Predictive Analysis Based on Goodreads Dataset

Abstract

This paper conducted Rating and Interaction Prediction and Spoiler Detection analyses using the Goodreads Book Graph Datasets. Text reviews were converted into word vectors as features, with Random Forest, LSTM [1], and BERT [2] models. Through the utilization of diverse classifiers and recommender systems, this study conducted a comparative analysis of predictive outcomes among various methods and established three effective models for the predictive tasks.

1 Introduction

As user-generated content and multi-dimensional information proliferate across various platforms, research on effective content detection and recommendation has expanded its scope to explore diverse approaches.

Our work concentrates on the following three tasks. First, we explore the methods to predict user-item interactions by comparing various methods and integrating the predictions into a unified response. The second task centers around predicting user ratings based on both user and book characteristics, as well as text content. The last research module involves the detection of spoilers in textual content by leveraging the Neural Network Model and Natural Language Processing techniques.

In summary, this research endeavors to contribute novel insights and methodologies to the fields of recommender system and spoiler detection, ultimately improving user experience across diverse online platforms.

2 Dataset

We performed data analysis using the Goodreads Book Graph Dataset that includes details about books, authors, reviews and book genres. After mapping data between the subsets and filtering out the missing values, it encompasses a total of 1,029,258 interaction data with 20 dimensions between 2007 and 2017.

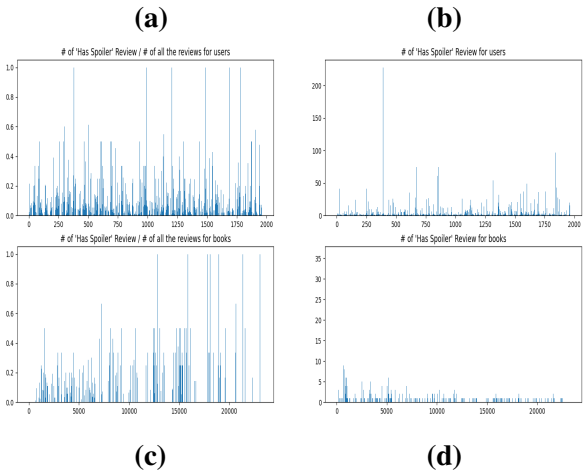

Figure 1: Spoiler Text Distribution Analysis(**a, c**): Proportions of Spoiler Reviews Per User/Item, (**b, d**): Number of Spoiler Reviews Per User/Item.

From Figure 1, we found that spoiler behavior is highly user-specific or item-specific; that is, some users or items have reviews that consistently contain spoilers.

Additionally, we observed no significant differences in the distribution of the text lengths or ratings between spoiler and non-spoiler reviews as in Figure 2. Therefore, we only used the proportion of spoiler reviews by users and items as two additional features.

In Figure 3.a, the popularity of books exhibits a bimodal distribution, as a small proportion of books can be really popular among users, which is essential for recommending.

Despite the relatively low amount of young-adult and fantasy books compared to fiction in Figure 3.b, their readers are more than other genres as in Figure 5.b, and the proportion of the reviews for fiction is decreasing as in Figure 4. This inspired us that certain book genres have more loyal fans, therefore is an important feature.

As in Figure 6, the number of votes and comments can be extremely high, indicating that some

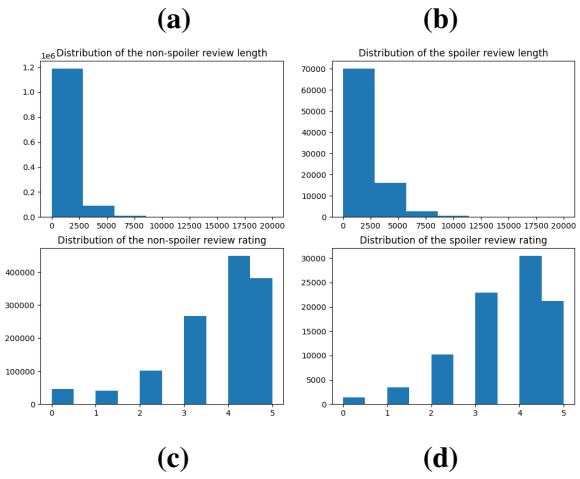

Figure 2: Spoiler Text Distribution Analysis(**a, b**): Distribution of Non-Spoiler/Spoiler Review Length, (**c, d**): Distribution of Non-Spoiler/Spoiler Review Rating,

