COMP20008, Elements of Data Processing, Semester 1, 2024
Assignment 2 - Report

Group Name: W14G2
Workshop: Wednesday 14:30
Tutor: Behzad Moradi
Group Members:
- Kerui Huang – 1463475 - keruih@student.unimelb.edu.au
- Peter Lu – 1462600 - houlinl@student.unimelb.edu.au
- Dylan Tran – 1462053 - dytran@student.unimelb.edu.au

Executive summary
This paper seeks to document the development of a book recommendation system that is able to predict a book's exact rating (or rating category) using data derived from the reader and the book. This rating prediction can be then used to provide insight into which books will likely be more popular in the future. The main objective of this report was also to outline which of the attributes from the reader and the book lead to the most accurate predictions. In the pursuit of this, many data issues hindered the overall exploration of the question, therefore a large component of the report is focused on the preprocessing aspect of the project. Through testing, we were able to fit two supervised learning models (KNN and DT) to predict book ratings, which both excel with binned rating predictions, but struggle with exact ratings. We determined that the root cause can be due to an imbalanced training dataset, and we've noted this for future improvement.

Introduction
Utilising a collection of datasets from an online bookstore containing information about book ratings and users, this paper intends to explore how different attributes from both books and users may influence ratings, to ultimately create a recommender system that can predict a book's rating. With this prediction, the managers of the bookstore are able to stock suitable books to boost future sales.

It is important to note that within the attributes of the provided csv files, the data collated was littered with irregularities, invalid entries, and even missing entries altogether. Because of this, our preprocessing stage needed to be well thought out and thorough in order to ensure accurate and high-quality data analysis. To begin our processing, the paper first extracted data sources from the following provided csv files:

Users	**Books**	**Ratings**
Within the users table, there contained data concerned with where the user is from (country, state, and city), the age of the user, and also a user ID.	The books table contains the unique book identifier ISBN (International Standard Book Number), book title, author, year of publication, and the publisher.	Within the ratings table, there contained information about the involved book (ISBN), User (UserID), and a respective rating out of 10.

Methodology
Preprocessing
Upon our initial assessment of the data quality we found that many entries were missing completely at random (MCAR), as well as discrepancies in its representation itself; with a large proportion of entries containing punctuation marks, extra whitespace, and inappropriate alpha-numeric values. If left untouched, this data would be near impossible to analyse, or at the very least lead to extremely inaccurate and poor data analysis.

In order to properly analyse and conceptualise the provided data, it was imperative to achieve standardised entries across all relevant dataframes. To do this, our group employed various different data preprocessing techniques to achieve increased data quality and consistency.
Key attention was directed towards preparing attributes pertaining to the users and books, where we employed various data preprocessing techniques including data imputation, scaling ranges, as well as grouping. All the steps involved in our data preparation for each attribute are documented below.

User Locations (City, State, Country)

As a brief summary, the data for each of these attributes is firstly cleaned, then standardised based on an external location database through fuzzy matching. The resulting location data is then encoded to integer values for imputation. After an analysis of Mutual Information between the data points to evaluate the viability of imputing using the respective data points, KNN imputation is lastly used to fill in missing or incorrect data. The final output from this block includes the fully processed and imputed location data.

The first step we took towards achieving consistent data was to regulate syntaxial inconsistencies within the provided location names, making use of regex. This included removing leading and trailing whitespaces, putting all entries in uppercase, removing all punctuation marks and all numeric values.

Following this, we then needed to account for diacritic marks within the text, which may cause a country recorded as “Curaçao” to be labelled differently to “Curacao” even though they are both referring to the same country. To remedy this, we normalised the text entries to the NFKD standard via the “unicodedata.normalize” function, hence standardising diacritic marks and enabling our program to recognise countries with/without diacritic marks as the same.

The same data-cleaning then is applied onto our externally sourced location database [LINK], which provides us with the basis to utilise fuzzy matching. After the provided location names and the external sources are cleaned to the same standard, we then utilised a string matching algorithm known as Fuzzy Matching (FM) on our data. In using Fuzzy matching, the algorithm will attempt to match the provided location names to similar existing locations via Levenshtein edit distance. The Levenshtein distance is a unit of measurement that measures how far two strings are from being identical to each other; with a higher number expressing a greater disparity between two strings. If a location name is unable to match with the documented existing locations, the name will be treated as an error and will be converted to np.nan for future imputation.

