Movie Recommendation System

Yishan Li
a356li@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Abstract

Nowadays, the explosive growth of digital information creates a great challenge for users, hindering timely access to items of interest available online. Recommendation systems help drive user engagement on the online platforms by generating personalized recommendations based on a user's past behaviour. In recent years, the recommendation system has become an effective approach to avoid information overload for users. How to effectively recommend a personalized movie to the target user becomes a significant research problem. Recent research has focused on improving the recommendation system's performance by extending the content-based approach, collaborative filtering approach and hybrid approach. However, in this paper, I focus on developing a data-driven hybrid solution to deliver recommendation tasks. The recommendation system trains three regression models, namely KNN, ridge regression model, and random forest models, on a new dataset produced by leveraging rating averages, content-based approach and collaborative filtering approach. Then, I empirically compared three hybrid regression models' performance with the baseline model, matrix factorization algorithm. The test result shows the remarkable effectiveness of my proposed hybrid solution over the baseline model. The best-performed model, the random forest algorithm, improves 10.8% compared to the baseline model. This study explores that enhancing the models' performance is not restricted to extending the models. However, the feature engineering procedure can also extensively boost the recommendation system accuracy and enable the regression models to deliver better recommendation tasks than a robust algorithm, matrix factorization without using feature engineering.

Introduction

Nowadays, we are living in a data-driven world. As more and more business activity is digitized, new sources of information and continuous generation of data bring us into a new era. Statistics show that Amazon(Dayton 2020) sells more than 12 million products online and on average it has more than 200 million different visitors per month. How to effectively gain insights from the users and product data to provide users a better user experience becomes a vital challenge to those online service providers. With the rapid development of data technology, the recommendation system becomes the effective solution to this problem.

Given that an explosive amount of information is available online, the users have countless choices of movies, music, products, etc. The recommendation engine turns out to be an essential strategy for facilitating a personalized online user experience. To recommend the item in a customized way, the service providers take the initiative to find out users' interest based on their past data(ex item features, user preference, other environmental factors like location, time, etc.) and suggest only the relevant items to users. Apparently, the introduction of the recommendation system has been bringing a great number of business values to the company. Statistics show that 75% of the content people watched on Netflix(Kasula 2020) are recommended by the recommendation system. Besides, Netflix saves more than 1 billion per year due to the effect of the recommendation system. The emergence of the recommendation system plays a dramatically significant role in the information access systems. This is a win-win technology that allows the service providers to guarantee customer satisfaction at the same time gain dramatic revenues. It also benefits the users significantly, as they can explore the online services more efficiently.

In this project, I plan to create some hand-crafted features using the content-based approach and collaborative approach. Then build up several regression models to learn from this new informative dataset and compare the performance of these models with the baseline model: matrix factorization, one of the most traditional collaborative models. Those regression models in this problem serve as hybrid models since the dataset they are learning from is generated by leveraging both the content-based approach and collaborative approach. The research question my project aims to solve is to evaluate whether my novel hybrid movie recommendation systems give more accurate movie recommendation, that is achieving lower RMSE (root mean square error) than my baseline model matrix factorization.Also showcase that feature engineering could support simple regression models to perform better recommendation tasks than matrix factorization without using feature engineering. In general, recommendation system algorithms typically fit into 3 categories, which are content-based systems, collaborative filtering systems, and hybrid recommendation systems. In this project, I will build up a collaborative filtering recommender using the matrix factorization algorithm, which takes advantage of user-movie interaction information to group a cluster of users sharing similar tastes and recommends the movies that other users like in this cluster to the target user. To build up a more robust and reliable hybrid model, I will craft some new features based on user similarity and movie similarity for the purpose of training the hybrid models. In this way, the hybrid models can learn from a more informative dataset that not only considers users' movie preferences but also takes into account the property of the users who share similar tastes with the target users. In this project, I will empirically compare the performance of the hybrid models with the baseline model and shows that the hybrid models should achieve lower RMSE than the baseline models.

The models will be trained and evaluated on the dataset MovieLens(ml-latest-small)(GroupLens 2018), a subset of movie dataset(ml-latest-large), containing 100836 ratings and 3,683 tag applications applied to 9,742 movies by 610 users. In addition, I will also take advantage of the movie meta-data available in the ml-latest-large for extracting the content-based features. Unlike, the traditional machine learning task, the recommendation algorithm cannot be evaluated using the traditional performance metrics, like accuracy, precision,etc. I will use root mean square error(RMSE) and mean absolute error(MAE) as the performance metrics, to measure how accurate the rating predictions of the given recommender system are. RMSE is measure of the difference between rating predicted by the recommender and the user rating observed and it is computed by the sum of all root square errors(RSE) divided by the total number of predictions, whereas MAE measures the average of the absolute difference between the actual rating and the predicted rating. Those two evaluation metrics tell how accurate the models predict the actual user ratings.The lower the RMSE or MAE the model predictions are, the better the model is and vice versa.

Main Results This study highlights the importance of feature engineering brings to model performance. A novel feature extraction strategy is proposed in this paper to capture the user's preference for movies. I provide an in-depth evaluation regarding three regression models training on the informative features and compare the models' performances with the baseline mode training on the raw dataset. Based on my finding, the proposed regression models perform significantly better than the baseline model. The best result achieved by the random forest model decreases 10.8% RMSE compared to the baseline model. This finding empirically outlines the importance of feature engineering and shows that feature engineering could extensively enable simple regression models to perform better recommendation tasks than a robust algorithm like matrix factorization without using feature engineering. The experiment result solidifies my assumption that feature engineering is an essential procedure in machine learning, and informative features could potentially enable the models to perform better. In addition, this study also empirically evaluates and compares the quality of the extracted features. The result shows that movie rating averages and user rating averages are two essential features for the proposed regression models. Surprisingly, the collaborative filtering features and content-based features are less important in the extracted features. The unexpected finding might be due to the cold-start problem, which causes the similarity matrix to become too sparse to capture a sufficient number of high-quality features.

Contributions The four main contributions made in this paper are stated below:

- I proposed a new feature extraction strategy that extracts the user rating behaviours from the raw dataset by leveraging averages, content-based approach and collaborative filtering approach.
- To retrieve high-quality content-based features, I proposed a recommendation system that considers a wide variety of movie factors that might impact user preference, including movie genre, keywords, director, and main characters. This feature extraction strategy allows a more reliable movie similarity matrix to be computed.
- I found that user rating average and movie rating average are the two most important features in all extracted features. However, combining all the extracted features could further enhance the model performance.
- I compared the proposed models' performance with one of the robust algorithms called matrix factorization on the same data set with different features. The proposed models achieve more accurate predictions than the baseline model due to a sound feature extraction strategy and model hyperparameter tuning.

Related Work

This section will briefly discuss the development of the recommendation system and summarize some of the prior work of the recommendation system. In 2009, Netflix sponsored a competition called Netflix Prize(Koren 2009), offering a grand prize of 1 million dollars to the team who had improved the 10% accuracy of the original performance. This competition is the primary driving force that energized the research in the recommendation system. There are mainly three categories of recommendations in this area: Content-Based recommendations, Collaborative recommendations, and hybrid recommendations.

