**A Comparison of Filtering Approaches in the Recommendation of Movies using MovieLens Data**

Nick Pasternak (nfp5ga), Kara Fallin (kmf4tg), Aparna Marathe (am7ad)

1. **Abstract**

Movie subscription services are becoming increasingly popular and competitive as several new streaming platforms are being developed. In order to fully capitalize on this market and retain users, it is essential that streaming companies have effective algorithms for recommending movies to new and existing users. Currently, the two most popular recommendation systems are collaborative filtering and content-based filtering. In this paper, we compare the effectiveness of both types of models in making movie recommendations. Following this, we found that the collaborative filtering recommendation system produced the best recommendations and had the lowest Root-Mean-Squared-Error (RMSE).

1. **Introduction**

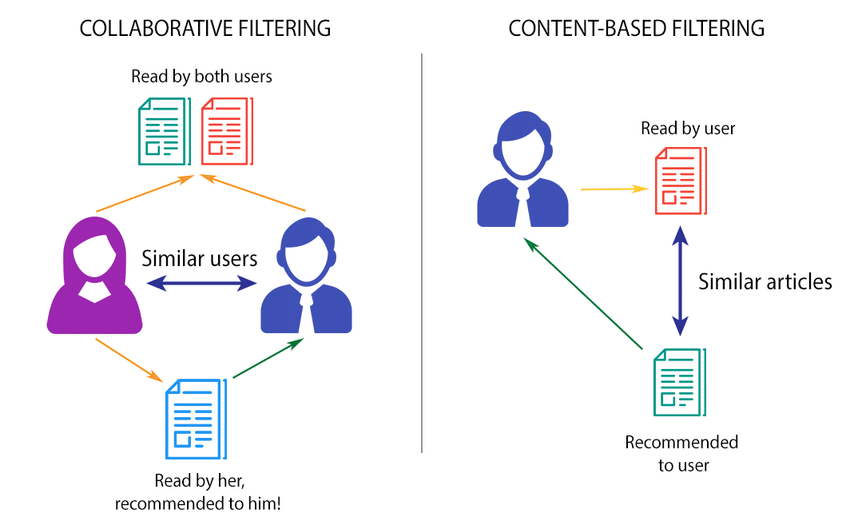
The main goal of recommendation systems is to generate personalized content based on user preferences, previous streaming history, and overall popularity. There are two main filtration strategies for movie recommendations – collaborative filtering and content-based filtering. The diagram below depicts the comparison between these two methods:

Figure 1: Comparison of collaborative filtering and content-based filtering

As indicated by the name, collaborative filtering takes into account the relations of one user’s preferences with other users’ behaviors. By comparing the interactions of several users with various movies, the algorithm is able to make personalized recommendations to specific users. Content-based filtering, on the other hand, focuses on past meta-data about the movies themselves, as well as information about a user’s past preferences. For example, if a user’s past streaming history mainly consisted of thriller and horror movies, then the recommendation system would likely suggest movies that fall into these genres. Content-based filtering tends to be easier to implement because the data required for it is more readily available than that required for collaborative filtering (i.e. explicit user ratings). However, collaborative filtering often produces better recommendations because of the use of detailed information on multiple users streaming histories.