To apply this to our program, we experimented with different similarity thresholds to find which one suited our requirements best. We found that too low of a threshold meant that the algorithm would match countries that should not be matched together, such as “america” being matched with “algeria” when the threshold is set at 0.3. However, if the threshold is set too high, such as 0.9, the program would be too harsh in selecting matches, failing to recognise simple cases such as “Bahamas” and “Thebahamas” as the same, negating the purpose of Fuzzy matching. Therefore our group agreed to set the threshold at 0.6, which we found brought the greatest success in string matching. Following this, we extended this experimentation out towards both cities and states, and found that the same threshold of 0.6 was also a viable figure for the project requirements.

Applying this method of Fuzzy Matching paid dividends to our data analysis as the alternative was going through the list of unmatched strings and individually changing names such as “Newyork” to “New York”. This would have been an extremely tedious process which would have diverted lots of time and resources away from the actual data analysis if we had used this method.

After applying Fuzzy matching to country, state, and city, we grouped all entries that failed to find a match against the external standard names together and set their values to Nan. By this standard, a total of 700 nan values were generated in countries, 2253 in cities and 2502 in the states column.

Next, in order to use the aforementioned process of imputation via KNN, the location names are required to be encoded (since KNN requires a numerical format to compute distances between data points). This will be done by the encode function defined in the "Encoding location strings with number" section of the code.

Finally, we imputed all the Nan values against the remaining attributes using KNN with k=3 as we believed such a k hyperparameter value to be the best compromise between accuracy and efficiency. This meant that if an entry had a missing value under country, the algorithm would assign a predicted country based on the state and/or city. For example, if an entry had "Melbourne" under city, "Victoria" under state, but a Nan value for country, the algorithm would consider the three nearest data points (neighbours) by Levenshtein edit distance and then assign the corresponding country.

Our final imputed columns are recorded and put back into the users dataframe for future use. As a test, we calculated the normalised mutual information score of the state and city data before imputation, and compared it against the score after imputation. After imputation, the score increased from approx. 0.86 to 0.91, which can be considered as a significant correlation. We also computed the Kendall tau correlation coefficient to confirm a non linear correlation, as shown in figure 1.

	country	state	city
country	1.000000	-0.085477	0.009568
state	-0.085477	1.000000	-0.026300
city	0.009568	-0.026300	1.000000

Figure 1. Kendall tau correlation coefficient

Book Title, Author, Publisher

Moving on, for the preprocessing of the books' author name, book name, and publisher, we decided to take a much simpler approach than we did for the location of the users. The main reason behind this was due to the fact that it was extremely difficult to find a csv file that contained every book/author/publisher name that we could match our entries to. Hence, we utilise a more general data cleaning method that we were already familiar with, simple RegEx filtering. Through this process, we aimed to achieve uniform data across these attributes by removing all numeric values, whitespaces, punctuation, as well as setting everything to uppercase. This helped us group cases where perhaps an author's name was entered as "JK. Rowling" vs "JK Rowling" or even "JK ROWLING" vs "jk rowling".

Book Year Of Publication (YOP)

Via research [reference x], we applied a valid range on the publication years, where the lower year is 1455 (earliest recorded year of book publication) and the upper is the current year 2024. The values that fell outside of this range were then set to Nan for imputation

In order to impute by KNN, we experimented with two different variables to predict missing book years. Initially, we were content with using the book-publisher to assign a value to the year of publication. However, upon calculating the normalised mutual information score between YOP and other logically related columns, namely book publisher and book author. We found that the book author column resulted in a higher score of approx. 0.53 than the book publisher with a score of 0.23. Acknowledging that the book author's substantially higher normalised mutual information score, we refocused the imputation to use the book author to infer the year of publication. Intuitively, this is a more logical approach, as a book author is more directly related to the book than the publisher company.

Finally we implemented KNN imputation on the book YOP along with the book authors, documenting the imputed column for future use.

User Age

Not only did the user age column include the most number of empty Nan values, it was also riddled with inconsistent formatting and values outside a logical range.

Due to this blaring issue, we first sought out to improve the data representation of User-Age through basic regular expression operations (or RegEx). Through this, we were able to remove leading and trailing whitespaces along with any non-digit characters from the entries. More importantly, we flagged all of the remaining non-numeric/Nan/blank entries as an error represented by a value -1. After changing the value of all defective ages to -1, we applied a range filter of [0,116] (oldest living person is 116) against all ages for scaling and removing outliers. Entries that fell out of this range were also changed to -1. We then grouped all -1 values together and then changed them back to Nan values once again. This complex process of locating all Nan values was to help the following imputation system run as smoothly and efficiently as possible.

