Sentiment Analysis of Goodreads Book Reviews with Machine and Deep Learning

by

Blanka Jarmoszko

A Bachelor's Thesis

Submitted to the Department of Statistics and Data Science

Northwestern University

April 2024

Supervised by Prof Emre Besler

Abstract

This thesis investigates sentiment classification in Goodreads book reviews using machine learning techniques. It explores the impact of preprocessing methods like TF-IDF and Bag of Words on classification accuracy, evaluates the effectiveness of various machine learning algorithms, including Logistic Regression, Support Vector Machines, Naive Bayes, Random Forest, AdaBoost, k-Nearest Neighbors, and Recurrent Neural Networks, and addresses challenges unique to book review texts. The study finds that TF-IDF consistently outperforms Bag of Words, with Logistic Regression achieving the highest accuracy, precision, and recall rates at 65%. Despite promising results, the complexity of book review texts highlights the need for alternative sentiment classification approaches. This research contributes to a deeper understanding of sentiment analysis in literature and suggests avenues for future exploration.

Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

In the contemporary digital age, the internet has become an indispensable part of our daily lives, profoundly influencing how we interact, communicate, and make decisions. One of the most significant aspects of our online existence is the proliferation of reviews, which have become a cornerstone in guiding our choices and shaping our perceptions of products, services, and experiences. Whether we are considering purchasing a product, planning a trip, or selecting a service, the first instinct for many is to turn to online reviews to inform their decisions. However, the vast amount of user-generated content available on the internet poses a significant challenge: how do we extract meaningful insights from this abundance of data?

Sentiment classification of reviews has emerged as a critical area of research, seeking to provide a structured approach to understanding and categorizing opinions expressed in these vast repositories of information. By analyzing the sentiment of reviews in a systematic way, researchers and businesses can uncover valuable insights into consumer preferences, trends, and satisfaction levels. Consequently, sentiment classification has become an indispensable tool for various applications, including e-commerce, market analysis, and social listening.

While sentiment analysis has been extensively studied in the context of product and movie reviews, book reviews, despite their growing prominence, have been relatively less explored. The sentiment and emotions expressed in literature are essential for understanding the impact and reception of literary works. In this regard, Goodreads, a popular social cataloging website for book lovers, offers a vast repository of user-generated book reviews, making it an invaluable resource for sentiment analysis in the realm of literature.

1.1 Problem Description

This thesis aims to explore the application of machine learning techniques for sentiment classification of Goodreads data. By leveraging machine learning algorithms, this study seeks to extract sentiment and emotions from Goodreads book reviews, thereby providing valuable insights into the reception and impact of literary works in the digital age. Specifically, this thesis will investigate the effectiveness of various machine learning models, such as Logistic regression, Support Vector Machines (SVM), Naive Bayes, Random Forest, AdaBoost, K-Nearest Neighbors (KNN) and Recurrent Neural Networks (RNN), in classifying sentiment in book reviews. Additionally, this research will explore the nuances of sentiment and emotion extraction, adapting and refining existing methodologies to suit the unique characteristics of literary reviews.

Through this exploration, we aim to contribute to a deeper understanding of sentiment analysis in the context of literature, shedding light on the reception, impact, and emotional resonance of literary works in the digital era.

1.2 Research Questions

1. What is the impact of various preprocessing techniques, such as TF-IDF and Bag of Words, on the accuracy of sentiment classification in Goodreads book reviews?

2. How effective are machine learning algorithms, such as Logistic regression, Support Vector Machines (SVM), Naive Bayes, Random Forest, AdaBoost, K-Nearest Neighbors (KNN) and Recurrent Neural Networks (RNN), in sentiment classification of Goodreads book reviews?
3. What are the most significant challenges in sentiment analysis of Goodreads book reviews, and how can they be addressed to improve accuracy and efficiency?

1.3 Thesis Summary

Chapter 3 provides an overview of the data source, the process of obtaining the data, data extraction/scraping, exploration, and preprocessing.

Chapter 4 covers the methodology, detailing the feature extraction techniques, architecture of the implemented Machine Learning and Deep Learning models used for classification, as well as the model tuning process and evaluation metrics used.

Chapter 5 discusses the results obtained from the implemented approaches. It presents the seven different types of models built, along with a comparison of feature extraction methods on model performance. Additionally, it analyzes misclassified reviews by all models and identifies possible issues impacting the model performance.

Chapter 6 offers conclusions of the work described in Chapter 5 and outlines future work.

Chapter 2

2 Literature Review

This section discusses some of the relevant work done on sentiment analysis of review texts. The first section looks at general approaches for text sentiment classification and data preprocessing. Second section is about relevant work on book review sentiment analysis and classification.

2.1 Sentiment Analysis on Review Data

Tan, Lee, and Lim (2023) conducted a comprehensive survey titled "A Survey of Sentiment Analysis: Approaches, Datasets, and Future Research," where they analyzed and compared various studies in the field of sentiment analysis that utilized Product review data from Twitter, Amazon, US Airline Twitter Data and IMBD movie reviews (Tan, Lee, & Lim, 2023). The work outlines valuable insights into the most common feature extraction methods, different modeling approaches, and offers comparisons of the performance of a variety of studies. In their study, Tan, Lee, and Lim (2023) stipulate that sentiment analysis algorithms can be classified into three distinct categories: Machine Learning, Deep Learning, and Ensemble Learning. Machine Learning classifiers such as logistic regression, naive Bayes, and SVM are most used to predict sentiment. However, boosting, and bagging models such as AdaBoost, Random Forest as well as KNN algorithm are also commonly used in the wide range of featured studies. The study also compared the performance of models that used feature extraction on the data against models that didn’t preprocess the text data. Most of the studies featured in the survey, which utilized data preprocessing techniques such as stop word removal, lemmatization, stemming, and TF-IDF, achieved 80% or higher accuracy on test data, overall performing better than their counterpart without preprocessing. The survey also indicates that there isn't one Machine Learning classifier that outperforms all others. Depending on the data structure, preprocessing, and feature extraction, different models demonstrate the highest accuracy. Notably, based on the summary of machine learning approaches, the best performing models on IMBD reviews utilized TF-IDF as the feature engineering method.

Furthermore, in their section discussing Deep Learning classifiers they outline the commendable performance of two class classification neural network-based models due to their ability to capture more complex trends and patterns in the text data that are often prevalent in reviews. One of the best performing models is an RNN model with GloVe pre-processing that achieved 84% accuracy on publicly available movie review dataset (Janardhana, Vijay, Swamy, & Ganaraj, 2020).

In "Sentiment Analysis by Using Recurrent Neural Network," Patel and Tiwari (2019) covered different sentiment classification techniques, emphasizing two primary approaches: machine learning-based and lexicon-based (Patel & Tiwari, 2019). The study concludes that the machine learning-based approach is more suitable than the lexicon-based approach. The paper surveyed various deep learning models for sentiment analysis, highlighting CNN and RNN as the most popular deep learning algorithms for opinion mining. Patel and Tiwari's paper concludes that RNN achieved an accuracy of 87.42% on the movie review dataset due to its algorithm that produces the output based on previous computations by utilizing sequential information.

2.2 Sentiment Analysis of Goodreads Reviews

Although sentiment classification of review data has increased in popularity over the last few years, book review sentiment studies make up a small percentage of the related literature.

One potential reason explored by Zhu investigates the difference in the structure of book reviews compared to product or movie reviews (Zhu, 2013). Online book reviews tend to be highly subjective and contain a higher degree of internet-specific vocabulary such as emojis or slang. Furthermore, a study on the book impact assessment of Goodreads reviews by Wang, Liu, and Han found that, unlike in other forms of reviews, approximately 70% of emotional words appear in the first sentence of the book review, and that reviewers mostly discuss the book's content rather than its external characteristics (Wang, Liu, & Han, 2019).

In "Sentiment Analysis and Recommendation of Book Reviews", the authors processed the user rating and reviews of multiple books by the same author (Devaki, Lokesh, Muthukumar, & Shree, 2022). The data was obtained from Amazon and preprocessed through HTML tags and URLs removal, punctuation and special character removal, lemmatization, and POS Tagging and Filtering. The feature extraction was done with TF-IDF. In this paper, they used algorithms such as Logistic Regression, Random Forest, LSTM, and CNN with variation to determine the most optimal model for the recommendation system and proposed a book recommender system.

In the study on 'Sentiment Analysis of Book Reviews using Unsupervised Semantic Orientation and Supervised Machine Learning Approaches' by Kaur, both unsupervised and supervised learning (SVM and Naive Bayes) were employed on openly available Goodreads and Amazon datasets (Kaur, Bangalore). The comparative analysis of the approaches on the datasets indicates that the unsupervised approach performs better on the Goodreads dataset with an accuracy of 73.23%, whereas the supervised approach gives better results on the Amazon dataset, with Naïve Bayes achieving the maximum accuracy, ranging from 73.72% to 74.73% in the case of 5-folds and 10-folds, respectively. Based on these results Kaur concluded unsupervised algorithms provided better results when the dataset contained long phrases, whereas supervised learning algorithms achieved higher accuracy on datasets containing short one-line reviews.

In "Emotion-Based Literature Book Classification Using Online Reviews", Luton and Badica proposed a system for the extraction of emotions from Goodreads book reviews (Luţan & Bădică, 2022). They designed entities with the fields of interest and created a scraper to collect the dataset from the Goodreads website. Subsequently, sentiments and emotions present in the online reviews were retrieved. Sentiments were extracted using the TextBlob Python library (Loria, 2018). An existing emotion extraction repository from GitHub was utilized and further tuned to meet the study’s requirements (Zajac, 2017). The method of extracting sentiments was refined to consider verbs or adjectives, and the extracted emotions were arranged to be easily accessible for different use cases.

