THE OHIO STATE UNIVERSITY

# Classifying Book by Genre Based on Reader Reviews with Machine Learning

Project Category: Text
Physics 5680, Autumn 2024

Author: Jenna Bittner

December 12, 2024

**Abstract**

This project investigates the use of machine learning for classifying book reviews into genres, with the goal of enhancing recommendation systems. A Goodreads dataset containing book reviews is preprocessed with tokenization, stopword removal, and lemmatization. The text is transformed into numerical features using TF-IDF, and SMOTE is applied to address class imbalance. A Random Forest classifier is trained and evaluated using metrics such as precision, recall, and F1-score. Results reveal strong performance for genres like *comics, graphic* and *romance*, while the model struggles with less represented genres, such as *poetry* and *young-adult*. The overall accuracy of the model is 0.51, with macro and weighted average scores reflecting a bias toward larger classes. This highlights the challenges of classifying imbalanced and nuanced categories and suggests the need for further optimization.

## 1 Introduction

In today's digital age, the abundance of books available can make finding books that match a reader's preference a daunting task. Traditional methods of finding a book, such as relying on ratings or similar books, can often fail to capture the diverse and evolving tastes of readers. An intelligent book recommendation system could be the solution. A critical component of building an accurate book recommendation system is the automatic classification of book genres from reviews. By effectively categorizing books into genres, personalized book recommendations can be enhanced, allowing readers to easily discover books that align with their individual interests.

The motivation for this project stems from the growing demand for a recommendation system that can aid readers in navigating vast digital libraries. While many recommendation systems already exist, classification techniques based on user-generated book reviews would provide a more precise way of categorizing books that can contribute to a more effective recommendation system.

This project's primary objective is to develop a genre classification model that takes textual reviews as input and predicts the genre of a book. The algorithm's input consists of book reviews, which are processed through text preprocessing, sentiment analysis, and machine learning techniques, such as TF-IDF vectorization. The output is the predicted genre of each book. This genre classification system will be a key component in a broader book recommendation system.

## 2 Related Work

In the paper *A Survey on Book Genre Classification System using Machine Learning* by Dr. M. Narendra et al., various Natural Language Processing (NLP) methods are discussed, including traditional machine learning models like Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Logistic Regression, along with the newer FastText model. While the paper uses entire book synopses as input, the same NLP methods can be applied to book reviews. FastText achieved impressive results with precision of 0.92, recall of 0.93, and an F-value of 0.92, suggesting its effectiveness in text classification tasks where word embeddings are key. However, the paper does not provide a full discussion on the process of building a genre classifier, which could offer insights on fine-tuning these algorithms.

The Project Gutenberg Book Genre Classifier by Ryan W. West uses decision trees, logistic regression, and random forests to classify texts by genre. While it provides accessible source code, its reliance on simpler models like decision trees may limit its ability to capture complex relationships in larger datasets such as Goodreads. West reported an accuracy of over 0.80, though the models' simplicity and lack of advanced techniques may hinder their effectiveness compared to more complex approaches like neural networks.

The ML-NLP-Goodreads-Book Classification project offers a detailed guide on creating a genre predictor from reviews, using a range of models like LSTM, SVM, Random Forests, Naive Bayes, and BERT. BERT, a transformer-based model, is notable for its ability to capture contextual relationships between words, which improves accuracy in genre classification. However, transformer models require significant computational resources, making them less accessible for users with limited resources.

Recent advancements in NLP, particularly transformer models like GPT-3 and T5, have shown remarkable improvements in text classification. These models leverage large-scale pretraining, enabling a deeper understanding of context, which is particularly useful for tasks involving reviews that contain sentiment and style. These models could significantly enhance classification accuracy, given sufficient computational resources.

