Predicting the Rating of a Book Based on the User's Age, Region, and Rating of Past Books Using Collaborative Filtering and K Nearest Neighbors

Group W{13}G{9}

Eden Aristo Tingkir
COMP20008 - 1480734
etingkir@student.unimelb.edu.au

Putera Sutrisna
COMP20008 - 1480731
psutrisna@student.unimelb.edu.au

Executive Summary

The goal of this project is to create a book recommender system that will provide bookstore managers with useful information about what kind of books to stock for optimal sales and what sort of books to suggest to customers for further reading. By using a user-to-user collaborative filtering technique, the system generated an estimate of a book's rating of one user based on another similar user's feedback. Our models utilise a machine learning technique called k-nearest neighbours, which computes similarity scores between users based on user reviews, age, and location. This kind of collaborative filtering assists in the prediction of books that will appeal to readers with comparable taste profiles. According to initial testing, the model is able to recognise user patterns and preferences, which results in recommendations for books that are both useful and relevant. The test results, which demonstrate how well the recommendations match user interests and possible purchases, have been satisfactory. This recommender system offers a big improvement in bookstore management technology since it can provide managers with insightful data. For further improvement, we need to concentrate on enhancing the algorithm's precision and incorporating more comprehensive consumer data that will improve our prediction.

Introduction

In this digital age, the volume of books to read have been growing exponentially. With this massive growth of books to read, it becomes increasingly harder for people to get a good recommendation of what books to read. To address this issue, the recommender system has become an essential tool in helping readers to books that match their interest. This report discusses a recommender system developed to predict the rating of a book for a particular user.

The objective of this report is to discuss the development and performance of our recommender system. This recommender system that is developed, uses the user's age, region, and rating of past books to accurately predict the rating that the user would give to a particular book. The system also uses a user-based collaborative filtering method and K nearest neighbours as the main machine learning algorithm. By doing so, the system is expected to give accurate recommendations that are personalised to the user's demography and preference.

The main datasets that we are going to use in this project are originally sourced from a website called *Kaggle*, which contains:

- BX-Books.csv

  These files include detailed information about books, such as ISBN, Book-Author, Year of Publication, and Book-Publisher. There are a total of 18,185 books in the datasets.

- BX-Ratings.csv

  It contains each book's ratings, where each entry includes ISBN, User-ID, and the given rating.

- BX-Users.csv

The datasets comprise user information, including their User-ID, City, State, Country, and Age. There are a total of 48,299 user data in the datasets.

In addition to those data, there are some more data which contains:

- BX-NewBooks.csv

  Provides information on 8,924 new books that is different from the BX-Books, the format is similar to the BX-Books dataset.

- BX-NewBooks-Users.csv

  Contains data on 8,520 existing users, including demographics and their historical rating data. The users in this csv file is not new and has already been included in the BX-Users.csv

- BX-NewBooks-Ratings.csv

  Lists actual ratings for the new books in BX-NewBooks.csv. The users that are included here are listed on BX-NewBooks-Users.csv.

This report will start by discussing the methodology used in the recommender system. We will then continue to explore the data and analyse our findings. We will then detail the result and interpretation of our model and research. Lastly, we will highlight some limitations and improvements that can be done to perfect this research. The ultimate goal is to provide insights that can help make a better recommender system by increasing its performance and guide future research on recommender systems.

Methodology

This section discusses the details of our methods, techniques, and tools used to prepare and analyze the data. In addition, we will point out the details in our machine learning algorithm to understand it better.

Data Preprocessing

We preprocessed each data table we have differently. We decided to merged the each old datasets with the new datasets (i.e. BX-Books.CSV with BX-NewBooks.CSV) except for users, since that BX-NewUsers is just a subset of the BX-Users.

1. Preprocessing Books

   In order to preprocess the two datasets BX-Books and BX-NewBooks simultaneously, we must first concatenate them. We conducted data filtering to eliminate duplicate ISBNs in order to preserve the integrity of our book dataset. This ensures that there is only one individually identifiable ISBN, which improves the accuracy of our collaborative filtering. In addition, we also assume that any books labelled with a zero for "Year-of-Publication" are not valid. We chose to retain the 314 data that had a year of publication of 0, but we infer the value to be NaN rather than 0.Additionally, we considered that a valid year of publishing could not exceed 2024. Lastly, we performed data normalisation on ‘Book-Author’, by making sure that each word is capitalised. For example, if there’s an author named “kevin stuart”, we change the name to be “Kevin Stuart”. For such special cases where the author name has “mc” or “mac” like ” Maggy mccarthur”, the name will be transformed to be “Maggy McArthur”.