Content-Based Filtering Approach

The content-based approach is a filtering algorithm that is designed to recommend the items to the users who liked in the past. The core of this algorithm is to measure the similarity between the movies. One of the most popular approaches is the vector space model, which captures the importance of the keywords in the movie metadata by calculating a TF-IDF score for each word.(Ma 2016) Content-based filtering finds the movies that are similar to users' history movie preference by measuring the similarity between the user content preference vector and the movie content matrix. Eventually, filtering out the most relevant and similar movies based on the similarity matrix.

Collaborative Filtering approach

Collaborative filtering is a technique that filters information by using the interactions and data collected by the system from other users who share similar tastes with the target users. Matrix factorization techniques turn out to be the dominant methodology with collaborative filtering recommender. This family of methods became widely known during the Netflix Prize challenge.Research(Koren, Bell, and Volinsky 2009) introduced the matrix factorization techniques that essentially paved a solid road for the collaborative filtering recommender. They illustrated the idea that the matrix factorization significantly outperforms the classical k nearest neighbor for movie recommendation. More specifically, it works by decomposing the user and movie interaction matrix into the product of the two lower dimensionality matrices - user matrix and movie matrix, respectively, using the stochastic gradient descent to minimize the cost function RMSE. Additionally, research(Ilhami and others 2014) cleverly builds up a recommender with the combination of matrix factorization and nearest neighbor algorithm. The experiment shows that the combination of matrix factorization, collaborative filtering, and k nearest neighbor algorithm can effectively improve the recommendations prediction accuracy after properly tuning the parameters k.

Hybrid Approach

A hybrid recommender is a recommendation system that leverages both content and collaborative approaches to overcome the common problems in pure approaches, such as cold start problems. To maximize the effectiveness of the recommendation system, ongoing research has constantly been proving the robustness that the hybrid approach brings. Several studies empirically compared the hybrid's performance with the pure collaborative and content-based methods and demonstrated that the hybrid methods could provide more accurate recommendations than pure approaches. There are several ways to implement the hybrid recommender; one possible hybrid approach works in a sequential manner where the hybrid recommender makes use of one pure recommender to provide additional information for the next recommender. In research(Melville, Mooney, and Nagarajan 2002), the hybrid model feeds the user-rating vectors into the content-based recommender to create the pseudo-user-rating matrix, which is the mixture of user actual rating vectors and prediction from the content-based recommender. The final recommendation predictions are produced by a collaborative filtering recommender based on the pseudo-user rating matrix. Besides, the weighted hybrid recommender is another way to combine the baseline models to provide a recommendation. In research(Miranda et al. 1999), a weighted hybrid approach provides recommendations by taking a weighted average of the predictions of content-based approach and collaborative filtering approach. Claypool et al initially set the equal weights to the content-based and collaborative filtering models and then gradually adjust the weighting of them based on the predictions of the ratings in the recommendation system. In research(Pazzani 1999), instead of assigning different weights to different models, Pazzani proposed a hybrid approach aimed at treating the output of the five recommendation baseline models as votes and providing the final recommendation using a consensus-based method. To check the influence of each recommender in the hybrid recommender, the consensus-based method is run five times, each time excluding one baseline learner, and measure the recommendation performance of the four other learners. The experiment result shows that consensus-based hybrid methods can effectively combine each baseline learner's strength and thus improve the overall recommendation accuracy.

Recommendation with Deep Learning

With the rapid development of deep learning algorithms, there is a hot trend that researchers put the focus on researching integrating the deep learning algorithm with the recommendation system. It turns out that the introduction of deep learning algorithms boosts the performance of recommendation systems well. Recent research(Devooght and Bersini 2016) explored the technique of converting the collaborative filtering to a sequence of prediction problems and introduced the recurrent neural network in particular LSTM to allow the model to take into consideration the evolution of users' taste. The researcher empirically compared the algorithm's performance with LSTM and collaborative filtering with the traditional nearest neighbor and matrix factorization approaches. The experiment result shows that the LSTM model outperforms that of the other state of art models. Using LSTM, but researching on a document recommendation task, Tan et al (Tan, Wan, and Xiao 2016) proposed a neural network approach to boost the performance of the content-based recommendation system in the domain of quote recommendation in writing. The research effectively leverages the LSTM to learn the distributed semantic representation of quotes vectors and the contexts and make the recommendation based on the relevance between quotes and contexts. The performance of the experiment shows the remarkable effectiveness of combining the LSTM framework with a content-based recommender. Other deep neural networks, like convolutional neural networks, have also been researched in recent years. In research(Kim et al. 2016), Donghyun et al developed a convolutional matrix factorization model that uses convolutional neural networks to capture the contextual information of users and movie meta-data. To test the convolutional matrix factorization model's effectiveness, the model is evaluated on a different dataset, for example, MovieLens and Amazon review datasets. The researchers compared the models with several other states of art collaborative learning competitors. The result shows that the convolutional matrix factorization outperforms the other models and can even overcome the sparsity problems. Moreover, the experiment result shows that a pre-trained word embedding model such as Glove can boost the model performance when the rating data is extremely sparse.

Cold Start Problem

The Cold Start problem is the most common problem in the recommendation system. It refers to when the recommendation meets a new user or a new movie. Some traditional approaches like the demographic-based method are applied to make recommendations by analyzing the user profile, such as age, gender, location, etc, or clustering the new movie based on the cast, crew, and genre. A more advanced hybrid approach(Basiri et al. 2010) called the optimistic exponential type of ordered weight averaging(OWA) operator, which is proposed to alleviate the cold start problem in the recommendation system efficiently. This hybrid approach works by first defining five traditional classifiers as the strategies- collaborative filtering method, content-based filtering, demographic-based method & collaborative filtering method,demographic-based filtering, collaborative filtering method & content-based filtering. Then feed the users' previous rates, demographic data, content information into those five classifiers. The final result is computed by feeding the predictions from the classifiers to the OWA algorithm. This proposed approach significantly outperforms the traditional recommendation system and provides a novel solution for the recommendation system to overcome the cold start problem in the future.

Methodology

Outline

In this project, I will construct a baseline model and several hybrid regression models for movie recommendations. The baseline model, matrix factorization, is trained solely on the original rating table where the dataset contains only userId, movieId, and ratings. However, for the hybrid models, I plan to create hand-crafted features to train the models, using the content-based and collaborative approaches, which can be effectively computed by measuring the user similarity and movie similarity matrices. After preparing the data, I construct three regression models: KNN, ridge regression, and random forest, to learn from this new informative dataset. Finally, I will empirically compare these hybrid models' performance with the baseline model and show the remarkable effectiveness of my proposed hybrid solution.

