**Building a Recommender System**  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**IST 718 Big Data Analytics

Final Project Report

Team: Devan Grey, Daniel Piston, Keeley Ables

Project Overview

This project focused on the most accurate possible method for building a recommender system that made movie content suggestions for users of movie streaming services. The team collaborated to use past user ratings data in creating and training various models to address this objective. Over the course of this project, the team built a mean model, a weighted mean model often used by IMDb, a Singular Vector Decomposition (SVD) model, and a content-based model. From these models, based on the Root Mean Squared Error (RMSE) values used to evaluate them, the team recommended that the SVD model be used as a recommender system as it boasted the lowest RMSE, at 0.9. These results suggest that the SVD model would most accurately predict user’s movie preference.

**Specification**

Ina figurative professional environment, a movie recommender system utilizing user ratings data would likely lower churn rates due to improved user experience. To address this business problem, the team’s objective is to solve the problem of how to create the most accurate recommender system using past user ratings data.

In support of the above objective, this team anticipates the exploration of these questions throughout the analysis.

* What kinds of patterns regarding preferences will be seen among users?
* Will these patterns accurately predict user movie preference?

It was the above objective, along with the supporting questions for exploration, that comprised the problem that necessitated this project.

Out of the possible models implemented, the team hypothesized that the SVD model would return the most accurate movie recommendations. This seemed intuitive, as it is the only model that truly used any sort of differentiation specific to users.

**The Data**

The data for this project was taken from the Kaggle website below.

<https://www.kaggle.com/laowingkin/netflix-movie-recommendation/data?select=qualifying.txt>

This data set, in its original format, was composed of user ratings from 480,000 users selected at random from data obtained between 1998 and 2005. In its entirety, the rows in the data totaled 100,000,000, with each row representing one user movie rating.

Originally, the data was divided into four files which comprised the training data for implemented models. The features in these four data files consisted of Movie ID, Customer ID, the rating itself, and the date. User ratings ranged from one to five, with five communicating the most favorable score and one communicating the least. These four files were formatted as Python dictionaries, with the Movie ID serving as the index.

Provided along with the training data was a data set containing movie titles. This data set could be joined with other provided data sets on the feature, Movie ID, in order to reveal which movie titles were associated with various ratings. This data set contained the features Movie ID, Year of Release, and Title. The movie titles file was provided in a comma separated values (csv) file structured in a traditional tabular format.

The probe data set was supplied to enable the analysis to run the selected model after it had been trained. This data set was also formatted as a Python dictionary, which included the Movie ID and Customer ID. The Movie ID acted as the index.

The qualifying data set acted as the test set included to enable the analysis to test the model. This too included the Movie ID and Customer ID in a Python dictionary format. However, it did not include the customer ratings associated with the Customer ID to allow the model to predict the ratings. These ratings could then serve to recommend movies which would likely suit the user.

**Data Wrangling**

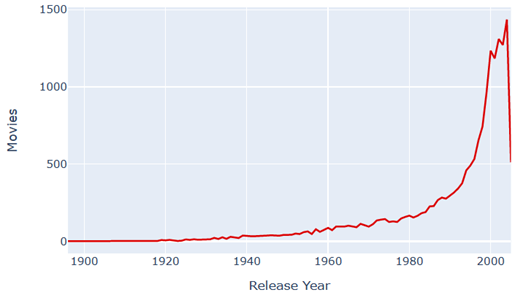
Unlike many professional scenarios, the user movie ratings data from the Kaggle source was quite complete. There was no need to discern strategies for filling NA values or removing incomplete rows from the data. Additionally, all the values in the data appeared to be in the same format. Furthermore, as mentioned above, several of the data files were formatted as Python dictionaries containing large sets of data with millions of rows. The data needed to be in an appropriate configuration for analysis, consequently, a focus on data wrangling rather than cleaning took precedence. To this end, upon import, the four training data sets were consolidated to create one extremely large data set. This data set was then exported to a pickle file in order to cache it more efficiently. Memory on the implementing machine was cleared, then the pickle file was imported once more. Finally, in order to obtain the tabular format needed for analysis, the Python dictionary was converted to a data frame, with the addition of the feature Movie ID.

**Observations About the Data**

The team determined appropriate exploratory analysis would include number of movies included in the data set by year of release, the distribution of all movie ratings across year, count of ratings over time, distribution of movie ratings frequency per movie count, and distribution of ratings frequency per user count.

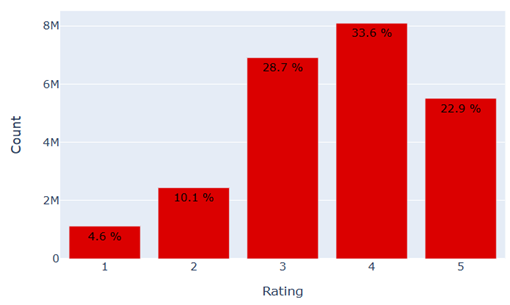
Below are visualizations representing this exploratory analysis, as well as insights gained.

