Example: <http://cs229.stanford.edu/proj2018/report/122.pdf>

Final Research Report DUE a week after the last day of class by 11:59PM EST:

A final research report (double-spaced in Arial 12 font) is required and needs to include the following:

Abstract: Paragraph synopsis of research

Introduction: Description of your project

Data: Description of the data used

Methodology: Description of technology used

Results: Summary of key findings

Conclusion: Restate what problem or question your project addresses

References: Citations, APA style

# Abstract:

Amazon Customer Reviews is one of Amazon's iconic products, which provide a place for customers to write reviews and provide a star rating. We used a subset of amazon customer reviews- amazon digital e-books purchase dataset and tried to explore whether reviews can help us understand the star ratings of e-books. By using sentiment analysis, topic modeling, logistic regression and classification models, we concluded that sentiment analysis score on reviews can be used to predict star ratings, and using topic probabilities and embeddings as representations of each review are strong predictors of the review rating category.

# Introduction

[more info on dataset, our interest and research questions; I just listed our existing 3 research questions here]

We have 3 research questions:

1. Can we use classification models on sentiment scores to predict star ratings?
2. What is the correlation between the polarity of reviews and star ratings?
3. Can topic probabilities and custom embeddings be used together to predict review rating category (positive, neutral, negative)?

* Data (Connor)
  + Preprocessing
* Methodology
  + Sentiment analysis + Predictive Analysis (Mengting, Xingchen)
  + Topic modeling, Embeddings, Classification Modeling- Melissa
  + (Exploratory Analysis- Danielle?
* Results
  + Topic modeling, Embeddings, Classification Modeling- Melissa
  + T M (Melissa, Danielle)?
  + SA + PA (Mengting, Xingchen)

# Data Wranglings:

We utilized an existing aggregated dataset of Amazon products reviews collected by Hugging Face Co. and made available to the public [ref]. The dataset was constructed to be representative of customer evaluations and opinions with various geographical regions of generating and promotional intents or bias. Due to limited computational resource, though the original dataset has over 130 million reviews, we confined our data into the e-book reviews category and randomly resampled 51017 entries out of the whole dataset, which is about 1% of the entirety of the e-book review data collection.

There are 17 variables in each entry: marketplace, customer\_id, review\_id, product\_id, product\_parent, product\_title, product\_category, star\_rating, helpful\_votes, total\_vote, vine, verified\_purchase, review\_headline, review\_body, review\_date, review\_category, and review\_body\_clean. Each variable by its name is hopefully self-exoplanetary of what it records. And we will talk more about each of the variables when we use them in the models we employed. But, in general, we focus on the “review\_body” and the “star\_rating”.

The “star\_rating” ranges from one to five (inclusive), and we re-categorized the ratings into three groups: Positive, Negative, and Neutral, whereas stars of one-to-two are considered Negative; three as neutral; and four to five stars as Positive. We created a new column named “Category” to denote the labels.

The “review\_body” contains the raw review inputs from customers on the product. This is the main research object of our natural language processing program. Here is an example of a typical entry of “review\_body” is: “Barbarians need love too ! Short stories work well with ebooks. The texts can be printed and read later. They can be read on a laptop. Bravo !” (index 4).

There are 51017 entries of reviews in our subsample and here we treat each of review as independently made.

Some descriptive statistics can be found in Graph [] [] []

We then used the “standard” pipeline we have seen in class to process the reviews. Graph [ref] shows the steps of our pipeline.

The clean corpus took out all punctuations and dis-sensitize cases. We used NLTK package for stop-words removal and the lemmatizing process. And for embeddings, we used Word2Vec. Different vector lengths have been used for data explorations and modeling.

We used the length of 100 to explore the basic structures of our data. For example, the word “true” has the following top similar hits in the vocabulary, with the similarity score reported.

[('faith', 0.7253068685531616),

('compassion', 0.7125968933105469),

('forgiveness', 0.7073450684547424),

('courage', 0.6939508318901062),

('heartbreaking', 0.6890562772750854),

('tragedy', 0.675846517086029),

('grace', 0.6746917963027954),

('truly', 0.6685964465141296),

('touched', 0.6662676334381104),

('sadness', 0.6608010530471802)]

Then we used T-SNE to conduct a dimension reduction analysis to have a general picture of the global as well as local structure of our corpus. A more detailed discussion of T-SNE can be found in the “Methodology” section later in this report.

