Requirements:

* Full, written analysis
* Intro the topic
* inspiration
* any hypotheses or expectations you had
* ML SECTION (discuss the entire modelling process)
* TABLEAU SECTION (discuss designs, how to use the dashboard, observations, etc)
* Web App Section (what does the website look like, and how can it be used?)
* Conclusions/call to action
* limitations and future work
* Include plenty of images, tables, etc. You can use an appendix

Project Overview:

Inspiration:

Machine Learning:

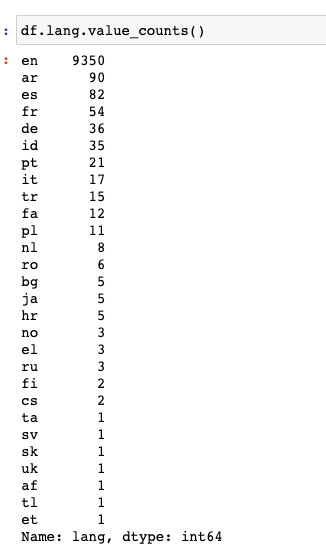
**Natural Language Processing: Data Preprocessing**

*Introduction*

Natural Language Processing (NLP) is “a branch of artificial intelligence that helps computers understand, interpret and manipulate human language,” as stated by SAS[[1]](#footnote-0). For the scope of this project, NLP allows us to analyze the book descriptions available in the data set for term frequency and sentiment in order to build a book recommendation list for the user (which is a content-based filter). Unlike the KNN model, this analysis will not be based on any numeric data, such as ratings, reviews, or rating count. Recommendations will be from similarity on book descriptions and genre.

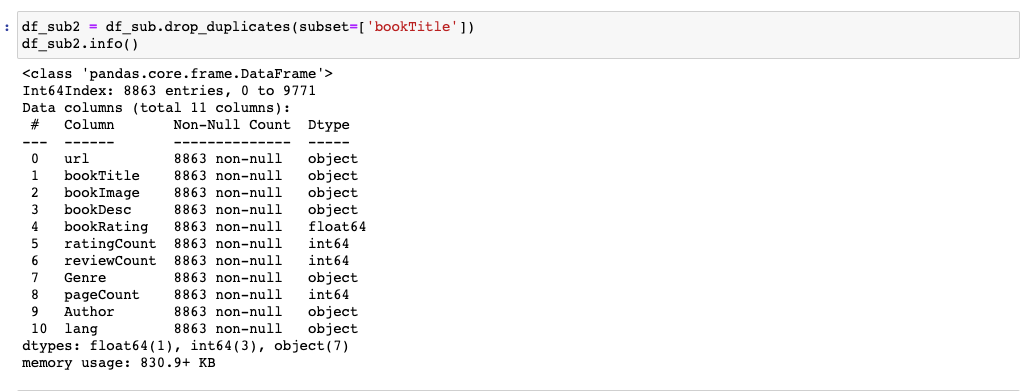
*Chosen Columns*

Because NLP is text driven, the columns from the data set that will be used for analysis are Genre and bookDesc. The other columns included are key identifiers or other additional information needed for our user experience. These columns are: url, bookTitle, bookImage, and Author.

*Data Filters*

Some of the books in the data set were non-english titles. Using the langdetect library[[2]](#footnote-1), we identified the 9,350 books that appeared to be in the english language (language breakdown pictured with code). We then dropped the rest.

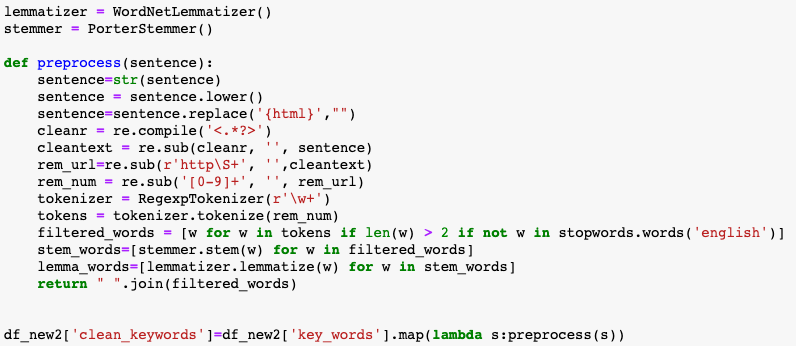
After this, we dropped duplicate titles. This dropped some times that had the different authors, such as *Apollyon* by Jennifer L. Armentrout and Tim LaHaye. We needed to make this decision due to time constraints, however, since we had 487 duplicate rows that we could manually review, nor could we make our user-function on our website complex enough to differentiate between two of the same titles. To drop rows, we dropped on the bookTitle column and .dropduplicates, as seen in the screenshot below:

At this stage, only 5,000 rows were kept in the data set due to size constraints of the website. The 5,000 titles were randomly chosen (using random state 42) in a jupyter notebook. 

*Keyword Extraction*

Using Rake (from the NLTK library), stopwords, punctuation, and white space were removed from the bookDesc and bookTitle column. All words were made lowercase and made a list for easier future processing. This was done so that text analysis is not done on common words such as the, on, as, a, and so on. The code was derived from a movie-recommender example[[3]](#footnote-2)

This process did not remove all symbols or numbers from the data. To address this issue, the following function was used:



In the function, symbols, urls, and numbers were removed. Keywords were also processed using PorterStemmer and WordNetLemmatizer libraries.

The Porter stemmer algorithm is “a process for removing the commoner morphological and inflexional endings from words in English,” as stated by the official PorterStemmer library page.[[4]](#footnote-3) In other words, it finds the root/stem/base words of terms in our keyword lists and replaces them with the stem word, such as replacing *making* with *make*.

WordNetLemmatizer is a more advanced text mining library than PorterStemmer.[[5]](#footnote-4) It is often considered a better alternative, but using them in conjunction with one another is common. Here, the keywords are lemmatized after they were stemmed to make sure that a wide enough net was cast to convert all of the relevant words in the dataset.

*Bag of Words*

Bag of Words in NLP is a technique of text modeling. It places all relevant words in the data (after processing) in a vector to be used for NLP models.[[6]](#footnote-5)

For this data set, the following preprocessing took place:

* Genre: made lowercase and one word
* Book Title: made lowercase, tokenized, and stop words dropped
* Book Description: made lowercase, tokenized, and stop words dropped (described above)

These columns were added to our “Bag of Words” column.

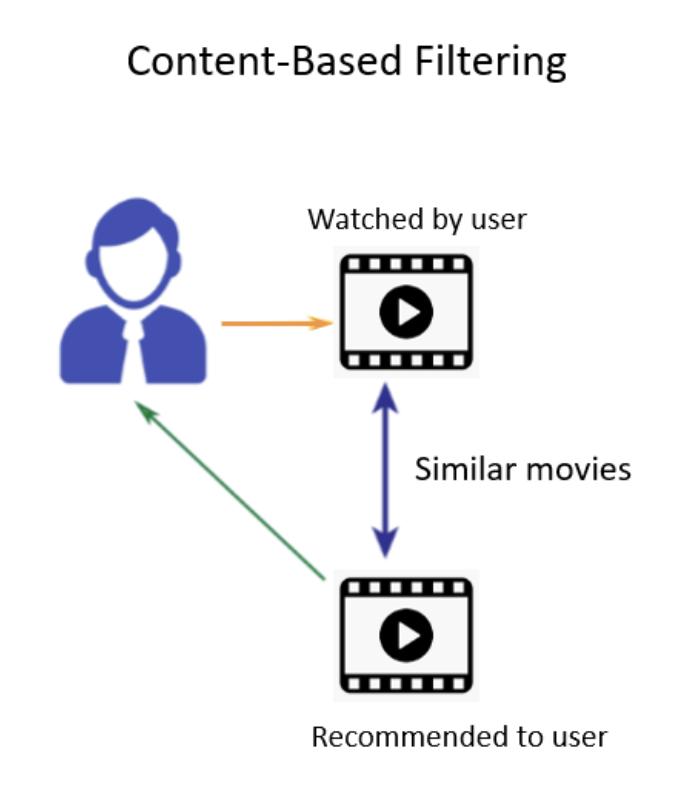
The disadvantages of this technique is that: it ignores the location information of the word, ignores the semantics (such as sarcasm), and, in the same vein, treats synonyms separately from each other (booklover and bibliophile can be interchangeable, but would not be treated as such with this technique).