Dataset

The dataset I used to train and evaluate the model is MovieLens(ml-latest-small), a subset of the movie dataset(ml-largest-large). This dataset describes 5-star ratings and the tagging activities from movieLens, a movie recommendation service. In the rating file, the dataset contains 100836 user ratings on 9742 movies created by 610 users between March 29, 1996, and September 24, 2018, where each row of the file represents the rating that a particular user gives to a particular movie. The user ratings are made on a 5-star scale, with half-star increments(0.5 stars to 5 stars). The timestamp is displayed seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970, but it is of interest in this project. The links dataset contains the information on identifiers that can link the movies to other movie data sources. Each row of the links file contains movieId,imdbId, tmdbId, where imdbId, tmdbId are identifiers for different movie sources. In this project, I will mainly make use of the tmdbId to access the movie metadata dataset. The movie metadata dataset contains 45467 movie metadata, including genre, overview, crew, cast, popularity, etc. The movies in the rating dataset are just a subset of the movies in the movie metadata dataset. In this project, I will mainly use the attributes crew, cast, genre, keywords and then combine them in a way such that I can effectively extract similar movies that the users like. I chose the smaller MovieLens dataset because the full dataset is too big, which causes some memory limit issues during development. In this project, the smaller dataset should be sufficient to make valuable insights and provide accurate movie recommendations effectively.

Data Preprocessing

This section will discuss the data preprocessing steps that need to be done before crafting the new features for training the hybrid models. A user-movie interaction needs to be built for preparing the steps for measuring user similarity. To prepare the steps for calculating the movie similarity, I have to merge the rating table with the movie metadata table accordingly and extract the movie metadata I am interested in.

Collaborative filtering approach I constructed a user-movie interaction spreadsheet-style table where user id are displayed row-wise and movie id are displayed column-wise. Each of the cells in the user-movie tables shows the particular user's rating to the specific movie. I will call this table a user movie interaction matrix in the later article. Given that the users may not rate all the movies and not all movies are rated by all the users, I fill the cell 0 if the users do not rate a particular movie. In this case, I can take advantage of this users-movie interaction matrix and apply the cosine similarity to measure the similarity of users' preferences later.

Content-Based approach After doing the exploratory data analysis and doing some sanity checking on the dataset, I found several ill-formatted movie ids in the movie metadata dataset. Those ill-formatted rows are removed before merging the metadata dataset with the rating dataset in terms of the link dataset. Afterward, I gathered all the necessary movie metadata information(cast, crew, genre, and keywords) I am interested in to form a new feature. More specifically, from the crew column, I only extract the director as my feature since I don't think other information could be the significant factor that affects users' movie preferences. Besides, from the cast column, I decided to extract the top 3 characters in the movie as my features since I don't think the lesser-known characters affect people's opinion of a movie. To standardize the characters' names and director's names, I convert all the letters to lower cases and strip the spaces between first name and last name so that each term is unique in the feature. Finally, I combined these two extracted features with the keywords and genre to form a new combined feature column, which will be later used as the main features for measuring the movie similarity.

Feature Extraction

This section will discuss how I craft the relevant features before training the models. I will explain how I apply the TFIDF to extract features from the text data. Besides, explain how I apply cosine similarity on the user-movie interaction matrix and content-based features to get user and movie similarity matrices. Lastly, I will demonstrate the strategy of filtering the most relevant information from the user similarity matrix and movie similarity matrix.

Average feature To better understand the users' rating pattern and overall movie popularity, I calculate the average ratings of all movies given by a specific user and the average ratings of a specific movie provided by all users. These two averages are added to the rating table for all users and movies in each row, which serve as new features to capture the users' potential rating.

Collaborative filtering feature We know that the user-movie interaction matrix captures the user rating on different movies. So we can take advantage of this user-movie interaction matrix to compute the similarity between each user. Cosine similarity is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing to roughly the same direction. See Equation 1 below.

$$\cos\theta = \frac{\vec{A}\cdot\vec{B}}{\left\|\vec{A}\right\|\cdot\left\|\vec{B}\right\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

The resulting matrix is a 610*610 matrix(we have 610 users in the dataset) that measures the similarity of users' tastes based on their rating to different movies. The resulting similarity ranges from -1, meaning exactly opposite taste to 1, meaning exactly same tastes with 0 indicating decorrelation. After obtaining the user similarity matrix, I extract the hand-crafted features for the target user and movie by filtering out the top three most similar users who have also rated the particular movie. Knowing that each row of the rating table indicates a specific user U rates a specific movie M, this operation will be applied to all the users and movies in the rating table. Those similar users' ratings towards the movie M will be saved as the new features in the rating table. If the movie M does not have more than three ratings rated by 3 similar users, I will replace the feature with the average ratings of the target user U.

Content-based feature I also take advantage of the movie meta-data in the movie dataset(ml-largest-large) to construct the keywords for each movie to train the content-based model. In particular, I extract the attributes - director, the first three main casts, genre, and keywords of each of the movies, combine all those word features for each movie in a new column called "combined feature". We can now take advantage of each movie's "combined feature" to better measure the movies' similarity. During the information retrieval phase, I extracted the unigram and bigram features for each of the movie's metadata, and TF-IDF is applied to the sequences of unigrams and bigrams to weigh the importance of the features. TF-IDF is a statistical metrics that evaluate how important a particular word is to a comment in a large corpus, which is achieved by multiplying IDF and TF. A high TF-IDF score is performed by a high term frequency in a particular movie's metadata and a low frequency of the term among all the movie metadata. A brief mathematical explanation of how TF-IDF works is shown below. Let's denote a term by t, a document by d, and a corpus by D.We first measure the value of TF(t,d), which shows how frequently a term t appears in document d. We then calculate the inverse document frequency (IDF), which indicates how much information a word carries. DF(t, D) measures the number of documents that contain the term t. Intuitively, IDF is computed using Equation 2.

$$IDF(t, D) = \frac{|D|}{DF(t, D)} \tag{2}$$

Words with low IDF scores indicate that they appear in most movie metadata and carry little information for particular movie metadata. If a term appears in all documents, this will result in IDF(t, D) = 0. As we cannot divide by 0, we smooth the value by adding 1 to the denominator and rescale the IDF by taking the logarithm, as shown in Equation 3.

$$IDF(t, D) = \log\frac{|D| + 1}{DF(t, D) + 1} \tag{3}$$

After obtaining IDF(t, D) and TF(t,d) for word d, I multiply those scores together and get a final TF-IDF score, as shown in Equation 4. Each word and bigram in a particular document contains a TF-IDF score indicating the importance of the term in movie metadata.

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D) \tag{4}$$

After extracting the TFIDF scores for each movie's metadata, the cosine similarity could be applied to each movie's TFIDF scores. The resulting matrix is a 9742*9742 matrix(we have 9742 movies in the dataset) that measures movies' similarity based on the movies' metadata. The resulting similarity ranges from -1, meaning exactly the opposite type, to 1, meaning the same type with 0 indicating decorrelation. After obtaining the movie similarity matrix, I create the hand-crafted features for the target movie by extracting the top three most similar movies that the target user has also rated. Knowing that each row of the rating table indicates a specific user U rates a specific movie M, this operation will be applied to all the users and movies in the rating table. Those similar movies that the target users U rated will be saved as the new features in the rating table. If the users U did not rate more than three similar movies, I will replace the feature with the average ratings of movie M.

Model

This section will briefly discuss how the baseline model works and describe the three algorithms I implement for building the hybrid regression models. Besides, discuss the rationale of why those algorithms are selected for the movie recommendation problem.