1. **Data and Methods**
   1. ***Data Processing***

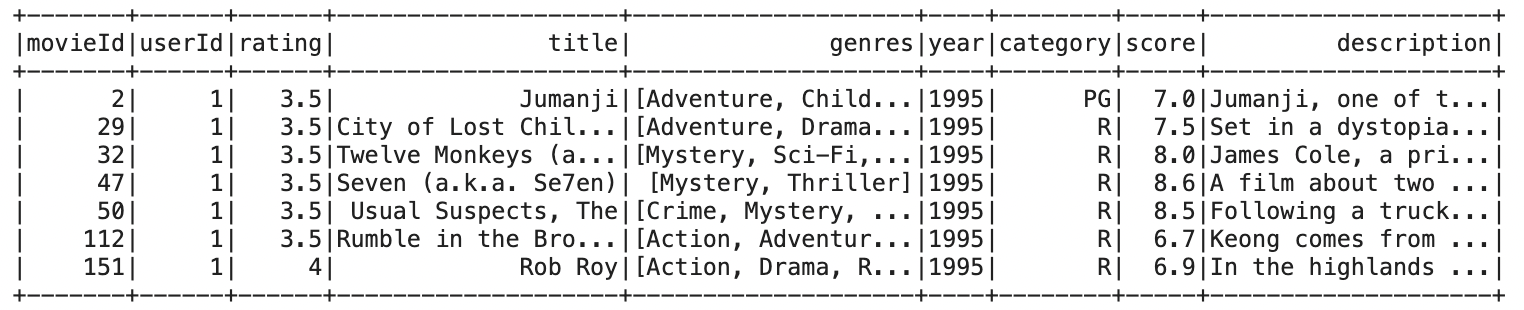
For this project, we used the MovieLens dataset, which details user streaming history. The dataset was provided by GroupLens Research in 2016. Four separate datasets containing features such as movie ID, user ID, user rating, title, genre(s), release year, and IMDB ID were merged to create a single dataset. In addition, we web-scraped the IMDB website using the IMDB ID to obtain movie categories (e.g. PG, R, PG-13), IMDB score, and movie description. This merged dataset contained over 20 million reviews, from which we dropped null values and then sampled 5% to divide into a train and test set. The train-test split was 70% and 30% respectively on the reduced dataset. Using these final data sets, we ran two types of collaborative filtering models and one content-based recommendation system. Below is a sample of the cleaned dataset. 

Figure 2: Dataset Sample

Before creating the recommendation systems, we performed basic exploratory data analysis to inspect the distributions of user ratings and IMDB scores. Both histograms were slightly skewed to the left as shown below. We also noticed the average rating for each movie genre was above three and the average IMDB score for each movie genre was above six.

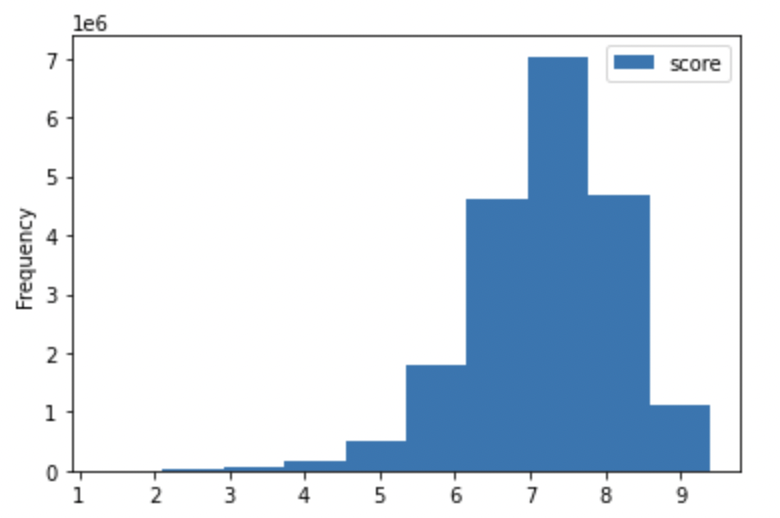
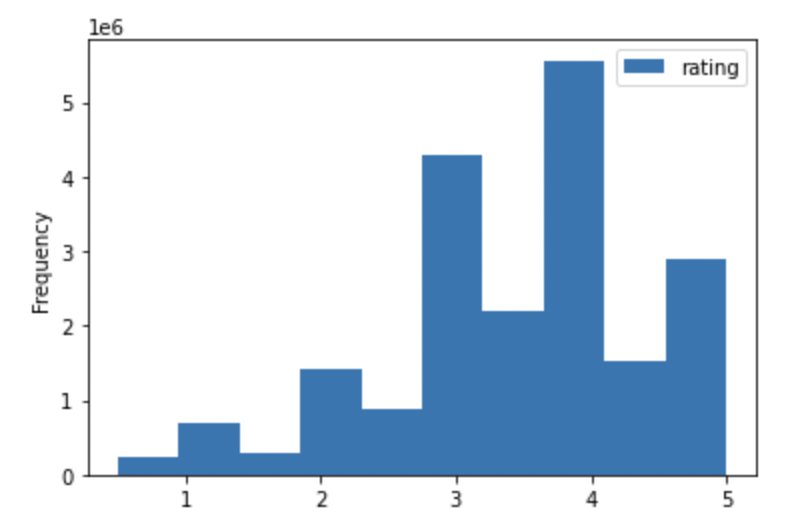