Chapter 3

3 Data Set

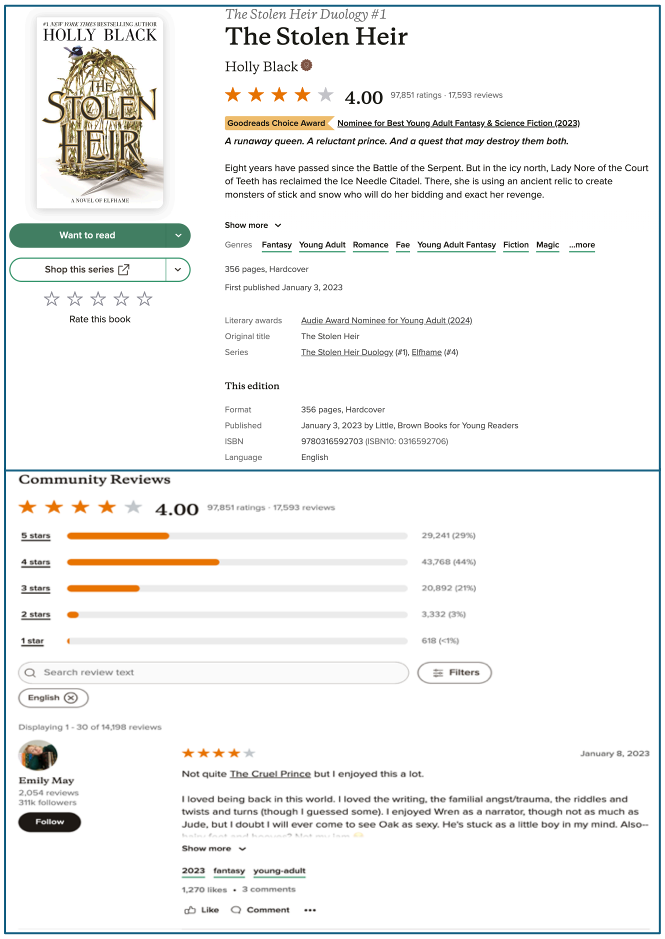
3.1 About Goodreads

Goodreads is an American social cataloging website and subsidiary of Amazon, it serves as a comprehensive platform for book enthusiasts to explore a vast database comprising books, annotations, quotes, and reviews. Founded in December 2006 and launched in January 2007 by Otis Chandler and Elizabeth Khuri Chandler, the platform quickly gained traction, boasting 650,000 members and 10 million books by December 2007. Over the years, Goodreads experienced exponential growth, reaching 10 million members and 20 million monthly visits by July 2012. Its acquisition by Amazon in March 2013 further accelerated its expansion, with the user base soaring to 20 million members by July 2013 and reaching 90 million members by July 2019. Goodreads offers users the ability to create personalized library catalogs, reading lists, and groups for book discussions, surveys, and blogs, making it a vital resource for literary engagement and community interaction.

3.2 Goodreads Data

To collect book and review data, we concentrated on individual book webpages within Goodreads. Each Goodreads book webpage is structured into two primary sections. The initial section encompasses essential details about the book, including a cover image, title, genres, plot description, publication date, information regarding the first edition, and indications of series affiliation.

The second section contains rating and review information. Within this section, the average star rating for the book is displayed, with ratings ranging from 1 to 5 stars, as provided by users. Additionally, the distribution of different star ratings is presented. Moreover, each web page showcases the top 30 reviews, prioritized based on the number of likes and comments they have received, along with the users' standing within the website for the specific titles. Although any user with a Goodreads account can leave a review after indicating they've 'Read' the book, this process lacks verification, allowing reviews from individuals who may not have read the book. Each of the 30 review cards features the reviewer's profile information, including their username, profile picture, total number of reviews, and followers.



The Stolen Heir Duology #1

The Stolen Heir

Holly Black

4.00 97,851 ratings · 17,593 reviews

Goodreads Choice Award Nominee for Best Young Adult Fantasy & Science Fiction (2023)

A runaway queen. A reluctant prince. And a quest that may destroy them both.

Eight years have passed since the Battle of the Serpent. But in the icy north, Lady Nore of the Court of Teeth has reclaimed the Ice Needle Citadel. There, she is using an ancient relic to create monsters of stick and snow who will do her bidding and exact her revenge.

Show more

Genres Fantasy Young Adult Romance Fae Young Adult Fantasy Fiction Magic ...more

356 pages, Hardcover

First published January 3, 2023

Literary awards	Audie Award Nominee for Young Adult (2024)
Original title	The Stolen Heir
Series	The Stolen Heir Duology (#1), Elfhame (#4)

This edition

Format	356 pages, Hardcover
Published	January 3, 2023 by Little, Brown Books for Young Readers
ISBN	9780316592703 (ISBN10: 0316592706)
Language	English

Community Reviews

4.00 97,851 ratings · 17,593 reviews

5 stars	29,241 (29%)
4 stars	43,768 (44%)
3 stars	20,892 (21%)
2 stars	3,332 (3%)
1 star	618 (<1%)

Search review text Filters

English

Displaying 1 - 30 of 14,198 reviews

Emily May
2,054 reviews
311k followers
Follow

January 8, 2023

Not quite The Cruel Prince but I enjoyed this a lot.

I loved being back in this world. I loved the writing, the familial angst/trauma, the riddles and twists and turns (though I guessed some). I enjoyed Wren as a narrator, though not as much as Jude, but I doubt I will ever come to see Oak as sexy. He's stuck as a little boy in my mind. Also--

Show more

2023 fantasy young-adult

1,270 likes · 3 comments

Like Comment

Figure 1: Example Goodreads Web Page representing Book and Review information.

Reviews themselves are limited to 15,000 characters and may include various media such as text, memes, embedded quotes from the books, and emojis. Each review is accompanied by the star rating, review text, number of likes and comments received, and the review date.

3.3 Obtaining Data/ Goodreads Scraper

In 2024, Goodreads no longer provides an available API for accessing book and review data. Consequently, data for this project was obtained using a custom Goodreads scraper we have specifically developed for this purpose. The scraper, written in Python, utilizes the Beautiful Soup library (bs4) for web scraping tasks (Richardson, 2007). Beautiful Soup is a popular Python library designed for parsing and extracting information from HTML and XML files. It simplifies the extraction process by enabling users to navigate through the HTML structure and target elements based on tags, attributes, or other criteria. By leveraging Beautiful Soup, the custom Goodreads scraper efficiently retrieved the required data from Goodreads webpages, facilitating the collection of book and review information for analysis.

The Goodreads scraper served two primary functions: retrieving book data and retrieving review data. Operating on an input text file containing Goodreads links, the scraper systematically extracted the desired information from the provided URLs. Subsequently, the scraped data was saved into separate JSON files: one file for book data and another for review data. To facilitate further analysis and integration into data preprocessing and machine learning models, the JSON files were then converted into CSV data frames, a format well-suited for such tasks. This systematic process, as seen in Figure 2, ensured the efficient collection and preparation of the scraped data for subsequent analysis and modeling stages of the project.

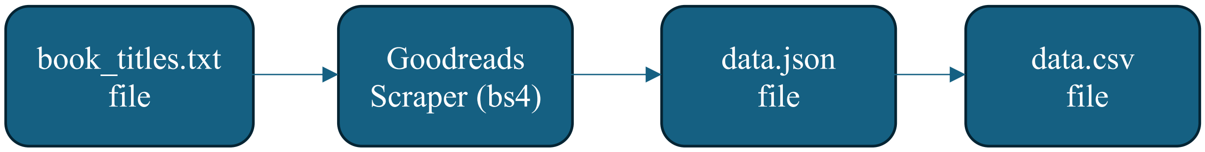


Figure 2: Diagram representing the process of obtaining data from Goodreads.

3.4 Datasets

For this project, it was necessary to narrow the scope of the book review data under analysis. Therefore, all books chosen for this analysis were nominees for the Goodreads Choice Awards. These awards, administered annually by Goodreads, allow platform users to cast their votes for the best books across various categories such as fiction, non-fiction, mystery, romance, fantasy, and more. The selection process involves multiple rounds of voting, commencing with nominations and culminating in the crowning of winners. Winning a Goodreads Choice Award holds significant prestige within the literary realm, often resulting in heightened visibility and increased sales for the recognized works.

Opting for books nominated for Goodreads Choice Awards ensured that the selected titles were already popular within their respective genres, thereby ensuring a sufficient number of detailed and insightful reviews to be analyzed. Furthermore, the Goodreads Choice Awards exclusively nominate books published in the corresponding year, guaranteeing that all reviews considered were composed within recent years, thus providing a fair basis for comparison.