To select the best predictor to use for user age, we again employed the normalised mutual information score to determine which column is potentially most correlated with user age, for future KNN imputation. By logic, we tried the columns for book rating, ISBN and Books author. The ISBN had the highest score of around 0.59 in correlation with the user age, while book author had a score of 0.32 and book ratings had a score of around 0.01. By this metric, we choose ISBN as the most relevant predictor for user age imputation. To use KNN imputation, we needed to first encode ISBN values. This was achieved using the same encoding function as previously stated.

Finally, mirroring the imputation methods for book publication year, KNN imputation is implemented using ISBN. The imputed user age column is documented and stored in the original column for future feature selection.

Feature Selection

Feature selection is a process concerned with selecting the most uniform, least redundant, and relevant features to use in a model construction. This section of the project is unparalleled in importance, as it ensures our research pertinent in measuring the correct attributes.

In order to find the most suitable features to apply in our recommendation system concerning the book ratings, we utilised a normalised mutual information score ranking method, which allows us to rank the attributes based on how relevant they are individually to the book ratings. Among these attributes are the previously processed information about the users' age, location (including country, state and city), ID, as well as information about the books' title, author, publication year, publisher and ISBN. The resulting scores are split into two categories: User-based features and Item-based features, as shown below.

User-based Features:

User-ID	0.418
User-City	0.113
User-State	0.015
User-Country	0.007
User-Age	0.005

Item-based Features:

ISBN	0.206
Book-Title	0.185
Book-Author	0.090
Book-Publisher	0.026
Year-Of-Publication	0.004

Figure 2. Split ranking of mutual information scores

For our recommendation system, we decided collectively to combine both user and item based features to predict the user ratings. A combined ranking of attributes that individually have the most significant correlation with the book ratings is shown in a descending order.

Although User-ID stands at the top of the list, due to the arbitrary nature of its connection with user rating, we decided to remove it from consideration.

Since the optimal number of features to select for rating predictions can vary by different circumstances, we tested using the top 3 ranked features, then increased the number of features to include the next highest ranked feature, until all features are used in our system. This ensures that the attributes chosen can operate at the optimal accuracy.

```
User-ID              0.418
ISBN                 0.206
Book-Title           0.185
User-City            0.113
Book-Author          0.090
Book-Publisher       0.026
User-State           0.015
User-Country         0.007
User-Age             0.005
Year-Of-Publication  0.004
```

Figure 3. Overall ranking of NMI to rating

Results and Modelling

Model Justification

Upon initial inspection of the Pearson correlation between data columns, it was immediately apparent that no features possess any significant linear correlation to the class label (ratings), as seen in Figure 4. This observation suggested that the use of classification algorithms would be more suitable for modelling the data. We decided upon two supervised machine learning classifications algorithms: K Nearest Neighbours and Decision Trees.

Figure 4. Pearson correlation

	City	Author	ISBN	Title	Publisher	Age	Ratings
City	1.000000	-0.001908	0.001005	-0.002609	0.007628	0.002093	0.011552
Author	-0.001908	1.000000	0.013733	-0.009224	0.059569	-0.015715	-0.012048
ISBN	0.001005	0.013733	1.000000	0.002909	-0.005317	0.004842	-0.006982
Title	-0.002609	-0.009224	0.002909	1.000000	-0.017696	-0.000350	-0.016328
Publisher	0.007628	0.059569	-0.005317	-0.017696	1.000000	0.013759	-0.025908
Age	0.002093	-0.015715	0.004842	-0.000350	0.013759	1.000000	-0.014507
Ratings	0.011552	-0.012048	-0.006982	-0.016328	-0.025908	-0.014507	1.000000

Analysis on feature selections

In our feature selection, it was mentioned that the number of features used for modelling can have an impact on the accuracy of the predictions. This is tested here. A DT and KNN model will be utilised to calculate the accuracy scores. The number of features used will be plotted against exact-rating accuracy scores obtained using the top number of features, until all features are used. The result is shown below:

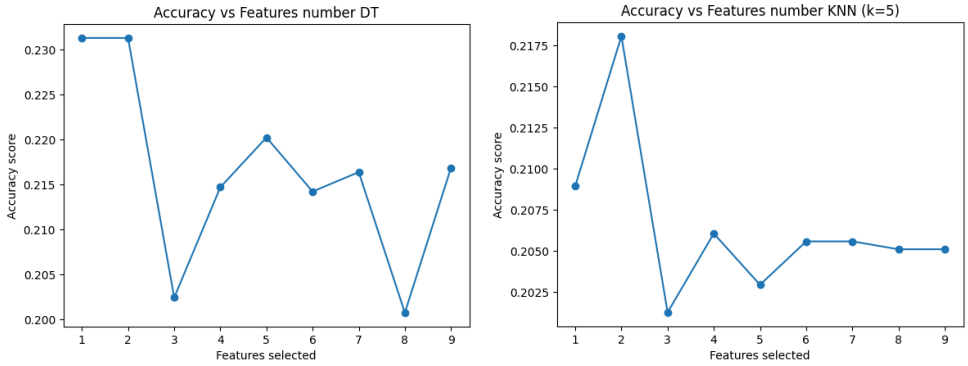


Figure 5. Accuracy score vs Number of features used (Left: DT, Right: KNN)

As seen in figure 5, a significant drop in accuracy occurred in both DT and KNN models after the third feature "User-City" was added as one of the prediction features. Otherwise, no clear pattern except for a general descending trend can be observed with accuracy scores as more features are selected.