*Sentiment Analysis*

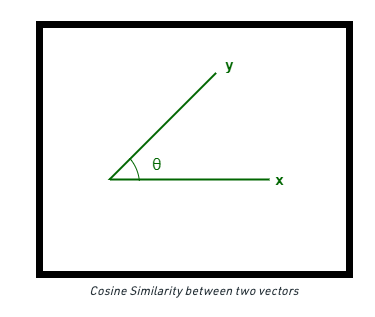
Sentiment analysis focuses on “defining the opinions, attitudes, and even emoticons in a corpus of texts. The range of established sentiments significantly varies from one method to another.”[[7]](#footnote-6) Text Blob is a Python package that generates polarity and subjectivity scores. Polarity is a float within the range [-1, 1], which returns negative sentiments (-1) or positive sentiments (1). A 0 is a neutral sentiment. It will also evaluate a subjectivity score between 0 and 1, where 0 is objective and 1 is subjective.

These scores were used in the KNN model filter.

**Natural Language Processing: Countvectorizer Recommender Model**

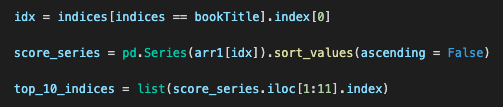
The models we chose to use for NLP are content-based filtering. This means that recommendations are generated based on product metadata, rather than input from other users, which is collaborative filtering.[[8]](#footnote-7)

CountVectorizer is a simple frequency counter for each word in the Bag of Words column. It will generate a matrix containing a count for all of the words, a cosine\_similarity function can be applied to compare similarities of counts between books.

Cosine\_similarity is helpful for determining how similar data objects are, irrespective of their size.[[9]](#footnote-8) In other words, even if the similar data objects are far apart by the Euclidean distance because of the size, there could be a small angle between them. The smaller the angle, the greater the similarity. This is a much better way of measuring similarity for a non-clustered data set. 

We ultimately chose to include the TF-IDF Recommender instead for our website for its prioritization of more targeted and rarer terms.

**Natural Language Processing: TF-IDF Recommender Model**



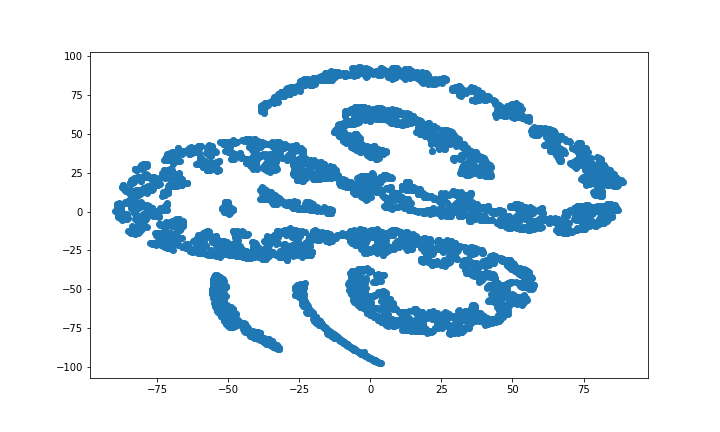
**Natural Language Processing: Launching on Flask App**

Cos file on aws (read in with s3fs)

Input (book title)

