Applying Natural Language Processing to Goodreads poetry book review dataset

Abel Thomas

Divya Thomas

Introduction

Our project is centered around the idea of books and analyzing trends with books and literature. We focussed on Poetry books for our analysis. Our project is divided into two parts

The idea for Part 1 of our project was to answer the questions that someone who is going to open a poetry book store would be curious about. To do this, we performed an exploratory analysis to answer questions such as which are the most popular types of books? Who are the more preferred publishers? Which are the most common words used in book reviews? Is there any relation between the number of pages and ratings?

In Part 2 of our project, we used the detailed book reviews to perform sentiment analysis on the review text and compare the sentiment that is extracted to the actual rating given by the user. We also used the review text as input in a variety of classification models to classify the text and predict the user rating

Part 1: Performing Exploration Analysis to Gain Insights on the Data Set

Problem:

In this first section, we will aim to perform some data analysis to better understand the data set that we currently have.

Some interesting questions that we will be looking to resolve in this section are:

1. What are the most popular poems?
2. Who were the most popular publishers?
3. What is the most preferred format?
4. What language medium is the most common?
5. How does the distribution of the ratings look like?
6. Are there correlations between ratings and the number of reviews?

Dataset:

The dataset that we are using for this project is from the UCSD Book Graph. The dataset is from the goodreads.com website and was scraped in 2017 from the public shelves of users. This dataset was then later updated in May 2019 and cleaned up to remove duplicates. Since the main dataset contains over 2.36M books, we decided to explore a specific genre of books. The genre that we decided to explore was 'Poetry'. This dataset (goodreads_books_poetry.json.gz) consists of information of 36,514 books. The information was divided up into 29 fields ranging from ISBN to the number of ratings.

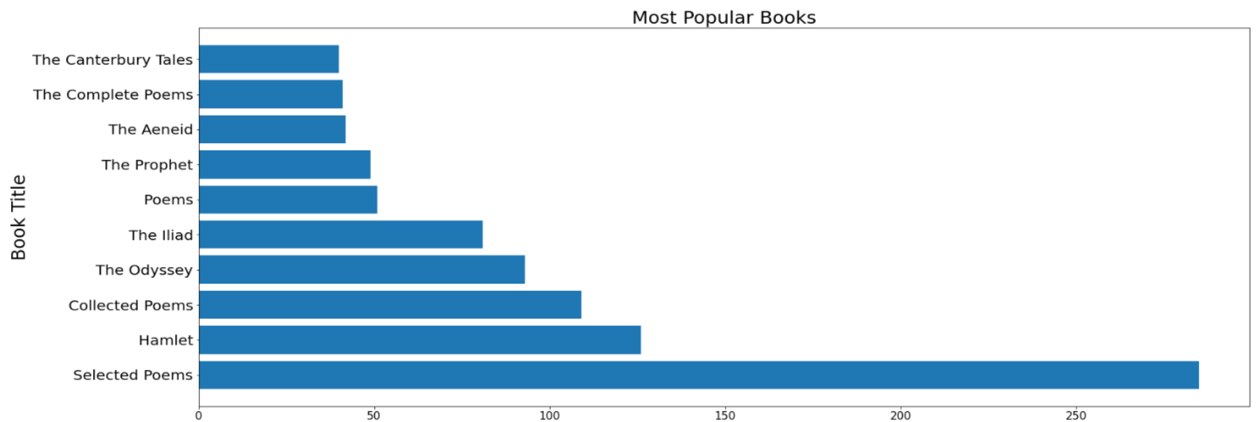
Data Cleaning:

The dataset here was relatively clean since it had been preprocessed. However, I still cleaned it to ensure that there were no rows that had NaN values, as it would muddy the end results.

Exploratory Data Analysis and Discussion

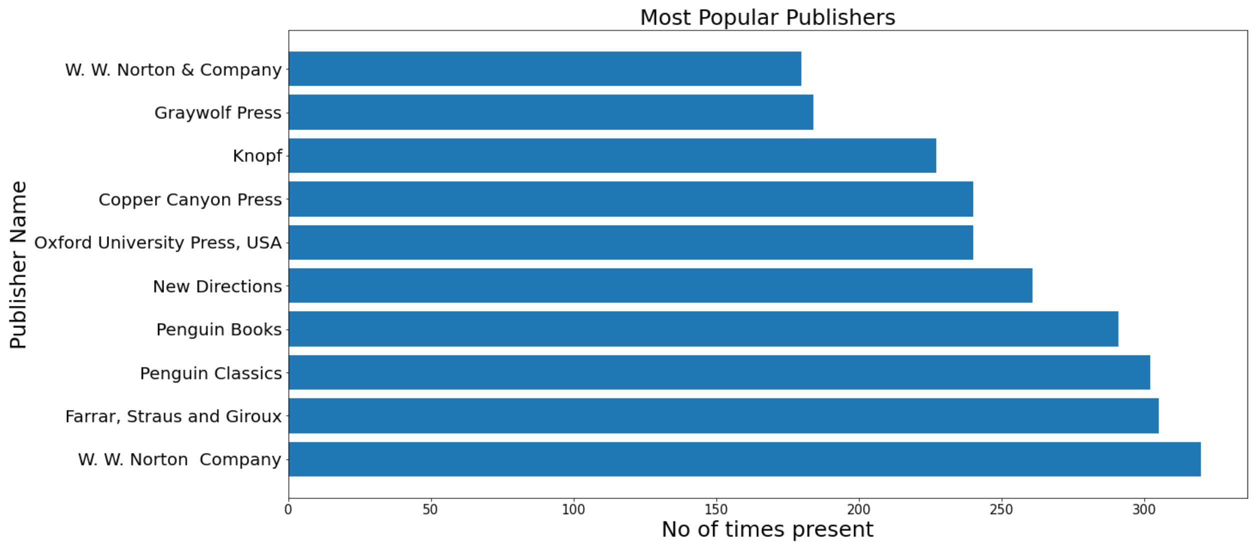
The Most Popular Poems:

To find what the most popular poem was, we had to use the `value_counts()` function which returns a pandas series containing counts of unique values. To get the most popular poems, I extracted the 'title' series and applied value_counts() on it. The following plot was obtained, from which we can understand that the most popular poem book was of the 'Selected Poems' type.



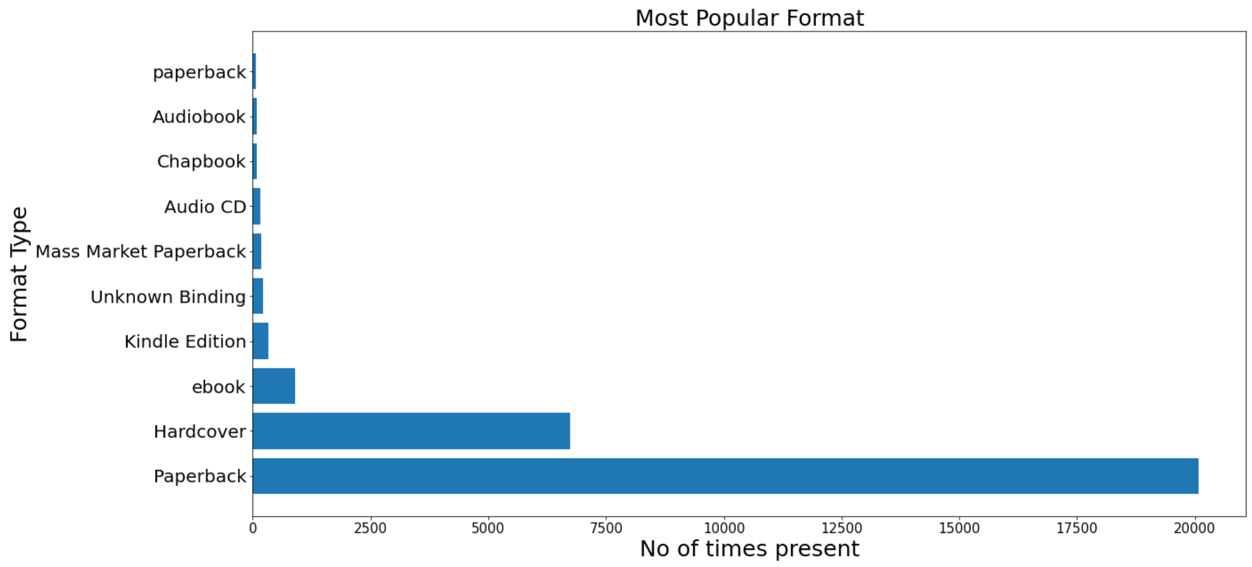
The Most Popular Publishers:

Similarly as above, to obtain the most popular publishers, I extracted a pandas series of ‘publishers’ from the original data frame. However, there were some entries for which the publisher was left as ‘ ‘, I then cleaned up the dataset to remove this using the `replace()` function and then applied the value_counts() function on it. From the obtained graph we can see that there are no majorly preferred publishers and that the books printed by them were relatively evenly split.



