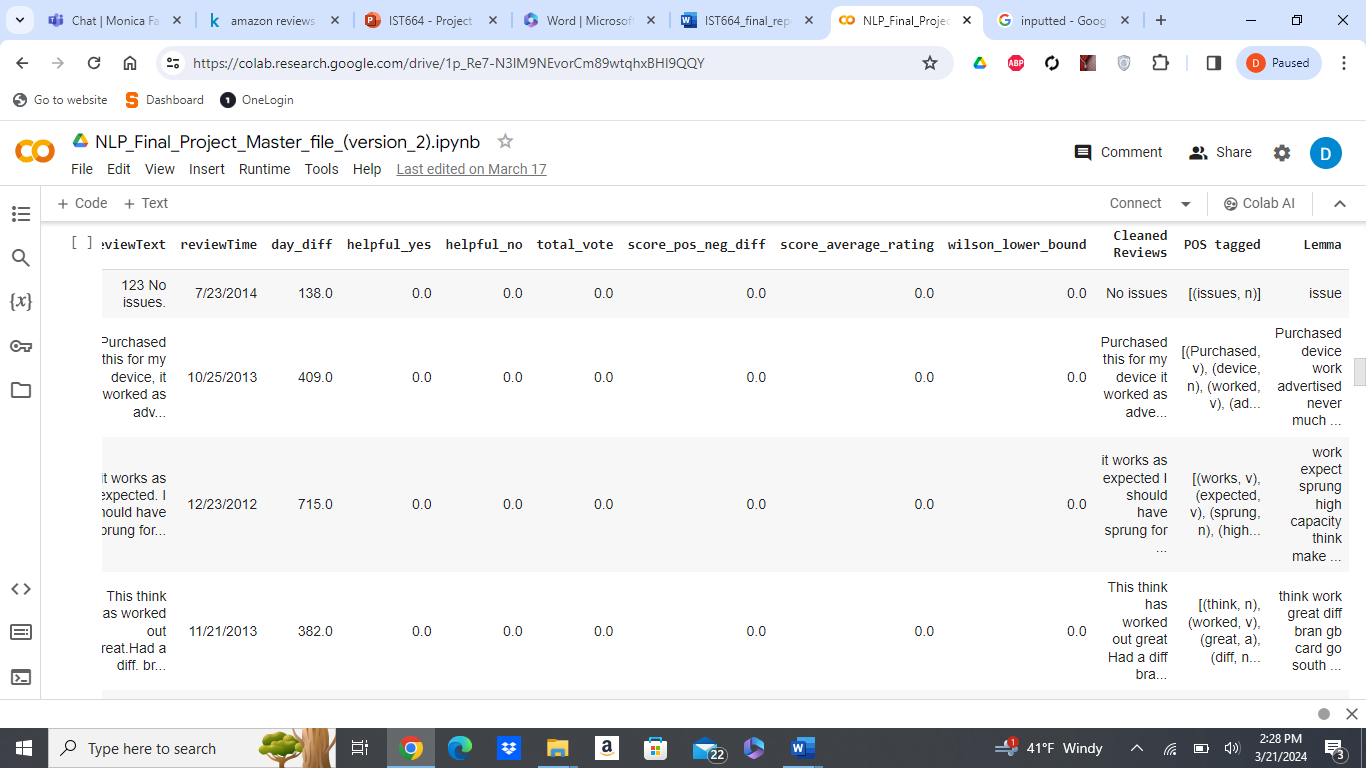
* reviewerName
* overall – star rating given by reviewers (1-5 stars)
* reviewText – product review left by user
* reviewTime – date the review was posted
* day\_diff – number of days since the review was posted
* helpful\_yes – number of users who found the review useful
* helpful\_no – number of users who thought the review was not useful
* total\_vote – number of votes given to the evaluation
* score\_pos\_neg\_diff – positive – negative score
* score\_average\_rating
* wilson\_lower\_bound

The key variables for us were the ID, reviewTime, reviewText, and overall.

**Pre-processing:**

The given dataset was already relatively clean, so when it came to producing our word features, we just took the fundamental approach. First, some cleaning was done to remove any special characters as well as numbers that may be included in the Amazon review(s). Next, tokenization on the reviewText column was done. The sentences were already separated into strings, so further word tokenization was done to return a list of lists of strings. This was followed by part-of-speech (POS) tagging, removal of stop words such as “and”, “the”, “is”, and lemmatization to group together different forms of the same word so they can be analyzed as a single item. Lastly, a new column was created to accompany each review with either 1 (positive), 0 (neutral), or –1 (negative). This was an important aspect for the training of our classification model so it could have a dependent variable to associate with the review texts being put in. To return the proper assigned sentiments for each review, a loop function was created to return a 1 for reviews with 4-5 stars, a 0 for reviews with 3 stars, and a –1 for reviews with 1-2 stars.

**Dataset with Fully Processed Review text (Lemma column)**



**About the Data:**

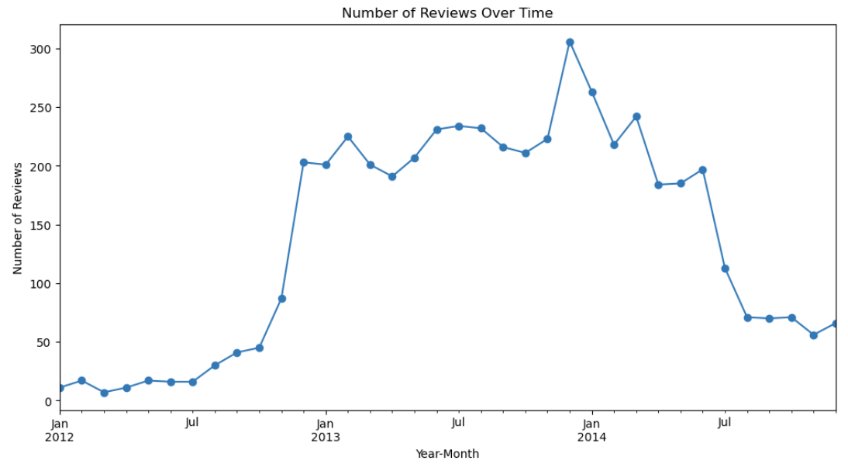
The dataset utilized in our analysis comprises 4,914 Amazon product reviews specifically for a SanDisk hard drive. Sourced from Kaggle, this dataset offers a comprehensive pool of reviews, allowing us to delve into the intricacies of sentiment analysis within the context of consumer feedback on a specific product. Each review includes essential attributes such as the reviewer's name, the overall star rating provided by the reviewer (ranging from 1 to 5 stars), the textual content of the review itself, and the timestamp indicating when the review was posted.