   After the pre-processing on BX-Books is done, we are left with 27,101 books.

2. Preprocessing Ratings

   Same procedure as books, we need to concat the old and new ratings datasets.

   We implemented a data validation to ensure that every unique ISBN can only be rated by the same user one time. This ensures that each user-ISBN pair is unique, which is crucial for the accuracy of user based collaborative filtering models. In addition, we also make sure that each

ISBN in ratings datasets, matches ISBN in the book datasets.Furthermore, we also ensure that ratings are on a scale of 1-10. In addition to that, we also created a graph “Unique ISBN vs Ratings per ISBN”, with the purpose of highlighting the relationship between book users. In the graph, we only show the number of ratings between 0-100 since we only wanted to see how the distribution looks. At first, we intended to filter out books that have very few ratings. However, seeing that most of the books that we have had a really small amount of ratings, we decided to not alter the number of books for the test, since it will make our model lose almost 30% of the datasets.

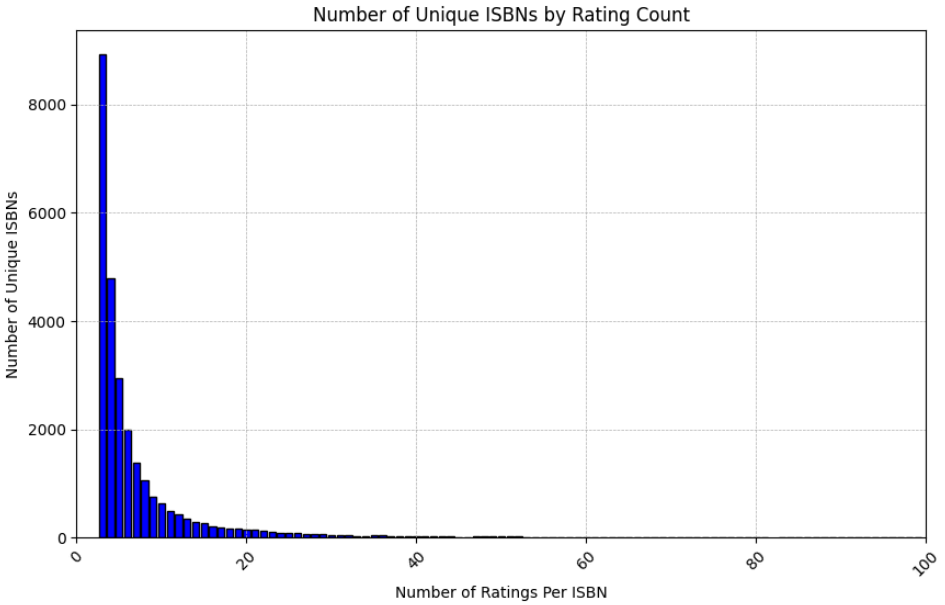

Figure 1: Number of Ratings per ISBN vs Number of Unique ISBN limited to 100 ratings

After the processing, we left with 230,936 ratings.

3. Preprocessing BX-Users.CSV

We removed any entries where User IDs are not integers or duplicated, ensuring each user is uniquely identified. Furthermore, if there’s any blank or entirely whitespace characters in regional data (city, state, country), replace them with NaNs instead of removing those entries. We performed imputation on states, where it's based on other states with the same city, but missing state information. In addition, countries were also imputed based on other entries with the same state but missing country. Post-cleanup, we dropped the city column since we decided to only use state and country. We applied one hot encoding to the state and country columns, to be included in our analytical models. Lastly, we remove non-integer IDs, by first stripping all non numeric characters from the age data. Afterward, we convert ages to integers and we mark them as NaN if they’re not integers. We also replace ages that are outside a realistic range by NaN to avoid skewing, and finally all entries with NaN are dropped from the datasets.

