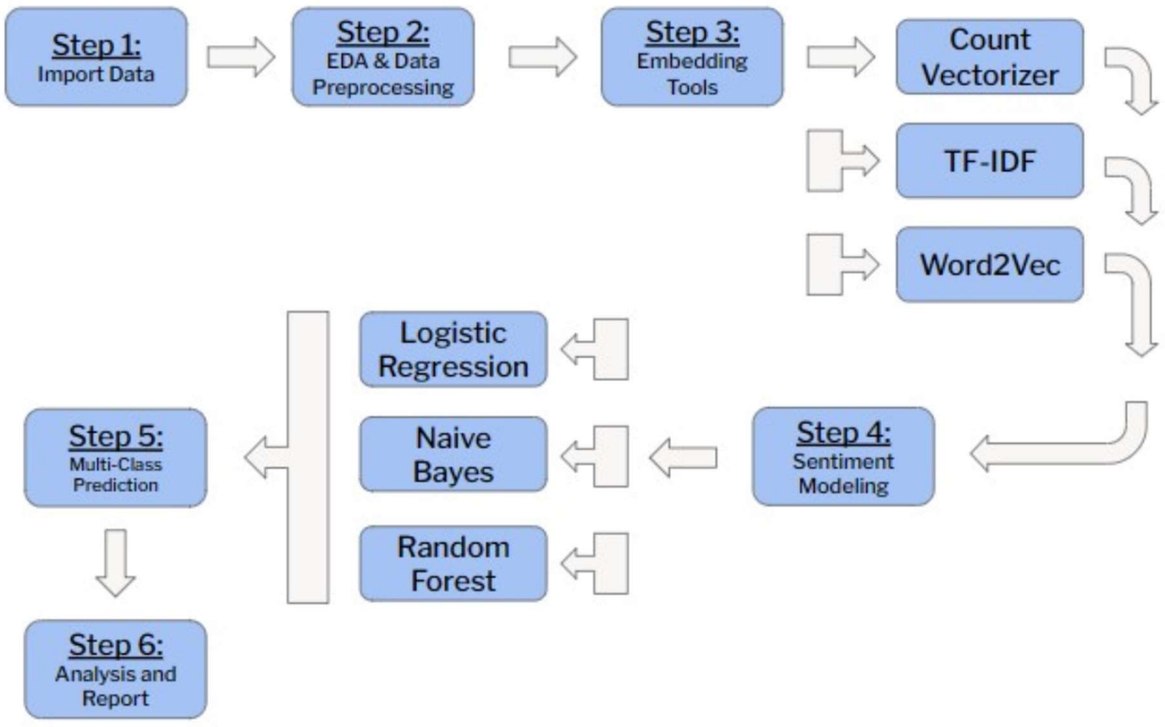
Sentiment Analysis from Amazon Reviews

Group 2: Carolyn Nina Fiore and Syed Hani Haider

BACKGROUND

Sentiment analysis is the process by which models process text strings to determine the emotion associated with the text. It uses natural language processing, or NLP, and models can range from the very simplistic (associating a single word with positive or negative emotion) to the very complex (for example, identifying sarcasm or contradictory statements). Applications for sentiment analysis include analysis of public opinion, publications (anything from news articles to tweets), to optimizing automated processes like recommender systems, customer service bots, and more. The dataset is from Stanford University[1] and consists of 28,978 rows with 8 feature variables, detailing various Amazon reviews. The combination of text and user rating in the dataset can be used as a proxy for sentiment. A high rating is a positive review, and therefore can be used as a "positive" label for that text string.