To amass enough books for analysis, we selected titles from the years 2019 to 2023. Spanning across 15 genres, with 20 nominees per genre, the total dataset encompassed 1664 books, offering a sufficient foundation for comprehensive analysis and modeling within this project. In terms of the review dataset, approximately 30 reviews were gathered for each title, resulting in a total of 46,764 reviews. The datasets comprise a range of variables, which are detailed in the table below:

Table 1: Dataset Information

Dataset	Number of Entries	File size (approx.)
book_data.csv	1664 books	446 KB
review_data.csv	46746 reviews	81.9 MB

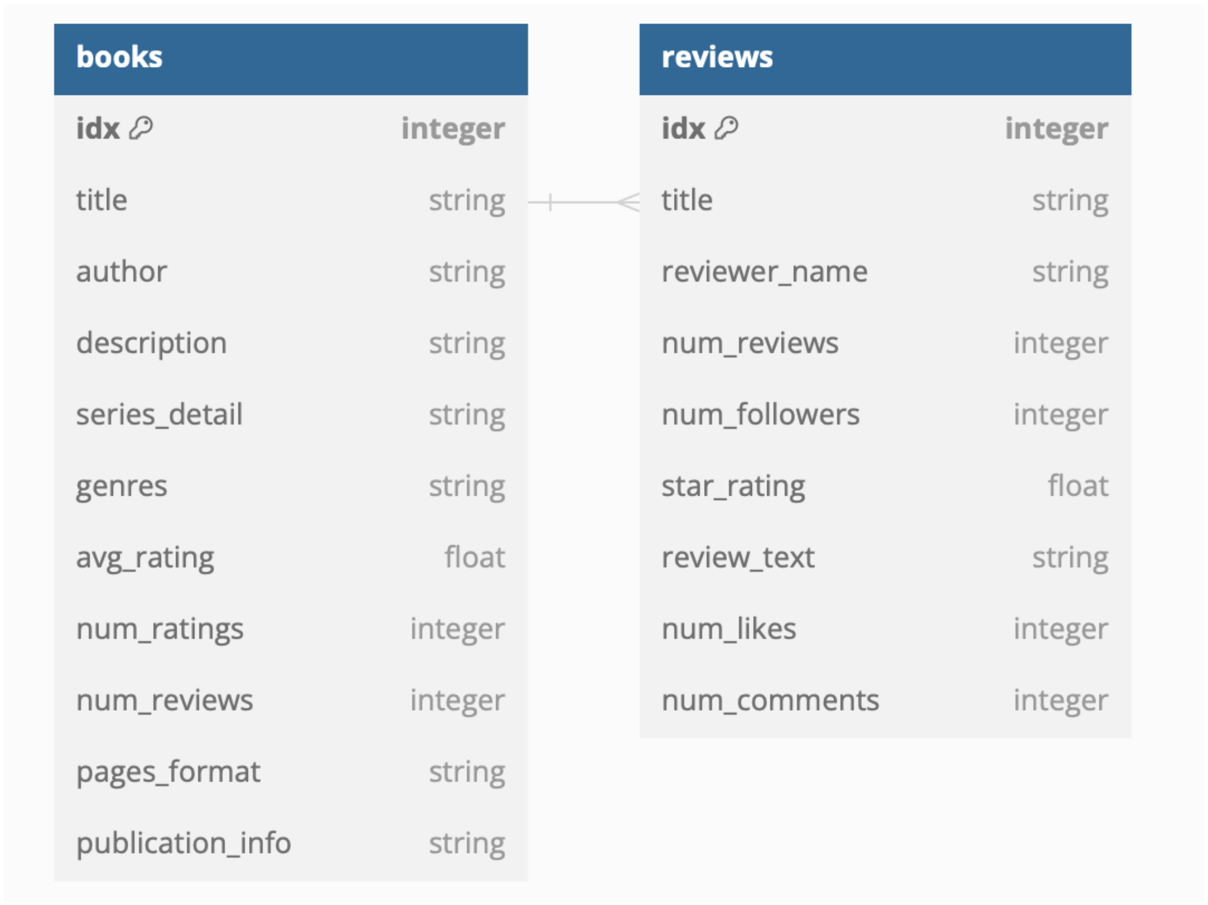


Figure 3: Entity and their relationships of Goodreads data.

3.5 Dataset Preparation: Polarity Ratings and Sentiment Mapping

Sentiment Mapping is a crucial step in Machine Learning Classification models, with sentiment classification models commonly employing either 2 class: positive and negative, or 3 class: negative, neutral, and positive sentiment. This project adopted a focus on 3 class classification, the classes: negative, neutral and positive were built upon the star rating assigned to each review. Despite the availability of star ratings ranging from 1 to 5 in the reviews, the distribution of these ratings was highly unbalanced, leading to increased complexity in the model as seen in Figure 4.

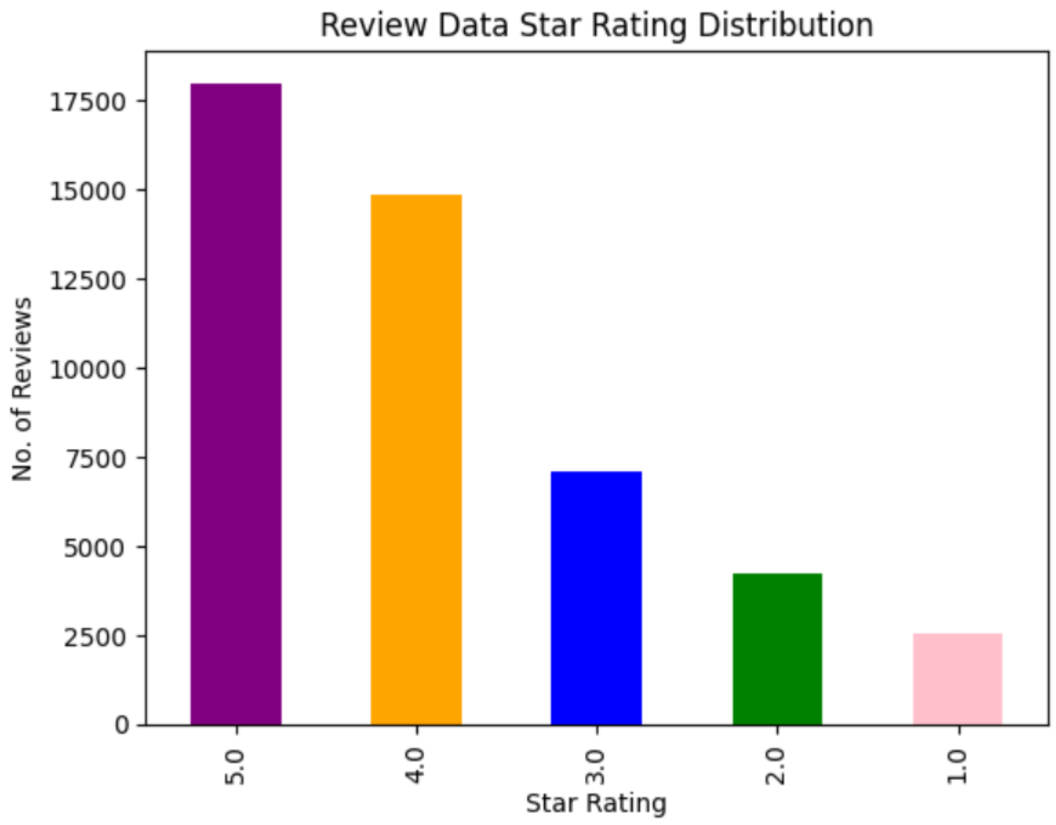


Figure 4: Distribution of Review Star Ratings

Polarity ratings serve as a metric for exploring the relationship between sentiment and star ratings and informing the mapping from star ratings to sentiment. TextBlob, a Python-based library, offers polarity ratings on text data. TextBlob simplifies natural language processing (NLP) tasks, such as sentiment analysis, by employing a lexicon-based approach.

TextBlob calculates polarity ratings by assigning each word in the text a polarity score, typically ranging from -1 (very negative) to +1 (very positive), based on its sentiment. These scores are aggregated to determine the overall sentiment polarity of the text. Sentiment lexicons or dictionaries, containing pre-defined lists of words along with their associated polarity scores, are often utilized for this purpose. TextBlob employs its built-in sentiment lexicon to compute polarity ratings.

When analyzing text sentiment, TextBlob calculates the average polarity score of all words to determine the overall sentiment polarity, with this average score serving as the polarity rating, indicating whether the text expresses positive, negative, or neutral sentiment. This idea can be formulated as follows:

$$\text{Polarity}(T) = \frac{1}{n}\sum_{i=1}^{n} \text{Polarity}(w_i)$$

Equation 1: Polarity of text T.

Where:

- Polarity(T) is the polarity of the text T.
- n is the total number of words in the text T.
- Polarity(w_i) is the polarity score of the i^{th} word in the text T.
- Polarity(T) is the polarity of the text T.

Figure 5 shows the distribution of polarity in text reviews for each star rating. There is a clear pattern showing that polarity ratings of reviews across all star ratings are very similar, with the majority falling within a small range.

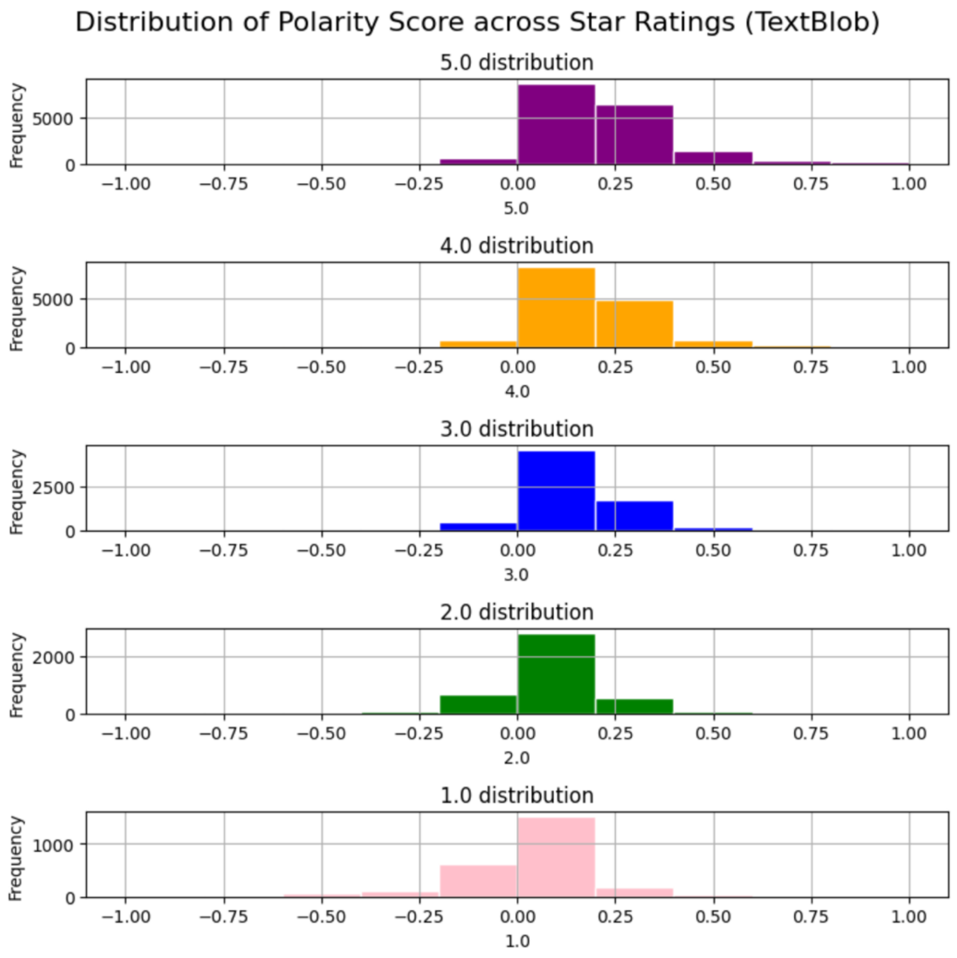


Figure 5: Distribution of TextBlob Polarity scores across Star Ratings.