## 3 Dataset

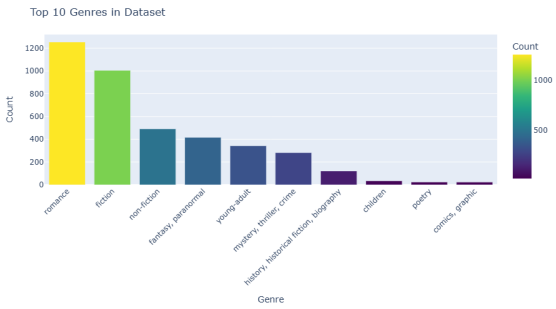
The dataset for this project is sourced from Mengting Wan's Goodreads dataset, specifically the Goodreads book reviews and genres. It includes about 15 million reviews for 2 million books. For this project, 10,000 books are randomly selected, filtered to exclude those with missing genres, and split into training and testing data. The same books are matched with their genres using book IDs and merged into a single dataset. While books may have multiple genres, this project focuses on a primary genre.

To prepare the data for training, the Natural Language Toolkit (NLTK) is used for preprocessing. Reviews are tokenized, converted to lowercase, and cleaned by removing stop words, non-alphanumeric characters, and short words. Lemmatization is applied to reduce words to their base forms. The cleaned text is then vectorized using the Term Frequency-Inverse Document Frequency (TF-IDF) technique via Scikit-Learn, retaining up to 1,000 features to ensure model efficiency. The reviews are then in a form suitable for machine learning, with noise reduced and meaningful words highlighted.

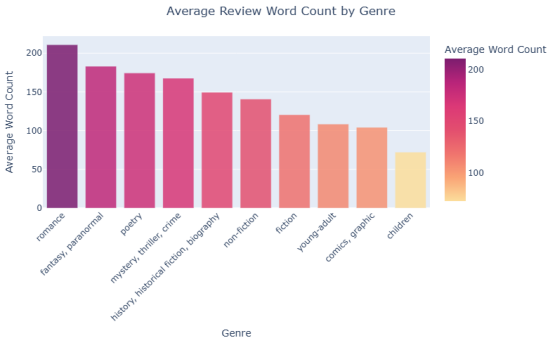
The input data consists of text reviews of books, where each review is a string of text that provides an opinion on the book. The reviews are processed and represented numerically using TD-IDF vectorization. The output of the data is the predicted genre of the book. Each book in the dataset is assigned a primary genre, which the model is tasked with predicting based on the review text.

Additionally, before modeling SMOTE was applied to the data. SMOTE (Synthetic Minority Over-sampling Technique) is a technique used to address class imbalance in datasets by generating synthetic examples rather than by duplicating existing examples.

Some visualizations were created to explore the data in Figure 1.




(a) This graph displays the distribution of genres in the dataset. The genres are ranked by their count, with the most common on the left. This highlights a class imbalance in which genres like romance and fiction make up a majority of the data, and children, poetry, while comics, graphic are clearly underrepresented.




(b) This graph illustrates the average word count of reviews for each genre. Romance receives the longest reviews on average, and it is a steady decline until children, which receives the shortest reviews. This visual reveals trends in how readers tend to be when reviewing books in each category.

Figure 1: Comparison of genre distribution and review length by genre.

# 4 Methods

## 4.1 TD-IDF Vectorization

Before using any model, the book reviews must be transformed into a numerical format. TF-IDF, Term Frequency-Inverse Document Frequency Vectorization, helps in transforming raw text data into a numerical format that allows machine learning algorithms to analyze the text effectively. Each word in the text gets a TF-IDF score that reflects its importance relative to the entire dataset within a document. The higher the score, the more important the word is. The score is a combination of Term Frequency (TF) and Inverse Document Frequency (IDF). TF measures how often a particular word, $w$, appears in a document, $d$. This metric quantifies the frequency of the term. The IDF measure how common a word is across the entire set of documents. Words that appear in many documents are considered less informative, while words that appear in few documents are considered more informative.

$$\text{TF} = \frac{\text{Number of occurrences of } w \text{ in } d}{\text{Total number of words in } d} \qquad \text{IDF} = \log\left(\frac{\text{Total number of } d}{\text{Number of } d \text{ in which } w \text{ appears}}\right)$$

With values for both TF and IDF, the TF-IDF score for a word can be calculated: $\text{TF-IDF} = \text{TF} \times \text{IDF}$

This final score is a numerical value that represents the importance of a word within the entire dataset.