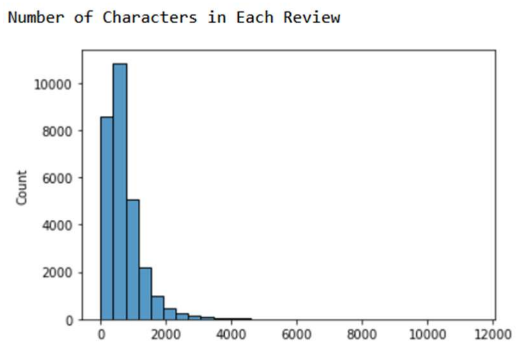
APPROACH

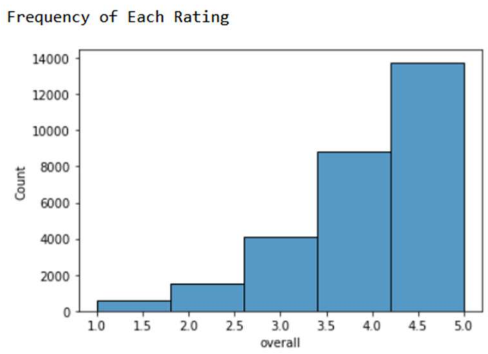


The project workflow is broken into 6 steps: Data Importing, Exploratory Data Analysis (EDA), Text Embedding, Modeling, Three-Class Embedding and Modeling, and Analysis. The initial round of embedding and modeling was conducted using binary sentiment analysis, separating the reviews into "positive" and "negative" categories. The multiclass embedding adds a third, "neutral" class ordered between positive and negative. Three embedding techniques, Count Vectorizer, Word2Vec, and TF-IDF, were used for both binary and multiclass embedding, and three algorithms were selected for the modeling: Logistic Regression, Naïve Bayes, and Random Forest. The project was successful, resulting in the completion of all goal milestones.

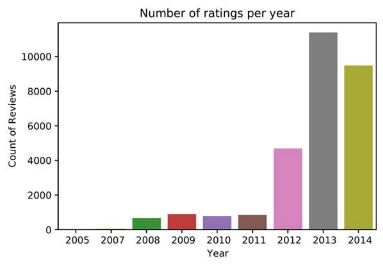
LOW-RISK GOAL (SUCCESS)

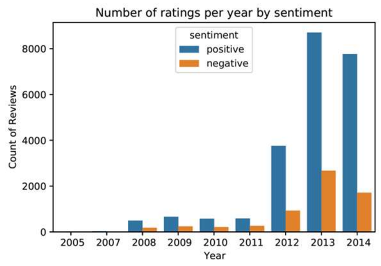
The low-risk goal included learning and executing all the tasks to prepare our dataset for embedding. Key tasks were to tidy the dataset, conduct EDA, and preprocess the text. The missing data were only in the reviewers' names and comprised less than 1% of that column, which was not relevant to the project and therefore did not require reducing the number of samples. The dataset is visibly imbalanced, with the 'positive' label (rating 4-5) vastly overrepresented. Due to the size of the dataset, the underrepresented class (rating 1-3) still contains thousands of reviews; the imbalanced nature led to the use of F1 score rather than accuracy as a measure of algorithm results.

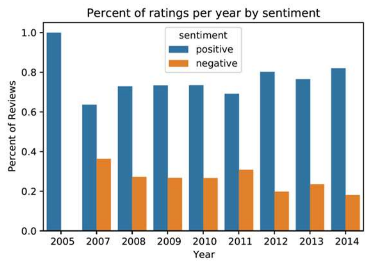




Additionally, the longevity of the data was explored due to possible changes in the way users interact with the platform over time. Amazon use was vastly different from 2005 to 2014; the goal was to determine how the data collection changed over time. As seen below, the number of reviews increased exponentially from 2011-2013. However, when viewing the breakdown by sentiment, both positive and negative grow and shrink with the overall changes. The percentage breakdown below right shows that with the exception of 2005, which has only 1 review, the positive reviews are consistently 60-80% and negative reviews consistently 20-40% of all reviews. The percent change per year is relatively small.







The desired output was a processed dataset that was ready for text embedding. Multiple functions were created to standardize the text, ensuring that the sentiment analysis tools would not get confused by contractions, accented characters, etc. Some of the pre-processing steps performed were: removing punctuation, removing accented characters and converting the characters into ASCII (é → e), removing URLs, removing stop words ('of', 'and', 'it', etc.), converting into lowercase (E → e), removing special characters ('@', '#', '*', etc.), expanding contractions to original ('don't' → 'do not'). These steps enabled all reviews to be standardized before Tokenization, Stemming, and Lemmatization were conducted.

skin
product
serum
result
help
day
one
good
work
regenerating
much
smooth
perfect
using