Based on this analysis and the existing class imbalance, the star ratings and sentiment classes were mapped as follows:

Table 2: Star Rating to Sentiment Mapping.

Sentiment Class	Star Rating
Negative (0)	1
	2
	3
Neutral (1)	4
Positive (2)	5

The outcome of this mapping was a much more balanced dataset as seen in Figure 6.

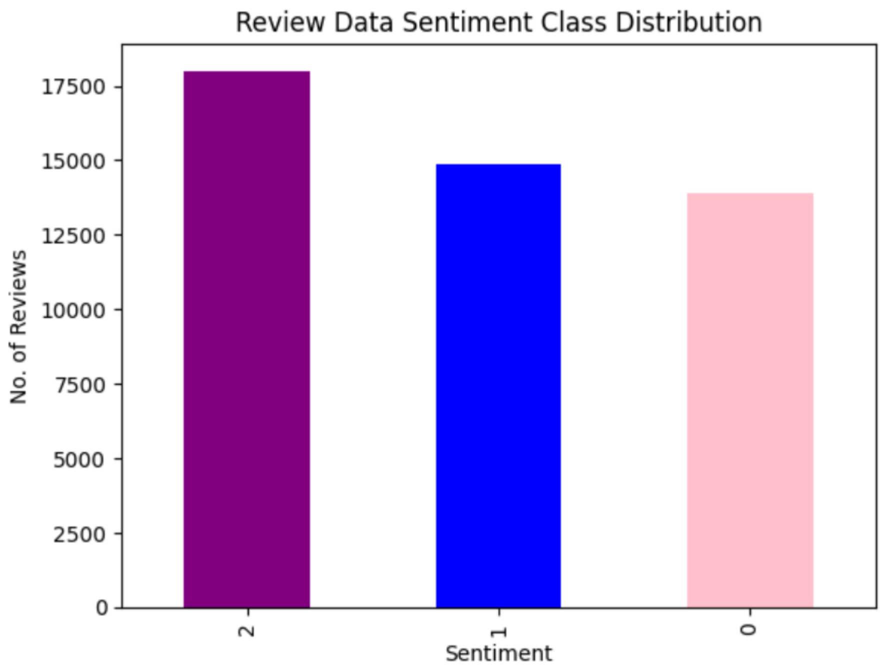


Figure 6: Distribution of Sentiment across reviews after mapping.

3.6 Dataset Preprocessing

A significant aspect of book reviews is the presence of emojis and other non-standard characters. Using the 're' library (Van Rossum, 2020), occurrences of emojis and symbols that deviate from standard formatting—such as letters, numbers, and special characters—were counted. It was observed that there were over 8000 different unrecognizable elements, ranging from emojis to cursive or special symbols. This phenomenon may be attributed to the encoding of special elements on the webpage that are not recognized as standard characters. The most frequently used emojis across all reviews were identified in Table 3:

Table 3: Emojis most often used in reviews.

Emoji	No. of Uses
✨	889
⭐	825
😭	599

It's noteworthy that many of the top emojis were variations of stars, which is logical given that reviewers often utilize them to enhance their reviews or as part of representing their numerical rating.

To address the challenge posed by emojis, an emoji library was utilized (Kim & Wurster, 2015), providing mappings between commonly used emojis and verbal descriptions. With this library, emojis were converted to their verbal descriptions within the review text. For instance, 😂 would be represented as "face_with_tears_of_joy". This approach was necessary as many text processing models struggle to handle emojis effectively.

3.7 Dataset Preprocessing: NLP Linguistic Data Processing

3.7.1 NLTK

Natural Language Toolkit (NLTK) is a leading platform for Python programming, designed to handle human language data efficiently (Bird, Klein, & Loper, 2009). NLTK provides a comprehensive suite of text processing libraries supporting various tasks, including classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

3.7.2 Data Processing

Following common practices in natural language processing (NLP), the text was first converted to lowercase, with numbers, punctuation, and special characters removed. Subsequently, the text was tokenized into a list of words or tokens, with each token representing a distinct unit of text. NLTK's tokenization process breaks down text into smaller units based on certain rules or patterns. For instance, NLTK's word tokenization breaks a text into individual words or tokens using regular expressions and language-specific rules to identify word boundaries. For example, the sentence "This book was amazing." would be tokenized into the list of words: ['This', 'book', 'was', 'amazing', '.'].

Tokenization was a necessary process that decreased the dimensionality of data and removed possible noise. To further optimize the data, common stop words, such as "the", "is", and "and", were removed using NLTK. These words typically did not contribute significant meaning to the text.

Table 4: All punctuation, special characters and stop words removed from review texts.

Punctuation Removed	Stop Words Removed
! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ ] ^ _ ` { | } ~	i, me, my, myself, we, our, ours, ourselves, you, your, yours, yourself, yourselves, he, him, his, himself, she, her, hers, herself, it, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very, s, t, can, will, just, don, should, now, d, ll, m, o, re, ve, y, ain, aren, couldn, didn, doesn, hadn, hasn, haven, isn, ma, mightn, mustn, needn, shan, shouldn, wasn, weren, won, wouldn
Special Characters Removed	
- \t (tab) - \n (newline) - \r (carriage return) - \f (form feed) - \v (vertical tab) - \x (hexadecimal escape) - \u (unicode escape) - \U (unicode escape)	

Lastly, stemming was applied, which converted words to their root form or stem. This preprocessing technique aimed to decrease the dimensionality of the data by reducing words to their base form. For instance, both "history" and "historical" were stemmed to "hisori". While stemming reduced the number of unique words processed by machine learning algorithms and is time-efficient on large datasets, it may have also led to a loss of contextual meaning, as the stemmed words may not have accurately reflected their original context.

Table 5: Comparison of review text before and after preprocessing.

Original Review Text	Post-processing Review Text
‘Every single mention of Oak’s hooves was a jumpscare’	‘everi,singl,mention,oak,hoov,jumpscar’

Chapter 4

4 Methodology

When building machine learning (ML) models on text data, it is necessary to vectorize the data. Vectorizing transforms the text data into a numerical and structured format. To classify sentiment on book reviews we explored various supervised learning algorithms: Naive Bayes, Logistic Regression, Random Forest, Support Vector Machines (SVM), k-Nearest Neighbors (kNN), AdaBoost, and a Recurrent Neural Network (RNN) model. Additionally, we implemented two techniques for feature extraction, Bag-of-Words (BoW), and Term Frequency-Inverse Document Frequency (TF-IDF). BoW represents the frequency of words in a document, while TF-IDF measures the importance of words in the document. By comparing these feature extraction techniques and models, this study aimed to determine the most effective approach for sentiment classification in book reviews. This methodology section provides an in-depth overview of the feature extraction, model building, and model tunning techniques employed.

4.1 Train Test Split

The data was randomly split according to the training/test split ratio, where 80 percent of the data was allocated to the training set and 20 percent to the test set. This split ensures that a sufficient amount of data is available for model training while also providing a separate dataset for evaluating the model's performance. Random splitting helps ensure that the characteristics of the review data are evenly distributed between the training and test sets, reducing the risk of bias in model evaluation.

4.2 Feature Extraction

4.2.1 TFIDF

Term Frequency - Inverse Document Frequency (TF-IDF) is a statistical method utilized in natural language processing and information retrieval to determine the significance of a term within a document relative to a collection of documents, known as a corpus.
TF-IDF transforms words within a text document into numerical importance scores through a process called text vectorization. It achieves this by multiplying the term's Term Frequency (TF) with the Inverse Document Frequency (IDF).

Term Frequency (TF) represents the frequency of a term within a document, calculated by dividing the number of times the term appears in the document by the total number of words in the document.
Inverse Document Frequency (IDF) reflects the rarity of a term across the corpus. Terms unique to a small percentage of documents, such as technical jargon, receive higher IDF scores compared to common words present in most documents, like "a", "the", and "and". The TF-IDF score of a term is obtained by multiplying its TF and IDF scores.

In summary a term's importance is high when it appears frequently within a specific document but infrequently across the entire corpus. TF measures the commonality of a term within a document, while IDF balances this by considering its rarity across documents. Consequently, the resulting TF-IDF score reflects the term's importance for a document within the corpus.

Equation 2: Term Frequency (TF) Equation

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

Equation 3: Inverse Document Frequency (IDF)

$$\text{IDF}(t, D) = lo\ g\left(\frac{\text{Total number of documents in the corpus } |D|}{\text{Number of documents containing term } t}\right)$$

Equation 4: Term Frequency-Inverse Document Frequency (TF-IDF)

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

Equations used to calculate the final TF-IDF score.