## 4.2 Random Forest Classifier

To achieve genre classification, the machine learning algorithm utilized is a Random Forest classifier, an ensemble method that combines the prediction of multiple decision trees to improve robustness. This algorithm works well for classification tasks by averaging the predictions of many diverse trees. Random forest is trained using bootstrap sampling, a technique used to create multiple subsets of the training data by sampling it with replacement, allowing the same object to be chosen multiple times. For each subset, a decision tree is constructed. At each node in a tree, the algorithm evaluates the best way to split the data based on the Gini impurity criterion, $G$. The following equation defines the Gini impurity criterion, where $p_k$ is the proportion of samples belonging to class $k$, and $K$ is the total number of classes:

$$G = 1 - \sum_{k=1}^{K} p_k^2$$

The Gini impurity split aims to maximize separation between genres, and the split that minimizes $G$ is chosen. Each tree in the forest is grown until a stopping condition is met, which could be maximum depth, minimum samples per leaf, etc. For each review, each tree will predict a genre. The forest aggregates these predictions using majority voting where $\hat{y}$ is the predicted genre, $T$ is the total numbers of trees in the forest, $y_i$ is the genre predicted by the i-th tree, and $I$ is the indicator function which is equal to 1 if the tree predicts class k and 0 otherwise.

$$\hat{y} = \arg\max_k \sum_{i=1}^{T} I(y_i = k)$$

The result is the final predicted genre.

# 5 Results/Discussion

## 5.1 Hyperparameters

The hyperparameters for the Random Forest Classifier were selected through a combination of understanding their functionality and experimentation. The number of trees (`n_estimators`) was set to 500, balancing model performance and computational efficiency. The maximum tree depth (`max_depth`) was left as `None`, allowing the trees to grow fully and capture complex patterns, with bootstrapping mitigating overfitting. The minimum samples required to split a node (`min_samples_split`) was set to 2, enabling detailed learning while leveraging Random Forest's robustness to avoid overfitting. To ensure sufficient data at each leaf node, `min_samples_leaf` was set to 4. The number of features considered for each split (`max_features`) was set to `log2`, promoting diversity among the trees by reducing feature correlation. Bootstrapping (`bootstrap`) was enabled to introduce randomness and enhance model robustness. The Gini impurity criterion (`criterion`) was used to evaluate splits, ensuring low impurity in nodes. Finally, `random_state` was set to 42 for reproducibility.

## 5.2 Metrics

| **Class** | **Precision** | **Recall** | **f1-Score** | **Support** |
|---|---|---|---|---|
| children | 0.45 | 0.68 | 0.54 | 819 |
| comics, graphic | 0.56 | 0.75 | 0.64 | 1212 |
| fantasy, paranormal | 0.51 | 0.42 | 0.46 | 3598 |
| history, historical fiction, biography | 0.52 | 0.33 | 0.40 | 2478 |
| mystery, thriller, crime | 0.49 | 0.46 | 0.47 | 2181 |
| non-fiction | 0.53 | 0.60 | 0.56 | 2345 |
| poetry | 0.14 | 0.66 | 0.23 | 191 |
| romance | 0.57 | 0.65 | 0.61 | 3926 |
| young-adult | 0.46 | 0.31 | 0.37 | 2189 |
| | | | | |
| accuracy | | | 0.51 | 18939 |
| macro avg | 0.47 | 0.54 | 0.48 | 18939 |
| weighted avg | 0.52 | 0.51 | 0.50 | 18939 |

Table 1: This classification report presents the precision, recall, f1-scores and support for each genre. Additionally, it includes average performance metrics across all genres.