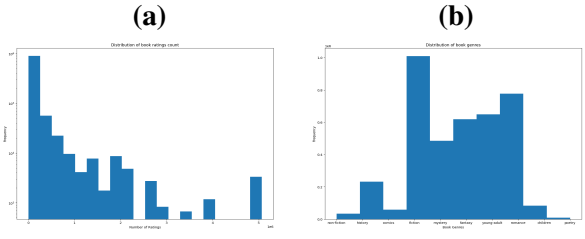

Figure 3: (**a**): Distribution of Book Ratings Count, (**b**): Distribution of Book Genres.

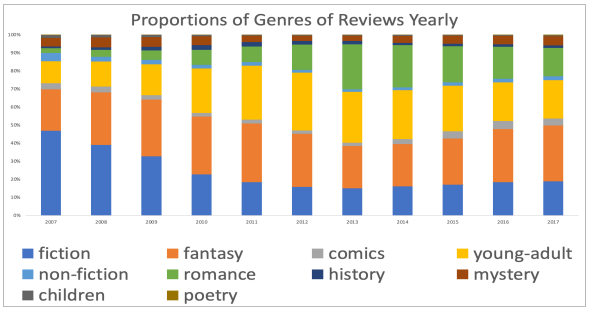

Figure 4: The Proportion of the Genres of the Reviews Yearly

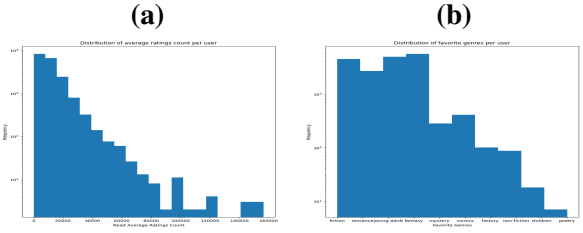

Figure 5: (**a**): Distribution of Average Ratings Count of the Books Read Per User, (**b**): Distribution of Favorite Genres of the Books Read Per User.

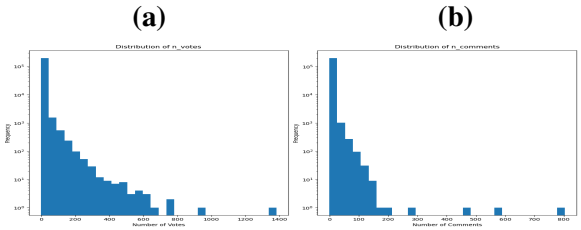

Figure 6: (**a**): Distribution of Votes for Reviews, (**b**): Distribution of Comments for Reviews.

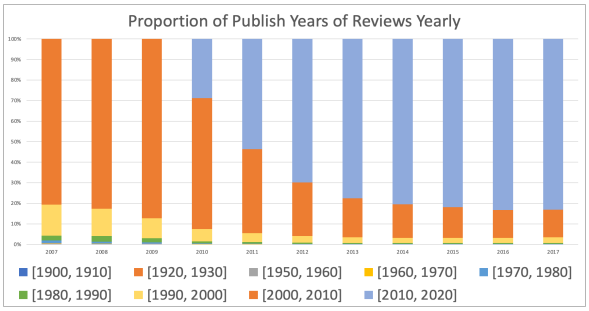

Figure 7: The Proportion of the Publish Years of the Reviews Yearly

key opinion leader comments receiving more attention can have a significant impact on ratings. Therefore they should be added to the features.

In Figure 7, we can tell there are always some classical books can receive consistent proportion of reviews. So it's important considering these nostalgia fans for certain eras.

3 Predictive Tasks, Features and Models

3.1 Predictive Tasks

We work on three predictive tasks on the dataset, including predicting if a reader would read a book, predicting book ratings, and detecting the spoilers in review texts. And we use Accuracy, Precision, Recall Rate, F1 Score and AUC score to evaluate the outcome of the interaction prediction and spoiler detection, and Mean Squared Error for rating prediction.

For comparison, We choose Bayesian Personalized Ranking as the baseline for interaction prediction, SVR as the baseline for ratings prediction, and BERT as the baseline for spoiler detection.