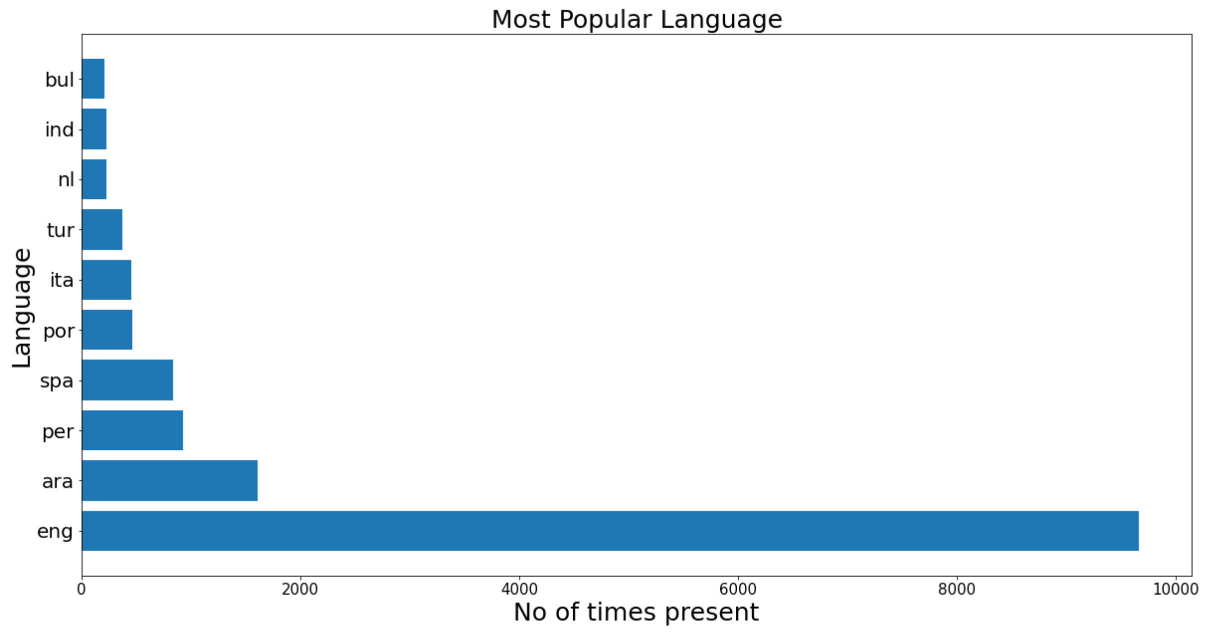
The Most Preferred Format:

To obtain the preferred format, the panda series of ‘format’ was extracted from the original data frame with dataset cleaning being once again performed before the value_counts() function was applied on it. From the obtained graph, we can see that for poem books, physical formats such as hardcover and paperback are mainly preferred over electronic formats.



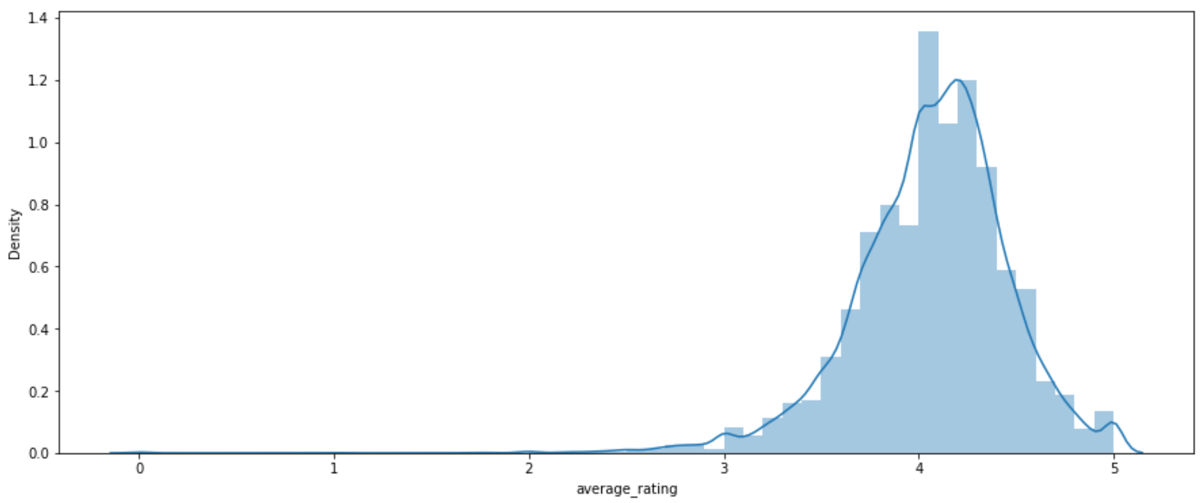
The Most Popular Language Medium:

To obtain the preferred format, the panda series of ‘format’ was extracted from the original data frame with dataset cleaning being once again performed. However, this time there is a unique problem that arises due to the different language codes that English has, ie: both en-US and en-GB are both English mediums and hence should be grouped together. To perform this, I utilized regex along with panda’s replace function to match and replace any English dialects under one common group. After this, I applied the value_counts() function and obtained the following graph. From this graph we can see that majority of the poems are written in English, followed by Arabic and Persian, which is not surprising since there were the main languages in which a lot of poems were written in during ancient times.