The above table summarizes the performances of the classification model across all genres using three key metrics: precision, recall, and f1-score. Precision measures the proportion of of positive predictions that are actually correct. A high precision indicates that the model produces few false positives. Recall measures the proportion of actual positives that are correctly predicted. A high recall indicates that the model captures

most of the relevant instances. For a given genre, precision and recall are calculated as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \qquad \text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The f1-score is the harmonic mean of precision and recall, providing a single measure that balances both. High f1-scores indicate there is a good balance between precision and recall, and the model is overall doing its job well. It is calculated as:

$$\text{f1-Score} = 2 \times \frac{\text{Precision} + \text{Recall}}{\text{Precision} \times \text{Recall}}$$

The model's performance varied across different classes. It achieves its strongest results in the *comics, graphics* and *romance* classes, with f1-scores of 0.64 and 0.61 respectively. This indicates both high precision and high recall for both classes. *Non-fiction* also exhibited solid performance with an f1-score of 0.56. On the other hand, the model really struggles with classes like *poetry* and *young-adult*. The *poetry* class in particular shows a very low precision of 0.14, suggesting frequent false positives despite a relatively high recall of 0.66. The *young-adult class* had similar challenges, with an f1-score of 0.37, reflecting low recall and difficulty distinguishing from other classes. For the larger classes such as *fantasy, paranormal*, and *history, historical fiction, biography*, the model's performance was moderate with f1-scores of 0.46 and 0.40 respectively. These results suggest challenges in accurately classifying high-support classes, possibly due to their ambiguous nature. Meanwhile, the *children* and *mystery, thriller, crime* classes each showed more balance but moderate performance with f1-scores of 0.54 and 0.47 respectively. Overall, these results suggest that while the model performs well on certain well-defined and larger categories, smaller and more ambiguous classes require further optimization.

In addition to the individual metrics, the classification report also provides overall metrics: accuracy, macro average, and weighted average. The accuracy measures the proportion of correctly predicted instances overall.

$$\text{Accuracy} = \frac{\text{Number of correct predications}}{\text{Total number of predictions}}$$

The overall accuracy was 0.51, meaning the model correctly classified 51% of the reviews. The macro average calculates the mean of the metric across all averages, treating all classes equally. It can be calculated as:

$$\text{Macro Avg} = \frac{1}{n} \sum_{i=1}^{n} \text{Metric}_i$$

where $n$ is the total numbers of genres in the dataset, and *Metric* represents the value of the specified metric, either precision, reccall, or f1-score. The macro precision is 0.47, indicating that across all genres, 47% of the model's positive prediction were correct. The macro recall is 0.54, indicating that across all genres, the model correctly identified 54% of actual positives. The macro f1-score is 0.48. Smaller classes, like *poetry* influence the macro metrics equally as larger ones like *romance* do. It highlights how the model performs with bias towards the dominant ones. The weighted average adjusts the mean by the support, the total number of instances, for each genre.

$$\text{Macro Avg (Metric)} = \frac{\sum_{i=1}^{n} \text{Support}_i \times \text{Metric}_i}{\text{Total Support}}$$

The weighted precision is 0.52, indicating 52% of the model's positive predictions were correct. The weighted recall is 0.51, indicating that the model correctly identified 51% of the actual positive instances. The weighted f1-score is 0.50. Since the weighted metrics give more weight to the dominant classes, it makes sense that the metrics are more closely aligned with the performance of larger classes. Classes with low support have minimal impact on these metrics. Overall, these metrics suggest the model performs better on larger classes, but struggles with smaller classes.