**Count of Movies by Year of Release**



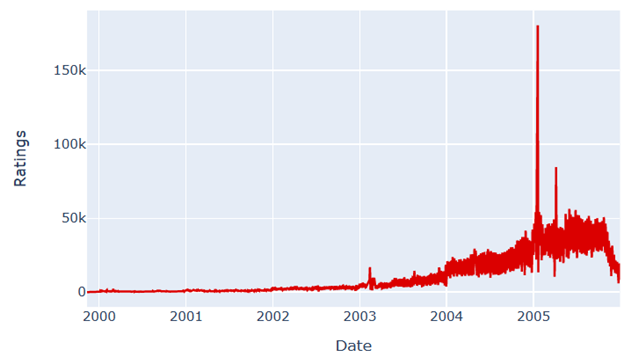
The years ranging from 2000 to approximately 2005 enjoyed the highest count of movies in our data. In contrast, the lowest count of movies in the data set centered around 1920. Interestingly, the highest count of user ratings across years also fell in a similar date range. This observation may suggest a pattern indicating users enjoy rating movies which have been released recently, as opposed to older movies.

**Distribution of All Movie Ratings Across Release Year**



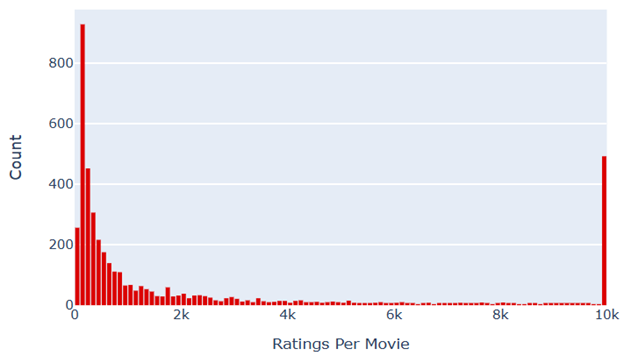
It seemed the bulk of movies in the data set were rated a 3, 4, or 5 with the highest frequency of ratings resting on four and the lowest frequency resting on one.

**Count of User Movie Ratings Over Time**



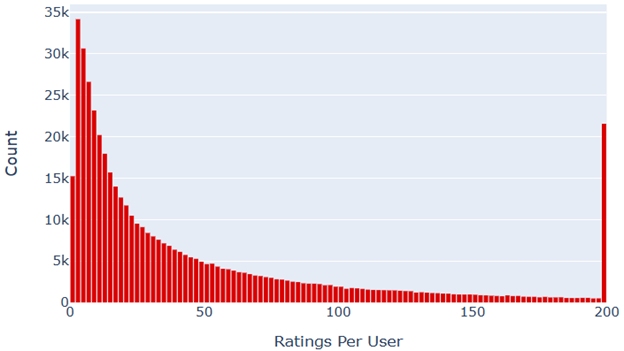
The above visualization portrayed the highest frequency of user movie ratings occurring in 2005. Observe the two acute increases in user movie ratings that year. It may also be notable that 2005 is the approximate year when Netflix began to gain popularity, which may explain why this time period enjoyed the highest count of user ratings.

**Distribution of Movie Ratings Frequency per Movie Count**



The above distribution communicated that few movies had high frequencies of user ratings. The exception of course being the observation in the graph associated with the 10k value. This is another anomaly in the data that may be worth investigating later, when analysis is driven by different objectives.

**Distribution of Ratings Frequency per User Count**



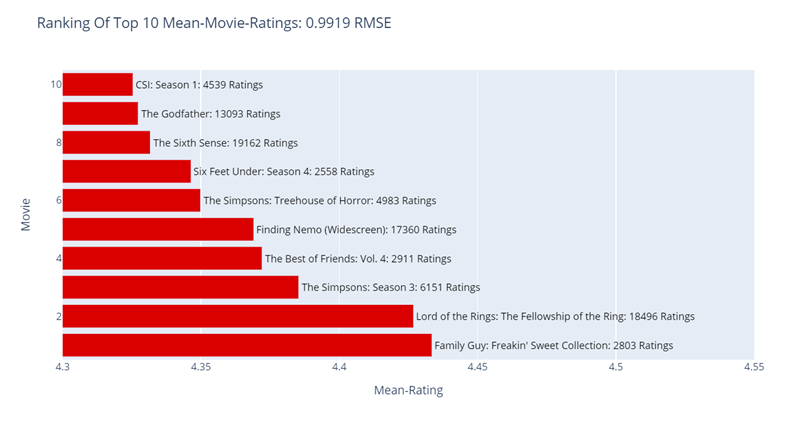
The shape of this distribution closely resembles the preceding. This graph also displays the same inconsistent observation aligning with the value 200, similar to the previous distribution.