Figure 3: User Rating Distribution Figure 4: IMDB Score Distribution

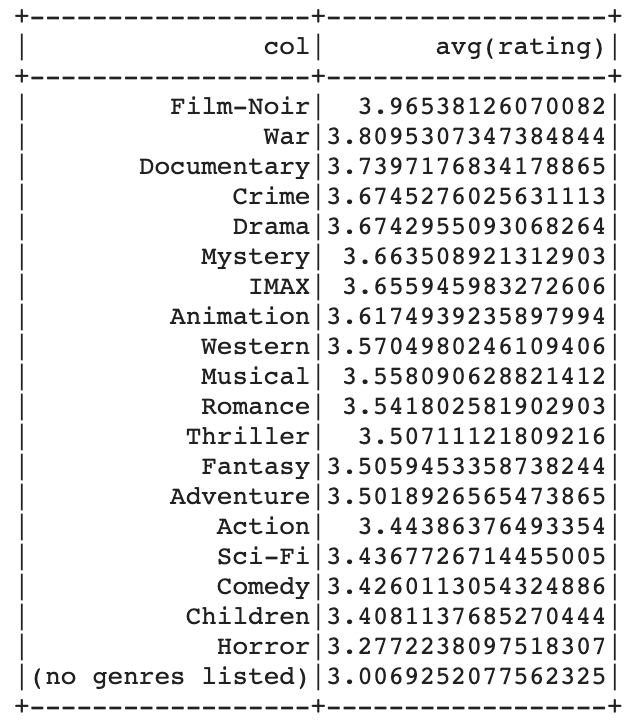
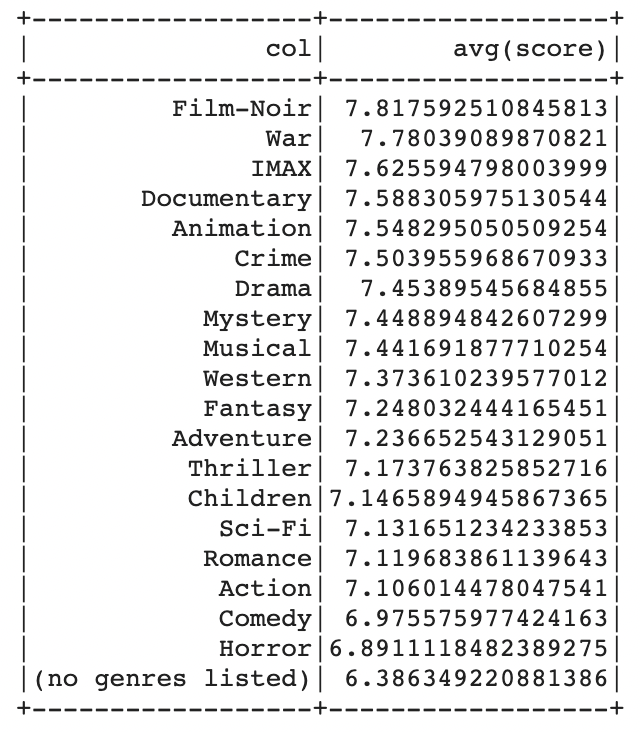


Figure 5: Average Ratings by Genre Figure 6: Average IMDB Score by Genre

* 1. ***Model 1: Collaborative Filtering with Alternating Least Squares Explicit***

The first recommender system was developed using collaborative filtering, which uses the interactions between user behavior and movie features to recommend movies and predict ratings. Because the data contains information on concrete user ratings, ranging from 0 to five, we used explicit Alternating Least Squares (ALS). This method generates predictions based solely on the user ID, movie ID, and user rating columns. Furthermore, this approach utilizes ALS for its form of matrix factorization to predict ratings for new user-movie interactions. To create the system, we used the PySpark ALS function to fit the model on the training data set, specifying implicit preferences as false. Before fitting the final model, we used the Param Grid Builder and Cross Validator methods to tune the model hyperparameters. We tested the following hyperparameter values: Rank = [10, 15, 20, 25], Max Iterations: [10, 15, 20, 25], Regression Parameter = [1, 0.1, 0.01]. We used three-fold cross validation to find the optimal values based on these inputs. Based on the results, the best model had a rank of 25, max iterations of 25, and a regression parameter 0.1. After fitting the final model using these hyperparameters, we created predictions on the test set using the transform function, and evaluated these predictions using the Regression Evaluator tool.