Algorithms and Models

Once the data was preprocessed, we employed specific machine learning algorithms to build the recommender system. The methodology followed for algorithm selection and model training includes:

1. Collaborative Filtering

Collaborative filtering is one of recommendation methods that works by examining prior relationships between users and interdependencies across items (Su & Khoshgoftaar, 2009). Collaborative filtering algorithms make predictions about user preferences. To be more precise, CF operates on the assumption that A is more likely to share B's viewpoint on a given topic than a randomly selected individual if A and B share similar tastes. In this case, we used the user-based collaborative filtering. This means we are using the hypothesis that users that like similar books are similar and have a similar interest and preference in books.

In addition, different to the regular collaborative filtering methods that use only the ratings, we also decided to use the age and the Country and States of the users as extra features to measure similarity between the users. This is intended to be the solution to the cold start problem. Where a recommender system that only depends on the ratings of the user will always be unable to make recommendations for new ratingless users. In our system, the recommendation for new users will be based on their similarity to other users based on their age and region.

The hypothesis is that people of the same age and same region will have approximately similar interest and preference in books. This may happen due to social behaviour and colleagues that often have similarities in taste. In addition, the region also provides similarity based on language that is evident where people will only read books that have the language that they understand.

1.1. Implementation

We implemented our collaborative filtering by first, making a user-feature matrix. This step is essential since we want to classify each user based on their features. The matrix that we made has rows of users and columns of features. The features include: age, encoded state, encoded country and ratings of each book in the database.

Once this matrix has been made, we added a weight for the region and the age of the users. This is done by multiplying the value of age and region by a certain constant. This is done to even out the impact of the extra features so not to overwhelm the rating features which are more important.

After all these preparations, we then start to make the model. We use the *sklearn* library to use the NearestNeighbors() function to make our model. Once the model is made, we can fit our training data into the model.

Our algorithm works by calculating each pairwise distance between all users. This distance is then used to calculate a similarity score between all the users.

2. K Nearest Neighbors

K Nearest neighbours is the machine learning algorithm we used to support the collaborative filtering system that we made. K Nearest neighbours works by finding k users that are most similar to the user for further analysis. By doing this, we are saying that a user is approximately an average of some users that is similar to them.

2.1. Implementation

Once we calculate each pairwise distance between all users, we can calculate a similarity score. The similarity score has the formula: similarity = 1 / (distance + 1). This means that the higher the distance the less the similarity score. For 2 users to be considered similar, their distance has to be very close. In this context, being really close means that they have similar ratings, age, and region in the table.

After calculating each pairwise similarity, the system will then be ready for input. We can input a userID to a book ISBN that we want to predict the rating of. After inputting the desired user and book, the system will then find k nearest neighbours that are closest to the desired user.

In our system we used the number of users in our system to determine the k value used. The formula for k value used is k=int(sqrt(number_of_training_data)). This is a convention and a safe value used to find k.

After finding the similar users, their rating for the specific inputted books will be averaged. The average used in this part of the system is a weighted average on their similarity score to the inputted user. This means that the more similar a user is to the inputted user, the higher their weight, thus creating a higher impact to the predicted score. This weighted average of rating from k similar users will then be outputted as the predicted rating for the inputted user.

Training and Testing Implementation

The objective of this part is to provide a robust way to split the data into train and test data. We will then train the model using the training data and test our model using the test data. This part of the implementation is very important since it will provide us with clear metrics on the performance and accuracy of our model.

1. Data splitting

   We first have to split the data into training and testing. This will be done by splitting the user-feature matrix. The user feature matrix that we have consists of users as rows and features which are encoded state, encoded country, and ratings of each book. The data that we need to split is the rating data since we want to use the test data as actual rating that will be compared to the predicted rating by the training data. There are 2 ways to split that depends on what we are trying to achieve

   1.1. Hot start testing splitting

   In the hot start testing, we want to test the capability of our model to predict rating while knowing the user's other rating. We use the test ratio 0.05 which is chosen due to hardware limitation.

   The splitting algorithm that we use is named user based splitting. Since we want to only test the hot start capability of our model, we only want to test users that already have a certain number of ratings in the system. In addition, we don't want to strip users off their rating since we are strictly not testing for cold starts.

   The number of ratings of each user that are going to be tested have a formula: number of rating tested for user x = int(0.05*(number of books rated by user x)). This means that only users that have rated 20 books or more will be considered to be tested. This also means that every single user will still have rated a book in the training data which shows that we only test for hot start capabilities. This is also done to make the test less random by using the ratio for each user and not for the whole data.