Baseline model-Matrix Factorization Matrix factorization is deemed to be one of the classical collaborative recommendation algorithms. The approach becomes widely known after the Netflix prize challenge. Since then, a lot of the research has been focusing on exploring this novel approach. In this project, I will treat this algorithm as the baseline model and empirically test my hybrid models to see if they show a remarkable improvement compared to the baseline model. Matrix factorization works by decomposing the user-movie interaction matrix into a product of two lower dimensionality rectangular matrices, which are user matrix and movie matrix, respectively. The decomposition is achieved in such a way that the product results of the user matrix and movie matrix are as close as possible to the rating in the user-movie interaction matrix. Once the user matrix and movie matrix are obtained, we can use those matrices to predict the user ratings. To effectively test this baseline technique, I use a built-in library called Surprise that uses a robust algorithm called singular value decomposition to minimize the root mean square error and thus predict the user ratings based on the product of the user matrix and movie matrix.

K-nearest neighbors regression KNN is one of the classical models that measures and groups similar items, and it is widely used in the recommendation application. KNN algorithm is capable of solving both classification and regression tasks. In this project, I will use KNN regression as one of the hybrid models. K nearest neighbors is an algorithm that stores all training data and predicts the numerical target based on a similarity measure. KNN regression approximates the association between independent variables and the continuous outcome by averaging the K nearest neighbors' ratings. It would be reasonable to apply KNN on the hand-crafted dataset and test to see if it can effectively predict how the users will like a specific movie based on the neighborhood's ratings. To choose the optimal hyper-parameter k, a set of the hyper-parameter k are predefined for parameter pruning. The technique,5-fold cross-validation, will be applied to select the optimal size of the neighborhood that minimizes the RMSE the most.

Generalized linear regression Linear regression is a widely used approach for predictive modeling. Linear Regression establishes a relationship between the dependent variable (Y) and one or more independent variables (X) using a best fit straight line. In other words, this model is used with the hope that the linear regression can find the best fit to measure the relationship between the features and actual user rating. To avoid the model suffering from the multi-collinearity issue, a model tuning method, ridge regression, is used in this project to fit the data, which introduces the penalty term with λ as a coefficient in the loss function(shown in Equation 5). The coefficient λ controls the amount of shrinkage, where the greater the amount of the shrinkage, the coefficient becomes more robust to collinearity.In this case, 5-fold cross-validation is applied to find the best λ that can minimize the loss function the most.

$$L_{ridge} = argmin_{\widehat{\beta}}(\|Y - \beta * X\| + \lambda * \|X\|_2^2) \quad (5)$$

Random Forest Random forest is considered a highly robust approach in machine learning, and it becomes increasingly popular in recommendation systems. Random forest is a supervised learning method for regression or classification tasks that works by building multiple decision trees at training time and predicting the final classes or values based on outputs of all the predefined decision trees, more specifically, by average predictions(regression) or majority voting (classification).Random forests construct decision trees on randomly selected data samples from the hand-crafted dataset and make final user rating predictions by taking the average of each prediction produced by the decision tree. I chose to implement a random forest rather than a single decision tree because the random forest prevents overfitting in such a way that it takes an average of all the predictions, which cancels out the biases. To optimize the model and minimize the RMSE, I plan to apply 5-fold cross-validation to tune the hyper-parameter and find the optimum max depth that each decision tree could explore.

Results

Experiment design

This section will discuss experiment design, present the experiment results, and compare the models' performance. The baseline model, matrix factorization, is trained and evaluated on the original rating table, containing the userId, movieId, rating. The new dataset is constructed with the consideration of both the content-based approach and collaborative filtering approach, containing eight features in total, which are movie rating average, user rating average, three similar movie ratings, and three similar user ratings. The three hybrid regression models will be trained and evaluated on the new dataset.
To prepare the experiment, I randomly shuffle the original rating dataset and split the dataset into a 90% training set and 10% testing set. Similarly, for hybrid models, I use the same set of training sets and testing sets as the baseline model experiment, but this time the same set of the dataset with the new features I crafted earlier.
Furthermore,to perform parameter tuning, I plan to apply 5-fold cross-validation for both baseline model and hybrid models. The surprise library for implementing matrix factorization has a robust CV iterator for cross-validation procedures. Similarly, 5-fold cross-validation is applied to do parameter tuning for my hybrid models. A set of the predefined hyper-parameters are defined and then split the training set into five subsets. For each value of the hyper-parameter, five rounds of learning will be performed where one of the five folds is the validation set, and the other four sets are the training set. The hybrid models are iteratively trained on the training set and evaluated on the validation set . After five rounds of training, I will calculate the average RMSE of each hybrid model on the validation set, where the average is taken over the five rounds. Finally, I will select the best hyper-parameter with the lowest RMSE on the validation set.

Evaluation metrics

Unlike the traditional classification problem, the recommendation system cannot be evaluated using traditional metrics like accuracy, or recall, since the movie recommendation task is not about predicting exactly what the user will rate for a particular movie. Instead, the task is more about approximating the rating that the user will rate in the future and minimizing the model's overall errors when making recommendation decisions. The recommendation system can recommend the movies that the user will potentially give a high rating based on its prediction. To evaluate the model's performance, I mainly use two evaluation metrics to measure the error of models: root mean square error(RMSE) and mean absolute error(MAE). Root mean square error(RMSE) measures the average magnitude of the prediction error, which is computed by taking the difference of the predicted rating and the observed rating and then averaged over the sample and finally taking the square root of the average, see the formula 6 below.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}} \tag{6}$$

Mean absolute error(MAE) is also a popular evaluation metric in recommendation system that measures the average magnitude of the absolute difference between the actual rating and the predicted rating, see the formula 7 below.

$$MAE = \frac{\sum_{i=1}^{n}|P_i - O_i|}{n} \tag{7}$$

The evaluation metrics, MAE and RMSE, measure the magnitude of the error the model has in different ways, but both tell how accurate the models predict the actual user ratings. The lower the RMSE or MAE the model predictions are, the better the model is and vice versa. In the real-world scenario, a sound recommendation system should have both low RMSE and MAE, and it is supposed to approximate the actual rating that a particular user will give to a movie. If the predictions are higher than a certain threshold, for example, 4.5, the recommendation system should recommend the movies to the users.

Experiment

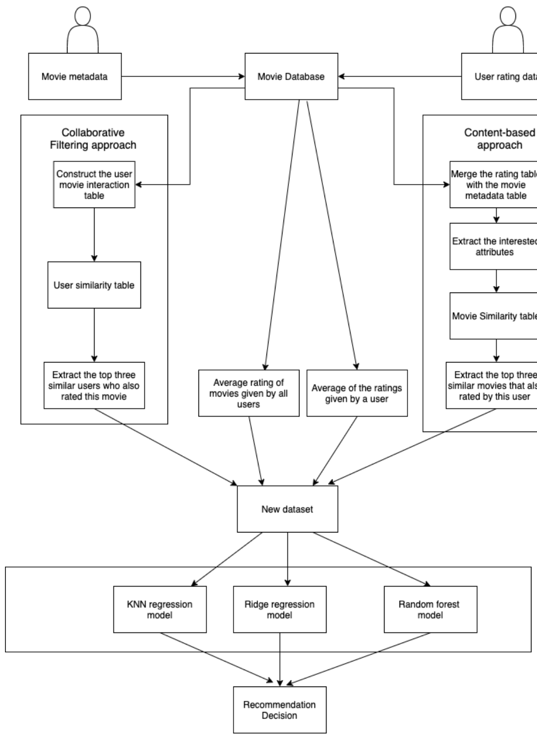

Movie metadata
Movie Database
User rating data
Collaborative Filtering approach
Construct the user movie interaction table
User similarity table
Extract the top three similar users who also rated this movie
Average rating of movies given by all users
Average of the ratings given by a user
Content-based approach
Merge the rating table with the movie metadata table
Extract the interested attributes
Movie Similarity table
Extract the top three similar movies that also rated by this user
New dataset
KNN regression model
Ridge regression model
Random forest model
Recommendation Decision