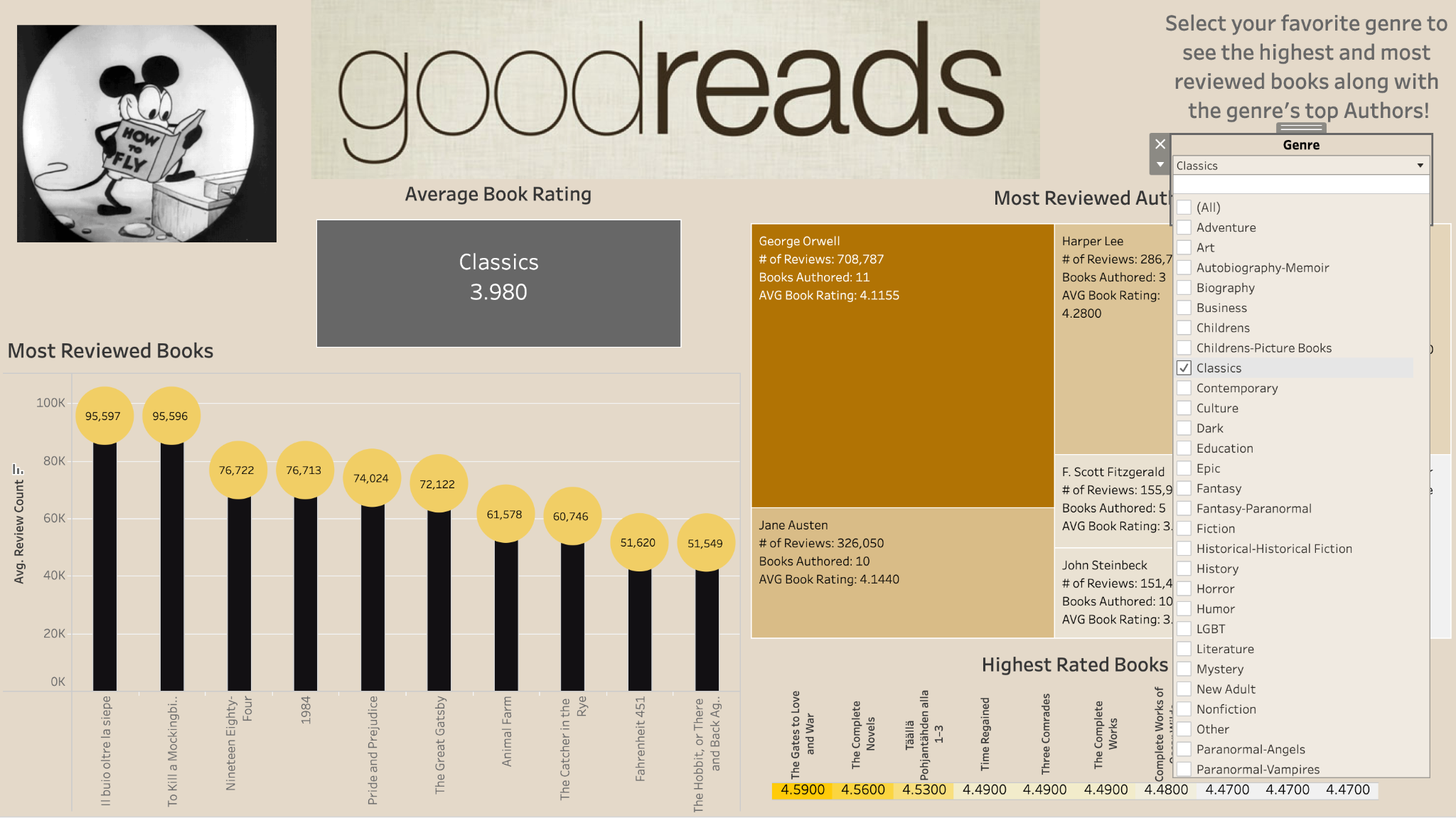
**Machine Learning KNN Model**

Due to the scope and type of problem this project is tackling, a KNN model was selected as the best model to recommend book titles to a user. KNN stands for k-nearest neighbors and is ‘is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.’[[10]](#footnote-9) In other words, the algorithm works by evaluating numeric features in the data set, and predicts the clustering of the data points. Due to the model only evaluating numeric data, we will only be using features/ columns that can be easily one-hot encoded and that are already in numeric form. When you ask the model to give you ten recommendations based on a data point, in our case a book, it pulls the ten nearest neighbors to that original data point.

The columns selected to become features for the KNN model differ from the TF-IDR recommender, numeric data is used rather than words. For this reason columns that could not be one-hot encoded or not in numeric format already were dropped. This left us with the book ratings, count of reviews, count of ratings, the sentiment polarity and subjectivity, and the one-hot encoded columns of genre and author. The three numeric fields that we started with, rating, count of ratings, and count of reviews were scaled. To look at clustering of the books, principal component analysis (PCA) was used on the same three columns that were scaled. PCA is the process of computing the principal components to reduce the dimensionality of the data[[11]](#footnote-10). The PCA columns along with the one-hot encoded genre and author columns were concatenated to produce a dataframe with all the columns needed to be run through the model. To visualize the clusters, the dataframe was transformed using t-sne to reduce dimensionality to just two dimensions. This reduced data is plotted onto a graph to give us a visual representation of the clusters.