   1.2. Cold start testing splitting

   In this test, we want to test for the cold start capability of our model. Since the model includes features other than the rating itself, it is expected to overcome the cold start problem.

   To test this capability, we need to split the testing data by only testing users that only have rated 1 book. This is done by using the test ratio that we determine to be 0.05. We chose this number due to hardware limitations. After determining the test ratio, we find 5 percent of users that only have rated 1 book and move their rating to the test data. On the train data, the user will have no rating which shows a cold start scenario.

2. Training the data

   We train the data by making a model using collaborative filtering and K nearest neighbours. By this part, the training data will still have all the original users, only some ratings are taken by the testing data. Before training, the train data needs to be imputed. NaN ratings are imputed using the mean rating of that particular user. For the cold start test, the students with NaN mean ratings are given the universal mean of rating to be imputed before the data is trained. The way the data is trained is already explained in the above "Algorithms and Models" section

3. Testing the data

   The testing data includes some ratings that have been stripped from the training data. Before testing the data, we need to process the training data. This is done by simplifying the test data which were made like the user-feature matrix, into a 2d table with columns user, isbn, and ratings. The rating column will be used as a realised value of the predicted value that is going to be determined. This is done to simplify the test and reduce hardware load. This new test data has the same format as the original BX-Ratings.csv table.

After simplifying the data, we can then test the model. The way to test the data is by looping over each user and based on the test data to find a predicted rating from the model. After it is complete, we will have 2 arrays. One array "predicted_rating" consists of all predicted ratings from the model and the other array "observed_rating" consists of realised values from the "ratings" column of the test data.

After the two arrays are ready, we can then check the performance of our data. The "errors" array can be made by subtracting the predicted rating from the observed rating. We decided to check the accuracy by using mean error (ME), mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). On top of that we used spread metrics which is the standard deviation of errors.

In addition to these metrics, we also used some visual representation of our testing which are predicted vs observed value scatter plot, and a Mean Absolute Error plot for different multiplier values and both the cold and hot start testing.

Data Exploration and Analysis

In this section we will discuss the outcome of the tests and evaluate our model accordingly. Our evaluation process is divided into 3 parts:

1st Analysis: Base Evaluation of The Model

First, we are trying to analyse the general accuracy of our model. This is done using the "base" model. We considered the base model to be the model with age multiplier = region multiplier = 0.5 and test ratio as 0.05. For this evaluation we will do it both for the hot start testing and the cold start testing

1. Hot Start

   In the output of our base hot start model we provided some performance metrics and spread metrics. Here is the output of the testing.

```
============================================================
=
||              Base Hot Start Model Evaluation
||
============================================================
=
|| Settings:
||
|| - Cold Start Testing           : Disabled
||
|| - Test Ratio                   : 0.05
||
|| - Region Multiplier            : 0.5
||
|| - Age Multiplier               : 0.5
||
============================================================
=
|| Performance Metrics:
||
|| ------------------------------------------------------
||
|| - Mean Error (ME)              : 0.023509956885575344
||
|| - Mean Absolute Error (MAE)    : 1.1109642876103158
||
|| - Mean Squared Error (MSE)     : 2.104066180368529
```

```
||
|| - Root Mean Squared Error      : 1.4505399616585988
||
============================================================
=
|| Spread Metrics:
||
|| ------------------------------------------------------
||
|| - Standard Deviation of Errors : 1.450349427653818
||
============================================================
=
```

Figure 2: A brief performance summary of the hot start model evaluation

Let's start by examining the performance metrics. Mean Error is used to find whether the model is biassed or not. A good model will have 0 mean error since it means that the middle of the model is right in the expected rating. On our output, it can be seen that the mean error is 0.02, which is very close to 0. This shows that our model is centred.

Next, we have a Mean Absolute Error of 1.11. This is relatively good. This shows that our prediction is expected to miss by 1.11 from the true observed rating.

We also have mean squared error and root mean squared error of 2.1 and 1.45 respectively.

For the spread, we used the standard deviation of errors which comes out to be 1.45. This spread is relatively high which shows that our data is not very accurate.