Distribution of the Average Book Rating:

To explore how the average book rating was distributed, a density plot was the ideal graph to examine. First, we had to calculate the average book rating for each book, this was easy since it was provided within the dataset. To then plot this, I used `distplot` from the `seaborn` library. From this, we can

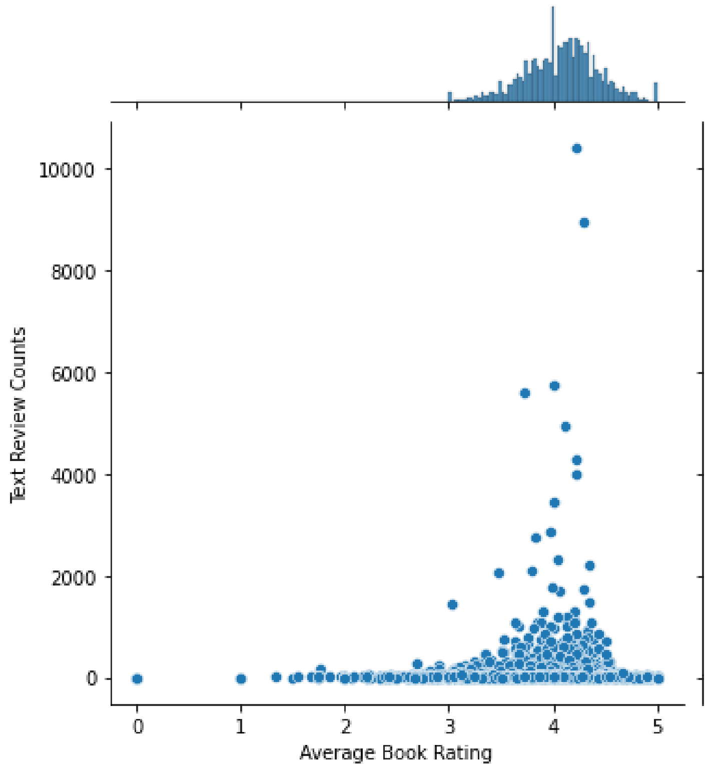


see that the average book rating was normally distributed with the mean average book rating being a 4/5.

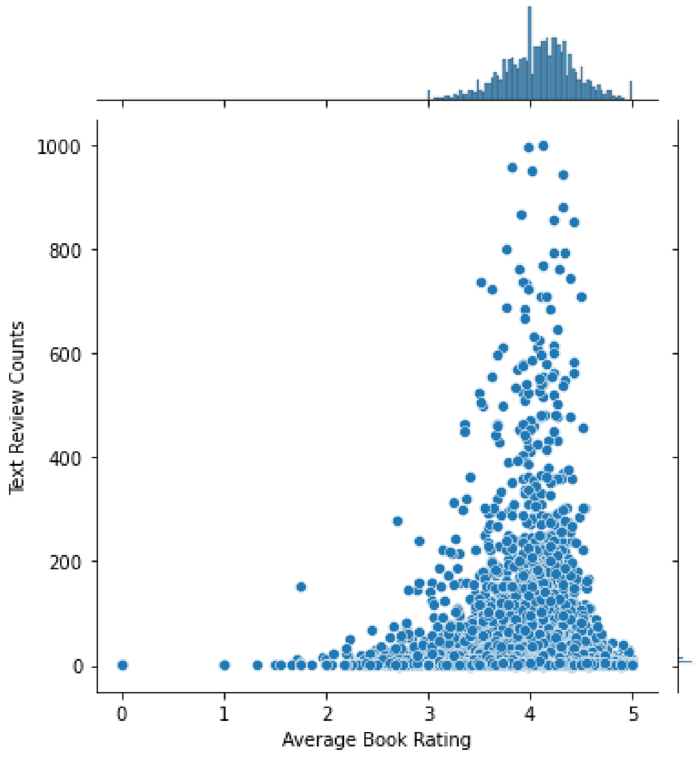
Correlation between Rating and Number of Text Reviews:

To check if there is a relationship between Rating and Text Review Counts, I decided to use I used `joinplot` from the `seaborn` library. Joinplot allows you to plot two variables with bivariate graphs allowing us to visualize and see if there is any relationship between the two.

In this plot, we can see that there seems to be some relation, but due to a few high text review counts, it is very difficult to examine the lower values. As a result, we have decided to exclude these outliers and plot for books that have Text Review Counts < 1000.



Once we exclude these outlying values, we are able to see that there is in fact a strong relationship between them. We can see that as the number of Text Review Counts increases, the more it seems that the average book rating seems to tend towards a value of 4. This would also explain why the mean of the average book ratings of all poem books is around 4.



Part II: Using Natural Language Processing on Book Reviews for Classification

Dataset

For this section, we used the Poetry book review dataset from UCSD Book Graph. The dataset (**goodreads_reviews_poetry.json.gz**) contains 154,555 detailed reviews on 36514 different books. In addition to the text review, the dataset also contained a numerical rating by the user for the book. The numerical rating ranged from 0-5 with 0 being the lowest rating and 5 being the highest rating. The dataset contained reviews in a variety of languages not just English.