In this project, the corpus specifically refers to all the book reviews across the dataset. Each individual review within the dataset constitutes a document in the corpus. Therefore, the entire collection of book reviews within the dataset serves as the corpus for the TF-IDF vectorization process. This means that the TF-IDF scores are calculated for each term within each review relative to the entire collection of book reviews present in the dataset. The TF-IDF vectorizer is used on the training data and test data, thus essentially leveraging the entire body of book reviews within the dataset as the corpus for the analysis. Furthermore, the TF-IDF vectorizer was tuned using a Multinomial Naive Bayes classifier and Grid Search with cross-validation to identify the most optimal parameters for the model, ensuring effective feature extraction and classification performance.

4.2.2 Bag of Words

Bag of Words is the second document vectorization method used in this project. The Bag of Words (BoW) approach involves representing each document (book review) as a numerical vector based on the frequency of words present in the document. The BoW approach disregards the order of words and focuses solely on their occurrence within the document.
The BoW representation begins by constructing a vocabulary that includes all unique words across the entire corpus (collection of book reviews). Each unique word in the vocabulary becomes a feature in the vector representation.

For each document in the corpus (book review), a vector is created where each element corresponds to the frequency of a word from the vocabulary in that document. If a word appears multiple times in a document, its corresponding element in the vector will have a higher value indicating its frequency.

Therefore, in the BoW approach, each document is represented as a sparse vector, where most of the elements are zero, except for the elements corresponding to the words present in the document.

The Bag of Words approach is simple and effective for text classification tasks, as it captures the presence of important words in each document while disregarding word order and grammar. However, it is considered a less complex algorithm than TF-IDF due to its simpler calculation methods.

4.3 Model Architecture and Learning

4.3.1 Naïve Bayes

Naive Bayes is a probabilistic classifier based on Bayes' theorem, relying on a "naive" assumption of feature independence (IBM, n.d.). This assumption simplifies the calculation process by assuming that the presence of a particular feature in a class is unrelated to the presence of any other feature. Mathematically, Bayes' theorem is represented as:

Equation 5: Naive Bayes Theorem

$$P(y \mid x_1, \dots , x_n) = \frac{P(x_1, \dots , x_n) \cdot P(y \mid x_1, \dots , x_n)}{P(y)}$$

Where:

- $P(y \mid x_1,\dots,x_n)$ is the posterior probability of class y given predictors $x_1,\dots, x_n$.
- $P(y)$ is the prior probability of class y.
- $P(x_1,\dots,x_n \mid y)$ is the likelihood which is the probability of predictors given class y.
- $P(x_1,\dots,x_n)$ is the total probability of the predictors.

The training algorithm of a Naive Bayes Classifier uses text data transformed into numerical representations. The classifier estimates the probabilities required by Bayes' theorem using the training data, including class prior probabilities and conditional probabilities for each feature given each class.

During classification, the trained model predicts the class label for new instances by calculating the posterior probability of each class given the feature values. The model employs the maximum a posteriori (MAP) decision rule to assign the class label that maximizes the posterior probability.

4.3.2 SVM

The Support Vector Machine (SVM) used for classification tasks, was developed in the 1990s by Vatnik et al. SVMs classify data by finding an optimal hyperplane that maximizes the margin between classes.

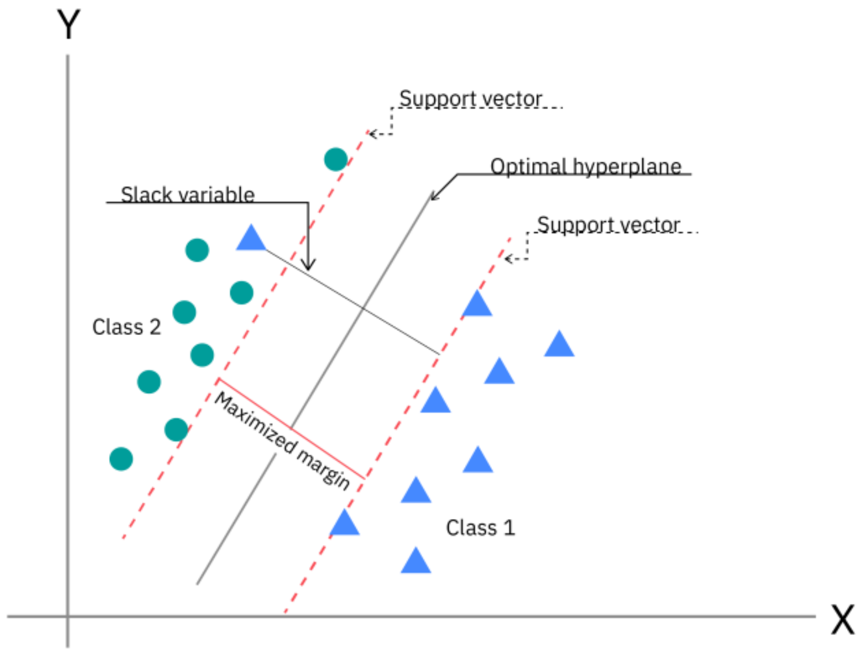


Figure 7: Graph representing SVM classification algorithm.

Linear SVMs are used with linearly separable data, without needing transformations, and rely on quadratic optimization problems. They distinguish classes by maximizing the margin, represented mathematically as $\boldsymbol{wx} + \boldsymbol{b} = \boldsymbol{0}$, where $\boldsymbol{w}$ is the weight vector, $\boldsymbol{x}$ is the input vector, and $\boldsymbol{b}$ is the bias term. The representation of the Maximized margin is portrayed in Figure 7 (IBM, 2023). Two approaches, hard-margin and soft-margin classification are used to calculate the margin. Hard-margin SVMs perfectly separate data points, while soft-margin classification allows for some misclassification. The choice of hyperparameter ***C*** adjusts the margin, controling the balance between margin width and misclassification. In the project, a linear SVM was utilized as part of the classification process.

4.3.3 Logistic Regression

Logistic regression estimates the probability (p) of an event based on independent variables, bounded between 0 and 1. It transforms odds using a logit function, represented by:

Equation 6: Logit Transformation applied to the odds formula.

$$\text{Logit}(p_i) = \frac{1}{1 + e^{-p_i}}$$

The features of the model are put into the linear regression equation which in turn is plugged in the logit function. The predicted probabilities are calculated by summing conditional probabilities for each observation. Afterwards, the entire equation is optimized by the finding of the best coefficients (β) with the use of the maximum likelihood estimation method.
For multinomial predictions, logistic regression employs the "one-vs-rest" approach, where separate models are trained for each class against the rest. Each model predicts the probability of belonging to its respective class, and the class with the highest probability is chosen as the final prediction. This method extends logistic regression to handle multiple classes effectively, allowing for the classification of observations into more than two categories.

Equation 7: Multinomial Logistic Regression Equation.

$$\ln\left(\frac{P(Y = k)}{P(Y = K)}\right) = \beta_{k0} + \beta_{k1}X_1 + \cdots + \beta_{kp}X_p$$

4.3.4 Random Forest

Random forest is a machine learning algorithm designed to address the shortcomings of individual decision trees (Freund & Schapire, 1997). Decision trees, while intuitive, can suffer from overfitting due to their tendency to memorize the training data. In contrast, random forests utilize ensemble learning by combining the outputs of multiple decision trees (Breiman, Friedman, Olshen, & Stone, 1984). Each tree is trained on a random subset of the data, created by a sampling method called bootstrapping, and considers only a subset of features at each node, known as feature bagging. As a result, it introduces variability and reduces correlation among the trees, resulting in more robust predictions. With random forests, hyperparameters such as number of leaf nodes, the number of trees, and the number of features sampled can be fine-tuned to optimize performance.

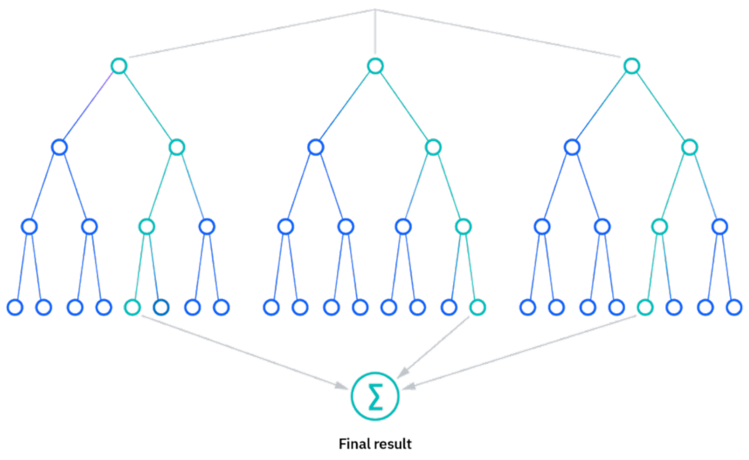


Figure 8: Graph representing Random Forest classification algorithm. (IBM, 2023)

After training, predictions are made by averaging outputs for regression or by majority vote for classification.

4.3.5 AdaBoost

AdaBoost, is an ensemble-based algorithm that improves the performance of weak learners by combining them into a strong classifier (Freund & Schapire, 1997). This is achieved through an iterative process where each weak learner focuses on instances that are difficult to classify for the previous learner, incrementally improving the overall predictive accuracy of the model. The algorithm begins by assigning equal weights to all training instances, treating them with equal importance.

During each iteration, a weak classifier, often a heavily regularized decision tree, is trained on the weighted training data. The classifier aims to minimize the weighted error rate, where the weights reflect the importance of each instance. AdaBoost then calculates the classifier's weight based on its ability to correctly classify instances, with higher weights assigned to more accurate classifiers. This weight α_t is calculated using the equation:

Equation 8: AdaBoost Weight Equation for Classifiers

$$\alpha_t = \frac{1}{2} ln\left(\frac{\epsilon_t}{1-\epsilon_t}\right)$$

Where:

- ϵ_t represents the weighted error rate of the t-th weak classifier.