* 1. ***Model 2: Collaborative Filtering with Alternating Least Squares Implicit***

The second recommender system was also developed using collaborative filtering, however, in this instance, rating was taken out as a feature. Thus, only implicit data regarding user preferences was used to train the model. Creating the system followed a similar process to the explicit model. We used the ALS function to fit the model on the training set, specifying implicit preferences as true. We once again used Param Grid Builder and Cross Validator to tune the hyperparameters, using the same parameter grid as the explicit model. Lastly, we created predictions based on the test set and evaluated them based on RMSE.

* 1. ***Model 3: Content-Based Filtering with Penalized Linear Regression***

The final model we created was a content-based recommendation system which makes recommendations based on genre(s), release year, IMDB score, category, and movie description. In order to run content-based filtering, we needed to vectorize the text columns into numerical features. This required further data pre-processing for the non-numeric columns. Specifically, the text in the genre column was converted into an array of strings (where the strings were different genres) and then vectorized via term frequency-inverse document frequency (TF-IDF). This feature vectorization method creates a numeric value of importance for words in a corpus. It was implemented using hashingTF followed by IDF in PySpark. The category column was indexed and then vectorized via one-hot encoding. The description column required the most pre-processing due to the large amounts of text for each movie. After converting the text to lowercase, we removed the punctuation and stopwords. Finally, we vectorized the column using TF-IDF. Following this data processing, we used the Vector Assembler method to combine all five predictors into a single predictor column to ultimately predict user rating for each of the movies. To develop the model, we split the data into training and test sets based on unique movie IDs. This was to ensure that users in the test set did not include movies already watched by test users. We then used the Parameter Grid and Cross Validator to test the following penalized linear regression parameters (lambda): 0.1, 0.3, and 0.5, and the following elastic net parameters (alpha): 0, 0.2, 0.5, 0.8, 1. Using maximum iterations of 10, we found that the best model had a lambda of 0.3 and an alpha of 0.

1. **Results**
   1. ***Model 1: Collaborative Filtering with Alternating Least Squares Explicit***

The final model, using a rank of 25, max iterations of 25, and a regression parameter equal to 0.1, had an RMSE of 1.047 on predicted user-specific ratings. See below for a sample of these predictions.

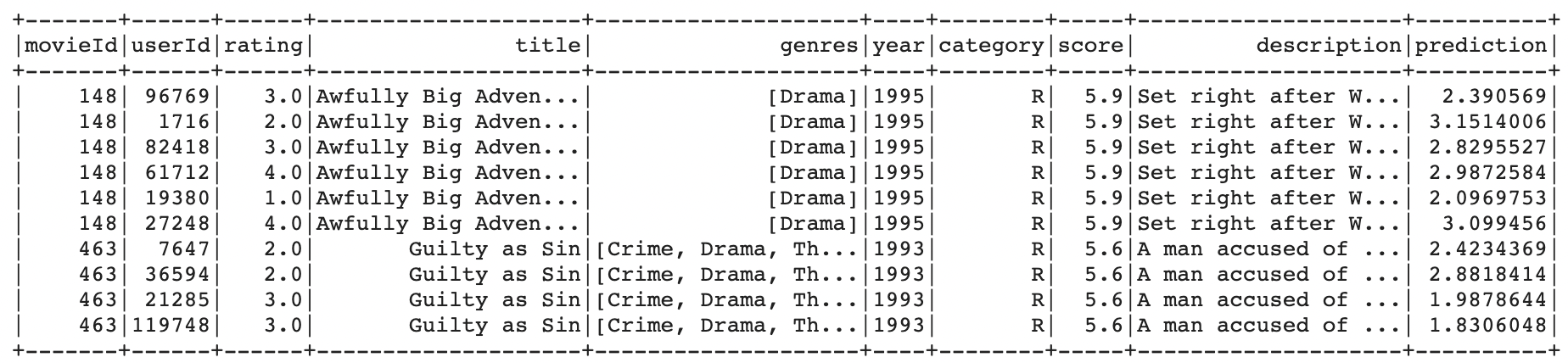


Figure 7: Sample of predicted user ratings (prediction) and actual user ratings (rating)

To further evaluate the model, we created movie predictions for a few users to categorically investigate how the recommendations compared to their previous viewing histories. For this analysis, we arbitrarily selected user #318. We first established what this user’s actual preferences were, based on the movies he/she rated the highest. This is shown below.