With further testing, it was found that selecting features highly correlated to the class label (via Mutual Information) did not necessarily improve the predictive accuracy of the model. We determined that "User-City" may have been negatively affecting model accuracy and decided to add the feature last. After this modification it was found that User-City did in fact reduce accuracy as seen in Figure 6 and 7, and it was removed from the prediction features. We hypothesise that the high correlation between User-City and the class label could be attributed to imbalances in data and pure coincidence. This hypothesis was made on two observations: the fact that User-State and User-Country both had negligible correlation with User-Rating, suggesting User location likely had little correlation with ratings in general; and the fact that a value count of User-City shows high skewness toward certain cities.

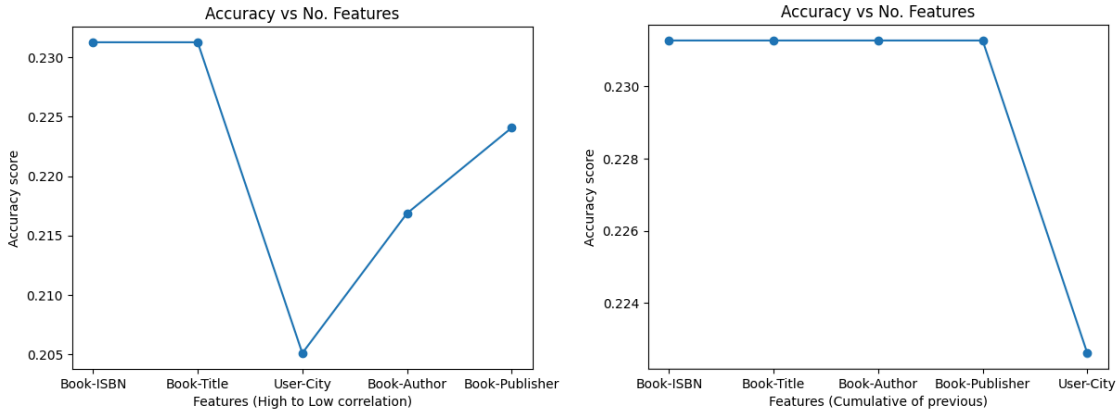


Figure 6. Effect of User-City on accuracy using DT
(Left: Added in order of correlation, Right: City is moved to last)

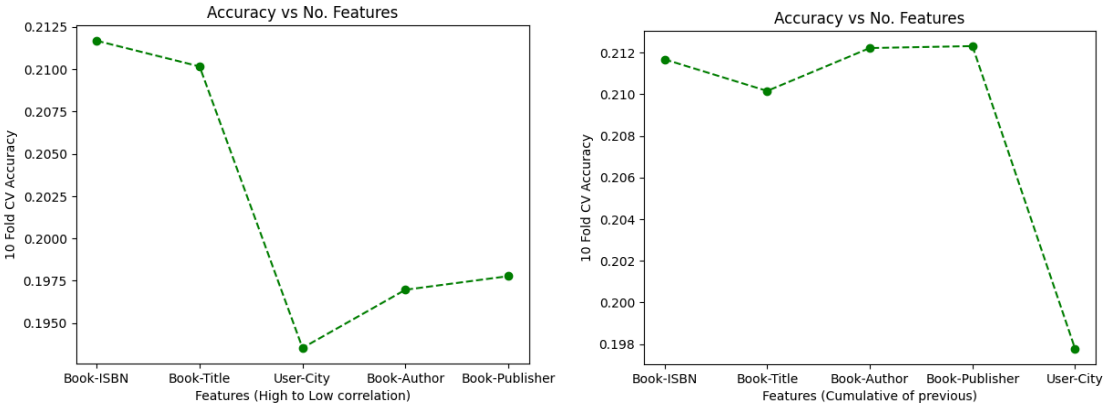


Figure 7. Effect of User-City on accuracy using KNN
(Left: Added in order of correlation, Right: City is moved to last)

Modelling via KNN

Our goal for using the KNN algorithm was to produce a prediction model capable of predicting Book-Rating given some features. Our model was initially trained on predicting exact ratings Users might give to Books, however we found that accuracy was low. We implemented a second model, with class labels categorised into groups of Low, Medium and High, corresponding to ratings of [0,3], [4,6] and [7,10], respectively. From here on out, we will refer to the model outputting exact ratings as the exact KNN model and the model with categorised outputs as the grouped KNN model.