2. Cold Start

For the cold start base model testing we have the output:

```
============================================================
=
||              Base Cold Start Model Evaluation
||
============================================================
=
|| Settings:
||
|| - Cold Start Testing           : Enabled
||
|| - Test Ratio                   : 0.05
||
|| - Region Multiplier            : 0.5
||
|| - Age Multiplier               : 0.5
||
============================================================
=
|| Performance Metrics:
||
|| ------------------------------------------------------
||
|| - Mean Error (ME)              : -0.14026662135446202
||
|| - Mean Absolute Error (MAE)    : 1.3400568829336441
||
|| - Mean Squared Error (MSE)     : 3.155567412486205
```

```
||
|| - Root Mean Squared Error      : 1.776391683296847
||
============================================================
=
|| Spread Metrics:
||
|| ---------------------------------------------------
||
|| - Standard Deviation of Errors : 1.7708451901337974
||
============================================================
=
```

Figure 3: A brief performance summary of the cold start model evaluation

First, it can be seen that the mean error is 0.14. Compared to the hot start model, this is worse since for cold start, our model is biassed to the lower values and tends to underestimate.

In addition the Mean Absolute Error is also higher at 1.34. We also got a Mean squared Error of 3.16 and Root Mean Squared Error of 1.78.

For the spread, we have a standard deviation of errors 1.77 which is higher than the hot start testing.

Based on these metrics, we can say that our model performs better in the hot start model than the cold start model.

2nd Analysis: Base Predicted vs Observed Value

Rather than just focusing on metrics, we also need to find the distribution of our model and how our model predicts the ratings compared to the real ratings. Here, we are trying to find the pattern of errors in the data. The more the plot resembles the x=y line, the better the model is. In addition, we can see if the errors of the model have a constant variance which shows a good model. By "base" we still mean the model with age_multiplier = region_multiplier = 0.5 and test_ratio = 0.05. Here we do the evaluation twice, one for hot start and one for cold start.

1. Hot Start

   Here we will output the predicted vs observed rating scatter plot:

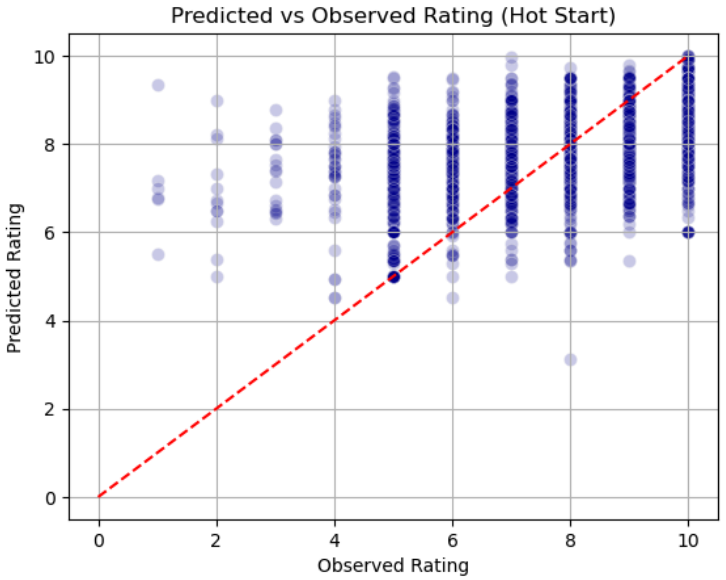

Figure 4: A scatter plot on Predicted vs Observed Rating for the Hot Start test

Here we can see that our model is not very good at predicting lower ratings. Our model is far from the red dotted line which is the expected “perfect” model.

Here we can see 2 patterns that emerged. On the right hand side (from observed rating 10 - 4) we can see that the overall trend of the predicted rating is downwards, this is a good sign, even though it does not accurately depict the red dotted line. On the other hand, the predicted rating from 4 - 0 looks like it is going upwards. This is a problem since that means that the model did not accurately predict the ratings for lower observed ratings.