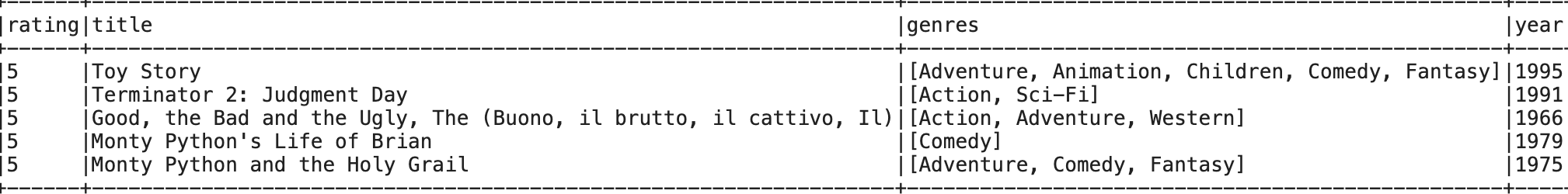


Figure 8: User 318’s actual preferences

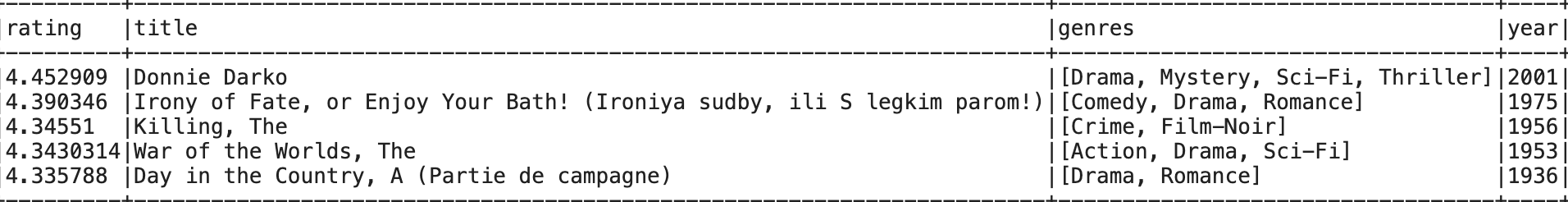
Using the recommendForAllUsers method, we then generated a series of recommended movies for this particular user. These predictions are shown below.

Figure 9: Explicit model output – top five recommended movies for user 318

* 1. ***Model 2: Collaborative Filtering with Alternating Least Squares Implicit***

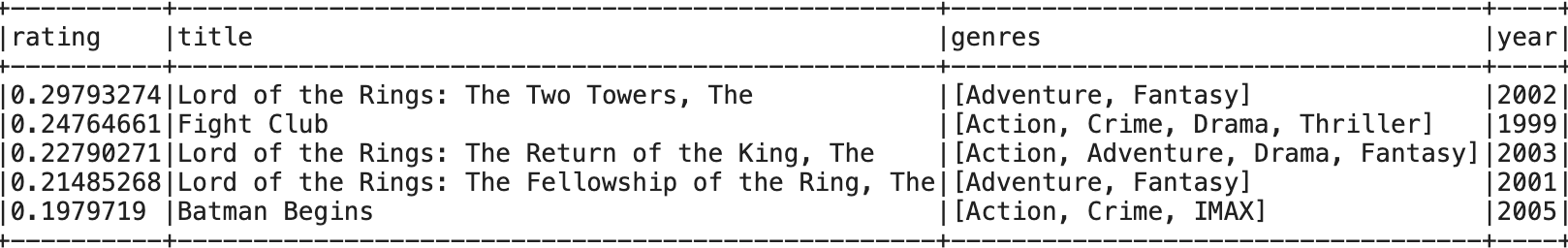
Hyperparameter tuning yielded the best parameters for optimal implicit model as follows: a rank of 10, max iterations of 25, a regression parameter of 0.01. This model yielded an RMSE value of 3.636 on predicted user-specific ratings. In order to further compare this approach with the other filtering methods, we analyzed the predictions that were outputted for the same user, user 318. The predicted movie preferences for user 318 are shown below, respectively.