Figure 1: Recommendation system architecture

The recommendation system first gathers the relevant dataset from the movie database and produces some new features by leveraging rating averages, collaborative filtering, and content-based approaches. More specifically, the system first introduces two general features into the new dataset: the average ratings of all movies given by a specific user and the average ratings of a particular movie provided by all users. After that, the system measures similar users' behaviours by constructing a user similarity matrix based on the user-movie interaction matrix. Then, it introduces the top three ratings of other similar users who also rated this movie into the new dataset. Similarly, the system extracts and combines all the relevant metadata, including keywords, crew, director, genre. Then the system applies TF-IDF with unigram and bigram to the metadata data and uses the Pearson correlation on the TF-IDF scores between each movie's metadata to measure the movies' similarity. After that, the system extracts the top three ratings that a specific user gave to other similar movies. The recommendation system organizes all the new features in a new dataset before feeding the new features into the three hybrid regression models. To tune parameters for each model, the regression models, including the KNN model, ridge regression model, and random forest model, are trained and validated with 5-fold cross-validation. In this project, I mainly focus on optimizing the models and comparing the recommender performances. However, in the real world, the recommendation system can add another ensemble learning layer upon the models' layer where each hybrid regression model makes a prediction of a user's future rating on a particular movie. If a model's prediction of the future rating is high enough; for example, the prediction is above a threshold of 4.5-star, we can assume this model agrees to recommend this movie to the user and vice versa. In this way, the recommendation system can make the final decision based on the majority votes of all models' predictions and recommend the movies that most models think the target user will like. The architecture of the recommendation system can be seen in Figure 1.

Data exploratory

Figure 2 shows the distribution of user ratings on different movies in the dataset. The histogram gives a big picture of the movie rating pattern. Most users gave relatively high ratings to the movies, while some users still give 1-star ratings to some movies. Figure 3 shows the number of movies a particular user rated. The red horizontal line marks the threshold - 50 movies a user rated. The scatterplot illustrates that most users tend to rate many movies, while a few users still give little ratings to movies. This pattern needs taking extra care later when doing data mining since it is harder to extract features from the users who give little ratings than those who give more ratings.

Experiment Result

Matrix factorization The matrix factorization as the baseline model is trained and evaluated on the original rating dataset using the built-in library surprise. The cross_validate iterator runs 5-fold cross-validation and trains on four folds of data for 20 epochs, and evaluates the RMSE and MAE of

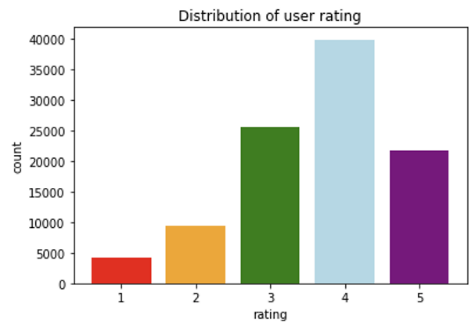


Figure 2: User rating distribution

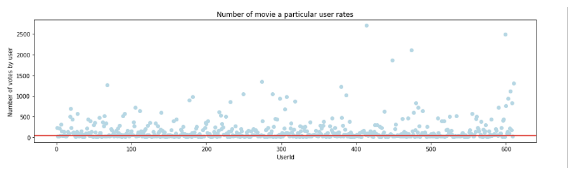


Figure 3: Number of movie a particular user rates

the algorithm SVD on one fold of the data. Table 1 shows the RMSE and MAE values on five validation sets, the training set and the testing set. The test result of the RMSE value is 0.8746, and the MAE is 0.6727. It is also noticeable that the baseline model fits the training data well but has a poor fit with the testing set, indicating that the model overfitting occurs.

	Baseline model	RMSE	MAE
	1st fold	0.8793	0.6765
	2nd fold	0.8779	0.6726
	3rd fold	0.8756	0.6725
	4th fold	0.8800	0.6791
	5th fold	0.8809	0.6759
	Training set	0.6913	0.5310
	Testing set	0.8746	0.6727

Table 1: RMSE and MAE value of baseline model

KNN KNN is one of the traditional models in the recommendation system. Based on the predefined neighbour threshold k, the algorithm groups the k nearest neighbour and compute an average of the neighbours' ratings. In this experiment, I set K to be 1, 2, 4, 8, 16, 32, 64, 128, 256 as the hyperparameter candidates and 5-fold cross validation is used to search for the k that best fits the model. Figure 4 shows the result of the 5-fold cross-validation. The blue curve shows the average RMSE of the KNN regression model on the validation set, whereas the orange curve shows the average MAE of the KNN regression model on the validation set. Both RMSE and MAE curves decline significantly and converge when k=8. Then, I trained the KNN model with k=8 using all the training data and test the model on the unseen testing set. The test result of the RMSE value is 0.8226, and the MAE is 0.6212, shown in the table 3.

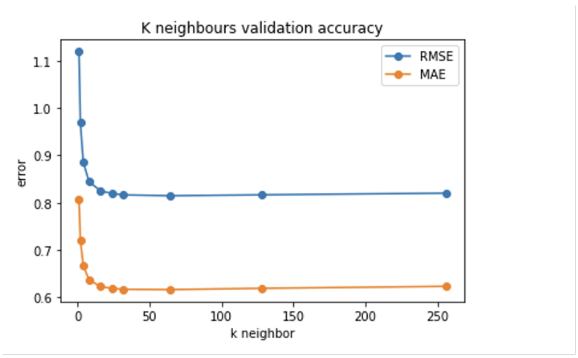


Figure 4: Average validation errors for KNN model

table 3.