2. Cold Start

   For the cold start base model testing we have the output:

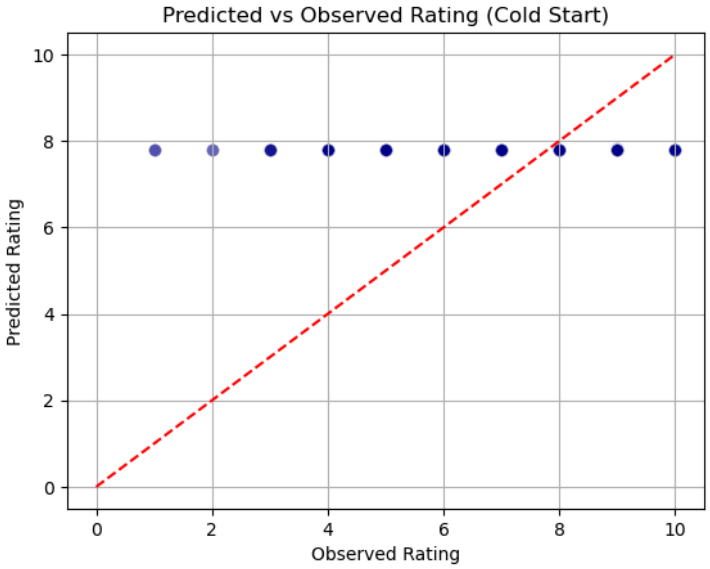

Figure 5: A scatter plot on Predicted vs Observed Rating for the Cold Start test

Here we can see that our model is very bad at predicting the observed rating. As can be seen, the blue dots resemble a line which is the universal mean line. The model is basically using a mean value of all the ratings in the data to guess every single cold start testing. This is a very bad model that doesn’t actually predict the rating.

3rd Analysis: MAE Plot for Different Age and Region Multiplier

Here, we are trying to find the relationship between different (age and region) multipliers and MAE as an accuracy metric of the model. We also provided 2 charts for each multiplier's value to show the relationship of higher multipliers to the cold start model testing as well as the hot start testing. The weights that we are going to test are 0, 0.5, and 1. 0 means that the model only uses the rating data without the age and region data of each user. 0.5 is the "base" model. Lastly, 1 multiplier is used to test the relatively high impact of the age and region multiplier to the data. Here is the output of the 3rd analysis:

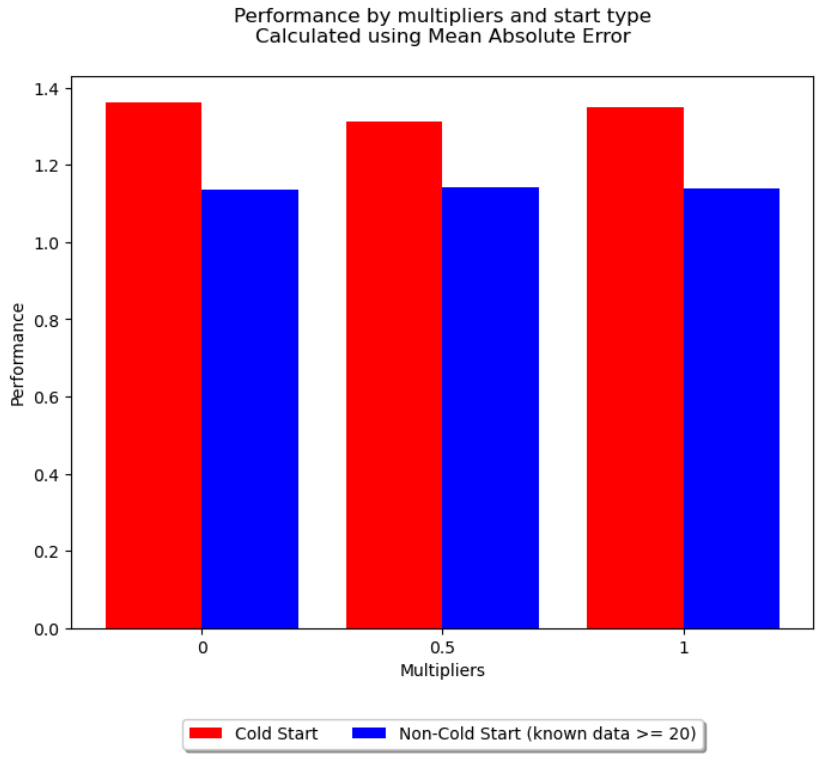

Figure 6: A comparison of different multipliers performance using MAE

As can be seen from the bar plot, there is no apparent correlation of the multipliers to the mean absolute error of each test. This can be seen as there is no increasing or decreasing trend of the MAE over different multipliers.