Analysis of K hyperparameter selection for KNN

The selection of a K hyperparameter for KNN is a balancing act between maximising model accuracy and minimising model training time. By creating plots between the K-value and 10-Fold Cross Validation Accuracy, we can employ the elbow method to find the optimal point at which the model has both acceptable performance and training speed. As seen in Figures 8 and 9, we found the optimal hyperparameter for the exact KNN model to be k=25, and for the grouped KNN model to be k=20.

Figure 8. K selection for Exact KNN model

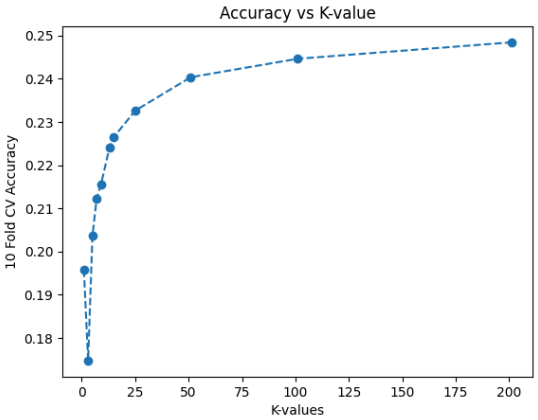
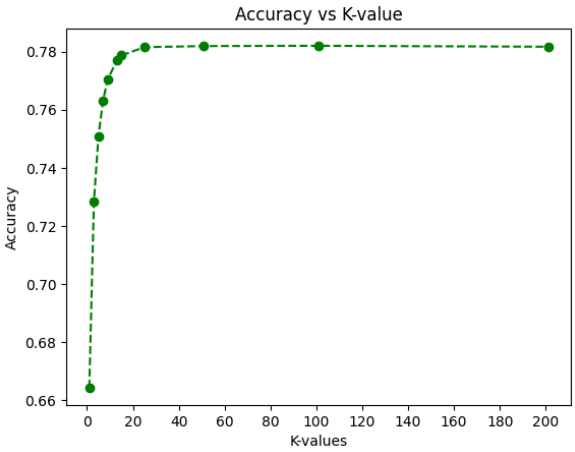


Figure 9. K selection for Grouped KNN model



Model evaluation and analysis

Using the k values of 25 and 20 as discussed before, we were able to obtain a 10-Fold Cross Validation mean accuracy of 30.11% and 78.25% for Exact and Grouped KNN models, respectively. To evaluate the Recall, Precision and F1 score of the models, we separated the original data into a training and validation set, with 180,000 samples in the training set and 24,164 samples in the validation set, to avoid data leakage.

Figure 10. Results of Exact KNN model

```
Recall       0.231791
Precision    0.199878
F1           0.203829
Accuracy     0.231791
dtype: float64
8     0.453816
7     0.190697
10    0.179565
9     0.112440
5     0.042791
6     0.017050
1     0.003352
4     0.000207
2     0.000083
dtype: float64
```

Figure 11. Results of Grouped KNN model

```
10 Fold CV: 0.780279449103358
Recall       0.781452
Precision    0.692587
F1           0.692529
Accuracy     0.781452
dtype: float64
High      0.991102
Medium    0.005628
Low       0.003269
dtype: float64
```

Although the Accuracy of the Grouped KNN model is drastically higher than that of the Exact model, both models perform poorly in comparison with their baseline. As seen in Figure 8, the baseline prediction accuracy of the Exact KNN model is 44.8%, compared to a model's predictive accuracy of 30.11%. This baseline considers the accuracy of a model if it were just to always predict the most common output: a rating of 8 (refer to figure 10), thus implying that if we had just predicted a rating of 8, we would have obtained a higher accuracy than our model, which is quite poor. Similarly the baseline for the Grouped KNN model is even higher, at 99.0% (refer to Figure 9), compared to the predicted 78.25%. This extremely high accuracy for the grouped baseline suggests that our categorisation of the class labels were inadequate and resulted in almost all of the output being captured by a single group: High (refer to Figure 11). However, this skewness in output is also caused by a skewness of the distribution of ratings in the input data. In future attempts, this factor should be fully taken into consideration. Overall, the ability for the KNN model to be used in a recommender system is heavily undermined by its low score compared to its baseline. Further investigation is required to ascertain the applicability of using such a model for recommendation.