Figure 10: Implicit model output – top five recommended movies for user 318

* 1. ***Model 3: Content-Based Filtering with Penalized Linear Regression***

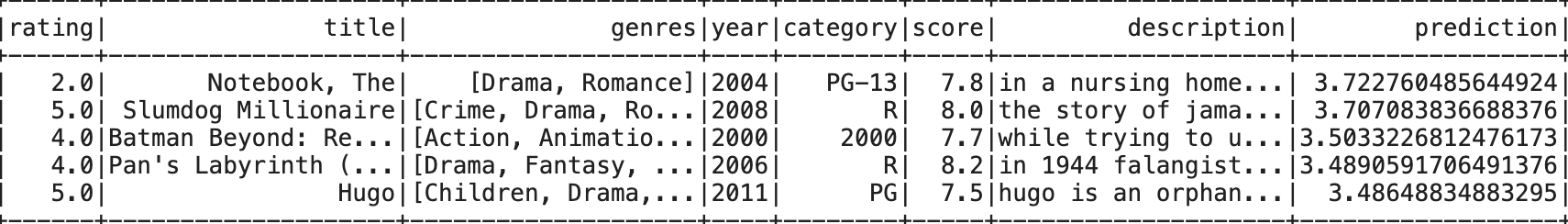
The best parameters for this specific model were found to be: a regression parameter of 0.3, and an elastic net parameter of 0 (indicating an L2 regularization). This optimal model yielded an rmse of 1.079 on predicted user-specific ratings. In order to further compare this approach with the other recommendation methods, we analyzed the predictions that were outputted for the same user, user 318. The predicted movie preferences for user 318 are shown below.

Figure 11: Content-based model output – top five recommended movies for user 318

1. **Conclusions**

We utilized both quantitative and qualitative measures in our model evaluation efforts. In our quantitative analysis, we chose to evaluate the models with root-mean-squared error (RMSE). After tuning and evaluating the various models, we concluded that the ALS explicit collaborative filtering model was the best overall recommendation system, with the lowest RMSE value of 1.047. Given the range of the ratings that were included in the data [0,5], this was a pretty good result. It was also evident that our collaborative model with implicit preferences yielded a much worse RMSE value than our collaborative model with explicit preferences. This result made sense, since the implicit model used less information (as compared to the explicit model, which incorporated the actual user ratings of movies).

As part of our qualitative analysis, we arbitrarily selected a specific user to see how our recommendations compared to that user’s preferences (the movies that he/she rated the highest). User #314, the user we selected, seemed to have a general preference for action, adventure, animation, and comedy genres. We found that our explicit model did well in recommending movies with action and comedy genres. Our implicit model recommended movies with drama, action, and thriller genres. Our content-based model yielded a set of genres that we felt most closely aligned with the user’s actual preferences. The recommendations captured genres such as action, adventure and animation, which was pretty similar to the user’s preferences.

Although the collaborative model with explicit preferences performed the best in terms of the RMSE value, we would also like to note that the content-based model yielded a similar RMSE value in addition to good movie predictions.

For future work, this model can be improved by normalizing the reviews column using z-scores to weight users that rate each movie differently as more influential than those users who tend to give fives to all movies. Some alternative model approaches can also be taken in the future, such as singular value decomposition instead of alternating least squares for collaborative filtering and cosine similarity instead of linear regression for content-based filtering. We would also hope to incorporate user information in our content-based model.

1. **Works Cited**
2. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=<http://dx.doi.org/10.1145/2827872>
3. “Build a Movie Recommendation System on Your Own.” *Analytics Vidhya*, 24 Nov. 2020, https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/.
4. *Building a Movie Recommendation Engine Using Spark ALS*. Binod Suman Academy, https://www.youtube.com/watch?v=EfHscLaMobY.
5. “Collaborative Filtering.” *Collaborative Filtering - Spark 2.2.0 Documentation*, https://spark.apache.org/docs/2.2.0/ml-collaborative-filtering.html.
6. “Content-Based Recommender Using Natural Language Processing (NLP).” *KDnuggets*, https://www.kdnuggets.com/2019/11/content-based-recommender-using-natural-language-processing-nlp.html.
7. “Movielens.” *GroupLens*, 8 Dec. 2021, https://grouplens.org/datasets/movielens/.
8. Nair, Snehal. “PySpark Collaborative Filtering with ALS.” *Medium*, Towards Data Science, 8 Sept. 2020, https://towardsdatascience.com/build-recommendation-system-with-pyspark-using-alternating-least-squares-als-matrix-factorisation-ebe1ad2e7679.
9. “Ratings, Reviews, and Where to Watch the Best Movies & TV Shows.” *IMDb*, IMDb.com, https://www.imdb.com/.