Additionally, auxiliary attributes such as the number of days since the review was posted, the number of users who found the review helpful or not, and various scores and metrics related to the review are provided. For our analysis, we focused primarily on the reviewText attribute, as it contains the textual content of the reviews, which is essential for sentiment analysis.

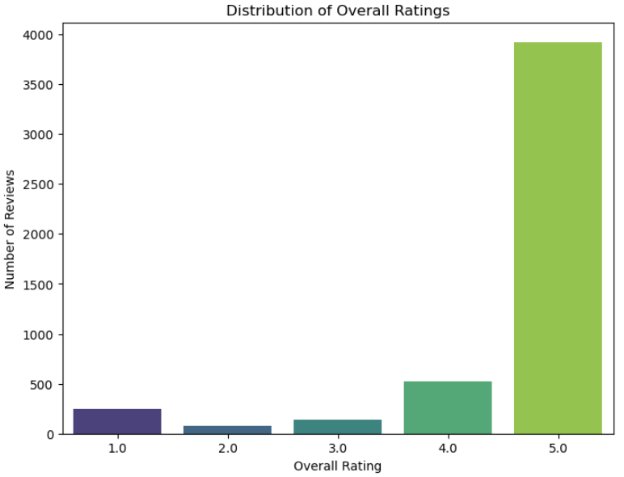
Furthermore, we performed exploratory data analysis to gain insights into the distribution of overall ratings and the length of reviews. Visualizations such as bar plots and histograms helped us understand the distribution of overall ratings and identify any anomalies, such as zero-length reviews, which were subsequently removed from the dataset to ensure data quality.

**Word Cloud**

**Plot of Reviews over Time**



**Bar Graph of Overall Ratings for SanDisk Hard Drive**



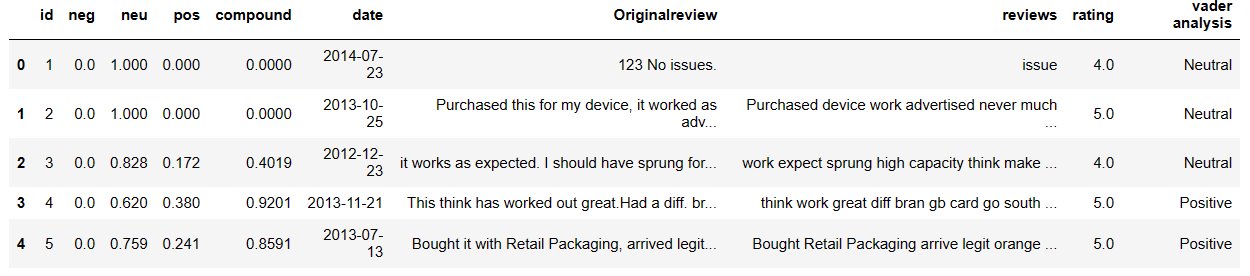
Overall, the dataset provided a rich source of information for our sentiment analysis, allowing us to explore various aspects of customer feedback and product perception over time. Through preprocessing and exploratory analysis, we uncovered valuable insights that informed our subsequent modeling and classification efforts, contributing to a comprehensive understanding of customer sentiments in Amazon product reviews.

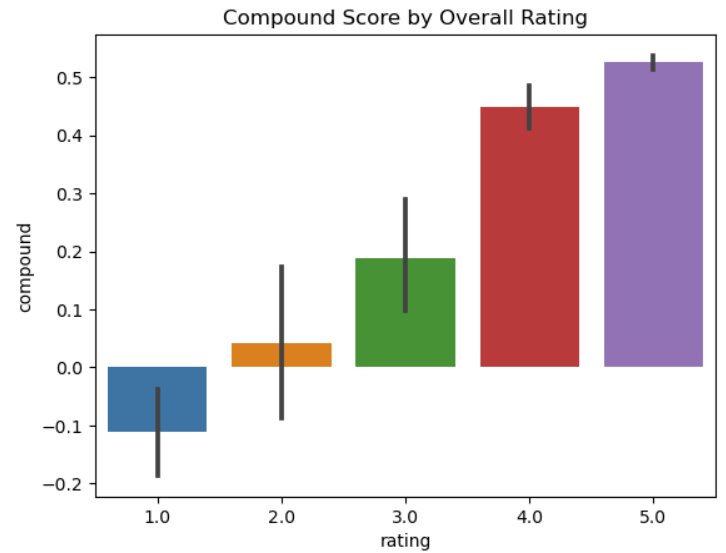
**Experiments:**

Lexicons comparison: For this portion of the project, we focused on two lexicons, Vader and TextBlob. Lexicons are another form of classification tools. What’s special about lexicons is that these models don’t require being trained, they have a built-in collection of words which are pre-assigned to certain sentiments. The downside to lexicons however is that they have usually been found to be not too accurate. The accuracy between lexicons may vary as many of them specialize with different types of texts and/or are built to approach comprehending texts in different ways. The first lexicon we used was VADER. VADER specializes in determining sentiments expressed in social media, focusing on punctuation and how words are placed with other words, considering context. Once VADER processes the text, it returns positive, negative, and neutral scores, with a reflecting compound score which can be interpreted as shown below:

* Greater than 0.5 = Positive Review
* Between 0 and 0.5 = Neutral Review
* Less than 0 = Negative Review

VADER results:



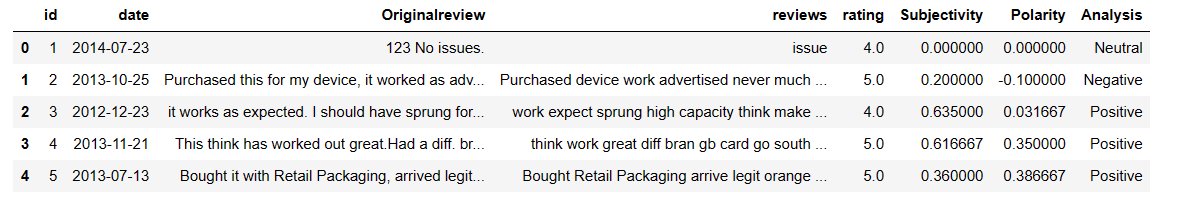


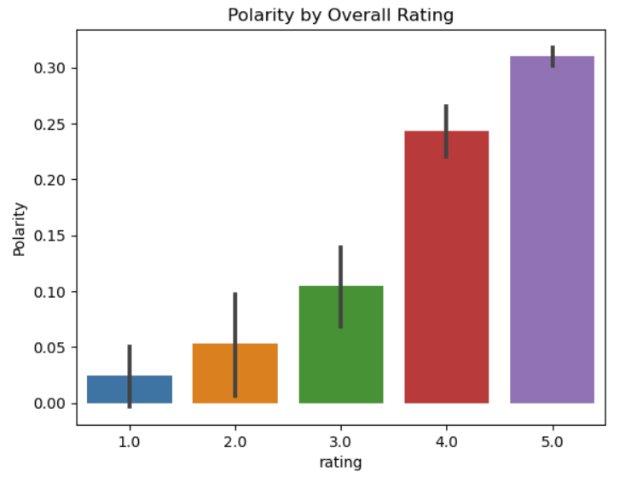
As shown in the bar plot above, most reviews were classified as either positive or neutral. Also, there is a strong distinction between positive and negative reviews, the 1-star reviews have a negative compound score.