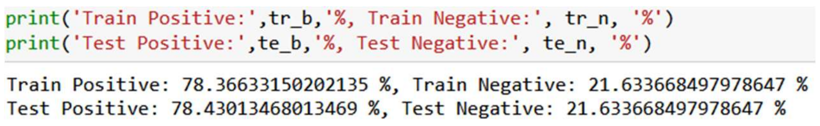
WordCloud After Preprocessing Data

Processing Tools:

- **Tokenization** is a natural language processing tool that takes input text paragraphs and sentences and splits the text into the smallest units (tokens) in order to process, transform, or analyze them. The tokens help the model interpret text by analyzing the sequence of words and sentences, creating context.

- **Stemming** reduces all variations of a word down to their root word. This way, separate words such as ‘variation,’ ‘variety,’ ‘varieties,’ and ‘varied’ can be stemmed to the non-word root ‘vari.’ This reduces data redundancy and reduces inefficiencies in the model.
- **Lemmatization** is another tool used in NLP to reduce words to their root. As opposed to stemming, lemmatization conducts morphological analysis using a complete vocabulary. This can reduce errors a stemmer might get from combining ‘operational,’ ‘operating,’ and ‘operative,’ all of which have the same *stem* root ‘operati’ but which have very different linguistic function and meaning from each other.

The resultant data was reviewed to ensure that the text strings are sufficiently processed for modeling and to ensure a well-stratified (although due to the data, imbalanced) train-test splits.



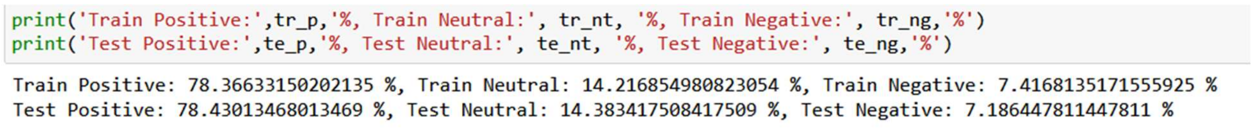
```
print('Train Positive:',tr_b,'%, Train Negative:', tr_n, '%')
print('Test Positive:',te_b,'%, Test Negative:', te_n, '%')

Train Positive: 78.36633150202135 %, Train Negative: 21.633668497978647 %
Test Positive: 78.43013468013469 %, Test Negative: 21.633668497978647 %
```

Train-Test Stratification (Binary)

MEDIUM-RISK GOAL (SUCCESS)

The medium-risk goal is to perform 3 different embedding techniques on 3 different models to conduct binary classification. We will compare the performance of Count Vectorizer, TF - IDF, and Word 2 Vec with Logistic Regression, Naive Bayes, and Random Forest. For the binary classification we will use ratings of 1-3 as the negative classification and 4-5 as the positive classification.



```
print('Train Positive:',tr_p,'%, Train Neutral:', tr_nt, '%, Train Negative:', tr_ng,'%')
print('Test Positive:',te_p,'%, Test Neutral:', te_nt, '%, Test Negative:', te_ng,'%')

Train Positive: 78.36633150202135 %, Train Neutral: 14.216854980823054 %, Train Negative: 7.4168135171555925 %
Test Positive: 78.43013468013469 %, Test Neutral: 14.383417508417509 %, Test Negative: 7.186447811447811 %
```

Train-Test Stratification (Multiclass)

Embedding Tools:

- **Count vectorizer** takes in the text string and counts how many times each word is used in the text. It returns a vector whose length is the number of unique words in the text, and each index is a word, with the value corresponding to how many times it occurred. This weights the term frequency for importance.
- **TF - IDF**, which stands for Term Frequency - Inverse Document Frequency, takes any given word within its text and quantifies its importance. To calculate this, it uses term frequency, which is the count of occurrences of the word in the text, multiplied by inverse document frequency, how often it occurs in other texts, for its weight. The algorithms will use these weights to generate their sentiment predictions.
- **Word 2 Vec** assigns each word a numerical value. Words that are similar, such as dog, puppy, hound, etc., will have similar values. Part of the way this is determined is that they have similar proximity words, such as fluffy, cute, fur, or bark. Reviews with similar vectors are more likely to have the same sentiment.