By implementing T-SNE, we constructed a 2-D display of the lemmatized words. Table [] demonstrate the components value for a few words in the vocabulary.

| Words | First Component | Second Component |
| --- | --- | --- |
| elmore | 1.370997 | -9.716476 |
| leonard | -11.130122 | 31.051918 |
| meet | -5.841982 | 60.820881 |
| cast | -29.960129 | 44.151703 |
| sierra | -3.428209 | 26.553064 |

Picture [] show the entire vocabulary on the reduced-dimension space. Each dot in the graph represents a word in our vocabulary, and the dots are close together in the graph should have a closer sematic connection.

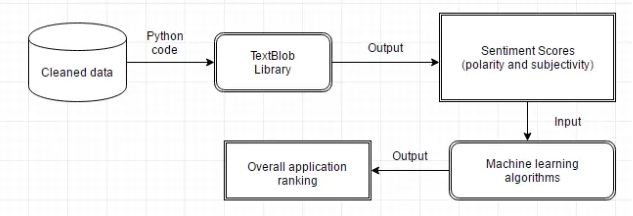
A zoom-in view of the 2-D plane is shown in Picture [ref]. In the bottom of the graph, we can see that “complied” and “database” are close to each other and intuitively, they have connected meanings.

# Methodology:

Sentiment Analysis Methodology

1. TextBlob

TextBlob is an open-source python library for processing textual data. it offers a simple API to access its methods and perform basic NLP tasks. TextBlob performs different operations on textual data such as noun phrase extraction, sentiment analysis, classification, translation, etc. It is built on top of NLTK and Pattern. It is very easy to use and can process the text in a few lines of code. TextBlob can help you start with the NLP tasks.



We can see the output is categorized between two — Polarity and Subjectivity. Polarity is a float value within the range [-1.0 to 1.0] where 0 indicates neutral, +1 indicates a very positive sentiment and -1 represents a very negative sentiment. Subjectivity is a float value within the range [0.0 to 1.0] where 0.0 is very objective and 1.0 is very subjective. Subjective sentences express some personal feelings, views, beliefs, opinions, allegations, desires, beliefs, suspicions, and speculations whereas Objective sentences are factual.

1. Vader Analysis

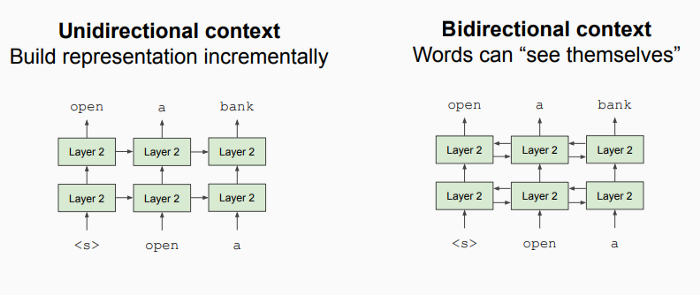
VADER ( Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. It is available in the NLTK package and can be applied directly to unlabeled text data.

VADER sentiment analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text.

For example- Words like ‘love’, ‘enjoy’, ‘happy’, ‘like’ all convey a positive sentiment. Also VADER is intelligent enough to understand the basic context of these words, such as “did not love” as a negative statement. It also understands the emphasis of capitalization and punctuation, such as “ENJOY”.

1. Bert Analysis

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained NLP model developed by Google in 2018. BERT was considered to be the most interesting model to work in deep learning NLP. The model, pre-trained on 2,500 million internet words and 800 million words of Book Corpus, leverages a transformer-based architecture that allows it to train a model that can perform at a SOTA level on various tasks. With the release, Google showcased BERT’s capability on 11 NLP tasks, including Stanford competitive QA dataset.



Classification Methodology:

# Results:

Classification prediction:

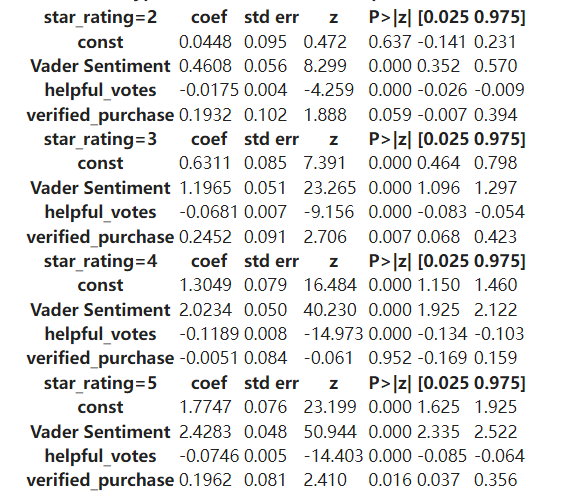
Multiple logistic regression:

After doing sentiment analysis with Vader and Textblob, we are interested in exploring the relationship between polarity score and star ratings.