Problem Definition

These were our goals for this section of the project:

1. Perform sentiment analysis on the review text and compare the sentiment that is extracted to the actual rating given by the user.
2. Use the review text as input in classification models to classify the text and predict the user rating

Techniques Used

1. Data preprocessing techniques were used to clean the data to normalize it and remove noise
2. Sentiment Analysis: We did this in 2 methods: by computing the polarity of the reviews and also using VADER which is a text sentiment analysis model part of the NLTK module.
3. Text classification: In order to train a machine learning model on text, we first need to tokenize the words and encode them into vectors of numbers so as to recognize the important words. We used 2 schemes. The first was a simple encoding (TF-IDF) and the second was a pre-computed word-based embedding (Doc2Vec) as research suggests perform a bit better. After feature engineering and selection we used a variety of classifiers for our predictions. These were multinomial logistic regression, Random Forest, Multinomial Naive Bayes classifier, Decision Tree Classifier, K Nearest Neighbours, Gradient Boosting and Neural Networks.

Data Preprocessing & Cleaning

- Since the dataset contained 154,555 records, a sample with frac =0,2 and without replacement was taken from the dataset in the interest of fast computation. This created a dataset with 30,911 records.
- After ensuring there was no missing data, we removed non-English text reviews from the data set using the corpora of English words in NLTK (nltk.corpus.words).

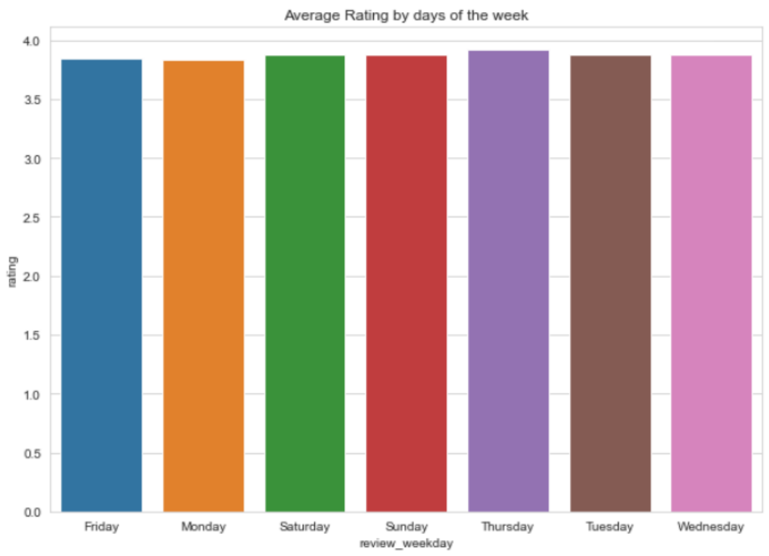
- To clean the remaining text reviews, we created a cleaner function that performed the following parts:
    1. Converting all characters to lowercase
    2. Expanded contractions (n’t to not)
    3. Tokenized the text and removed punctuation
    4. Removed alphanumeric words
    5. Removed stop words
    6. Removed empty tokens
    7. Perform Part-Of-Speech (POS) tagging to tax word to correct part of speech such as noun, verb etc
    8. Lemmatize the text by replacing the word with the one from the wordnet word(from nltk.corpus) corresponding to the POS tag
    9. Remove one letter words
- Remove texts that are just empty strings

After the cleaning, the dataset contained 27727 records.

Basic Dataset Exploration

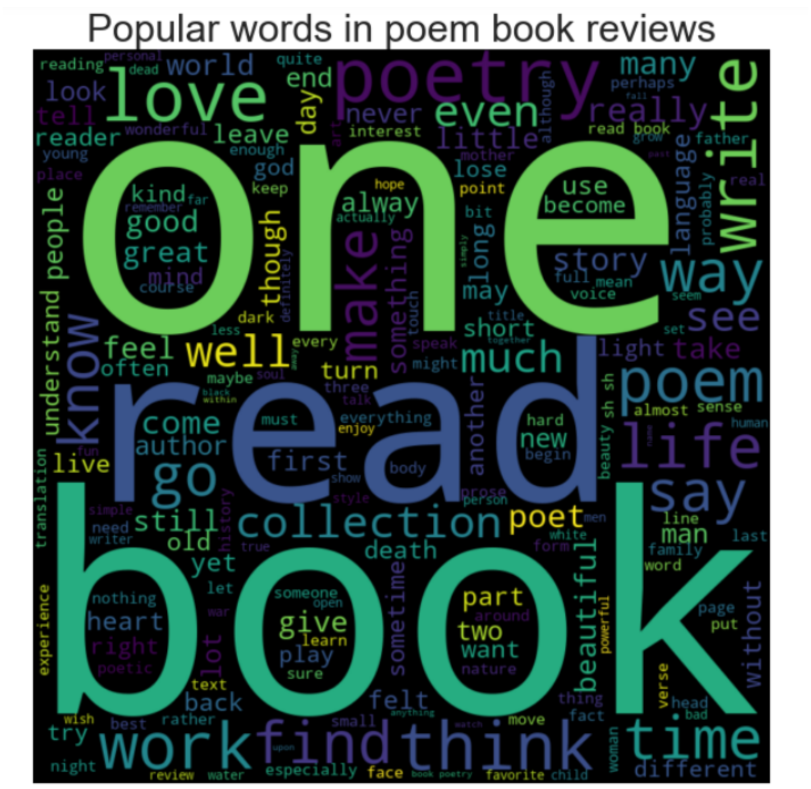
Ratings vs day of the week

We were curious if a user was likely to give a higher rating on any particular day of the week. So we extracted the day of the week when the review was posted and computed the average review rating. As you can see in the bar chart, there does not seem to be any appreciable difference in ratings based on the day of the week it is posted.