And for assessment of the validity of the three models, we also employ them on several similar datasets and find the similar conclusion.

3.2 Features and Models for Interaction Prediction

For the task of predicting if a reader would read a book, our work compare the result of Bayesian Personalized Ranking (The Baseline model) with Logistic Regression, Linear SVM (with Stochastic Gradient Descent optimization), XGBClassifier, LightFM, and Multi-Layer Perceptron to identify the most effective method. These models have always been proved to be effective for such prediction as our task.

We also explore further application of model integration by incorporating the models' result to optimize the predicting outcome.

3.2.1 Singular Model Comparison

We separately select the features for users and books as shown in Table 1 and Table 2 and construct the feature by concatenating.

Table 1: Features for Book

Category	Variables
Popularity	The Number of Ratings
Genre	The Most Labeled Genre
Belonged Era	Published Year
Acclaimed	Average Rating

Table 2: Features for User

Category	Variables
Favorite Genre	Most Genre that Read
Read Rating	Average Rating of Books Read
Read Popularity	Average Ratings Count of Books Read

For the book genres, we select the genre labeled most in the Platform as the book genre and used one-hot encoding to transform them into binary vector. For user features, we extract their past behaviors and calculate the features for each user.

For the LightFM model as Table 3, we add user and item features to take the explicit factors into consideration. The model transforms these explicit and latent factors through gradient descent, enabling the model to better capture the relationship between users and items.

We also use the rating to add weight to the interaction instead of labeling 1 and -1. The interaction matrix is as Eq. 1 where rows stand for users and columns stand for books.

Table 3: LightFM Model Structure

Category	Variables
Colabrative Filtering	Interaction Matrix
Interaction Weights	Book Rating
Content Based	User & Item Features

$$(u,i) \in U \times I = \begin{bmatrix} r_{1,1} & 0 & \cdots & r_{1,i} \\ 0 & r_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & r_{u,2} & \cdots & r_{u,i} \end{bmatrix} \quad (1)$$

Therefore, we can get the embedding vectors of the user and item features e^U, e^I which is the sum of their series of features, and the corresponding bias vectors b^U, b^I derived from features. The final prediction is the dot product of two embedding vectors, plus their biases, as Eq. 2.

$$\hat{r}_{ui} = f(e^U \cdot e^I + b^U + b^I) \quad (2)$$

With the logistic loss function, LightFM gives the final score of interaction tendency by solving the maximum likelihood function as in Eq. 3 [3]. The S^+ is the set of positive interactions (with ratings), and S^- is the set of negative interactions.

$$L = \prod_{(u,i)\in S^+} \hat{r}_{ui} \times \prod_{(u,i)\in S^-} (1 - \hat{r}_{ui}) \quad (3)$$

For the multi-layer perceptron, we add one input layer, two hidden layers and one output layer for prediction, and train it for four epochs to avoid the overfitting.

3.2.2 Ensemble Learning Model

As the result showing that each model gets its strengths and weaknesses, it gets us thinking if we can combine the strengths. With the same features, we let each model generate their predictions, and then try to incorporate them for final response.

There are many ways to infuse the predictions, and we testify the voting and stacking methods. Voting means that the final prediction follows the majority choices of the models. But a significant flaw is that there are always more models performing poor in certain cases.

Meanwhile, Stacking allows us to assign weights on models based on our needs. In our experiments, we train the Random Forest Classifier and use the

AdaBoostClassifier to form a strong classifier, in order to dig out the best weights in various cases. We also employ this model on other Dataset for validity.

3.3 Features and Models for Rating Prediction

For the task of predicting book ratings, we first preprocess the text with TF-IDF vectorization, which highlights content-specific terms by combining term frequencies and inverse document frequencies. The cosine similarity between document vectors is then computed to quantify the text similarity. In addition, user and item deviations are calculated through iterative optimization to adjust for deviations from the global average rating. Finally, features such as comment lengths, number of votes, and number of comments are normalized to ensure that all features have a uniform scale, which is crucial to effectively improve model performance.

With the features, regression models including Random Forest, Gradient Boosting, Support Vector Machine, and Linear Regression are trained and compared to predict user ratings.