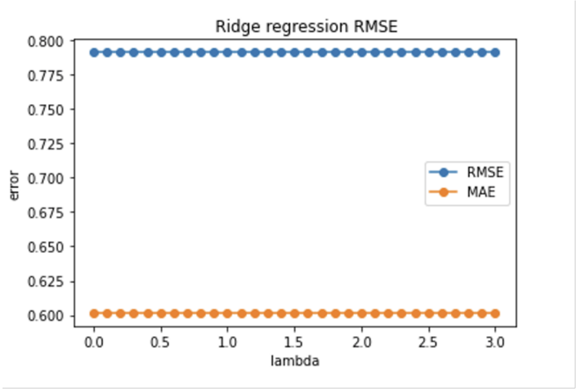
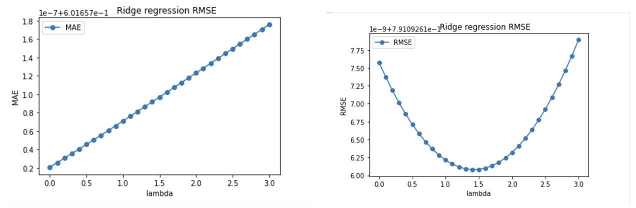


Figure 5: Average validation errors for ridge regression



(a) Average validation MAE (b) Average validation RMSE

Figure 6: Average validation errors for ridge regression

Ridge Regression In the ridge regression model, I introduced the hyperparameter lambda into the objective function to prevent overfitting. I chose 0 to 3 with 0.1 increments as my hyperparameter candidates. In the experiment, I tested the effect of lambda on the model and attempted to find an optimal lambda that can prevent overfitting while reducing the error. Figure 5 shows the validation error of the ridge regression model. It is clear from this plot that lambda's magnitude does not contribute much to the model itself. The curves' change might be too subtle to capture on the same scale in the same plot. Plotting one single curve at a time can largely magnify the effect. Figure 6 shows that the MAE value increases as we increase the lambda's magnitude, whereas the RMSE decreases as the lambda increases, reaches the minimum when lambda equals 1.5 and then shoots up again after 1.5. In this case, I chose 1.5 as my hyperparameter to train the ridge regression model using all the training data and test the model on the unseen testing set. The test result of the RMSE value is 0.7879, and the MAE is 0.6001, shown in the

Random Forest The random forest algorithm is an ensemble learning model that prevents the model from being overfitted by constructing multiple decision trees and allows each decision tree to train on samples(with replacement) of the training data. This ensures that the regression model does not overly rely on any individual features. In this experiment, I treated the max depth of each decision tree as the hyperparameters to tune. I set 1,2,4,8,10,15,20,25,30 as my hyperparameter candidates.After repeated experiments, I found that the model converges in the range between 10 to 20. Then I decided to shrink the max depth domain accordingly. Figure 7 shows the trend of the average validation error of the random forest algorithm. The RMSE and MAE trends maintain similar tendencies. Both decrease significantly and converge when max depth equals 10. In this case, I trained the random forest model with max depth 10 using all the training data and test the model on the unseen testing set. The test result of the RMSE value is 0.7798, and the MAE is 0.5879, shown in the table 3.

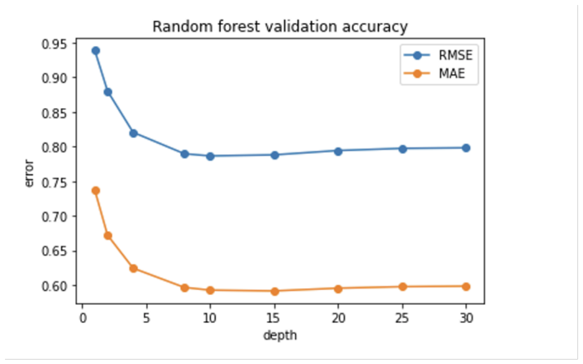


Figure 7: Average validation errors for random forest model

Experiment Comparison This section focuses on comparing the model performances by showing the baseline and hybrid models' experimental results. Table 2 shows the RMSE and MAE of all models on training data, and Table 3 shows the RMSE and MAE of all the models on the testing dataset. The baseline model is likely to overfit the training set since it performs the best among all the models during training time but does not generalize well on the testing data. All three hybrid models training on the new dataset perform better than the baseline model on the testing set. In particular, the best result was achieved by the random forest model with 0.7798 RMSE value and 0.5879 MAE value on the unseen testing dataset, which is an improvement of 10.8% compared to the baseline model.

	model	RMSE	MAE
	Baseline model	0.6913	0.5310
	KNN	0.7703	0.5808
	Ridge Regression	0.7910	0.6016
	Random Forest	0.7452	0.5652

Table 2: RMSE and MAE of all the models on training data

	model	RMSE	MAE
	Baseline model	0.8746	0.6727
	KNN	0.8226	0.6212
	Ridge Regression	0.7879	0.6001
	Random Forest	0.7798	0.5879

Table 3: RMSE and MAE of all the models on testing data

	model	P_value
	Baseline model and KNN	8.5193e-05
	Baseline model and Ridge regression	3.2586e-07
	Baseline model and Random Forest	8.2102e-07
	KNN and Ridge regression	4.9866e-05
	KNN and random forest	3.5890e-04
	Ridge regression and Random Forest	0.0132

Table 4: 5x2cv paired t-tests result

I also applied 5x2cv paired t-tests on all the candidate model pairs to compare whether the models' performances are significantly different. 5x2cv paired t-tests is a built-in library that compares two models' performance and tests whether the difference between models is real or due to statistical chance. This test is done under the null hypothesis that the two models have equal performance. Given that this API only takes the dataset with the same feature, I have to write an extra test function to adjust the dataset format to do the 5x2cv paired t-tests on the baseline model and hybrid models. Table 4 shows the p values after applying 5x2cv paired t-tests to all the candidate models pair. P_value < 0.05 indicates that we have evidence against the null hypothesis that the two models have equal performance. The experiment result shows strong evidence against the null hypothesis that all three hybrid models have equal performances with the baseline model. Besides, the ridge regression and random forest models are significantly different from the KNN model. Also, we have some evidence against the null hypothesis that random forest and ridge regression models have equal performance. In other words, the result that the random forest model performs slightly better than the ridge regression model is not due to statistical chance. In summary, the test results effectively show that the hybrid models deliver the recommendation task more accurately than the baseline model. In particular, the random forest model performs the best among all hybrid regression models.

Figure 8 shows the first 20 predictions comparisons of the baseline model, hybrid models, and the user's actual ratings in testing data. The plot shows that the baseline model fails to predict the actual user ratings well. In contrast, the

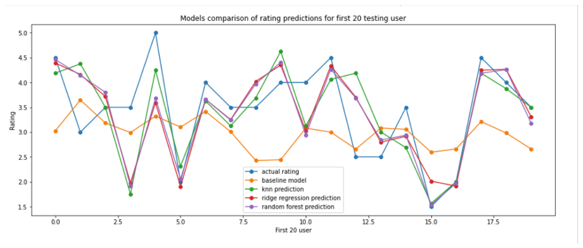


Figure 8: Model predictions compared with actual ratings

hybrid models could still capture each user rating's gross pattern and tend to make good predictions of the user ratings for most cases. This observation is consistent with each model testing result of RMSE and MAE values shown in Table 3. However, it is also noticeable that the hybrid models fail to capture some user's preferences. Figure 9 shows the performance comparison of training the hybrid models on different sets of new features. The bar plot illustrates that user average and movie average are the two most significant features in the new dataset. The model does not generalize well when training solely on features extracted by the collaborative filtering approach or content-based approach. This observation is consistent with the experiment conducted by M.Chenna Keshava et al (Keshava et al. 2020). However, by combining all the features, the models can find more insightful correlations in the dataset and effectively boost the model predictions. Compared with my models' performance to M.Chenna Keshava et al (Keshava et al. 2020), my novel hybrid regression models show great effectiveness of delivering movie recommendation tasks, improving around 27% performance than their best model (RMSE=1.0675).