After training each weak classifier, AdaBoost adjusts the weights of training instances. It increases the weights of misclassified instances, making them more influential in subsequent iterations, while decreasing the weights of correctly classified instances. This process focuses the subsequent classifiers on the instances that were previously difficult to classify.

The final prediction is made by combining the predictions of all weak classifiers through weighted majority voting. Each weak classifier's contribution to the final decision is weighted based on its performance, represented by the classifier's weight α_t. The algorithm continues the iterative process until a predefined number of weak classifiers have been trained or until a satisfactory level of performance is achieved.

4.3.6 KNN

K-Nearest Neighbors (KNN) algorithm uses a majority voting mechanism. Its foundational principle lies in the belief that akin instances often share similar outcomes (Gongde, Wang, Bell, & Bi, 2004). For classification, KNN assigns a class label to a new data point based on the majority class affiliation of its proximal neighbors.

KNN begins by archiving the training data. Then for each new data point, it calculates distances to all other training instances utilizing designated distance metrics. For this project we utilized the Euclidean distance. For two points p and q in an n-dimensional feature space, the Euclidean distance is calculated using the following formula:

Equation 9: Euclidean distance equation for KNN.

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

The algorithm then identifies the K nearest neighbors, where "K" represents the hyperparameter to be tuned. In KNN classification, the determination of a new data point's class label $\hat{y}$ hinges on a majority vote among its K nearest neighbors. Let y_i denote the class label of the i^{th} nearest neighbor of x. The expression for $\hat{y}$ is:

Equation 10: KNN predicted value equation.

$$\hat{y} = arg\,\max_{j} \sum_{i=1}^{K} w_i \cdot I(y_i = j)$$

Where:

- $I(\cdot)$ serves as the indicator function, returning 1 if the condition holds true and 0 otherwise.

- w_i represents optional weights assigned to the neighbors, usually uniform weights ($w_i = 1$)

- j traverses all unique class labels in the dataset.

4.3.7 Recurrent Neural Network

A recurrent neural network (RNN) is a type of deep learning model designed to process sequential data and generate sequential or numeric outputs (Lipton, Berkowitz, & Elkan, 2015). The sequential data for this project was review text, where the order of elements matters. RNNs consist of interconnected neurons organized into input, output, and hidden layers. They work by passing sequential data through the hidden layers one step at a time, utilizing a self-looping mechanism to remember and use previous inputs for future predictions.

During training, RNNs adjust their weights through a process called backpropagation through time (BPTT), where errors are calculated and used to update the model's parameters.

At the output layer of an RNN, we used the softmax function to convert the raw output scores into probabilities. The softmax function takes a vector of arbitrary real-valued scores as input and outputs a vector of probabilities that sum to one. The softmax function is defined mathematically as follows:

Equation 11: Softmax Equation

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

Where:

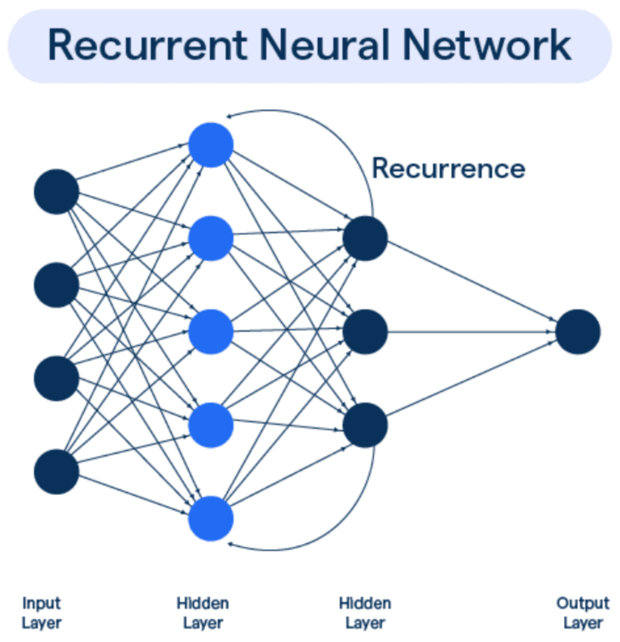
- $\sigma(z)_i$ is the i-th element of the output probability vector.
- z_i is the i-th element of the input vector (logits).
- N is the total number of classes.

The softmax function ensures that the output probabilities are normalized and interpretable, making it easier to interpret the model's predictions. During training, the model learns to optimize its parameters to maximize the likelihood of the correct class labels given the input data.

One of the challenges with traditional RNNs is the vanishing gradient problem, where gradients become extremely small during training, leading to difficulties in learning long-range dependencies. To address this issue, we utilized Long Short-Term Memory (LSTM) architecture, implemented by the Keras library in Python, which are a type of RNN architecture designed specifically to overcome the vanishing gradient problem (Hochreiter & Schmidhuber, 1997). LSTM networks incorporate specialized mechanisms for gating and memory management, enabling them to effectively capture long-term dependencies and mitigate the vanishing gradient problem. By utilizing LSTM layers in our model, we can ensure that the network is better equipped to retain and update information over longer sequences, making it more capable of capturing the complex relationships and dependencies present in text data.

The RNN requires all inputs to be the same length. Thus, the tokenized word review text data was converted into numerical tokens, selecting only to keep the 1000 words that appear most often in the corpus. To be passed in as input to the RNN, the sequences were padded to the length of 50, truncating all sequences longer than that and adding padding to those shorter. The output of the network was a 3-neuron layer representing each sentiment class.

The RNN model was trained using the Adam optimizer, which computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the



F

gradients (Kingma & Ba, 2015). This adaptive learning rate helps the model learn better, especially with LSTM layers, where it's crucial to have an optimizer that can handle different learning rates effectively. In the provided RNN, the Adam optimizer adjusted the learning rate during training, mitigating the vanishing gradient problem and ensuring effective learning. The simplified diagram of the RNN structure can be seen in Figure 9 (BotPenguin, n.d.):

4.4 Model Fine Tuning

4.4.1 Grid search CV

The Logistic Regression, Random Forest, AdaBoost, and KNN models were initially trained with default parameters. After that, grid search with 5-fold cross-validation was used to tune the hyperparameters of the models to increase performance accuracy on the test data. Grid Search CV divided the training dataset into training and validation sets. In this case, it performed 5-fold cross-validation (cv=5), dividing the dataset into 5 parts; four-fifths of the data were used for training, and one-fifth was used for validation. This process was repeated five times so that each subset of the data was used as the validation set exactly once. It was performed for each combination of hyperparameters provided in param_grid, and for each combination, grid search built a model. After evaluating all models, Grid Search CV selected the set of hyperparameters that produced the best result (in this case, the highest accuracy) on the validation data. Finally, the best model was fitted to the entire training dataset using the optimal hyperparameters.
Using Grid Search with cross-validation ensured the model was trained with the optimal set of hyperparameters, thus increasing its performance. The model generalized better to unseen data since it was evaluated across multiple folds, ensuring consistent performance across different subsets of data. The risk of overfitting was reduced as the model was assessed on different subsets of the data, and the best hyperparameters were chosen. Similarly, underfitting was mitigated by exploring various hyperparameter combinations.

We were able to successfully tune the KNN and Logistic Regression models. The Naive Bayes model does not take in extra hyperparameters; thus, its performance could not be improved through hyperparameter tuning. The SVM, Random Forest, and AdaBoost models were not tuned due to computational limitations as the dataset was too large and created classifiers that took an unreasonable amount to time for hyperparameter tuning.

4.4.2 RNN Model Tuner

We used the Keras Tuner to tune the hyperparameters of the RNN LSTM model. The custom model_builder function was defined to create four LSTM layers with specified hyperparameter ranges and values. The model was set up with an embedding layer, followed by a variable number of LSTM layers, and ended with two dense layers. The hyperparameters, including activation functions, the number of LSTM units, and the learning rate, were defined. Keras Tuner then searched through different combinations of hyperparameters (O'Malley, et al., 2019), using the defined model_builder function, to maximize the validation accuracy. Finally, the best hyperparameters found by the tuner were used to compile the model.

4.5 Evaluation Metric

Accuracy was chosen as the main metric to evaluate the performance of the sentiment classification model. Accuracy represents the ratio of correctly predicted sentiments to the total number of sentiments and is calculated using the formula:

Equation 12: Accuracy Equation

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100\%$$

By using accuracy as the main metric, it provides a straightforward and intuitive measure of the model's overall performance. Moreover, it allows for a direct comparison with the results of other sentiment classification studies, enabling a comprehensive evaluation of the model's effectiveness in sentiment analysis tasks. This approach ensures that the results obtained from this study are not only reliable but also directly comparable to similar research.

Chapter 5

5 Results and Discussions

5.1 Comparing BoW vs TF-IDF

To investigate and compare the performance of the vectorizers used, each model was trained with default parameters on both BoW data and TF-IDF data. We used accuracy as a comparison metric in determining which vectorizer resulted in better-performing models. Accuracy was calculated on both the test and train data; the values can be seen in Table 6.

Table 6: Accuracy metric for model performance on train and test data.