## 5.3 Plots

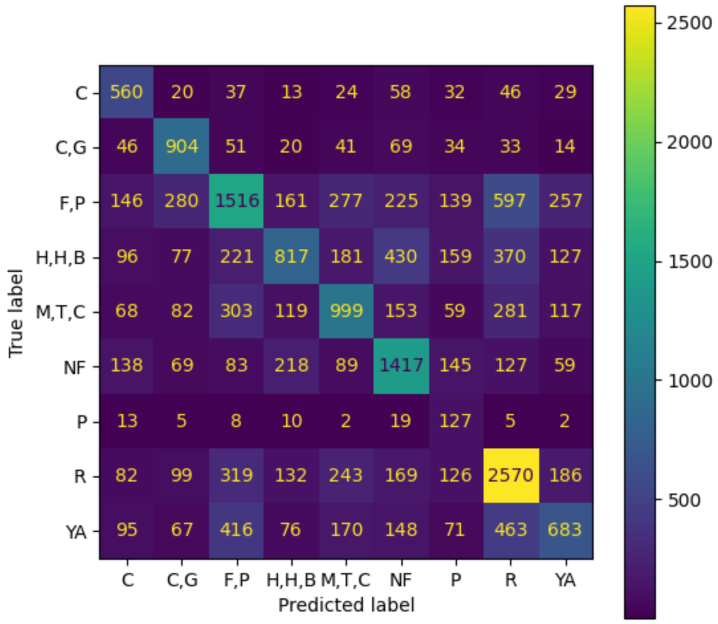



Figure 2: The confusion matrix shows how well the model performed for each genre. Diagonal values represent correct predictions, and off-diagonal values represent misclassifications. Looking at each row reveals what genre the misclassifications fall under.

The above confusion matrix shows highest values along the diagonal, which is generally the desired result. However, there are still a significant number of high values in the off-diagonals, representing many misclassifications. The confusion matrix reveals that the model does a decent job of classifying the genres, but struggles where there may be genre overlap.

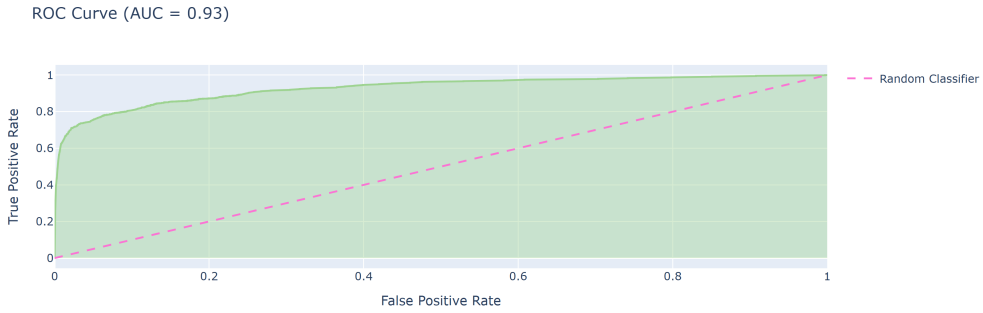



Figure 3: This plot shows the model's ROC curve in green, and the performance of a random classifier with the pink, dashed line.

A Receiver Operator Characteristic (ROC) is a tool for evaluating the performance of classification tasks, and is particularly useful for distinguishing between classes. It plots the true positive rate (TPR) against the false positive rate (FPR). The area under the curve (AUC) value of 0.93 indicates the model has a strong ability to distinguish between the positive and negative classes. An AUC of 1.0 represents perfect classification, while of score of 0.5 (shown with the pink, dashed line) represents complete randomness in classification. With an AUC so close to 1.0, the model clearly performs significantly better than random guessing. The curve's initial steepness and position above the random classifier indicates that it maintains

a low FPR and high TPR. The model has a high sensitivity to true positives and is relatively accurate at rejecting false positives.

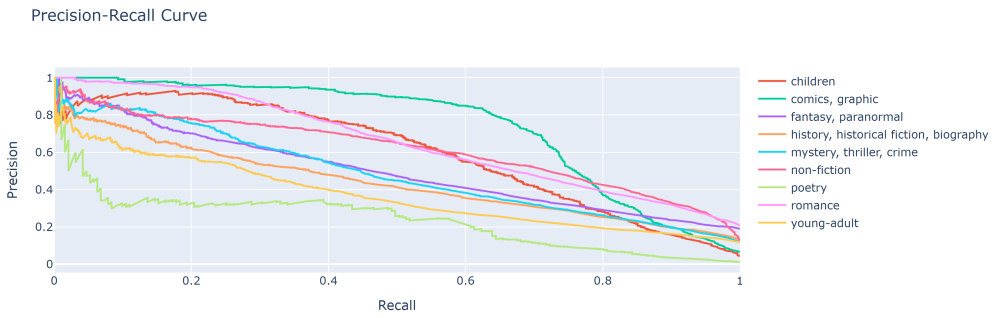



Figure 4: This is a plot of precision against recall for each genre. The genres are represented by different colors shown in the top right.