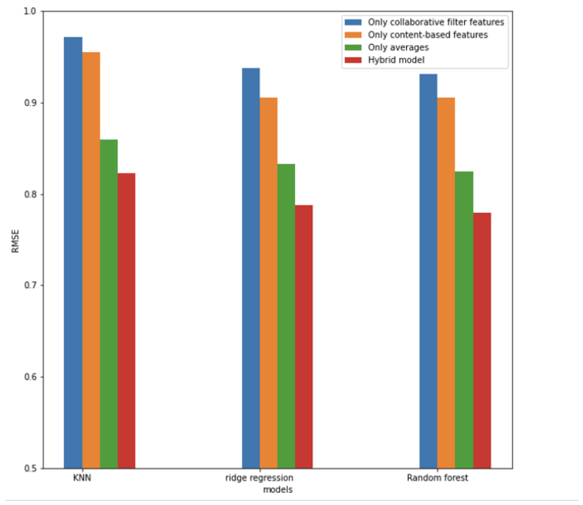


Figure 9: Model performance after training on different features.

Challenge and optimization

This section discusses how several optimization strategies have been carried out during implementation to cope with different implementation challenges.

Content-based features selection Initially, I chose the movie overview attribute as my content-based features to find the correlation between movies. This operation is extremely time-consuming and even hits the memory limit. I realized some movies have very long descriptions but with only several essential keywords in the context. Also, I realized the overview should not be the only factor that impacts the users' movie preference. After carefully exploring the movie metadata, I found that other significant factors, including genre, director, main characters, and the movie's keywords, are good representatives of movies. These movie components are likely to produce more meaningful information than the overview attribute. Compared with the previous approach, this optimization plan tends to extract more insightful features and also significantly boost the efficiency of the feature selection process.

Feature selection efficiency issue Extracting new features from the user similarity matrix and the movie similarity matrix are very time-consuming procedures since this operation needs to be done for every single pair of users and movies, respectively. More specifically, we need to locate the similarity vector for every user or movie and sort the vector to extract the top three most correlated users and movies, which is an O(N^3) operation. In this case, there is no alternative approach to boost efficiency other than optimizing the sorting algorithm. But the process is still very slow after optimizing the sorting algorithm. To improve the implementation efficiency, I stored those newly-extracted features inside a new CSV file. In this case, I do no need to do the feature selection phase over and over again. Instead, I can directly read all these pre-prepared new features from the CSV file every time before training the model.

Ways to select the hyperparameters candidates Parameter tuning is also one of the optimization phases I have done to improve my models. I carefully chose the candidates' hyperparameters for each model. For KNN, I defined the hyperparameter k to grow exponentially from 1 to 256 to test a bigger domain space and check how the neighbour size impacts the final error. For ridge regression, I chose the lambda to increase from 0 to 3 with a 0.1 increment. Incrementing lambda a little bit each time gives me a bigger picture of how the magnitude of the lambda continuously impacts the validation error. For the decision tree, building multiple trees with large depth is very time-consuming. To improve this, I first tested the model with max depth, a wider range from 0 to 60 with an increment of 5, noticing that there is no apparent model improvement after max depth equals 30. Then I decided to shrink the hyperparameters domain accordingly, which enables me to find the best hyperparameter more efficiently in the later cross-validation process.

Lesson learned In this project, I conducted several experiments to show the effectiveness of the hybrid regression model training on new features. These new features play significant roles in helping regression models learn since they capture the potential correlation between user and movie interaction and uncover some unseen patterns in the original rating dataset. From the experiment result, the random forest regression model outperforms the baseline model and the other two hybrid models. This might be because the random forest regression model has an effective mechanism to prevent overfitting. The random forest algorithm constructs multiple decision trees and determines the final decision based on every single decision made by each decision tree. Introducing extra randomness to the model ensures that the regression model does not overly rely on any individual features and thus largely prevents overfitting.

Nowadays, many movie recommendation research focuses on leveraging the models to improve the recommender's performance. However, the recommendation models I built were hybrid in the sense that the regression models learn the hybrid features produced by averages, collaborative filtering and content-based approaches. This project proves how effective the data mining procedure is when delivering the recommendation task. Data mining is essential for discovering the raw data patterns and preparing more insightful information for the model training. I have learned the lesson that improving the models' performance is not restricted to leveraging the models. However, extracting more insightful data can also give the models a boost in error reduction.

Discussion

This paper proposed a data-driven hybrid recommendation system that trains three regression models, namely KNN, ridge regression model, and random forest models, on a new hybrid dataset. This study mainly tests on the assumption that the hybrid movie recommendation system gives more accurate movie recommendation, that is, achieving lower RMSE or MAE than the baseline model, matrix factorization. The experiment result shows the remarkable effectiveness of the hybrid recommendation regression models in delivering recommendation tasks. In particular, the best-performed random forest model achieves the lowest RMSE or MAE values among all models. RMSE and MAE are two popular evaluation metrics measuring the average magnitude of the prediction error. Lower RMSE or MAE values achieved by the model imply that the model, on average, predicts the user rating on a particular movie more accurately than those models having high RMSE and MAE values. In essence, these results suggest that the random forest algorithm inherits a robust ensemble-learning mechanism for preventing models from overfitting and thus predicts the user rating more accurately than others. This finding is consistent with the work done by Sathiya Devi et al (Sathiya and Parthasarathy 2019). They proposed completely different feature engineering strategies. Their experiment result shows that the ensemble methods perform better in all the proposed feature engineering techniques, which is consistent with my finding that the ensemble learning model achieved best in my experiment setup. However, my proposed random forest model still shows greater effectiveness than the gradient boost regression ensemble method they proposed, showing a 7.8% improvement.

My study's finding provides a powerful argument for exploring relative importance of the extracted features. It is noticeable that the combination of those eight features can give the model a boost in model performance, as the model can find more correlation between the features and thus enable a more robust model to fit the data. However, It is surprising that the user rating averages and movie rating averages are the most important features in the dataset. User rating averages are considered a good representation of how likely users will rate on movies on average, while the movie averages can summarize the popularity of the particular movie as a whole. These features could be powerful for models to capture the correlation. This finding is consistent with the work done by M.Chenna Keshava et al (Keshava et al. 2020). The feature extraction strategy they proposed also verifies that the user rating averages and movie rating averages are the two most important features. The only difference is that they compared the features' importance by comparing the F score each feature achieved on different models. My novel hybrid regression models show the remarkable effectiveness of delivering movie recommendation tasks, improving around 27% of their models' best performance. Besides, It is also surprising that content-based features and collaborative filtering features do not contribute much to the new dataset. We know from Figure 3 that a group of users only rated a small portion of movies. Insufficient interaction with the system can cause the data sparsity problem in the user similarity matrix or movie similarity matrix. Because the similarity matrix is large and sparse, it could be a great challenge to find sufficient similar users who rated the same movie or similar movies rated by the same users. Therefore, the data sparsity problem causes the quality of the extracted features to become questionable and limits the usefulness of the content-based features and collaborative filtering features.