After considering the above exploratory analysis, the team concluded that this data was focused on movie ratings, most of which were obtained from circa 2005. These ratings scored the enjoyment of movies made circa the same year. This means that newer releases were the movies enjoying the highest frequency of ratings. Additionally, the data shows that relatively few users frequently participated in the movie-rating-opportunity.

**Models for Analysis**

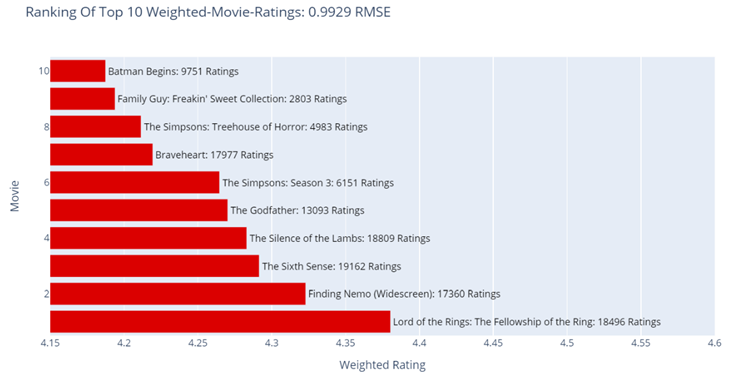
When considering which model(s) would address the objective for this project, the team examined various models which utilized an assortment of methods. For all the models implemented, the evaluation measure used was the Root Mean Squared Error (RMSE). The lower the RMSE score, the lower the error rate between the expected outcome and the actual outcome.

The most evident method for movie recommendation would be simply to determine the movies with the highest mean across users. In other words, these would be the most popular movies. Below is the return from the highest mean model implementation.



This analysis revealed the various genres of content suggested by the algorithm. There did not seem to be any obvious patterns in types of genres or specific actors that may explain the popularity of these movies. Notice suggestions such as “Lord of the Rings: The Fellowship of the Ring” and “Finding Nemo.” Both movies were released in the early 2000’s, and both were extremely popular in their time. Although this model returned a low RMSE of 0.9919, meaning this is an accurate model, it may be difficult to predict content for current users.

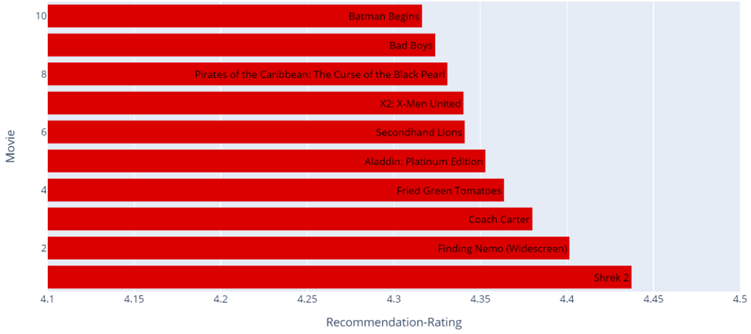
As with any analysis scenario, the team was compelled to utilize different models to improve accuracy. At this early stage in building, implementing, and evaluating models the team approached the objective with highest mean models, which would yield generally popular recommendations, before a more targeted approach. To improve the accuracy of the highest mean model, the team implemented the weighted mean model, used often by IMDb (a free content-streaming service). This algorithm assigns less weight to anomalous data that could skew the model’s return. The purpose of this practice is to maintain the quality and integrity of the model’s return. Below, the results returned by the weighted mean average model are visualized, along with its evaluation.



Interestingly, the RMSE did not significantly change. Many of the same movies are recommended in this model, although some hold different rankings. The RMSE evaluation measure of 0.9929 remained low, once again, proving the effectiveness of the model. However, this model was still a general recommender and did not differentiate for user preferences.

The team then ran a content-based model. This model learned user preferences, based on ratings, and suggested movies with similar content/genre, etc. It then grouped users it discerned to be “similar.” Its recommendations suggested movies enjoyed by users it deemed to be similar. Below are the results returned from the content-based model.