Our hypothesis is that there is a positive relationship between polarity of reviews and star ratings. The independent variable is polarity score(vader and textblob sentiment analysis results), and the main dependent variable is star rating of digital e-books. Other controlled variables include helpful votes(how many people are in favor of your reviews) and verified purchase(whether or not this is a verified purchase).

1. Multiple logistic regression with Vader analysis

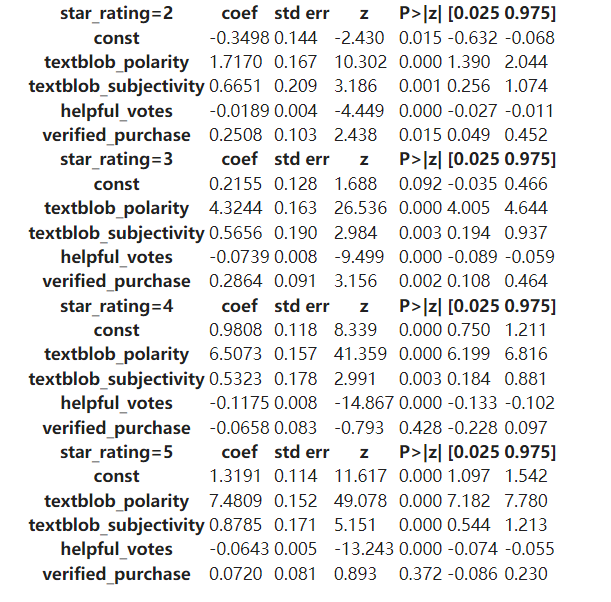
The multiple logistic regression results are shown below. Our reference group is chosen to be "star\_rating=1". From the coefficient of “Vader Sentiment” , we can see how many units in the log of ratio between the probability of star\_rating in other groups vs. the probability of a star\_rating=1 is caused by increasing the polarity by one unit.



Increasing the polarity by one unit will result in an increase by 0.46 units in the log of the ratio between the probability of star\_rating=2 vs. the probability of a star\_rating=1; Increasing the polarity by one unit will result in an increase by 1.20 units in the log of the ratio between the probability of star\_rating=3 vs. the probability of a star\_rating=1; Increasing the polarity by one unit will result in an increase by 2.02 units in the log of the ratio between the probability of star\_rating=4 vs. the probability of a star\_rating=1; Increasing the polarity by one unit will result in an increase by 2.43 units in the log of the ratio between the probability of star\_rating=5 vs. the probability of a star\_rating=1. Therefore, increasing the polarity score of reviews will significantly increase star rating.

1. Multiple logistic regression with Textblob analysis

Similarly, the reference group is chosen to be "star\_rating=1". We add a controlled variable “subjectivity” derived from textblob sentiment analysis, which refers to personal opinion, emotion or judgment.



Increasing the polarity by one unit will result in an increase by 1.72 units in the log of the ratio between the probability of star\_rating=2 vs. the probability of a star\_rating=1; Increasing the polarity by one unit will result in an increase by 4.32 units in the log of the ratio between the probability of star\_rating=3 vs. the probability of a star\_rating=1; Increasing the polarity by one unit will result in an increase by 6.51 units in the log of the ratio between the probability of star\_rating=4 vs. the probability of a star\_rating=1; Increasing the polarity by one unit will result in an increase by 7.48 units in the log of the ratio between the probability of star\_rating=5 vs. the probability of a star\_rating=1. The results are the same with vader sentiment analysis that increasing the polarity score of reviews will significantly increase star rating.

# Conclusion

There is a positive correlation between star rating and polarity score: increasing the polarity score of reviews will significantly increase star rating.