Figure 12. Confusion matrix of Exact KNN model

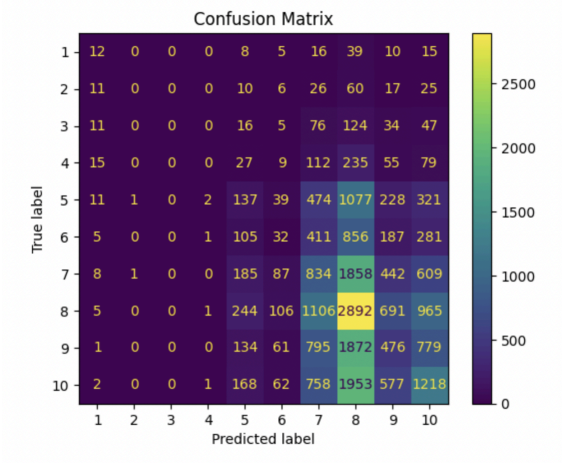
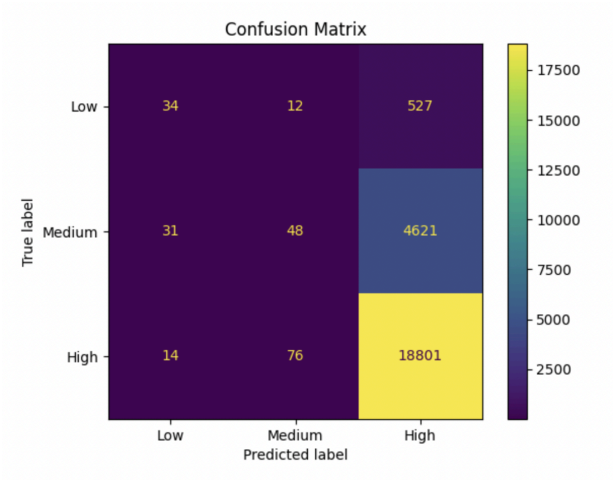


Figure 13. Confusion matrix of Grouped KNN model



Modelling via Decision Tree (DT)

In addition to KNN modelling, we also applied the DT model in order to predict user ratings. Similar to KNN, DT is a supervised learning method which utilises classification and regression to predict target variables. The model achieves this by seeking to reduce entropy (uncertainty in a dataset) at every branching node. The resulting model allows us to pass values in and predict the target values based on the classification of the input.

Parallel to the previous process, we will create two DT models, using both grouped ratings (low, medium, high) and exact ratings (1 to 10). To assess the effectiveness of each model, we will compute the metrics including the accuracy score, recall score, precision score as well as the f1 score. To ensure consistency, the top 4 features listed in feature selection, excluding user city (for reasons mentioned in feature analysis) will be used, this includes the encoded “Book-ISBN”, “Book-Title” and “Book-Author” and “Book-Publisher” The results are as follows:

Grouped Ratings

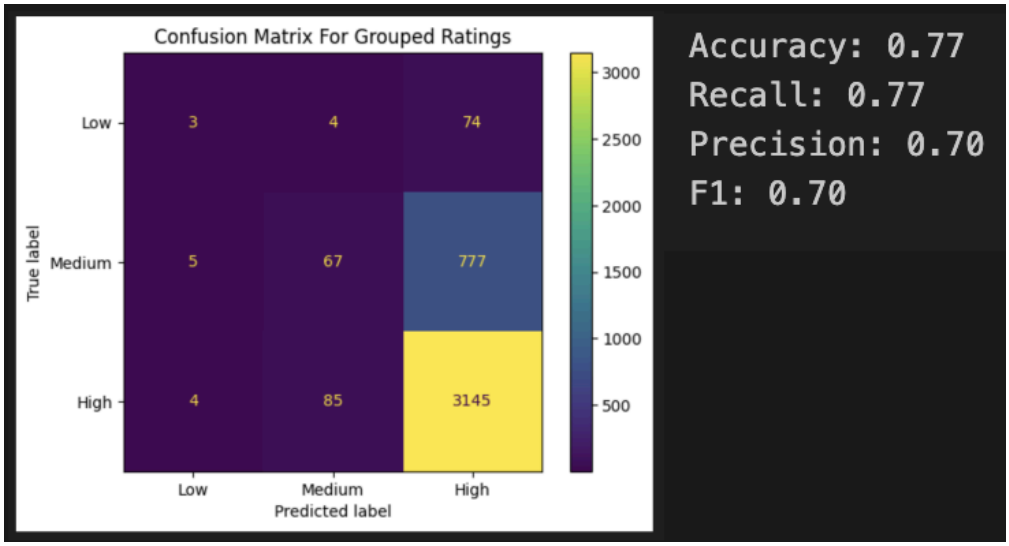


Figure 14. Confusion matrix of Grouped DT model

By a standard interpretation of the accuracy scores, scores above 0.7 can be generally considered "good accuracy". As shown above, the DT model is able to produce moderately correct predictions of rating labels, with relatively "good" accuracy, recall, precision and F1 scores. Although the testing data itself tends to skew towards the medium to high categories, there seems to be a general uncertainty in our predictions of these two labels, with over 800 false ratings being mixed up between them. The majority (97%) of high ratings were predicted correctly, while only 8% of medium values were accurate. Alarmingly, 92% of medium values were falsely predicted to be high. This highlights a significant bias towards high ratings in our model.

Exact Ratings

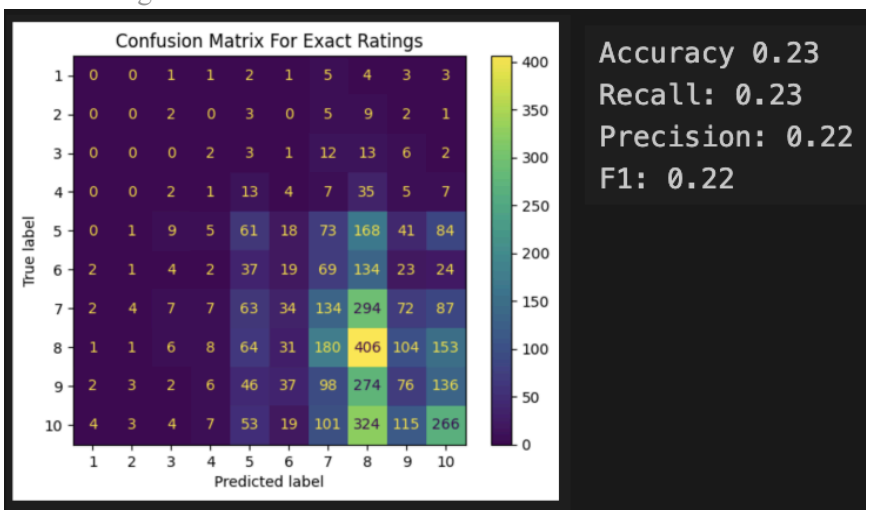


Figure 15. Confusion matrix of Exact DT model

By the same standard, the accuracy of the DT using exact ratings is considerably lower than the grouped ratings. Again, the general uncertainty in the medium to high ratings is clear. Specifically, there appears to be a bias towards 8 within predicted ratings. For true labels 10 to 2 (excluding 8), the majority of the ratings were unanimously falsely predicted to be 8. At true labels 10, 9, 7, 5, a multimodal distribution of predicted labels occurred, with the other peaks occurring at 5 and 10. This phenomenon could correspond directly to the frequency of ratings in the training data, where 8 is the most commonly selected rating among all ratings (with over 50,000 "8" ratings, taking up approx. 25% of all ratings), which incites the

model to favour choosing 8. Possibly due to this, 8 is shown to be the most falsely predicted label of all labels. In addition, the peaks occurring in 5 and 10 ratings also correspond to the peaks in 5 and 10 predictions, which further consolidates this theory.

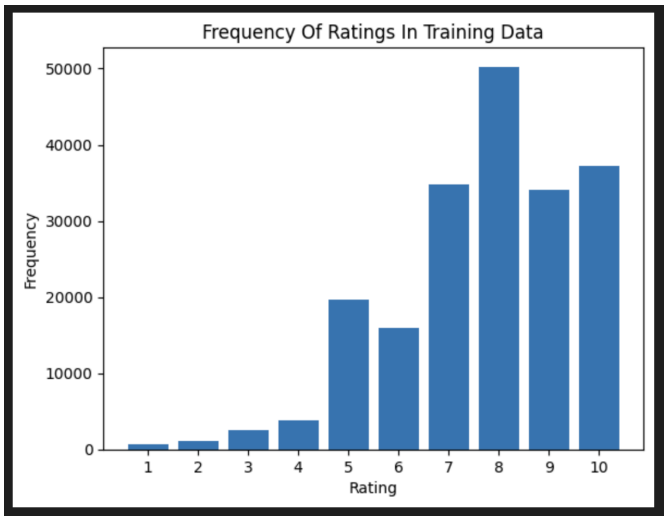


Figure 16. Frequency of Ratings vs Ratings

Evaluation

From comparison, it is clear to say that due to the DT model being unable to predict accurate exact results, a fuzzy prediction using grouped ratings is optimal for increasing the effectiveness of the model. While the data exhibits a training bias, the accuracy using binned target values is noteworthy.