The second lexicon used was TextBlob. The TextBlob lexicon focuses closely on the individual words and their part-of-speech tags. Unlike the VADER lexicon, TextBlob does not take context into consideration. Once the lexicon processes the text, it returns subjectivity and polarity scores and can be interpreted as shown below:

* Greater than 0= Positive Review
* Equals to 0 = Neutral Review
* Less than 0 = Negative Review

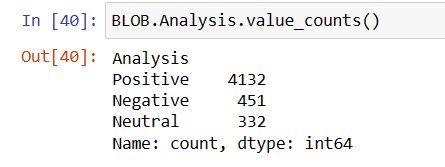
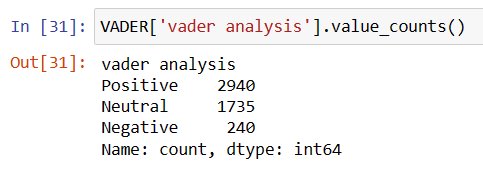
TextBlob results:



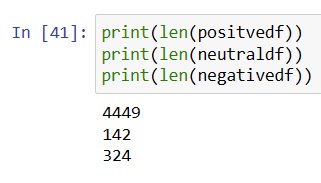


The bar plot above shows most of the reviews classified as positive similar to Vader. Unlike Vader, there’s less confidence in magnitude of how negative a 1-star review is. Also, Negative reviews were categorized as only barely negative.

**VADER** **TextBlob**



**Count of each assigned sentiment**

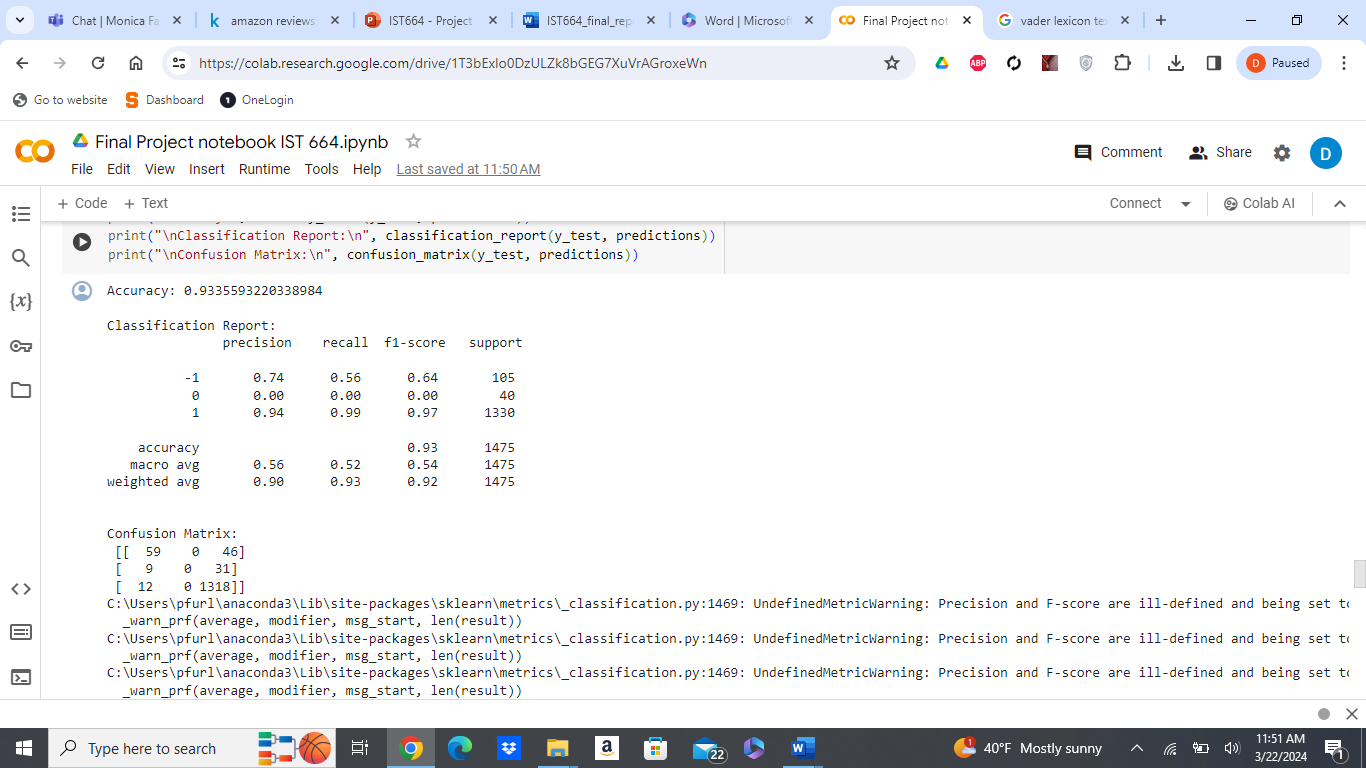


Comparing the value counts of the returned classifications for each lexicon, we can see there’s a big difference between VADER’s results and TextBlob’s results. Looking specifically at VADER’s results, it seems the model struggled to define many of the reviews and categorized them as neutral. TextBlob’s results proved to be more accurate when compared to length counts of each originally assigned sentiment.

**Naive Bayes:**

In an effort to create our own classification model, we decided to go with naive bayes from the scikit-learn package. We chose to use a 70/30 split of the data to generate training and testing datasets.