Word Cloud

We created a word cloud to find the most common words used in poetry book reviews. This is shown below. As expected most popular words include read, one, book love, poem, etc



Feature Engineering

- Adding a number of characters and number of words: For each text review, we added the number of characters and number of words as we wanted to use this in our sentiment analysis later
- The review ratings ranged from 0-5. I binned them into 3 buckets. Bad (0,1,2); Neutral (3); Good (4,5). I felt that this would help get out more meaningful classification. The dataset was heavily biased towards 4 and 5 ratings and so later when we trained the models we used stratified sampling to ensure this distribution is preserved

Sentiment Analysis

VADER

We used VADER which is a text sentiment analysis model part of the NLTK module to extract sentiment from the text. Vader uses the sentence context and a lexicon of words to return for each review text record a neutrality score, positivity score, negativity score, and a compound score which is an overall score based on the previous three scores. In order to determine how well the model is performing, we looked at the top 10 reviews that got the highest positive sentiment scores and 10 that received the highest negative sentiment scores. We compared these to the actual rating given by the user (Good,Bad, Neutral). We found that the model performed pretty well on interpreting positive reviews but did not perform very well on negative reviews. It interpreted words like “shy”, “bury dead”, “where was this when I was heartbroken”, “powerful and hard” etc as negative sentiment even though it is not used negatively in the review.

Computing Polarity Using Textblob

We decided to use another method to extract the sentiment to see if this might perform better. We computed the polarity using the Textblob package. This gave a polarity score with 1 being most positive and -1 being most negative. Again we looked at the top 10 reviews that got the highest positive sentiment scores and 10 that received the highest negative sentiment scores. We compared these to the actual rating given by the user

(Good, Bad, Neutral). In this case, we found that compared to VADER, many positive reviews were not identified correctly. But it performed much better in identifying negative reviews. Most of the reviews with the highest negative polarity (-1) were indeed negative reviews.

Transforming Text into Vectors

Doc2Vec

In order to extract vector representations for each text review, we used Doc2Vec from the Gensim module. A numerical vector was generated for each review using word vectors. We decided Doc2Vec would work better than Word2Vec in this case because Doc2Vec would give similar texts similar vector representations. After inputting the text reviews into the model, the model gives a numerical representation for it.

TF-IDF

We also used a simple encoding method namely TF-IDF(Term Frequency - Inverse Document Frequency). In this method, the number of times a word appears in a review is computed, and then the relative importance of this word is computed by looking at the number of reviews in which the word appears

Building Classification Models

Target Variable and Feature selection:

In order to build the model, our target variable was the rating(Good, Bad, Neutral). The features we used were the number of characters of review, number of words of review, Polarity computed by Textblob, four scores returned by VADER (pos, neg, neutral, compound), vector representations from Doc2Vec and TF-IDF scores. I decided to include both the sentiment scores since one performed well on good reviews and the other on bad.

Training and Text data sets
The dataset was split into test(20%) and training data. Since the dataset was very unbalanced(lots of good ratings), stratified sampling was used. We also used MinMaxScaler to create scaled test and training data for those models that can’t tolerate negative value in features.

Classifier Models

After feature engineering and selection we used a variety of classifiers for our predictions. These were multinomial logistic regression, Random Forest, Multinomial Naive Bayes classifier, Decision Tree Classifier, K Nearest Neighbours, Gradient Boosting and Neural Networks.

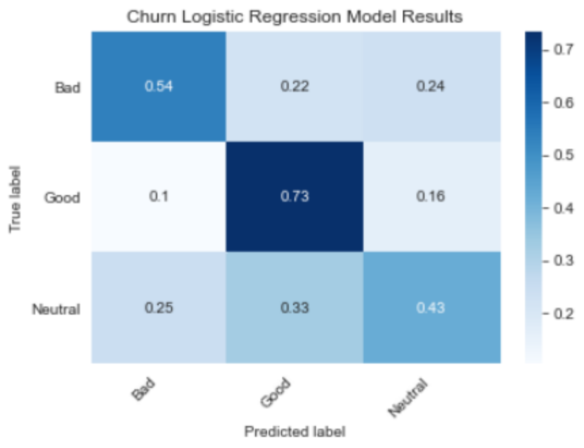
Below is the performance of the models summarized