3.3.1 Feature Extractions

First of all, we convert the texts into vectors with TF-IDF Vectorization as follows.

$$\text{TF}(t, d) = \frac{\text{Number of times } t \text{ appears in } d}{\text{Total number of terms in } d} \tag{4}$$

$$\text{IDF}(t, d) = \log\left(\frac{\text{Total number of } d}{\text{Number of } d \text{ with } t}\right) \tag{5}$$

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) * \text{IDF}(t, d) \tag{6}$$

Next, we capture textual similarities between different comments. We measure the cosine similarity between the TF-IDF vectors.

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{||A|| \times ||B||} \tag{7}$$

The predicted rating $\hat{r}_{ui}$ comprises the global average rating μ and the user/item biases,

$$\hat{r}_{ui} = \mu + b_u + b_i \tag{8}$$

and we calculate the bias terms by iterative optimization:

$$b_u \leftarrow b_u + \eta \cdot (e_{ui} - \lambda \cdot b_u) \tag{9}$$

$$b_i \leftarrow b_i + \eta \cdot (e_{ui} - \lambda \cdot b_i) \tag{10}$$

where e_{ui} is the error between actual and predicted ratings, η is the learning rate, and λ is the regularization term.

In addition to textual features and user- and item-specific biases, multi-dimensional features including comment lengths, number of votes and number of comments were incorporated to add in additional information.

3.3.2 Models Selection

We employ and compare the performance of Random Forest, Gradient Boosted Regression, Support Vector Regression (SVR), and Linear Regression.

Random Forest performs well dealing with high-dimensional spaces by constructing a large number of decision trees and outputting the average result.

Gradient boosting model can optimize bias and variance by building one tree at a time, with each new tree helping to correct errors made by previous trees. However, overfitting may occur if not carefully tuned.

SVR is also effective in high-dimensional feature spaces, especially after TF-IDF vectorization of textual data, which typically produces sparse feature vectors. However, its computational requirements can be a major drawback.

Linear regression has good interpretability. However, it has limitations in capturing nonlinear relationships.

3.4 Features and Models for Spoiler Detection

For spoiler detection in review texts, we initially employ Long-Short Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT) to process the review texts, with BERT as the baseline. Then to improve the detection performance, we integrate the features of users and items with the text vectors to take more information into model.

3.4.1 LSTM

LSTM is a recurrent-neural-network-based (RNN) model. It mitigates the issues of gradient explosion and vanish during the training process of RNN, thus enhancing the model's performance

on longer sequences. The model employs several gated units to control forgetting, information input, and output:

$$z = tanh(W_{\hat{C}}[h_{t-1}, x_t] + b_{\hat{C}}) \quad (11)$$
$$z^i = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (12)$$
$$z^f = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (13)$$
$$z^o = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (14)$$
$$C_t = z^f \odot C_{t-1} + z^i \odot z \quad (15)$$
$$h_t = z^o \odot tanh(C_t) \quad (16)$$

In the above equations, t is a timestep in a sequence. x_t is the input from the outsize. h_t and C_t are state information transferred between different timesteps, and W are the parameters.

In Equations (1)-(4), the model maps the input and previous hidden states into a latent space. The terms z^i, z^f, and z^o denote the gate units that regulate the input, forgetting, and output. Equation (5) illustrates the process of selective forgetting and selective remembering. Subsequently, Equation (6) computes the present hidden state by reweighting and scaling C_t.

The bidirectional LSTM (bi-LSTM) comprises two distinct LSTMs that process the input sequences in both forward and reverse order. While these two LSTMs share the same word embeddings, all other parameters are independently learned. The output of the bi-LSTM is constructed by concatenating the features extracted from both LSTMs. This architecture enables the bi-LSTM to integrate contextual information from both the past and future, enhancing its performance on tasks where future information is needed.

3.4.2 BERT

BERT is a transformers-based model pretrained on unlabeled text by joint conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of natural language processing (NLP) tasks, such as question answering and sentiment analysis.

BERT's novelty lies in its utilization of the Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. Unlike directional models, which read the text input sequentially, BERT reads the entirety of the text at once, thereby allowing the model to learn the context of a word based on all of its surroundings. The model's structure is inherently bidirectional, making it uniquely suited to understand the full context of a sentences.