**Naive Bayes with Processed data**

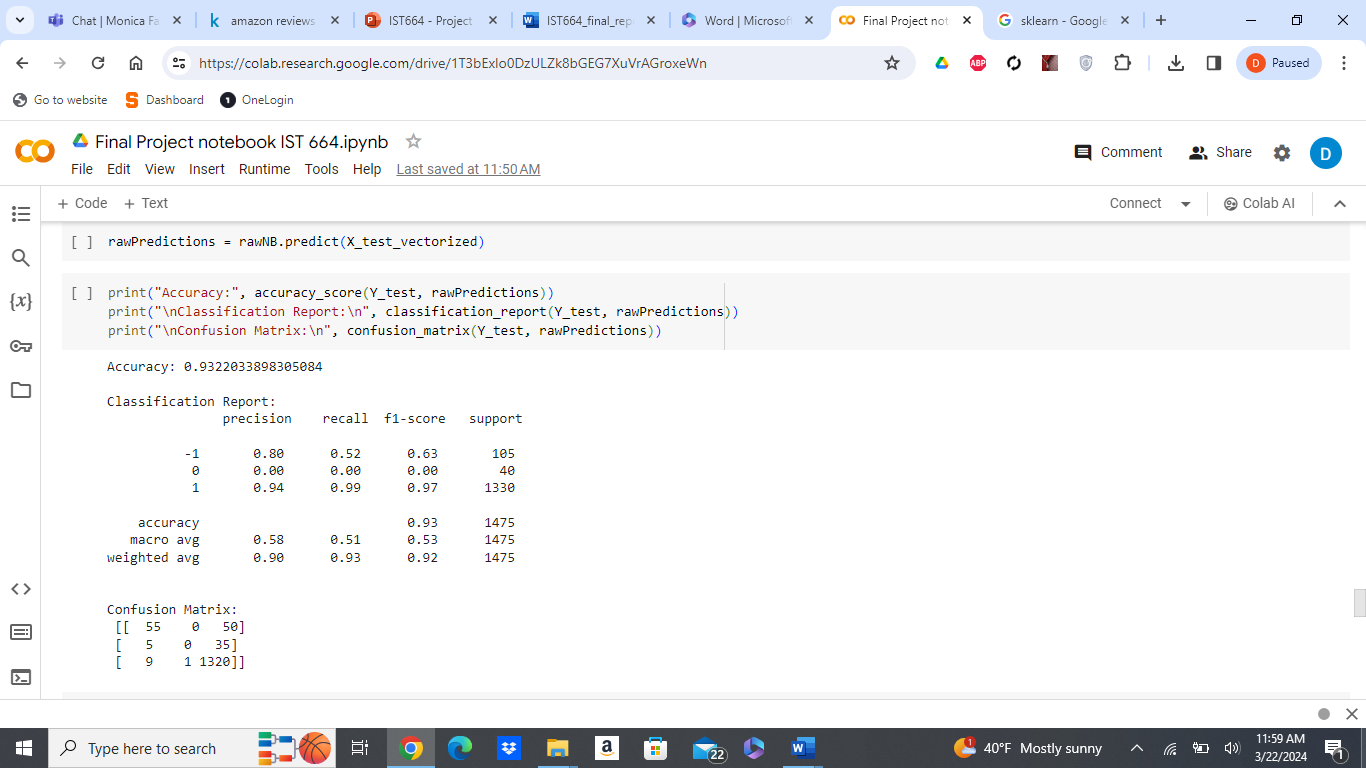


The results exhibit a mix of promising accuracy and noteworthy limitations. The classification report reveals that the model achieved an ok precision of 0.74 for negative sentiment (-1) and an outstanding precision of 0.94 for positive sentiment (1). These results indicate that the model performed well in correctly identifying positive sentiments while being serviceable in identifying negative sentiments in the reviews. Additionally, the high recall values for both negative (0.56) and positive (0.99) sentiments suggest that the model effectively captured a significant portion of the instances belonging to these categories.

However, a crucial issue emerges when considering the neutral sentiment class (labeled as "0"). Despite the overall accuracy being high, the precision for neutral sentiment is reported as 0.0, indicating that the model failed to correctly identify any neutral reviews. This lack of precision could be attributed to the insufficient amount of neutral and negative reviews available for training the model. With only a small number of negative and neutral reviews in the training data, the model might not have learned enough distinguishing features for these classes, leading to poor performance in identifying neutral sentiments.

The next experiment that was conducted was to compare our naive bayes model trained with our fully processed data to a naive bayes models trained with the unprocessed, raw data.

**Naive Bayes with unprocessed data**



The classification above shows the model trained with unprocessed data is only slightly less accurate than the original model trained with fully processed data. The two models share a lot of similarities when it comes to how well positive sentiments are identified and the struggle to classify neutral sentiments. This model, however, does seem to do a better job at classifying negative statements as it does have a higher precision of 80% compared to the original model’s precision of 74% when distinguishing negative statements.

**Insights and data rework:**

To address this limitation and improve the model's performance, it is imperative to obtain a more balanced dataset with an adequate representation of all sentiment classes, particularly the ones with fewer samples, such as neutral reviews. By augmenting the training data and fine-tuning the model, we can strive to enhance its capability to accurately classify neutral sentiments, thereby creating a more robust and reliable sentiment analysis tool for Amazon reviews.

**TF-IDF:**

A graph of a number of colored bars

Description automatically generated with medium confidence

For the third experiment, another model was, this time incorporating the TF-IDF approach. TF-IDF (Term Frequency-Inverse Document Frequency) used for feature extraction, measuring the importance of terms in documents relative to the entire corpus. Using scikit-learn's TfidfVectorizer, the raw data is transformed into numerical vectors. Splitting the data and running a naive bayes model with this newly prepared data resulted in the report below.