ACCURACY ON TRAIN DATA		
Model	BOW	TFIDF
Naïve Bayes	0.76	0.99
Logistic Regression	0.87	0.97
SVM	0.97	*
KNN	0.64	0.69
Random Forest	0.99	0.99
AdaBoost	0.59	0.59

ACCURACY ON TEST DATA		
Model	BOW	TFIDF
Naïve Bayes	0.60	0.62
Logistic Regression	0.60	0.65
SVM	0.57	0.63
KNN	0.46	0.46
Random Forest	0.56	0.57
ADA Boost	0.58	0.58

*.*SVM was not calculated on TFIDF train data due to computational restraints.*

Although both vectorization methods achieved similar performances across the test data, TF-IDF achieved equal or greater accuracy for each of the models. Furthermore, TF-IDF performed with significantly higher accuracy on the training data as well. However, based on the values for Naïve Bayes, Logistic Regression, and Random Forest, it is evident that the default models overfitted greatly, indicated by the accuracy close to 100%. TF-IDF is more complex than BoW; thus, the models might have trained too closely to fit the more complex data, lacking the ability to generalize well to unseen -test- data.

5.2 Performance of tuned models

To get a more detailed look into the tf-idf, which was the best feature extraction option, various models were evaluated based on their accuracy, precision, and recall. Table 7 presents the performance metrics for each model. Among these models, the Logistic Regression model with TF-IDF achieved the highest performance with respect to all three metrics. This highlights the effectiveness of TF-IDF as a feature extraction method in improving the overall performance of the Logistic Regression model in this context.

Table 7: Accuracy, Precision and Recall on Test Data for each model.

	Accuracy	Precision	Recall
Naïve Bayes	0.61	0.62	0.62
Logistic Regression	0.65	0.65	0.65
SVM	0.63	0.62	0.62
KNN	0.58	0.58	0.57
Random Forest	0.57	0.56	0.57
ADA Boost	0.58	0.57	0.57
RNN	0.55	0.55	0.55

5.3 Misclassified reviews sentiment and polarity

Although the performance of the best model was comparable to the benchmarks established in the literature review (referencing the paper), it was still significantly lower than that of some of the other studies that utilized the same vectorization and models to classify sentiment.

One potential factor could be the nature of book review texts. Other studies focused on product reviews and Twitter data, which vary significantly in structure from book review texts. To investigate further, we looked into the test dataset reviews that were misclassified by all the best-performing versions of Naïve Bayes, Logistic Regression, KNN, Random Forest, and AdaBoost models. Out of the 9349 reviews in the test data, 1634 of the reviews had their sentiment misclassified by all five of the models. Among the book titles present in the test data, "Spare" by Prince Harry, an autobiography published in 2023, had the highest number of misclassified reviews. A few other notable titles with the highest number of misclassified reviews can be seen in Table 8.

Table 8: Books with highest number of misclassified reviews by the models.

Book Title	Review Count	Author	Maine Genre	Year Published
Spare	7	['Prince Harry', 'J.R. Moehringer']	‘Nonfiction’	2023
Chain-Gang All Stars	5	['Nana Kwame Adjei-Brenyah']	‘Fiction’	2023
The Wishing Game	5	['Meg Shaffer']	‘Fiction’	2023
Upgrade	5	['Blake Crouch']	‘Science Fiction’	2022
Surely You Can't Be Serious: The True Story of Airplane!	5	['David Zucker', 'Jerry Zucker', 'Jim Abrahams']	‘Nonfiction’	2023

Due to the Goodreads tagging system, each book receives multiple genre tags. For the sake of the analysis, we have chosen the first tag to represent the main genre of the text as it is the tag that also represents the genre under which the book was nominated for Goodreads Choice Awards. Since the whole dataset had an equal number of reviews from each genre category and the data was split to train and test at random, we have assumed the genre distribution will also be equally distributed between test and train datasets. Based on this, we have looked at the genres of the reviews misclassified by all models, which can be seen in Table 9.

Table 9: Genres with highest count of misclassified reviews by all models.

Genre	Review Count
Fantasy	107
Nonfiction	79
Fiction	55
Romance	46
Historical Fiction	35

To gain further understanding of any possible trends among the misclassified reviews, we have also investigated the distribution of true sentiment label vs. predicted sentiment label in Figure 10.

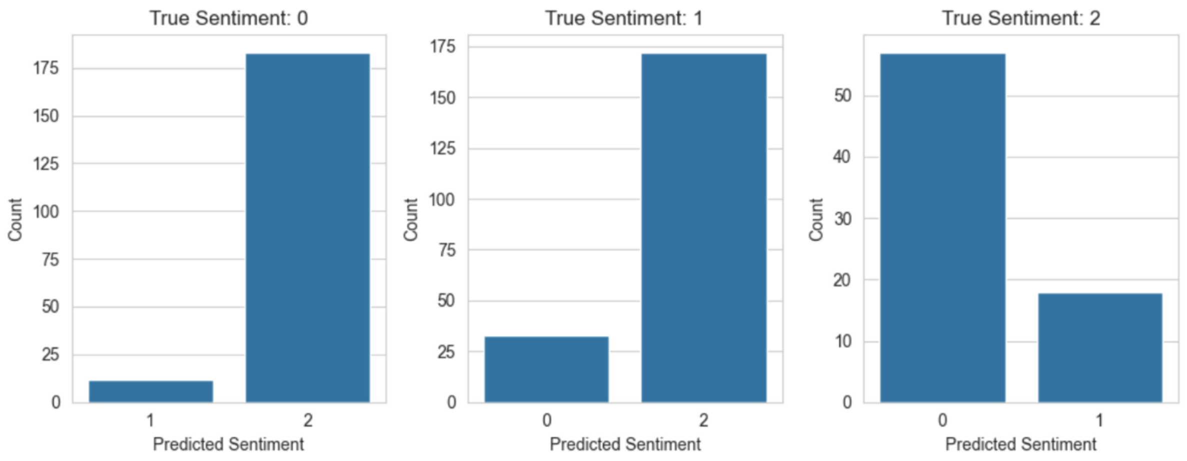


Figure 10: Distribution of True vs Predicted Sentiment of Misclassified Reviews.

Interestingly, many of the reviews with true negative or neutral sentiment were misclassified as positive sentiment, whereas true positive sentiment was primarily misclassified as negative. This raises an important issue considering that most reviews were misclassified to be the opposite sentiment, versus the sentiment closer in value.

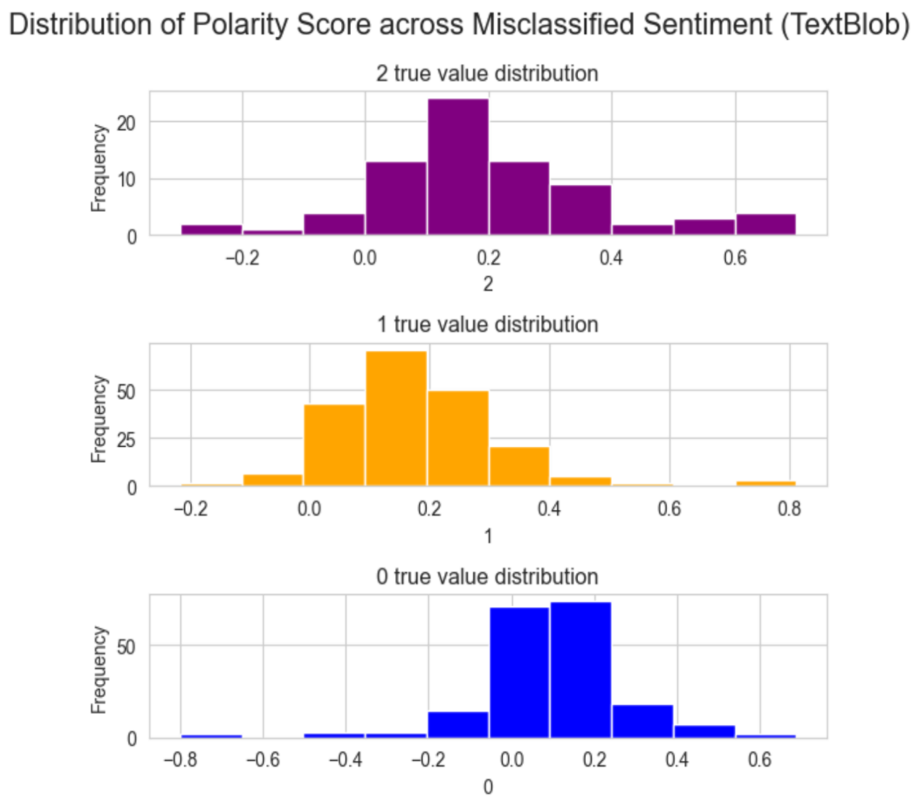


Figure 11: Distribution of Polarity scores vs True Sentiment of Misclassed Reviews.