In addition, it can be seen that the base model is better in cold start performance. It can also be seen that hot start test performance is always better than cold start performance. This shows that our model is much better at doing hot start prediction rather than cold start.

Results

Based on the results above, we can interpret that our model works poorly in predicting observed ratings. We found that our model performs better in hot start than in cold start tests. We also found that, while our model seems to work well in traditional measures, it cannot capture the overall distribution of user ratings accurately. The associated histograms in data analysis give a visual representation of this issue, which compares the actual and expected rating distributions.

Discussion and Interpretation

We can see from the first analysis that our model did fairly well for the hot start test. This shows that our model, given enough data for a particular person, can fairly predict the rating by only knowing the user's age, region, and book ratings.

The biggest problem to our model is that on average, each user rated very few books. This is very bad since our system uses user based k nearest neighbours. This means that we lack information on each user to be able to fairly say that some users are similar. In addition, the use of imputation using the mean of each user made the similarity score irrelevant since people with similar mean scores will be considered similar even though the books they rated are very different.

On the hot start test, the similarity score is helped by the additional features (region and age). This is because it helps users have some sort of similarity other than ratings which are imputed and inaccurate.

On the other hand, the cold start test didn't perform as expected. This is due to the usage of universal mean of ratings to be imputed for the users being tested. Every user tested ended up having 100% similarity between the users since they all have the same ratings across the board. This ended up very bad since the users tested are now clustered and the predicted ratings tend to go towards the mean which is the universal mean. This is why at the end, the predicted vs observed value of the cold start test resembles a straight line, which is suspected to be the universal mean.

The strength of our model is in the hot start testing. The metrics of the hot start testing being better than every metric on the cold start testing shows that our hot start testing is better than just guessing the universal mean for every prediction. In addition, the mean absolute error of 1.1 is fairly good enough for a recommender system.

In addition, we expected the higher the multipliers the better the cold start test and the worse the hot start test is. This is not the case because of the mistakes explained above. In our model, the multipliers did nothing for the performance of each test.

Limitation and Improvement Opportunities

Throughout this project, we've discovered some limitations and improvements opportunities such as:

1. Hardware Limitations

   Our existing hardware has limitations on the volume and speed of data processing. This affects how effectively we can handle massive volumes of data. Due to these constraints, we had to select a reduced test dataset with a ratio of 5%, which might not accurately represent the entire dataset and prevent our model from performing to its fullest. In the future, we ought to think about handling more thorough testing by utilising cloud computing services or more dependable hardware.

2. Sparse Data Per User

   Although there is a lot of data in the databases, there is not as much data for each individual user. This influences our user profile construction accuracy, which in turn impacts our user-based collaborative filtering. To improve the model's capacity to understand specific preferences, more data should be collected from each user.

3. Unexplored Collaborative Filtering Methods

   Our existing method made use of user-based collaborative filtering in this model. Other collaborative filtering techniques, such as item-based collaborative filtering, should be taken into account as it's possible that using an item-based approach will result in a recommendation system that is more effective in this situation.

4. High Proportion of Missing Data

   The databases contain a large number of missing data points in a variety of fields, particularly in the geographic data. We did not try to impute the numerous missing regional data points (cities, states, and nations). This reduces the models' efficiency and may produce inaccurate findings. In the regional section, an advanced imputation method should also be taken into consideration for better suggestions and enhanced demographic profile. In order to reduce the likelihood of missing data, greater caution should be used when retrieving data.

Conclusion

Our model performed adequately well in a hot start test. With a Mean Average Rating of 1.11, the user-based collaborative filtering model is fairly suitable to use for predicting book rating for users given that the users already have rated some books in the system. Even though the distribution of predicted value is not accurate, this issue can be negated by having more rating data per person in the data. In contrast, it performed badly under cold start test, this shows the weakness of user-based collaborative filtering model in predicting a rating for users with low rating count. For future research, it may be important to have more powerful hardwares, more rating data per user, better data retrieval

methods, and better imputation techniques. Overall, the system can be used for a book store where some loyal customers often buy books from the books store. This system can improve and help customers get recommendations on what book to buy, thus increasing sales and customer satisfaction.

References

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, *2009*.