A screenshot of a graph

Description automatically generated

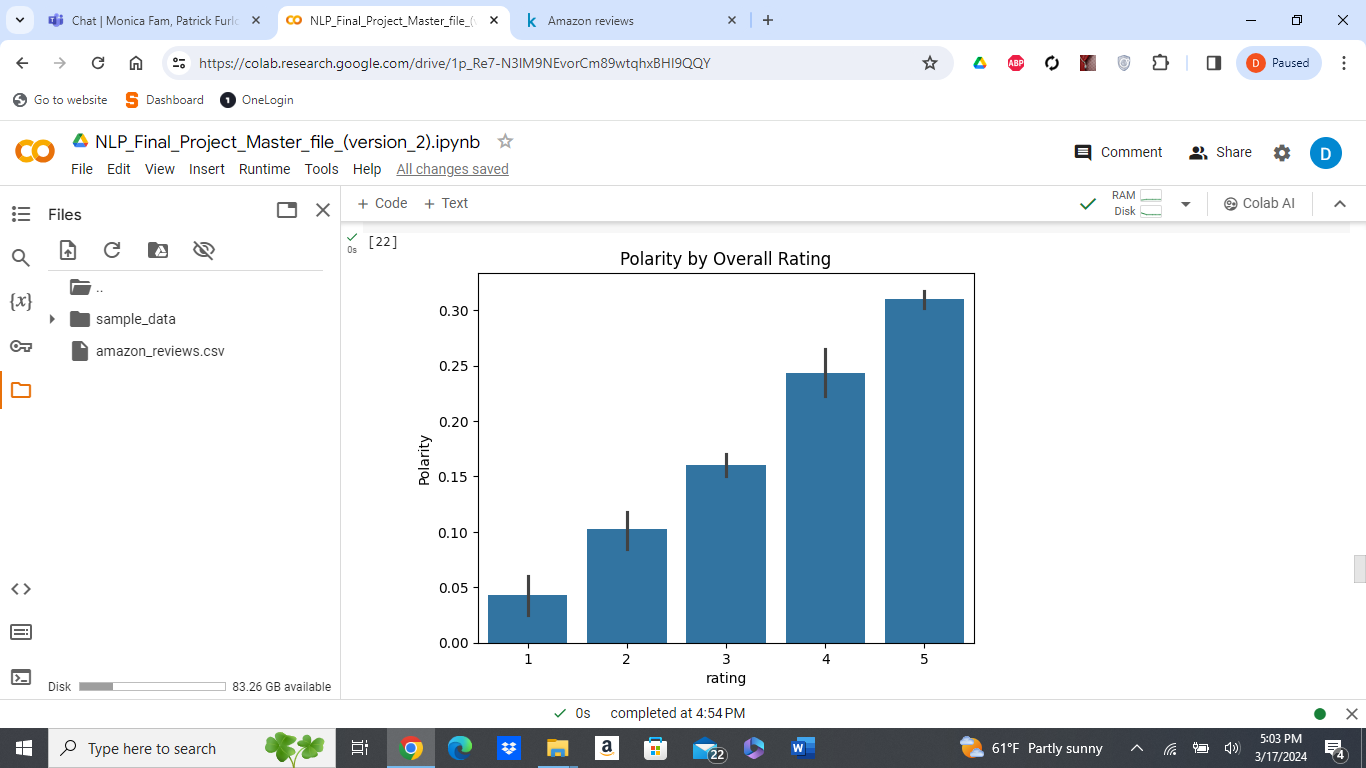
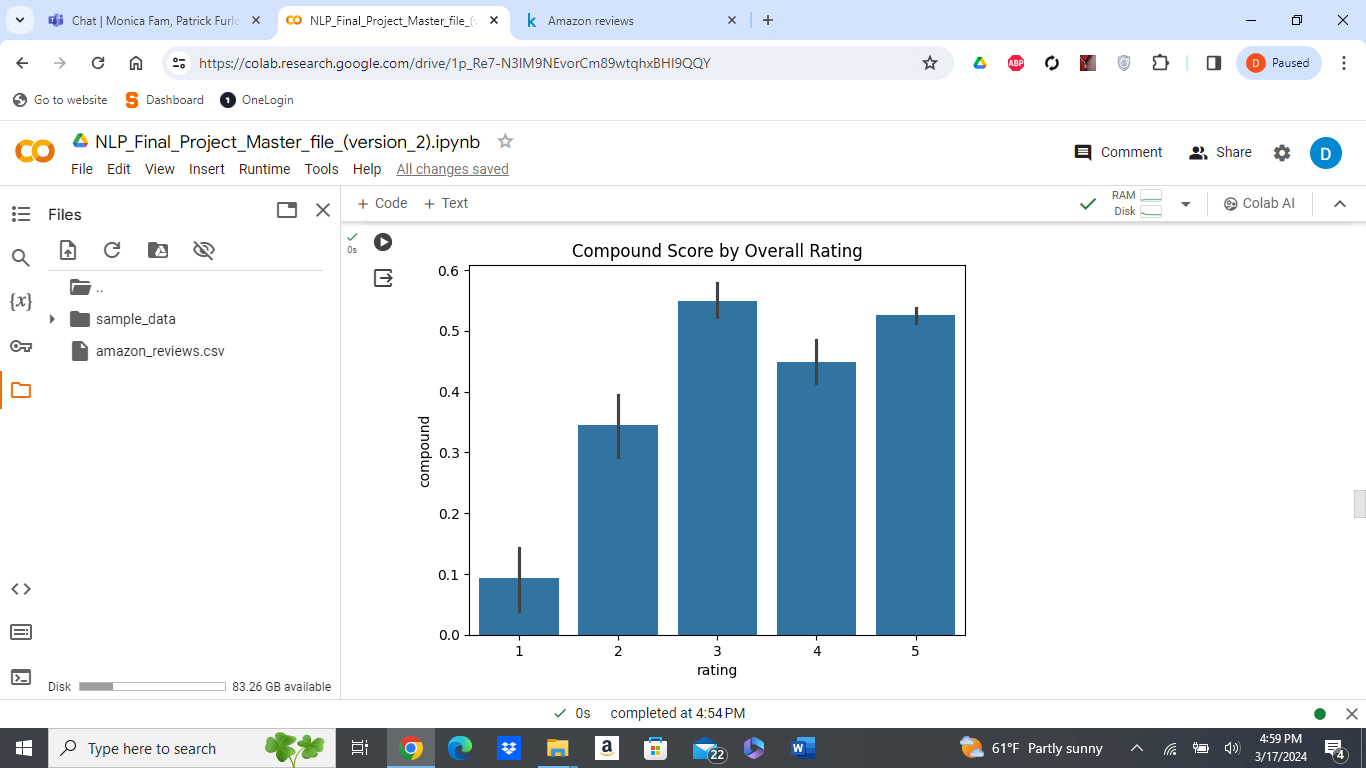
The model achieved an accuracy of 90%, indicating the model's overall correctness. The model also demonstrates high precision, recall, and F1-score for distinguishing positive sentiments (1). Unfortunately like the previous naive bayes models, there was limited success in predicting the other sentiments, which again is likely due to imbalanced data.

**Improving the Models:**

The lack of precision when it came to classifying negative and especially neutral sentiments led us to look back out our data. It was apparent the root of the cause was the imbalance in the dataset, a vast majority of the reviews as highlighted earlier were positive while neutral and negative reviews were much sparser in comparison. To address this problem, we expanded the original data set. This was done by merging data from another amazon reviews dataset, taking 2,000 records of 1-3 star rated reviews and putting it into our original dataset.