Upon looking at the distribution in Figure 11, of the sentiment values for each true sentiment label, it was clear one of the issues causing the severe misclassification is the similarity of reviews across all three sentiments in their polarity. Based on other studies referenced in the paper that included polarity scores, their distributions were much more varied and closely mapped to the true sentiment label. The severe similarity of polarity for the misclassified ratings seems to be caused by the phrasing and workings of Goodreads reviews feature. Unlike in product reviews where the goal of the review is to convey the sentiment about the product and describe its usability, if the product seems similar to the description and fits the requirements, Goodreads reviews appear to be much more subjective. On average, Goodreads reviews vary in length, but many of them tend to be on the shorter side as seen in Figure 12:

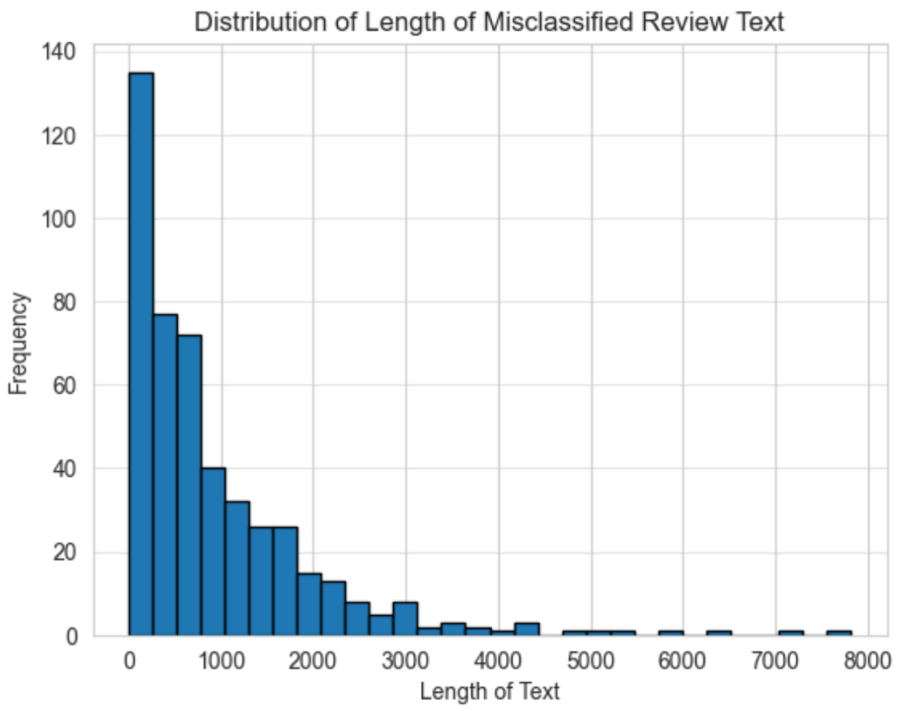


Figure 12: Distribution of Length of Misclassified Reviews after preprocessing.

Some of the main features of Goodreads reviews are quotes and descriptions of the plots of the books. This may lead to misclassification as the plot summary may use negative words due to the contents of the book being as such, but the actual sentiment is positive as the reader thoroughly enjoyed it as seen in Table 10.

Table 10: Example review text which features negative content and words (in bold) in an overall positive review.

Review Text	Predicted Value	True Value
'Oh my god! Amazing and brilliant! Brave and compelling! And that ending collision ~~~~~~~~~~~~~~~~~~When I turned the final page, I sat and tried to absorb the feelings and emotions from this impactful and powerful story. Harris has taken a story that needed to be told and **ripped** my heart to shreds. Fair **warning** – you need to be open and willing to read about the aftermath of a sexual **assault**. Trust me when I say, I was **nervous** too, but this author takes a **tragedy** and creates a miracle. It is truly a stunning story. Harris has taken the raw and gritty and gifted us with heart, compassion, healing, forgiveness, and love. Set aside a quiet weekend and read this with tissues and an open mind. One Summer in Savannah and Harris is amazing and brilliant! Brave and compelling! And that ending is nothing short of bomb-tastic! This is the best read of 2023 so far! Harris has just put her name in lights! * Copy received for review consideration*	0	2

Furthermore, Goodreads reviews do not have a direct goal such as product reviews, which aim to recommend or dissuade someone from purchasing the item. Instead, their purpose varies. Many users use the reviews as a personal diary of sorts, keeping track of their feelings about the book through a short phrase that does not mean much to other readers. However, the spectrum of reviews is wide as there are also sponsored/gifted reviews where the reviewer is writing the review with intentions to inform others about if the book is worth reading, their general thoughts, etc.

Chapter 6

6 Conclusions

In this thesis, the performance of six Machine Learning Models: Naive Bayes, Logistic Regression, Random Forest, Support Vector Machines (SVM), k-Nearest Neighbors (kNN), AdaBoost, and a Recurrent Neural Network (RNN) was investigated for classifying sentiment in Goodreads book review texts. Additionally, the experiment included the use of two feature extraction methods, TF-IDF and BoW, to investigate their impact on the models' performance. The primary tuning and evaluation metric used was accuracy; however, the final models were also compared based on precision and recall on both test and train data for further insights. The real dataset was collected from thousands of book reviews on the Goodreads website, as well as information about the books themselves. From the results achieved in this thesis, the following conclusions can be drawn:

- TF-IDF consistently achieved equal or greater accuracy than BoW across all models.
- Logistic Regression with TF-IDF emerged as the top-performing model, achieving the highest accuracy, precision, and recall rates, all at 65%.
- Although the overall accuracy was promising, it was lower compared to benchmarks in other studies cited in the literature review, reflecting the unique challenges posed by the structure of book review texts.
- The complexity of review text data is attributed to its varied structure and readers' focus on describing the book's content rather than its specific features in the reviews.
- Given the similarity of polarity scores across all sentiment categories, it suggests the need for alternative methods of sentiment classification that do not rely solely on star ratings.

6.1 Future Work

To tackle the challenge of sentiment mapping, an alternative approach to sentiment labeling can be employed. VADER, a lexicon and rule-based sentiment analysis tool tailored for interpreting emotions in text, assesses negative, neutral, and positive sentiments within the input text, yielding a composite score (Hutto & Gilbert, 2015). This score serves as a robust indicator for categorizing sentiment classes. Extensive research has demonstrated VADER's effectiveness, often outperforming traditional review star ratings and offering more dependable sentiment analysis outcomes (Adarsh, Patil, Shubham, & Veena, 2019).

To further develop models for accurate sentiment classification pre-trained models such as BERT could be deployed (Devlin, Chang, Lee, & Toutanova, 2019). BERT (Bidirectional Encoder Representations from Transformers) models are a type of neural network architecture designed for natural language processing tasks. BERT models have had success in past studies at accurately capturing complex patterns in review sentiment (Sayeed, Mohan, & Anbananthen, 2023). They are pre-trained on large text corpora and can be fine-tuned for specific NLP tasks like text classification, question answering, and language generation. BERT models are known for their ability to capture bidirectional context and semantics in text. In turn providing them with the ability to understand the nuanced language often found in literary reviews. For example, BERT can discern subtle differences in sentiment, tone, and review writing style that may vary widely across different genres and authors. This enables more accurate sentiment analysis and classification of book reviews, even when they deviate from standard patterns or contain complex expressions.

7 Bibliography

(n.d.). Retrieved from https://www.goodreads.com

Adarsh, R., Patil, A., Shubham, R., & Veena, K. M. (2019). Comparison of VADER and LSTM for Sentiment Analysis. *International Journal of Recent Technology and Engineering (IJRTE)*.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit.* Retrieved from https://www.nltk.org

BotPenguin. (n.d.). Retrieved from Recurrent Neural Network: https://botpenguin.com/glossary/recurrent-neural-network

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Wadsworth, Belmont, CA.

Devaki, P., Lokesh, K. R., Muthukumar, S., & Shree, K. A. (2022). *Sentiment Analysis and Recommendation of Book Reviews.* Bangalore: IEEE.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*

Freund, Y., & Schapire, R. (1997). *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.*

Gongde, G., Wang, H., Bell, D. A., & Bi, Y. (2004). *KNN Model-Based Approach in Classification.*

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 1735–1780.

Hutto, C. J., & Gilbert, E. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *Proceedings of the 8th International Conference on Weblogs and Social Media*.

IBM. (2023, 12 27). Retrieved from What are support vector machines (SVMs)?: https://www.ibm.com/topics/support-vector-machine#:~:text=A%20support%20vector%20machine%20(SVM,the%201990s%20by%20Vladimir%20N.

IBM. (2023, 12 27). Retrieved from What are support vector machines (SVMs)?: https://www.ibm.com/topics/support-vector-machine

IBM. (n.d.). *What are Naïve Bayes classifiers?* Retrieved from https://www.ibm.com/topics/naive-bayes

Janardhana, D. R., Vijay, C. P., Swamy, G. B., & Ganaraj, K. (2020). Feature Enhancement Based Text Sentiment Classification using Deep Learning Model. *2020 5th International Conference on Computing, Communication and Security (ICCCS)*, 1-6.

Kaur, V. D. (Bangalore). *Sentimental Analysis of Book Reviews using Unsupervised Semantic Orientation and Supervised Machine Learning Approaches.* 2018: IEEE.

Kim, T., & Wurster, K. (2015). Retrieved from https://carpedm20.github.io/emoji/docs/

Kingma, D. P., & Ba, J. L. (2015). *ADAM: A Method For Stochastic Optimization.* San Diego.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). *A Critical Review of Recurrent Neural Networks for Sequence Learning.* San Diego.

Loria, S. (2018). *Simplified Text Processing*. Retrieved from https://textblob.readthedocs.io/en/dev/.

Luţan, E. R., & Bădică, C. (2022). Emotion-Based Literature Book Classification Using Online Reviews. *Electronics*.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., & Invernizzi, L. (2019). *KerasTuner*. Retrieved from https://github.com/keras-team/keras-tuner

Patel, A., & Tiwari, A. K. (2019). Sentiment Analysis by using Recurrent Neural Network. *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) 2019*.

Richardson, L. (2007, April). *Beautiful soup documentation.* Retrieved from https://beautiful-soup-4.readthedocs.io/en/latest/

Sayeed, M. S., Mohan, V., & Anbananthen, K. S. (2023). BERT: A Review of Applications in Sentiment Analysis. *HighTech and Innovation Journal*.

Tan, K. L., Lee, C. P., & Lim, K. M. (2023, April 3). *A Survey of Sentiment Analysis: Approaches, Datasets, and Future Research*. Retrieved from Applied Sciences: https://doi.org/10.3390/app13074550

Van Rossum, G. (2020). Retrieved from The Python Library Reference: https://docs.python.org/3/library/re.html

Wang, K., Liu, X., & Han, Y. (2019). Exploring Goodreads reviews for book impact assessment. *Journal of Informetrics*, 874-886.

Zajac, Z. (2017). *https://github.com/zygmuntz/goodbooks-10k*. Retrieved from https://github.com/zygmuntz/goodbooks-10k

Zhu, Z. (2013). *Muti-document summarization of online book reviews based on information classification* . Nanjing: Nanjing University Press.