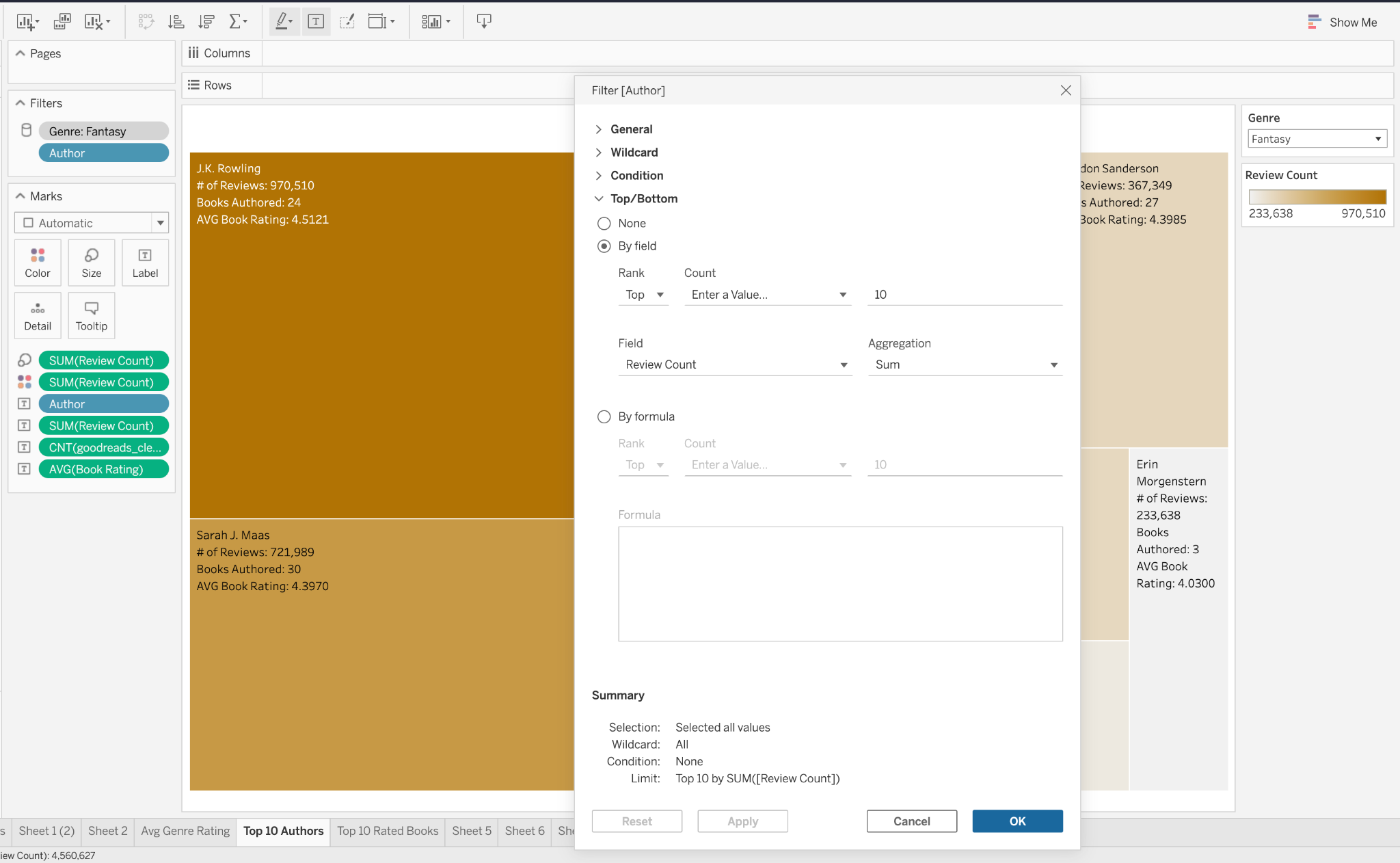
For the final deployed model we ended up using a more simplified version of the model. We dropped the PCA operations on the data. This gave us results that made more sense as some of the books that were recommended did not make too much sense. For our recommender, we wanted to pull the ten closest neighbors to the book that the user is inputting. Due to the model spitting out the same book that is being asked to compare, the number of neighbors we are asking the model to show is actually eleven. Since this model does not take into account the book description and book name, there is a very small chance that the model will recommend books that are in the same series, thus eliminating obvious choices that the user may already know of.