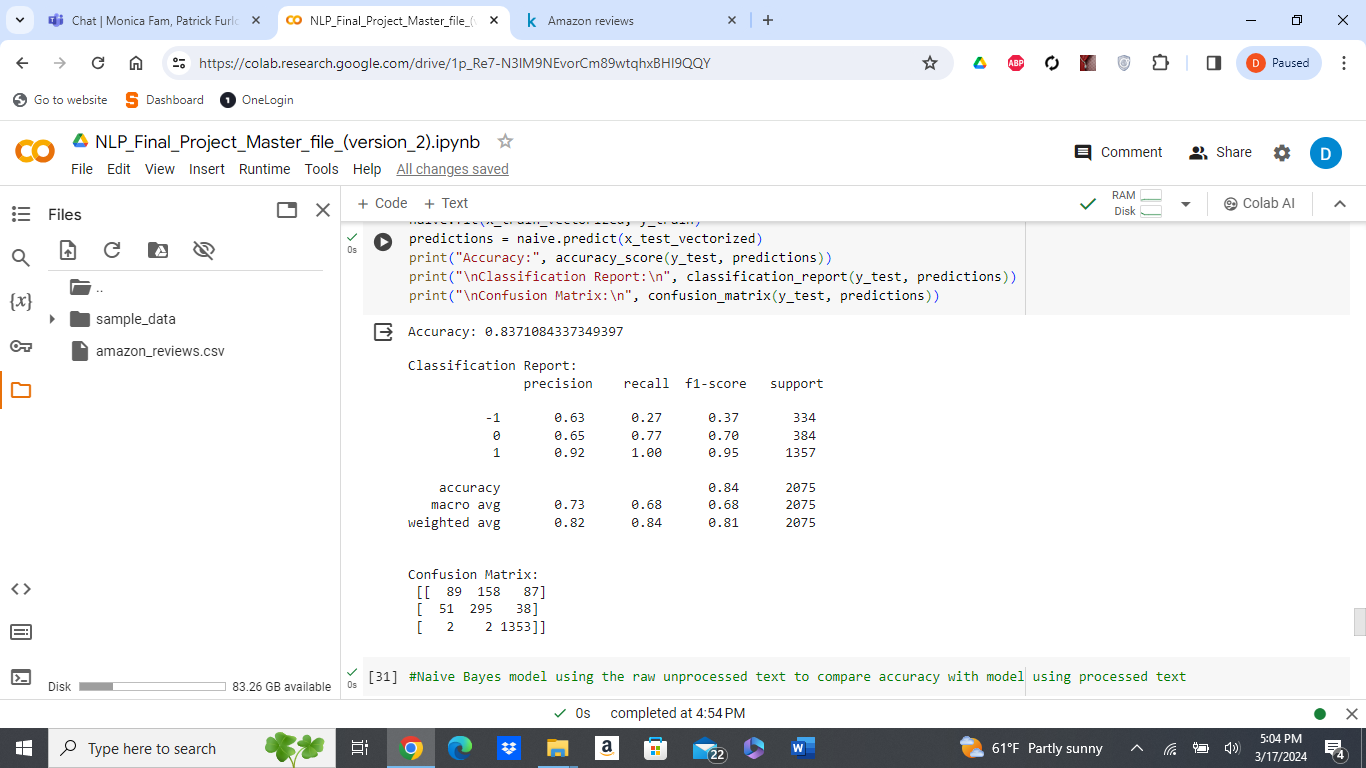
**Re-ran Results with Expanded Data:**

**VADER** **TextBlob**

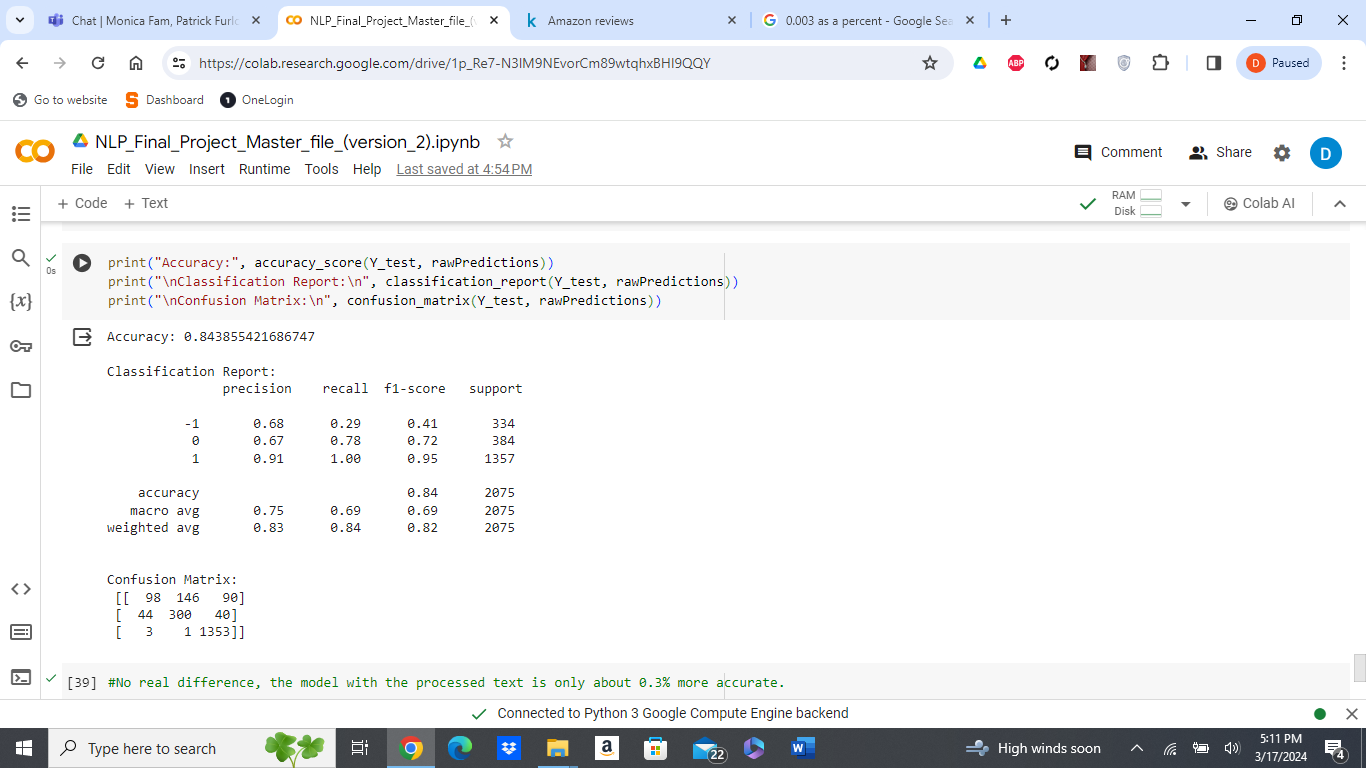


Redoing the lexicons with the newly expanded data, the VADER bar plot has drastically changed but the TextBlob bar plot looks the same. This VADER plot does a better job visualizing the impurities of the results, getting a bunch of unexpected neutral classifications. The TextBlob plot remained much more accurate, giving us the upward trend we would expect, as the polarity score gets higher so does the user’s rating. In this case, our takeaway stands as before.