4 Performance Evaluation

4.1 Interaction Prediction

4.1.1 Singular Model Performance

After analyzing the performances of the six models presented in Table 4, we observe that the SGDClassifier excels in precision but performs bad in overall accuracy. the LightFM model stands out for its exceptional accuracy and AUC score, but has a relatively low recall. MLP performs especially well in recall and F1 score, which indicates its great performance in handling positive cases. Because of Bayesian Personlized Ranking model's lack of scalability as it would only take implicit feedbacks into consideration, it performs poor in our task.

Table 4: Comparison of would play prediction results using Bayesian Personalized Ranking, Logistic Regression, Linear SVM Classifier (with Stochastic Gradient Descent optimization), XGB Classifier, LightFM and Multi-Layer Perceptron. The entries in bold highlight the best comparison results.

	BPR	LR	SVM	XGB	LFM	MLP
Accuracy	0.606	0.677	0.671	0.668	**0.692**	0.678
Precision	0.689	0.657	**0.758**	0.728	0.725	0.653
Recall	0.387	0.743	0.502	0.536	0.614	**0.759**
F1 Score	0.496	0.697	0.604	0.618	0.665	**0.702**
AUC	0.607	0.677	0.671	0.668	**0.690**	0.677

In summary, due to the highly heterogeneous users and books, their overall performance are not as well as expected even though they may perform really well in some aspects.

4.1.2 Ensembling Model Performance

To ensemble all the strengths of the models, we came up with an idea about incorporating the predictions of each model. And as discussed in Section 3, we test the Voting and Stacking methods as following.

As shown in Table 5, we find that the Stacking Model does learn the strengths from individual models and outperforms them significantly, with a better balance in handling both positive and negative samples.

Meanwhile, the voting model doesn't performance well in any aspect, as it is affected by the

Table 5: Ensembling Model. The entries in bold highlight the best comparison results.

	Voting	Stacking
Accuracy	0.674	**0.746**
Precision	0.672	**0.754**
Recall	0.682	**0.732**
F1 Score	0.677	**0.743**
AUC	0.675	**0.747**

fact that there is always less models performing well in certain cases.

Therefore, by combining the predicting result of each model and exploring the best weights, the stacking model can well predict if an user would read certain books based on a series of features of users and books.

4.2 Rating Prediction

The performances of each model are shown in Table 6. The random Forest Model achieves a good balance between bias and variance with an MSE of 0.8821, outperforming the Linear Regression. The Gradient Boosting Model performs similarly but with a slightly higher MSE. SVR is theoretically well suited for this task and so was used as a baseline to evaluate the individual models, but its poor performance in practice may be due to the limited computational resources available, which affects the optimization process.

Table 6: Comparison of MSE results using different models. The entries in bold highlight the best (lowest) MSE results.

Model	MSE
Random Forest	**0.8821**
Linear Regression	0.8843
Gradient Boosting	0.9412
SVR	1.3876

In summary, our comparative analysis highlights the efficacy of ensemble methods in dealing with complex datasets, where the Random Forest regressor stands out as a powerful model that is capable of dealing with feature-rich and complex datasets while remaining generalizable across different data distributions.

4.3 Spoiler Detection

In the experiment we use more than 200k data and divide it into training, testing, and validation in 8:1:1 ratio. Following one of the two models mentioned before, we use a Fully Connected Layer to classify the text embeddings and yield the final results, and the performance of the models are evaluated using metrics including precision, recall, accuracy, and F1 score. Notably, the parameter counts are significantly different between LSTM and BERT. This necessitates the adoption of different training methodologies. We train an LSTM model from scratch and fine-tune a pre-trained BERT model using Adam optimizer. During training, we encounter the overfitting issue and we solve it by applying larger regularization term, increasing the dropout rate, and early stopping method. The comparative results of both approaches are presented in Table 7.

Table 7: Comparison of spoiler detection results using the LSTM and BERT. The entries in bold highlight the best comparison results.

	LSTM	BERT
Accuracy	0.756	**0.790**
Precision	0.757	**0.790**
Recall	0.756	**0.800**
F1 Score	0.757	**0.790**

As mentioned in Section 2, we manually add two additional user and item features. They are concatenated with the BERT vectors to form the final feature vector, which are then processed through the classification head to yield the final results. We fine-tune this model for three epochs, and the performance is presented in Table 8.