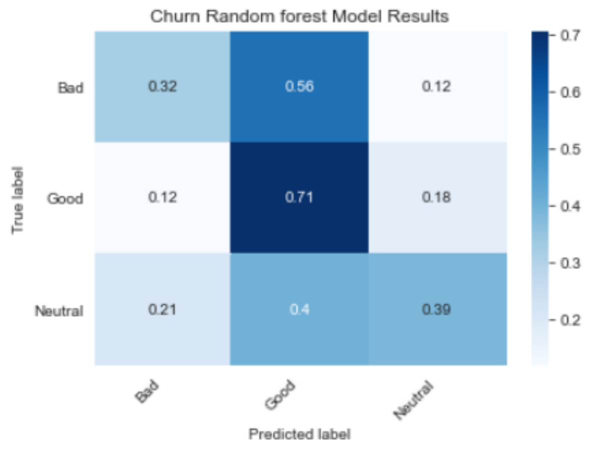
Model Name	Accuracy	Precision	Recall	F1-Score
Multinomial Logistic Regression	0.714	0.66	0.71	0.64
Random Forest	0.697	0.60	0.70	0.59
Multinomial Naive Bayes	0.696	0.62	0.70	0.63
Decision Tree Classifier	0.589	0.58	0.59	0.58
K Nearest Neighbours	0.685	0.56	0.69	0.59
Bayesian Classifier	0.322	0.59	0.32	0.36
Gradient Boosting	0.700	0.49	0.70	0.58
Neural Networks	0.656	0.63	0.66	0.64

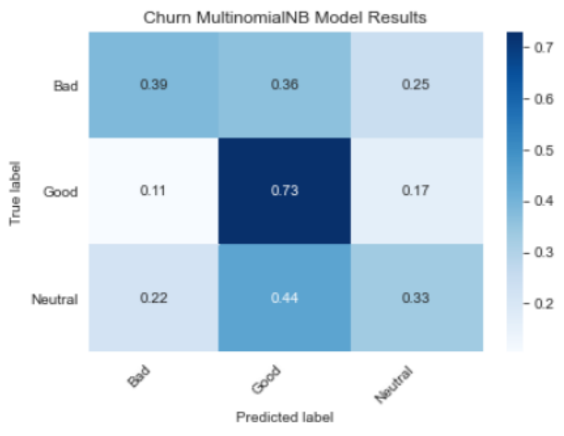
Discussion and Results

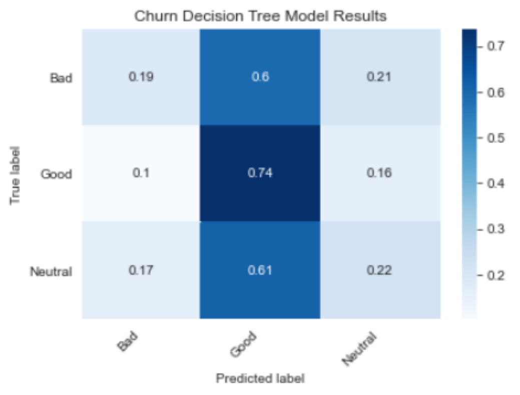
Looking at the feature importance table of the random forest (Figure 11), we can see that vector representation, polarity, sentiment scores as well as a few words are high in importance. Looking at the Model Performance summary table, Multinomial Logistic Regression performs best in terms of accuracy at 71.4%. F1 Scores might be also a good measure to use to choose the best model if we want a balance between precision and recall and especially in case of the presence of uneven class distribution as in this case. Looking at F1 Scores, it again looks like Multinomial Logistic Regression, Multinomial Naive Bayes and Neural Networks seem to perform the best. Due to the imbalance in the dataset towards good reviews, the model tends to "guess" good more frequently and is correct and so accuracy is higher. Looking at Churn Results figures also this is obvious. As we can see it gets a most percentage of "good" reviews correctly. For Multinomial Logistic Regression, Multinomial Naive Bayes, and Neural Networks, it seems to get neural reviews least correctly and this makes sense as it is difficult to identify neutral as opposed to good and bad. Looking at churn results for all the other models, we can again see that the model is guessing more percentage of reviews as good. This is due to the imbalance in the dataset as discussed before.