**Naive Bayes with Processed Expanded data**



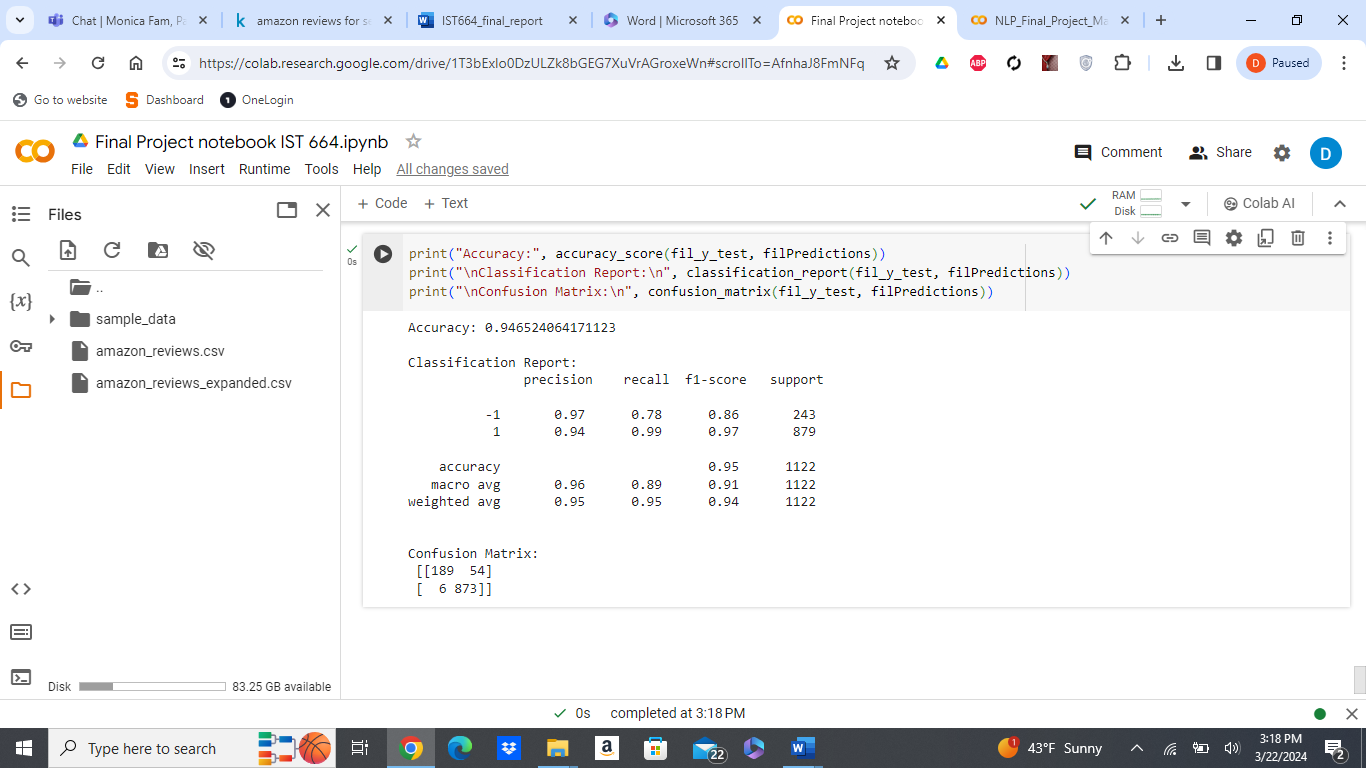
**Naive Bayes with Unprocessed Expanded data**



Training our models with the expanded dataset resulted in a loss of accuracy but still decent ones at 83.7% and 84.3% respectively. However, the models do now have precision when it comes to classifying neutral statements albeit not too high and the support numbers look better. The models’ ability to distinguish positive sentiments remained commendable in the 90 percent range, but the models’ ability to identify negative statements did take a hit.

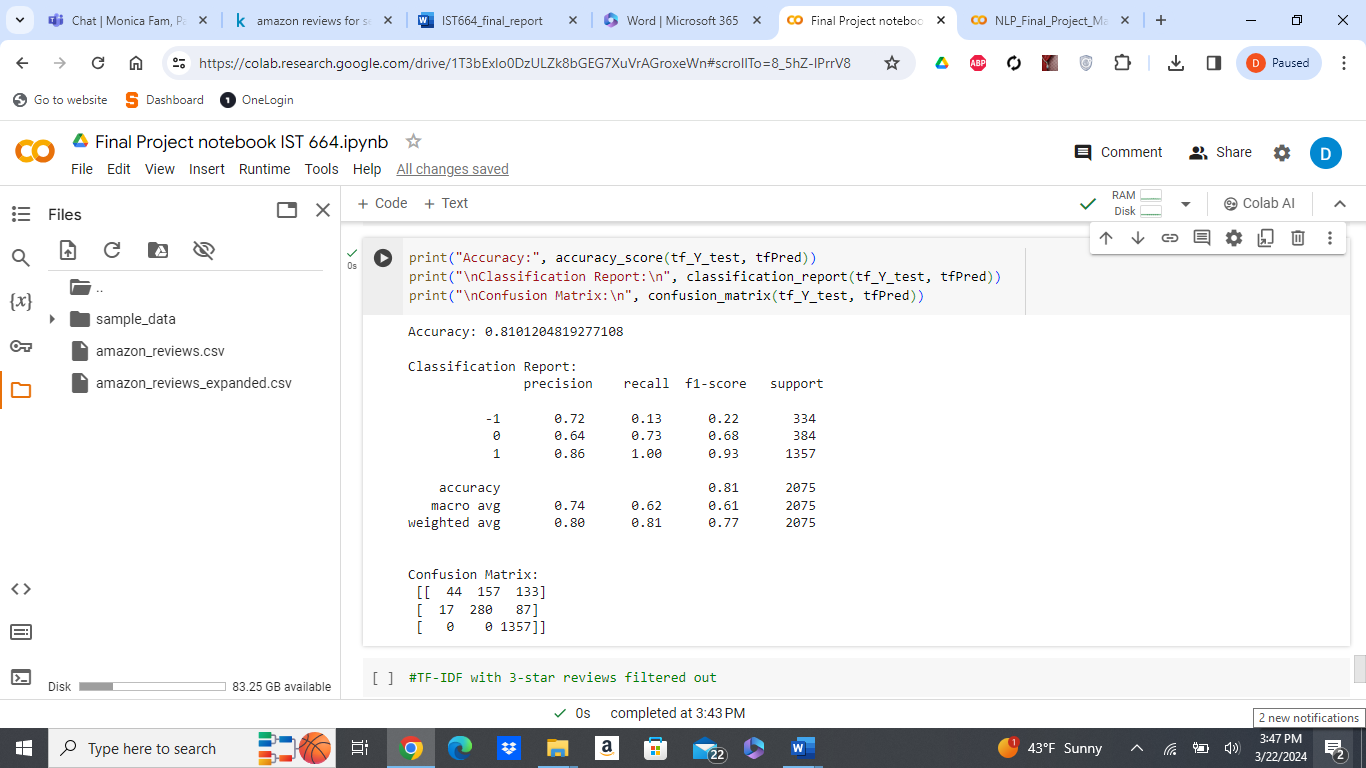
To ensure the reliability and accuracy of our sentiment analysis, we chose to filter out 3-star reviews from the dataset. While 3-star ratings may indicate a neutral sentiment, we aimed to focus our analysis on clearly positive and negative sentiments represented by 4- and 5-star ratings and 1- and 2-star ratings, respectively. By excluding 3-star reviews, we aimed to create a more distinct classification between positive and negative sentiments in our analysis.