The precision-recall curve plots precision against recall, and is especially useful when dealing with imbalanced datasets. Most classes exhibit a gradual decline, which is expected for classifiers as they begin to favor precision over recall. This suggests the model maintains a relatively consistent balance between precision and recall. The *comics, graph* line shows a wide peak which indicates the model performs well for this genre, as it achieves both high precision and high recall until the inevitable drop. Additionally, *romance*, *non-fiction*, and *children* all display a similar but less prominent pattern. On the other hand, the *poetry* curve displays a significant immediate drop. This suggests the model is initially overconfident and classifies a large number of instances as positive, leading to a big dip in precision as many of those instances are false positives.

# 6 Conclusions/Future Work

The classification model shows strong performance in genres like *comics, graphic* and *romance*, with high precision, recall, and f1-scores. However, it struggles with smaller or ambiguous genres such as *poetry* and *young-adult*. The overall accuracy of 0.51 indicates moderate performance, with room for improvement in distinguishing overlapping genres. The ROC curve with AUC of 0.93 highlights the model's ability to differentiate between classes, while the precision-recall curve shows a consistent balance between precision and recall. The confusion matrix further emphasizes challenges in classifying overlapping genres.

Initially, a Convolutional Neural Network (CNN) was used, but the Random Forest Classifier outperformed it. CNNs are better for image-based tasks and require substantial feature engineering for text classification. The Random Forest model, suited for handling complex feature relationships, was a better fit for this task.

Despite being a better fit, there is still room for improvement. Methods like SMOTEENN were attempted to address class imbalance, but computational constraints prevented completion. Parameter optimization was also limited by time and resources. Exploring BERT (Bidirectional Encoder Representations from Transformers) could enhance performance by capturing deeper semantic relationships, despite its computational expense.

Another approach is multi-label classification. The current model targets a single primary genre, which causes issues when books belong to multiple genres. Misclassifications occur when secondary or tertiary genres are predicted, affecting performance. The confusion matrix highlights this problem. Combining and redefining genre categories could address overlaps, especially for books spanning multiple genres, like a fantasy novel with a romantic subplot. By implementing multilabel classification, the model could better capture these complex, real-world scenarios where books are often categorized into more than one genre. This approach would allow the model to more accurately represent books with multiple thematic elements, ultimately improving its performance in classifying a broader range of books.

## References


[1] Wan, Mengting. "Goodreads Dataset." https://mengtingwan.github.io/data/goodreads.html, Accessed 2024.

[2] Dpm-a. "ML-NLP-Goodreads-Book-Classification." GitHub. https://github.com/Dpm-a/ML-NLP-Goodreads-Book-Classification/, Accessed 2024.

[3] Suresh, P. and Manoharan, R. "A Survey on Book Genre Classification System using Machine Learning." *Philippine Statistics Association*, 2024. https://www.philstat.org/index.php/MSEA/article/view/1597, Accessed 2024.

[4] West, Ryan W. "Project Gutenberg Book Genre Classifier." GitHub. https://github.com/ryanwwest/book-genre-classifier, Accessed 2024.

[5] Devlin, J., Chang, M.W., Lee, K., and Toutanova, K. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *arXiv preprint arXiv:1706.03762*, 2019. https://arxiv.org/abs/1706.03762.

[6] McKinney, Wes. "Data Structures for Statistical Computing in Python." *Proceedings of the 9th Python in Science Conference*, 2010. URL: `https://pandas.pydata.org/pandas-docs/stable/getting_started/citation.html`

[7] Bird, Steven, Edward Loper, and Evan Klein. "Natural Language Toolkit." 2001. URL: https://www.nltk.org/

[8] Pedregosa, Fabian, et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, 2011, 12:2825–2830. URL: https://scikit-learn.org/stable/citations.html

[9] Chawla, Nitesh V., et al. "SMOTE: Synthetic Minority Over-sampling Technique." Journal of Artificial Intelligence Research, 2002, 16:321–357. URL: https://www.jair.org/index.php/jair/article/view/10302

[10] Plotly Technologies Inc. "Plotly: Collaborative Data Science." 2015. URL: https://plotly.com

[11] Hunter, John D. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering*, 2007, 9(3):90–95. URL: `https://matplotlib.org/stable/users/citing.html`