# Melissa sections- methodology and results summaries

**Methodology**

1. **Topic Modeling Methodology**

Topic modeling is a type of unsupervised machine learning that can be used to detect recurrent themes in a body of text. This has applications in classifying text, finding similarities between texts, and in recommendation systems. An algorithm commonly used for this purpose is the Latent Dirichlet Allocation (LDA). Defined from the class 9 slides, LDA is ‘a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of latent topics. Each observed word originates from a topic that we do not directly observe. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities.’ This method can be thought of as similar to clustering, except instead of numerical features defining the clusters, the clusters are defined by tokens (words) [link 1]. The goal is to find the hidden, or latent, structure within the text. This latent structure in the text is defined by a distribution of topics and distribution of words in each topic. This distribution of distributions is known as a dirichlet. Once the dirichlet is discovered, the topics will be allocated to the documents, and the words of the documents will be allocated to the topics. LDA believes that each word in each document comes from a topic, and the topics are selected from a per-document distribution over topics. From the lecture slides and [link 2], the conditional probabilities determined by LDA are as below:

*p(topic t | document d)*

* The proportion of words in a document *d* that are assigned to topic *t*
* Tries to capture how many words belong to the topic *t* for a given document *d*

*p(word w | topic t)*

* Proportion of assignments to topic *t* over all documents that come from this word *w*
* Tries to capture how many documents are in topic *t* because of word *w*

*p(word w with topic t) = p(topic t| document d) \* p(word w| topic t)*

* Probability a word *w* belongs to topic *t*

To evaluate the performance of LDA in defining the latent topics, the coherence of each topic can be calculated. For a single topic, the coherence score measures the degree of semantic similarity between high scoring words in the topic. The c\_v coherence of the model can also be calculated to gauge the overall quality of the model This helps to understand whether or not the topics are cohesive. In addition, the perplexity score can also be computed, which is another indicator of how good the entire LDA model is. To improve model performance, there are various parameters that can be tuned, including number of latent topics, the learning decay, as well as the alpha (document-topic density) and beta (topic-word density). High values of alpha result in documents being composed of more topics, while low values of topics result in documents being composed of fewer topics. High values of Beta result in topics composed of large numbers of words from the entire corpus, while low values of Beta result in topics composed of few numbers of words from the entire corpus[link 3].

* Class slides- <https://courseworks2.columbia.edu/courses/135369/files/folder/slides?preview=12617512>
* Link1- <https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045>
* Link 2- <https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2>
* Link 3- <https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/>

1. **Embeddings Methodology**

From the class slides, Word embeddings are learned representations for text where words that have the same meaning have the same representation. Words are represented as a vector of real numbers, typically in dimensions of 50, 100 or 300. These vectors enable comparisons of word similarities, machine translation, named entity recognitions, relation extraction, sentiment analysis, and coreference resolution. The dimensions used to describe a text can be used to relate it to other texts in however many dimensions the embeddings occupy. Texts that have similar vectors are thought to be more similar than those with less similar vectors. A common way to create these embeddings is to use Word2Vec. Word2Vec is a two-layer neural network that takes a corpus as an input and outputs a vector representation of the corpus, according to the vector dimensions specified.

* Class slides- <https://courseworks2.columbia.edu/courses/135369/files/folder/slides?preview=12299521>

1. **Topic Modeling and Embeddings Classification Model Methodology**

Representing each review by the probability of containing each topic, along with the 300 dimensions of embeddings, provided an opportunity for multiclass classification through supervised machine learning. Each review assigned a star rating indicating the reviewer’s satisfaction, from 1 to 5, with 1 being least satisfied and 5 being most satisfied. Categorizing each review into a rating category of positive (4-5 star rating), neutral (3 star rating) or negative (1-2 star rating) simplified the task from a 5-category classification to a 3-category classification. These categories were also deemed more meaningful than five separate categories, where it can be difficult to determine the recurrent difference between ratings separated by one star. These three rating categories, positive, neutral and negative, were used as the dependent variable. The topic probabilities and embeddings used to describe each review were used as the independent variables.

Another factor to consider during the classification was the imbalance of the data. The data was heavily skewed in favor of positive reviews (nearly 34,000), with much smaller numbers of neutral and negative reviews (about 4,000 and 3,000 respectively). To avoid bias in the model, the training data was upsampled using SMOTE (Synthetic Minority Oversampling Technique) to provide the model with equal counts of each review category on which to train. This upsampling generated synthetic data of the neutral and negative review categories, making the number of reviews in each category in the training data equal, to just under 34,000 each.