**Tableau**

Our Website has 2 interactive tableau dashboards. Both dashboards respond to a genre filter where you can select any genre that is contained in the goodreads dataset and both pages will display 4 visualizations. All visualizations across these two pages respond to any change in the genre selector, whether it be representing one genre, multiple or all but one. 

The first dashboard presents the top 10 in the most reviewed books, the most reviewed authors, and the highest rated books. For the color theme, shades of brown, yellow and gold gives the page the aesthetic of wooden bookcases filled with books. The author visualization is a highlight table displaying the author, number of reviews, books authored, and their average book rating. The size of each chunk of the table each author represents is determined by the total number of reviews for each author divided by the total number of reviews of the cumulative top 10. The greater review count presents a larger area in the table. Books with the highest review count are represented by a lollipop chart with the number of reviews presented on each lollipop. Top 10 highest rated books are presented in a highlight bar in the lower right corner of the page, with the higher rating in a brighter colored bar. Lastly, the genre’s average book rating is displayed underneath the goodreads title in a gray box.

In building this dashboard, it was a challenge to get the 3 top 10 visualizations to respond with their respective top 10 at the same time with different measures. This was resolved by adding a second filter to each sheet used in the dashboard.

All 3 of the added filters displayed the top 10 of the measure involved while keeping the genre filter as the primary filter. Keeping the genre filter as the primary filter allowed the new filter to respond to the genre selection, then display the top 10.

Our second tableau dashboard also generates 4 visualizations contingent on the genre f

filter. To contrast the wood aesthetic, this dashboard has shades of green. The top 10 books for each genre by page count, top 10 authors by measure of books written, and the average page count and number of books in the dataset for each genre is displayed.

Website:

Nlp model:

Works cited:

Conclusion:

Limitations and future work:

Tying in titles without the same case throws an error. We would like to accept titles in a case or

have autofill in our search bar.

References:

Editing mark labels in Tableau: <https://help.tableau.com/current/pro/desktop/en-us/annotations_marklabels_showhideworksheet.htm#:~:text=the%20label%20text%3A-,On%20the%20Marks%20card%2C%20click%20Label.,text%20and%20then%20click%20OK>.

Generating Top 10 Lists in Tableau: <https://www.youtube.com/watch?v=nG2EDnXyN3M>

1. https://www.sas.com/en\_us/insights/analytics/what-is-natural-language-processing-nlp.html [↑](#footnote-ref-0)
2. <https://pypi.org/project/langdetect/> [↑](#footnote-ref-1)
3. <https://www.kdnuggets.com/2019/11/content-based-recommender-using-natural-language-processing-nlp.html> [↑](#footnote-ref-2)
4. <https://tartarus.org/martin/PorterStemmer> [↑](#footnote-ref-3)
5. https://aparnamishra144.medium.com/lemmatization-in-nlp-using-wordnetlemmatizer-420a444a50d [↑](#footnote-ref-4)
6. [https://www.mygreatlearning.com/blog/bag-of-words](https://www.mygreatlearning.com/blog/bag-of-words/#:~:text=What%20is%20a%20Bag%20of%20Words%20in%20NLP%3F,a%20method%20of%20feature%20extraction%20with%20text%20data) [↑](#footnote-ref-5)
7. https://stackabuse.com/sentiment-analysis-in-python-with-textblob/ [↑](#footnote-ref-6)
8. https://www.kdnuggets.com/2019/11/content-based-recommender-using-natural-language-processing-nlp.html [↑](#footnote-ref-7)
9. https://www.geeksforgeeks.org/cosine-similarity/ [↑](#footnote-ref-8)
10. https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point. [↑](#footnote-ref-9)
11. https://builtin.com/data-science/step-step-explanation-principal-component-analysis [↑](#footnote-ref-10)