Table 8: Performance of the spoiler detection model after concatenating user- and item-specific features.

	BERT
Accuracy	0.816
Precision	0.815
Recall	0.821
F1 Score	0.817

As shown in Table 8, including specific information about users and items results in an improvement in the model's classification metrics. This confirms that spoiler behavior is indeed related to

the characteristics of the users as well as the features of the books.

5 Literature Review

5.1 Alternative Datasets

Besides the series of Goodreads datasets sourced from Wan et al. [4] [5], we also use several alternative datasets for validation.

For interaction prediction and rating prediction, we use Douban book and movie data, sourced from the largest review website in China[6]. The dataset contains 383,033 unique users, 89,908 unique books and 34,893 unique movies, as well as 13,506,215 book reviews and 1,278,401 movie reviews.

For spoiler detection, we find the IMDB spoiler dataset [7] and the Large-Scale Spoiler Detection Dataset (LCS) [8], derived from IMDb reviews. The IMDB spoiler dataset comprises 570,000 reviews, of which 150,000 contain spoilers, and The LCS dataset spans approximately 1.9 million reviews with approximately 460,000 of them being spoiler-containing comments.

5.2 Would Play Prediction

Traditional techniques use collaborative filtering to make predictions, which can be further enhanced by incorporating additional implicit factors, like considering shared latent factors between users and items [9]. The state-of-the-art performance remains the same, except incorporating more dimensions of information [10]. But it is difficult to address diverse needs of its heterogeneous users, which is exactly what we conclude from the comparison of the models [11].

Hybrid methods overcome this problem by aptly combining recommendations obtained by different methods [12]. Many research have shown its outstanding performance in various field. Therefore, we work on a method by stacking the predictions of several models into a final response, which improves the overall performance, similar to conclusions in [13][14].

5.3 Rating Prediction

In exploring sentiment analysis in user reviews, a study by Batista et al. [15] delve into how text analytics can be integrated into predictive models to improve the accuracy of rating prediction. In addition, Zhang et al. [16] further advanced the field by using neural networks to capture features of user-generated content, and techniques such as NLP for sentiment analysis of textual comments have been widely adopted to improve prediction accuracy. Rendle et al. [17] uses Bayesian analysis for personalized recommendations and their findings highlight the importance of personalization bias in recommender systems.

Inspired by these ideas, we combine the TF-IDF vectors and user- and item-specific biases, and then use machine learning models for rating prediction.

5.4 Spoiler Detection

In the original research where Goodreads dataset sourced from, Wan et al. also explores spoiler detection with text vectors and specific biases that is similar to our work [5].

And at present, the state-of-the-art (SOTA) performance in this domain is achieved by the Multi-View Spoiler Detection (MVSD) framework introduced in [8]. This paper initially extracted a knowledge base (UKM) from the LCS database, utilizing pre-trained RoBERTa for text feature extraction from reviews, script descriptions, casts and user biographies.

The authors constructs three subgraphs including movie-review, user-review, and external knowledge, encoding these separately using graph neural networks (GNNs) [18] and employing attention mechanisms to fuse features. After multiple layers of MVSD, the comment node representations are passed through an MLP to obtain the final results. This approach not only leverages textual information but also considers the relationships among users, movies, casts, and reviews.

In comparison to various baseline models, including language models (BERT, RoBERTa, BART, and deBERTa), GNNs (GCN, R-GCN, Simple-HGN), and spoiler detection models (DNSD, SpoilerNet), this method outperforms others and exhibits a 2-3% improvement in F1 score on both the IMDB spoiler dataset and LCS.

The article emphasizes the importance of considering both textual information and the inherent characteristics of users and movies in spoiler detection, aligning with the underlying principles of our approach. An external knowledge base is constructed to provide additional context, and GNNs are employed to encode different graph relationships. In contrast, our method relies on manual feature engineering for extracting user and item characteristics, potentially resulting in lower infor-

mation density and complexity compared to the approach presented in the article.

6 Conclusion

6.1 Would Play Prediction Conclusion

During the experiments, we find that the rating, genre and publish year features work especially well as expected, while the author-specific features and the book format work poorly. It shows that people tend to continue reading the books of their favorite genres and eras.