Three types of classification algorithms were used, including Random Forest, K-Nearest Neighbors, and Multilayer Perceptron Networks. A brief description of each classification model’s methodology is below:

* Random Forest Classifier is an ensemble method that includes multiple decision trees, built on a bootstrapped sample of the training data. Each tree is composed of internal nodes, which are split, and leaf nodes, which are terminal nodes and do not split. Each time a split in one of these decision trees is made, a random subset of the total features are selected, with the split using one of those features in the subset. The split is greedy, meaning that the decision for each split is based on the potential increase in purity for that specific step, and not for future splits. This process repeats for all trees, with the goal of increasing the purity of samples in each node with each split. Some of the tuning parameters for Random Forest include the number of trees (called estimators) in the Forest, the maximum depth of each tree, the criteria to motivate or prevent a split (either minimum observations required to perform the split, or minimum observations permitted in a leaf node), maximum number of leaf nodes, and splitting criteria.
* Random forest slides- <https://courseworks2.columbia.edu/courses/135307/files/folder/Slides?preview=12214979>
* K-Nearest Neighbors Classifier is a simple algorithm where observations are classified based on the classifications of their most similar k neighbors. The similarity to other observations in the training data is calculated by a distance measurement, such as euclidean distance. Once the k nearest neighbors to an observation are determined, the majority vote of these neighbors is taken to assign the category to the observation in question. There is only one tuning parameter, that is, the number of k neighbors to consider when determining the majority vote.
* The Multilayer Perceptron Network Classifier is a type of neural network. These networks can learn non-linear relationships and discover other relationships that other models may miss. However, they require a large number of parameters, including the number of hidden layers, hidden nodes in the hidden layers, and iterations. These networks are composed of an input layer, which takes in data flattened into a vector, hidden layer(s), and an output layer. The nodes in the hidden layer(s) use nonlinear activation functions. These networks are fully connected, meaning that every node in a given layer is connected to all the nodes in the next layer. The strength of the connection between two nodes is called a weight, which is initially randomly assigned. The data is fed forward through the network, where the input values are multiplied by these initial weights and then summed. Each layer also includes biases which are constants, and adjusts the output along with the weighted sums of the nodes. A technique called backpropagation is then used to adjust these weights to improve model performance. Some of the tuning parameters for these networks include the number of hidden nodes in each hidden layer, the learning rate, the batch size of training data, and the number of epochs (number of times the algorithm will work through the entirety of the training data).

**Results**

1. **Topic Modeling Results**

As an alternate representation of the reviews, each product review can be represented by topic probabilities. After removing non alphanumeric characters, removing words comprised of fewer than three characters, removing custom stopwords, and performing lemmatization, Latent Dirichlet Allocation (LDA) was used to define the number of recurrent topics found within the body of the reviews. Multiple iterations of the LDA algorithm were run to test different possibilities. The LDA algorithm provides parameters for minimum and maximum token thresholds. These were used to ensure that only meaningful tokens, which appeared a reasonable number of times within all of the reviews, were used to define the topics. Different values for the token parameters were applied, including:

| Iteration | Tokenization | Maximum Token Threshold\* | Minimum Token Threshold\*\* | Ideal Topic Count\*\*\* | C\_v Coherence Score | C\_v Plot |
| --- | --- | --- | --- | --- | --- | --- |
| Model 1 | Unigram | 500 | 0.75 | 5 | 0.4170 | Appendix |
| Model 2 | Unigram | 500 | 0.50 | 5 | 0.4252 |  |
| Model 3 | Unigram | 5000 | 0.50 | n/a | 0.3710 |  |
| Model 4 | Unigram | n/a | 0.75 | 7 | 0.4220 |  |

*\*Exclude tokens that appear in more than x reviews*

*\*Exclude tokens that appear in less than x% of total reviews*

*\*\*\*Topic counts of 4 through 15 were tested*

Although Model 2 produced the highest coherence score, which indicates similarity among words associated with each topic, Model 4 produced the most interpretable results. The topics produced by Model 2 were not as clearly differentiable as those produced by Model 4.