Limitations and improvement opportunities

The available information on the books and users were somewhat constrained and shallow, leading to limitations in predicting book ratings against factors such as user location or book age. This is because it is logically arbitrary to determine a book's rating using these determinants as they are seemingly disconnected and do not influence one another. For example a user being from Melbourne does not necessarily mean that they will rate a book similarly to another user in Melbourne. This is because the connection between a user and book ratings is very obscured. For a more meaningful and accurate recommender system, it is suggested that the researchers seek to collate more data against the user and the book, such as book category or past user purchases. Indeed, such data would lead to substantially more accurate and deeper data analysis due to their greater relevance in regards to book enjoyment.

Another limitation of this paper was the method in which the predictor column was chosen for imputation. Indeed, when choosing which column to use for predicting missing values, we decided that comparing the potential columns' normalised mutual information scores would lead to the best results. What we failed to consider however is that we were calculating the scores in the presence of invalid ages. If you recall during the data cleaning process for missing ages, all invalid ages were computed to -1, hence, a large proportion of ages would be -1, which is illogical. This would have led to an inaccurate mutual information score, hence hindering the achievement of meaningful data analysis. To improve this, we could have done a quick filtering process of removing all -1 age values to avoid a bias.

A big assumption that this paper made was that User-ID's, ISBN's, and Book-Ratings were presumed to be standardised, requiring no preprocessing or data cleanup. In reality however, it should be expected that data anomalies such as book ratings falling outside the range may very easily happen, and therefore should be checked.

Another way this paper could be improved is by accounting for potential imbalanced data types. This was made clear to us after the training data for book ratings was positively skewed towards medium to high ratings, with 8 being the most frequent. Ideally, we would want the training data to have a similar amount of data at each rating to avoid problems associated with data imbalances such as biases or accuracy paradoxes.

Conclusion

In this paper, we developed a book recommendation system which predicts book rating using user and book attributes. Preprocessing and data cleaning was imperative during the project, given that the provided data was littered with significant data issues such as missing values and inconsistent formatting. Because of this, techniques such as fuzzy matching and KNN imputation helped standardise the data, ensuring the robust foundations for later analysis.

As the modelling methods used (KNN & DT) were both undermined by a significant data imbalance, the prediction accuracy of the grouped target ratings was significantly higher than exact ratings.

To improve, we strongly recommend identifying and addressing potential data imbalances, as it can seriously impact the accuracy of the findings and results. In conjunction to this, it is also suggested to source a broader range of data pertaining to both the user and the book in order for better data analysis.

References

Book-Crossing: User review ratings. (n.d.). Www.kaggle.com. https://www.kaggle.com/datasets/ruchi798/bookcrossing-dataset

Brownlee, J. (2020, June 23). *kNN Imputation for Missing Values in Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/knn-imputation-for-missing-values-in-machine-learning/

DATAtab Team. (2024). *t-Test, Chi-Square, ANOVA, Regression, Correlation...* Datatab.net. https://datatab.net/tutorial/kendalls-tau

Gada, D. (2024, February 9). *countries-states-cities-database/csv/countries.csv at master · dr5hn/countries-states-cities-database*. GitHub. https://github.com/dr5hn/countries-states-cities-database/blob/master/csv/countries.csv

Gupta, A. (2020, October 10). *Feature Selection Techniques in Machine Learning*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/

Kaushik Roy Chowdhury. (2020, July 13). *KNNImputer | Way To Impute Missing Values*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/07/knnimputer-a-robust-way-to-impute-missing-values-using-scikit-learn/

Mahmoudi, A., & Jemielniak, D. (2024). Proof of biased behavior of Normalized Mutual Information. *Scientific Reports*, *14*(1), 9021. https://doi.org/10.1038/s41598-024-59073-9

nvidia. (n.d.). *What is a Recommendation System?* NVIDIA Data Science Glossary. https://www.nvidia.com/en-us/glossary/recommendation-system/#:~:text=A%20recommendation%20system%20is%20an

Silva, E. (2022, July 15). *What is Fuzzy Matching?* Redis. https://redis.io/blog/what-is-fuzzy-matching/#:~:text=July%2015%2C%202022

Publisher, A. removed at request of original. (2016). 3.2 History of Books. *Open.lib.umn.edu*. https://open.lib.umn.edu/mediaandculture/chapter/3-2-history-of-books/#:~:text=The%20earliest%20example%20of%20a