My proposed solution scales well on users with a decent amount of interactions with the system. However, one possible limitation is that limited user interaction with the recommendation system impedes the system from drawing meaningful inferences for users or items. In this study, the system design sidesteps the cases where the users or movies have no rating history and mainly focuses on the cases where users have a decent amount of interaction with the system. The insufficient rating for a user or a movie is deemed untrustworthy and is likely to introduce bias to the system. Hence, I introduced the movie rating averages and user rating averages in the new dataset to alleviate the bias effect. The experiment result suggests that movie rating averages and user rating averages are likely to provide sufficient information for models to draw inferences from those with very few interactions with the system. However, It is still problematic to deal with cases where the user or movie has no rating history. A point for improvement lies in the way that we could recommend the most popular movies to new users without knowing their preference. Alternatively, We could pre-group the new user with the existing users based on the demographic information before recommending the movies to new users.

Due to the limited computational resource, I could not test my proposed solution on a larger dataset. There is a great demand for computational resources when extracting features from the similarity matrix as users and movies increase. As the number of users or items increases, the designed recommendation system tends to suffer from scalability issues since the user movie interaction matrix I deal with has more than 90% of the values being 0. Storing such a sparse matrix could be problematic. In this case, the proposed recommendation system might also be hard to accommodate millions of data in a real-time fashion efficiently. Therefore, a large amount of computation power and a more scalable strategy need to be acquired to support handling a large amount of rating data in real-time application.

The proposed solution also overlooked the selection of the train-test set splitting strategy. My experiment design randomly split the dataset into a 90% training set and 10% testing set. Given that some users rated many movies and some users had very little interaction with the system, the training models might be biased towards the users with more interaction with systems. Another extreme case is that if the training set contains those users who only rated one movie, the model would not have the chance to estimate those users' preferences in the testing set. Likewise, suppose the testing set includes those users who only rated one movie. In that case, the model might need to predict the user rating without prior knowledge of their history preferences. Research (Meng et al. 2020) shows that different data splitting approaches could have a strong impact on the ranking of recommender systems. Ideally, a more mature data splitting strategy should be extensively explored in the future so that the model can be trained and evaluated in a more standardized manner.

In conclusion, my proposed solutions perform reasonably well in delivering movie recommendation tasks when the users have a decent amount of interactions with the system. The advantage of adequately doing feature engineering gives the regression models more capabilities to learn the correlations between each feature. However, more investigation of overcoming the potential limitations needs further exploration in future work. Furthermore, my results build on existing evidence that feature engineering could extensively boost the recommendation system's performance. My novel solutions combining a sound feature engineering strategy with a proper model selection phase could further enhance the system's performance compared to the prior work. In practice, my study shows promising practical implications. The proposed feature engineering algorithm could help automate the process of gathering valuable features for any movie recommendation system, as long as a decent amount of user and item interaction are available. To make this project applicable to a more versatile recommendation system, for example, friend suggestion or ad recommendation on social media, the extracted rating features might need to change to a more standardized form to capture how likely a user interacts with a particular item.

Conclusion

This paper proposed a data-driven hybrid recommendation system that trains three regression models, namely KNN, ridge regression model, and random forest model, on a new hybrid dataset. The new hybrid dataset is produced by leveraging the rating averages, content-based approach and collaborative filtering approach from the user rating and movie metadata dataset. The proposed models are trained and evaluated on the new dataset with 5-fold cross-validation to tune the hyperparameters. This study provide solid evidence that feature engineering could extensively enable regression models to perform better recommendation tasks than a robust baseline model, matrix factorization without using feature engineering. The test result shows that the hybrid regression models are significantly reliable, as seen by the low RMSE compared with the baseline model. In particular, the random forest model achieves the best performance among all candidates, improving 10.8% compared to the baseline model. The recommendation system's remarkable effectiveness could be attributed to the feature engineering procedure, which uncovers some hidden patterns in the raw dataset. In addition, the experiment result also showcases the importance of each extracted feature and compares how effective the models become after training on the new features. User rating averages and movie rating averages are the two most important features among all. Surprisingly, the content-based features and collaborative filtering features are less important due to the data sparsity problem.

Future research could expand the system in many ways. Data sparsity problems originate from the user or moving having limited interaction with the system. This problem is also called the cold start problem. Apparently, It is impossible to extract useful features of users or items new to the system since they have no rating history. The demographic recommender approach is a promising strategy to mitigate the cold start problem. In future work, It would be interesting to introduce the users' demographics information, for instance, gender, age, location, to group the new users with existing users to provide the relevant recommendations.

In addition, It would be interesting to try a deep learning model like RNN. The current models and the prepared dataset are a static framework. But in reality, users' preferences keep evolving, leading them to redefine their taste. Similarly, movie popularity constantly changes as new movies emerge. Thus, the system should consider the temporal effects reflecting the dynamic nature of user-item interactions. RNN is a deep neural network that is good at modelling sequential tasks and can better adjust the recommendation decision based on the evolution of user preferences. It is robust in the sense that the model itself has a memory of the history computation and thus can better capture the sequential pattern of user behaviour. In this case, the timestamp attribute needs to be introduced during the data-preprocess procedure. Each interaction between a particular user and the corresponding movies needs to be sorted based on the timestamp to serve as input to the RNN model. The recommendation system can now define each user as the function of the item interaction history and recommend personalized recommendations to users dynamically.

References

Basiri, J.; Shakery, A.; Moshiri, B.; and Hayat, M. Z. 2010. Alleviating the cold-start problem of recommender systems using a new hybrid approach. In *2010 5th International Symposium on Telecommunications*, 962–967. IEEE.

Dayton, E. 2020. Amazon statistics you should know: Opportunities to make the most of america's top online marketplace.

Devooght, R., and Bersini, H. 2016. Collaborative filtering with recurrent neural networks. *arXiv preprint arXiv:1608.07400*.

GroupLens. 2018. Movielens latest datasets.

Ilhami, M., et al. 2014. Film recommendation systems using matrix factorization and collaborative filtering. In *2014 International Conference on Information Technology Systems and Innovation (ICITSI)*, 1–6. IEEE.

Kasula, C. P. 2020. Netflix recommender system — a big data case study.

Keshava, M. C.; Reddy, P. N.; Srinivasulu, S.; and Naik, B. D. 2020. Machine learning model for movie recommendation system. *International Journal of Engineering Research and* V9(04).

Kim, D.; Park, C.; Oh, J.; Lee, S.; and Yu, H. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM conference on recommender systems*, 233–240.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Koren, Y. 2009. The bellkor solution to the netflix grand prize. *Netflix prize documentation* 81(2009):1–10.

Ma, K. 2016. Content-based recommender system for movie website.

Melville, P.; Mooney, R. J.; and Nagarajan, R. 2002. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai* 23:187–192.

Meng, Z.; McCreadie, R.; Macdonald, C.; and Ounis, I. 2020. Exploring data splitting strategies for the evaluation of recommendation models. In *Fourteenth ACM Conference on Recommender Systems*, 681–686.

Miranda, T.; Claypool, M.; Gokhale, A.; Mir, T.; Murnikov, P.; Netes, D.; and Sartin, M. 1999. Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*. Citeseer.

Pazzani, M. J. 1999. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review* 13(5):393–408.

Sathiya, D. S., and Parthasarathy, G. 2019. Feature engineering based approach for prediction of movie ratings. *International Journal of Information Engineering and Electronic Business* 11(6):24.

Tan, J.; Wan, X.; and Xiao, J. 2016. A neural network approach to quote recommendation in writings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 65–74.