Algorithms:

- **Logistic regression** generates a prediction for the sample label by applying the logistic sigmoid function to weighted input values to estimate probability of the event’s occurrence.
- **Multinomial Naive Bayes** creates different tags for the text and then uses the Bayes Theorem to determine the likelihood of all the tags for a given text. Once it has calculated the possibilities, it returns the tag with the highest likelihood value.
- **Random Forest** is a classification algorithm consisting of many decision trees. It uses bagging and feature randomness when building each individual tree to create an uncorrelated forest of trees whose

prediction by committee is more accurate than that of individual trees.

Although Random Forest produced the best F1 score using Word2Vec embedding, overall Logistic Regression outperformed it for the TF-IDF and Count Vectorizer embeddings, both with higher F1 scores than the Word2Vec-Random Forest combination. Therefore, for the binary class, TF-IDF-Logistic Regression is the top performing combination with an F1 score of 84%.

BINARY (POS/NEG)	W2V	TF-IDF	C. V.
Logistic Regression	78%	$\boxed{84\%}$	$\boxed{83\%}$
Naïve Bayes	22%	78%	82%
Random Forest	$\boxed{80\%}$	68%	79%

Results of Binary Algorithms (F1 Score)

Excellent Failure:

The low results for the combination of Word2Vec and Naïve Bayes warranted further examination to determine why it performed so poorly. The confusion matrix showed that Naïve Bayes had failed to predict any test samples into the 'negative' category. When viewed across the various embedding tools, this issue was systemic to the implementation of Naïve Bayes on this dataset, including the later iterations of multiclass sentiment analysis during the high-risk goal. For this reason, it was concluded that Naïve Bayes was a poor model to predict labels on this dataset. One possible reason is that, since similar text is used for reviews of various sentiments, the recurrence of these words may be causing confusion as the algorithm attempts to assign tags to the sample reviews.

HIGH-RISK GOAL (SUCCESS)

The high-risk goal was to attempt to replicate the methodology of a 2022 paper by Vernikou, Lyras, and Kanavos[2] on this dataset. Their research resulted in higher performance metrics when using three-class sentiment analysis compared with two-class. For this dataset, "negative" label is assigned to ratings of 1-2, "neutral" to ratings of 3, and "positive" for ratings from 4-5. This phase successfully conducted the multiclass embedding using all three tools and all three algorithms. While Naïve Bayes implementations did attempt to assign labels to the neutral and negative classes, the amount was 42 out of 5760, or 0.7%. Across all three multiclass text embeddings, Logistic Regression had the highest performance. Although two Wave2Vec-algorithm combinations did produce F1 score improvement with multiclass embedding (W2V-LR: +2%, W2V-NB: +55%), the other combinations did not produce increased performance, nor did any of the multiclass embedding-algorithm combinations outperform the binary TF-IDF-LR combination that resulted in F1 score of 84%.

MULTI (POS/NEU/NEG)	W2V	TF-IDF	C. V.
Logistic Regression	$\boxed{80\%}$	$\boxed{80\%}$	$\boxed{80\%}$
Naïve Bayes	79%	79%	77%
Random Forest	79%	78%	78%

Results of Multiclass Algorithms (F1 Score)

IMPACT

Ultimately, while the methodology from Vernikou, Lyras, and Kanavos was reproduced successfully and a few similar increases were noted, there was not sufficient improvement to support the conclusions of their paper. It is likely that part of the variance is based on the difference in emotional subtext between reviews and tweets; Twitter users are more likely to use words connoting extreme emotion debating about COVID-19 than Amazon users reviewing shampoo. Therefore, it is possible that the linguistic separation between the classes of sentiment is smaller in this dataset than the COVID-19 dataset.

[1]Amazon Reviews Dataset [April 26, 2016]. Stanford University. [Online] available: http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Beauty_10.json.gz [accessed February 26, 2023].
[2]Vernikou, S., Lyras, S. and A. Kanavos. "Multiclass sentiment analysis on COVID-19-related tweets using deep learning models" [August 6, 2022]. Neural Computational Application 34(22). [Online] available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9362523/ [accessed February 23, 2023].
[3]Codebase can be found at https://github.com/insomninina/ds5500/blob/main/project2/.