For Model 2, some of the top words associated with each topic include:

* Topic 1: series, love, character, next, wait
* Topic 2: time, know, think, make, going
* Topic 3: story, character, little, romance, part
* Topic 4: life, love, family, friend, real
* Topic 5: recommend, author, interesting, reader, easy

For Model 4, some of the top words associated with each topic include:

* Topic 1: kindle, easy, idea, use, work
* Topic 2: character, novel, plot, interesting, action, mystery
* Topic 3: story, love, romance, heart, character
* Topic 4: book, one, would, think, know, time
* Topic 5: life, people, child, world, god, history, time
* Topic 6: family, year, man, woman, friend, mother
* Topic 7: book, series, one, next, loved, wait, recommend

From the associated words, it might be reasonable to infer that from the topics produced by Model 4, the first might be related to (Kindle) e-books and their ease of use, the second might be related to the plot of action or mystery books, the third might be related to romance stories, the fifth might be related to philosophy, the sixth might be related to family, and the seventh might be related to series of books (as opposed to single book releases). It is not immediately clear what is being described in topic 4.

Compared to the topics produced by Model 2, those from Model 4 are clearer. Topic 1 in Model 2 might be related to series of books, topic 3 might be related to romance stories, and topic 5 might be related to books that reviewers would recommend to others. It is not immediately clear what is being described in topics 2 and 4. This further emphasized that in machine learning, it is important to consider the interpretability of models in addition to their objective results. Objectively, one could say that Model 2 had a slightly better performance than Model 4, based on coherence score. However, interpretability of the topics produced by Model 4 seems clearer. This same intuition guided the number of topics tested in the models, where anything less than 4 might not capture topic variability and anything greater than 15 might be too specific to certain reviews.

1. **Embeddings Results**

Embeddings were intended to serve as multidimensional representations of each review. To create the embeddings, there are multiple parameters that can be customized, including the minimum count in the corpus per token, the embeddings vector size and neighboring word window size. For the embeddings created here, the minimum count in the corpus for each token was 1, the embeddings vector size was 300, and the neighboring word window size was 5. Together, these parameters intended to extract information about each review, and how it relates to the other reviews, in 300 dimensions. Using pre-trained embeddings related to product reviews were considered, but there was concern that these might not be specific enough to either e-books or even Amazon purchases. For this reason, it was determined that training custom embeddings would provide the most relevant information.

1. **Topic Modeling and Embeddings Classification Model Results**

Using the dataset structure with the topic probabilities and the embeddings dimensions as independent variables and the review rating categories as the dependent variable, three different classification modeling approaches were tried. Random Forest, K-Nearest Neighbors, and Multilayer Perceptron Networks were tuned, trained and cross-validated. The optimal models found from GridSearchCV were also applied to the test sets to gauge their performances.

Below are the training set accuracy, test set accuracy, and test set weighted f1 scores of each of the three models. The optimal parameters for each, discovered during GridSearchCV, are also noted.

| Model | Optimal Model Parameters | Training Set Accuracy | Test Set Accuracy | Test Set Weighted F1 Score |
| --- | --- | --- | --- | --- |
| Random Forest | 100 estimators, no max tree depth | 0.9401 | 0.8012 | 0.8023 |
| K-Nearest Neighbors | 5 neighbors | 0.8430 | 0.5387 | 0.6184 |
| Multilayer Perceptron Network | 3 hidden layers- hidden nodes per layer (500, 500, 100) | 0.9160 | 0.7876 | 0.7864 |

From the results above, the Random Forest had the strongest performance among all of the models. The Multilayer Perceptron Network is the second best performing model, not too much of a difference from the Random Forest. The K-Nearest Neighbors model performed the worst out of the three models tested. The decline in accuracy from training set to test set in all of the models is indicative of overfitting to the training data. This occurs when a model learns the training set too well and fails to generalize to new and unseen data, such as that of the test set. It seems that the K-Nearest Neighbors model may have the worst degree of overfitting due to the large drop from training set accuracy to test set accuracy. The weighted f1 scores for the Random Forest and Multilayer Perceptron Network are fairly similar, indicating that these models both have adequate balances of precision (true positives/actual results) and recall (true positives/model predictions) when exposed to test data. Although the class disparity in the training data was resampled using SMOTE, the high weighted f1 sores for these models indicate that these models are able to recognize instances of the different review rating clases in the test set, which contains mostly positive review ratings.

On the contrary, the K-Nearest Neighbors model displays the worst test set performance in regards to both accuracy and weighted f1 score.

Due to these results, to predict the review rating category of new reviews, it would be best to use the Random Forest model. This model performs well in terms of accuracy and does a good job of recognizing instances of each class, despite their differences in frequency. To use this Random Forest model, new review data would have to first be processed in the same manner that the data used to train the model was- by considering the topic probabilities of each of the seven topics found using LDA in each new review, and transforming the text of the new reviews into embeddings of the same dimensions.