**Naive Bayes with Filtered Expanded data (no 3-star reviews)**

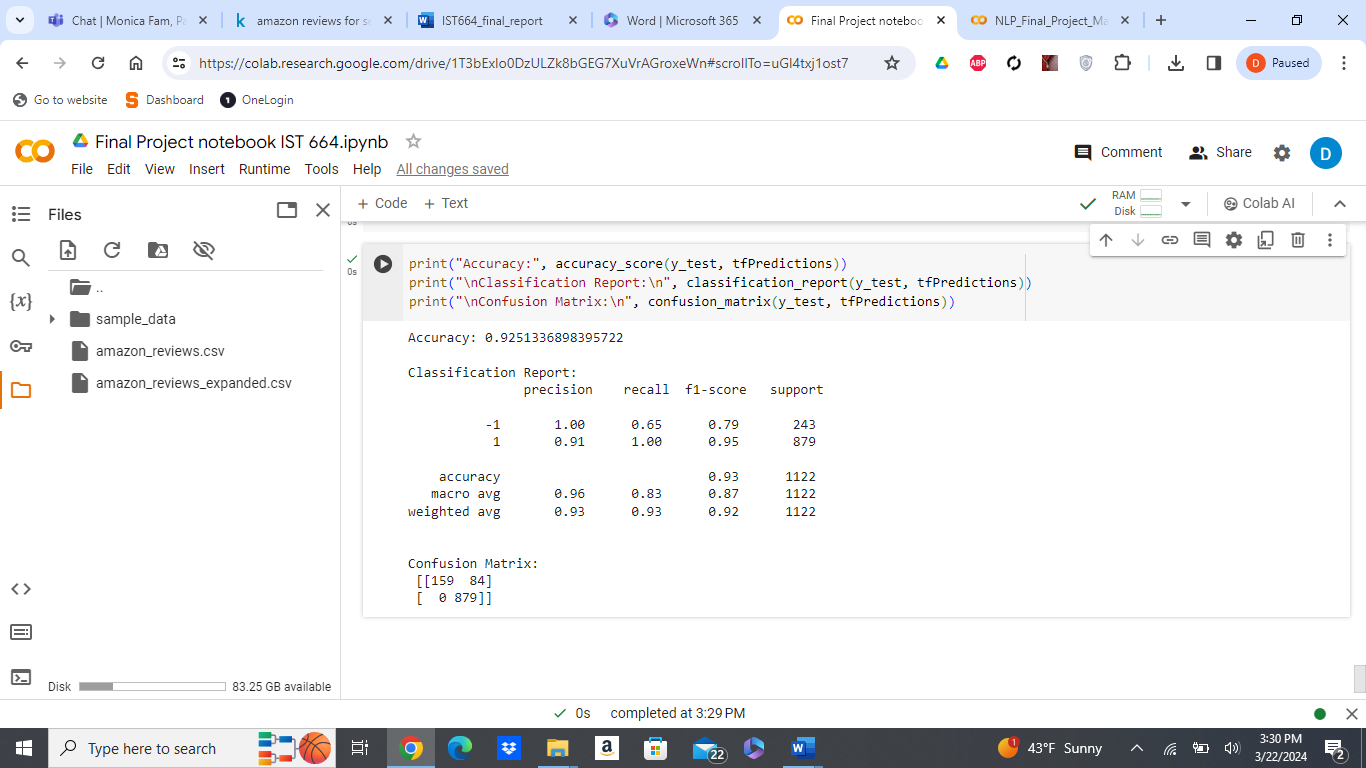


Removing the 3-star reviews brought the accuracy of the model back up to a high 94%. Doing so also greatly improved the model’s ability to classify negative reviews while still being proficient in classifying positive reviews. Assigning 3-star reviews as neutral most likely confused the model due to grey area those specific texts may hold. Although we assigned those as neutral, there very well may be cases where wording sides more with a positive sentiment and other cases where they side with a more negative sentiment. Eliminating those 3-star reviews seems to have eliminated that grey area.

**TF-IDF Naive Bayes with Expanded data**



**TF-IDF Naive Bayes with Filtered Expanded data (no 3-star reviews)**



Like the naive bayes models in our second experiment, training the TF-IDF models with the expanded data resulted in increased precisions and support numbers for the neutral and negative reviews. Eliminating the 3-star reviews also helped the TF-IDF model, returning a classification report with higher accuracy and better overall precisions and f1-scores.

**Takeaways and Conclusion:** Going through multiple phases in our project, we were able to gather some insightful takeaways. The most important one is ensuring the dataset being used is a balanced one to avoid a model that may result in deceiving results. Looking at the results, although TextBlob did a decent job processing the data, it is evident that our naive bayes model returned more accurate results. This was expected considering our model was trained using the dataset unlike the lexicon which doesn’t require training as mentioned earlier. Another takeaway, at least in our case, would be the lack of difference between the model trained with processed data and the model trained with unprocessed data. This was likely a result of the dataset being pretty clean before any pre-processing was done. Comparing our naive bayes models in our second experiment to the TD-IDF models in the third experiment, it seems incorporating TD-IDF vectorization slightly hinders the accuracy and overall ability to classify the sentiment. Lastly, eliminating the 3-star reviews and getting rid of the middle ground to focus strictly on positive and negative review texts gave us our best model, resulting in our highest model accuracy and overall precision and f1-scores.