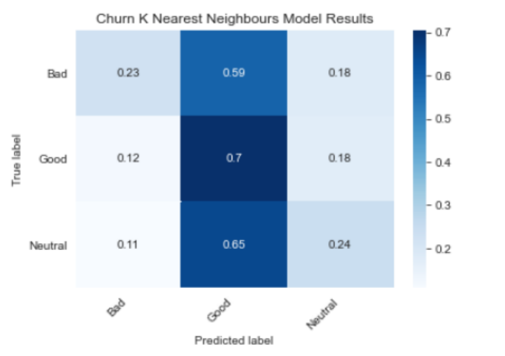
Churm Results For All Models

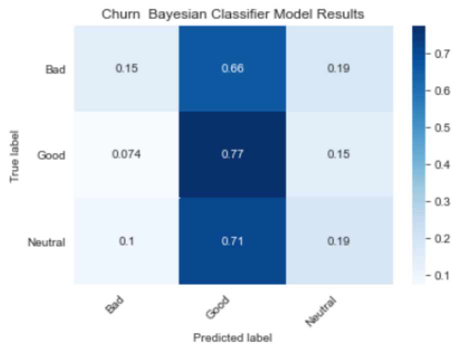


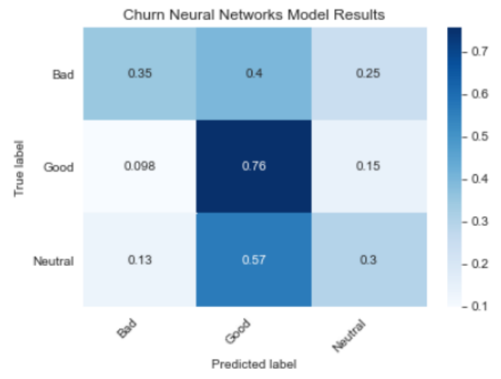


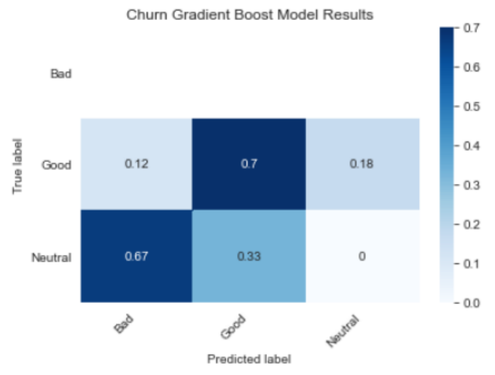


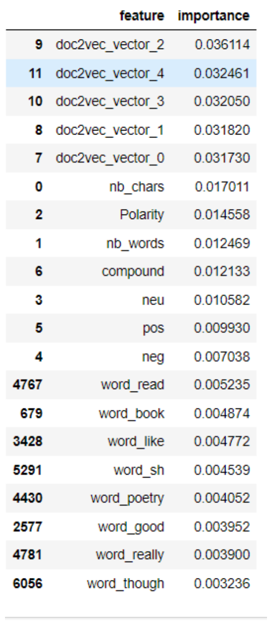












	feature	importance
9	doc2vec_vector_2	0.036114
11	doc2vec_vector_4	0.032461
10	doc2vec_vector_3	0.032050
8	doc2vec_vector_1	0.031820
7	doc2vec_vector_0	0.031730
0	nb_chars	0.017011
2	Polarity	0.014558
1	nb_words	0.012469
6	compound	0.012133
3	neu	0.010582
5	pos	0.009930
4	neg	0.007038
4767	word_read	0.005235
679	word_book	0.004874
3428	word_like	0.004772
5291	word_sh	0.004539
4430	word_poetry	0.004052
2577	word_good	0.003952
4781	word_really	0.003900
6056	word_though	0.003236

Figure 11: Feature importance for Random Forest

Limitations and Challenges of our project

There were a few limitations and challenges in this project.

Our original idea was to work with the entire book dataset from the UCSD Book graph. But due to the enormous size of the dataset 2.36M books and 15M reviews, we decided to focus on only poetry books but even the large size of this dataset was a challenge to work with on a local machine. Most models could not be run on the huge dataset and therefore a sample of the dataset was used instead so that work could be done on a local machine and even in this case it often took a while to run.

Another major shortcoming of this project is that different people associate ratings with different sentiments. That is the difference between 2 and 3 rating or 4 ratings might be different for different people. A Generally positive review with minor points of discontent might be a 4 rating for one person while a 3 rating for another. This will create some in-build error in the model

Another issue that was touched upon earlier is the tendency of sentiment analysis models used to misinterpret negative words or double negatives to be a bad sentiment even though in the context it is not. Also to get confused in mixed sentiment reviews where someone says “It was nice but the cover was horrible…”. Similarly removing capitalization could have potentially led to some loss of sentiment as “I did NOT like the book” should have a lower rating than “I did not like the book”.

Project Experience Summary

Abel Thomas

- Was responsible for performing exploratory data analysis by first cleaning the data and carefully exploring and analyzing for various trends within the data set.
- Performed preliminary analysis, to see if there were any interesting questions that could be examined or worth exploring in Part 2 of the project.
- Created and set up the git repo and was responsible for looking over the code that was submitted to ensure and maintain code quality
- Wrote section 1 of the report
- Organized the entire report and did final formatting
- Submitted Final report

Divya Thomas

- Cleaned poetry book reviews by normalizing the texts and removing noise
- Performed sentiment analysis by computing polarity scores using TextBlot and VADER
- Transformed text into Vector representations using TF-IDF and Doc2Vec to prepare data for machine learning
- Performed feature engineering and feature selection and ran 7 models(multinomial logistic regression, Random Forest, Multinomial Naive Bayes classifier, Decision Tree Classifier, K Nearest Neighbours, Gradient Boosting, and Neural Networks)
- Created plots of Churn results for models and selected the best model.
- Wrote section 2 of the report