According to our experiments, it's important to apply the right method when dealing the heterogeneous cases. Therefore, we build a Stacking Model by incorporating the predictions of the various models to generate final predictions. The performance of such combination significantly outperforms all the other models, as it can capture the heterogeneity of the multi-dimensional data and reassign the weights among the models.

Additionally, as the parameters showing that the hinge loss function works best, there are noticeable linear characteristics in the dataset.

6.2 Rating Prediction

In the study of predicting book ratings, various models and features are evaluated to understand their impact on prediction accuracy. Random Forest is the most effective, with lower MSE values compared to SVM and Linear Regression. This superiority can be attributed to their robustness in dealing with the high-dimensional and complex nature of our dataset, especially the TF-IDF vectors.

The key to our model is the combination of TF-IDF vector of review texts using cosine similarity, which effectively captures the nuances of user opinions. In addition, incorporating user and item biases into the model allows for more personalized predictions that reflect users' individual preferences and book-specific attributes. In summary, our study highlights the importance of thoughtful feature selection and the advantages of ensemble approaches in predictive modeling, especially for complex datasets such as book reviews. Further feature acquisition from review texts might be a direction to further enhance the model.

6.3 Spoiler Detection

In our work, we have developed a spoiler review detection framework based on a text classification task. We employ pre-trained language models to process text and combine their outputs with features extracted through feature engineering as the basis for classification. On one hand, manual feature engineering provides the model with greater interpretability for decision-making. However, on the other hand, the number of features we utilize is limited, making it challenging to manually extract complex relationships, and feature engineering methods may require adjustments according to the dataset.

Hence, we can draw inspiration from other studies by constructing subgraphs and leveraging graph neural networks to process them. This approach can potentially address the limitations of manual feature engineering, allowing the model to capture more intricate relationships within the data.

References

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. In Toine Bogers and Marijn Koolen, editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, volume 1448 of *CEUR Workshop Proceedings*, pages 14–21. CEUR-WS.org, 2015.

[4] Mengting Wan and Julian McAuley. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 86–94, New York, NY, USA, 2018. Association for Computing Machinery.

[5] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. Fine-grained spoiler detection from large-scale review corpora. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2605–2610, Florence, Italy, July 2019. Association for Computational Linguistics.

[6] Bin Cui. DOUBAN BOOK RATING DATA, 2015.

[7] Rishabh Misra. Imdb spoiler dataset, 2022.

[8] Heng Wang, Wenqian Zhang, Yuyang Bai, Zhaoxuan Tan, Shangbin Feng, Qinghua Zheng, and Minnan Luo. Detecting spoilers in movie reviews with external movie knowledge and user networks, 2023.

[9] Mark Claypool, Anuja Gokhale, Tim Miranda, Paul Murnikov, Dmitry Netes, and Matthew Sartin. Combing content-based and collaborative filters in an online newspaper. In *Proc. of Workshop on Recommender Systems-Implementation and Evaluation*, 1999.

[10] Kapil Saini and Ajmer Singh. Original research article comparative analysis of collaborative filtering recommender system algorithms for e-commerce. *Journal of Autonomous Intelligence*, 7(2), 2024.

[11] Murat Göksedef and Şule Gündüz-Öğüdücü. Combination of web page recommender systems. *Expert Systems with Applications*, 37(4):2911–2922, 2010.

[12] Robin Burke. *Hybrid Web Recommender Systems*, pages 377–408. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[13] Michael Jahrer, Andreas Töscher, and Robert Legenstein. Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, page 693–702, New York, NY, USA, 2010. Association for Computing Machinery.

[14] Pigi Kouki, James Schaffer, Jay Pujara, John O'Donovan, and Lise Getoor. Personalized explanations for hybrid recommender systems. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 379–390, New York, NY, USA, 2019. Association for Computing Machinery.

[15] Huoston Rodrigues Batista, José Carmino Gomes Junior, Marcelo Drudi Miranda, Andréa Martiniano, Renato José Sassi, and Marcos Antonio Gaspar. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2:1–135, 2008.

[16] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis : A survey, 2018.

[17] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback, 2012.

[18] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications, 2021.