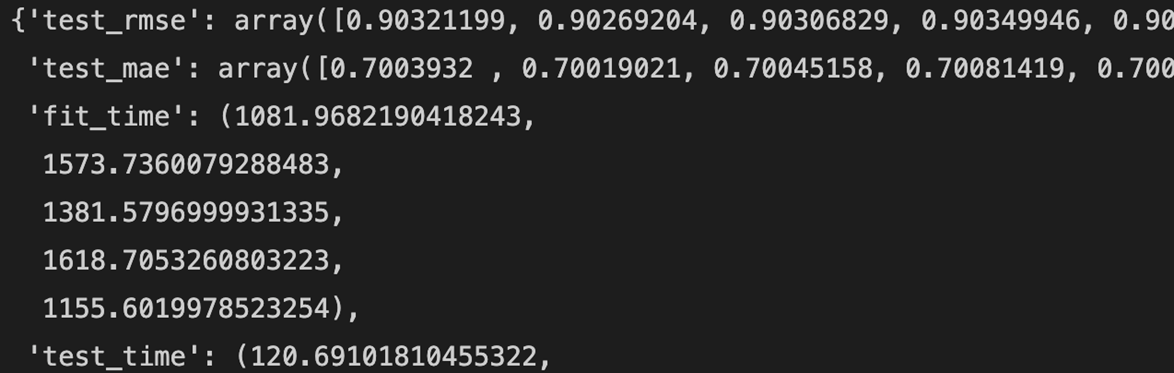
Ranking of Top 10 Recommended Movies for a User Based on Similarity



RMSE 1.3354

Like the two previous mean models, this content-based model boasted a low RMSE score. This is evidence attesting to its efficiency. However, it is notable that the RMSE is slightly higher than the highest mean model. This suggests less accuracy than the model previously run.

To build a model with an improved accuracy, that differentiated users according to preferences and recommended content accordingly, a Singular Vector Decomposition model was built. This collaborative model extracts patterns found in users based on their ratings, patterns in high-rated/low-rated movies, as well as the strength of their relationships, which are the basis for its recommendations to users. Included below is the return from this model.

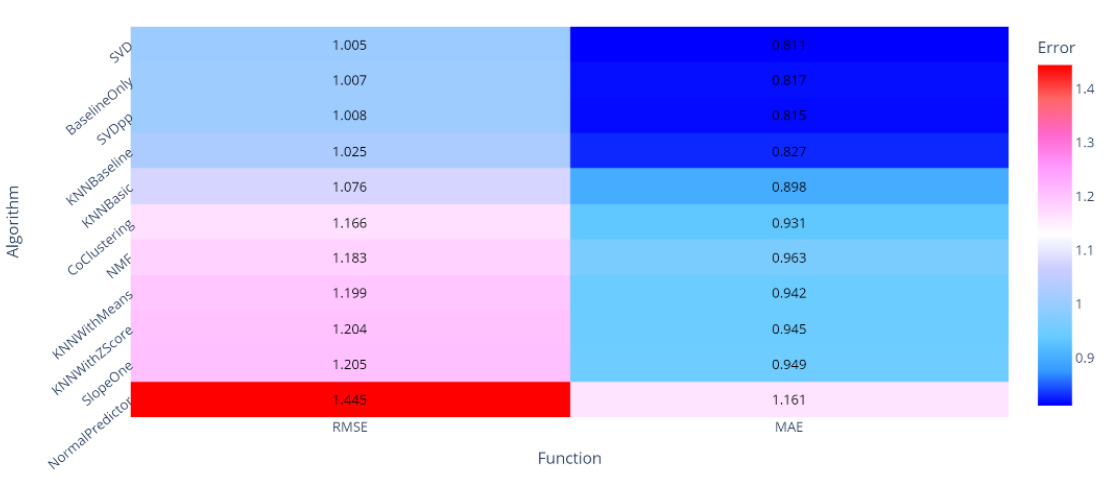


RMSE: 0.90+

The RMSE value returned by this model was the lowest score of the models implemented. Although all the models returned similar RMSE values, the SVD model did perform most efficiently.

To further compare the performance of the SVD model others, a variety of algorithms were run the data and their evaluation RMSE scores compared to each other. The following visualization is the result of this comparison.

Cross-Validated Comparison of Surprise Algorithms



Because these algorithms were run on different samples of the data than what was used to run the previous SVD model above, the RMSE value is different than previously seen. However, it is evident from this visualization that the SVD model does enjoy the lowest RMSE score. Therefore, it is the most accurate model.

**Recommendation**

The team’s recommendations regarding this project contains several facets. In the exploratory analysis of the data, we saw increased user engagement with movie content that was somewhat current at the time of rating. As a result, user rating data samples reflecting more modern users and choices would likely be beneficial to obtaining more accurate results for present customers. Furthermore, the exploratory analysis revealed that relatively few customers were participating in rating movies. As this could present bias when making predictions, we suggest initiatives to encourage more users to participate in rating movie content. Perhaps more advertisement of how ratings benefit users or creating a pop-up one-click rating option after each movie watched could create a convenient, simple opportunity for the user to provide this ratings data.

To address the initial objective of creating a recommender system, the team maintains that the company would need to assess the benefits and tradeoffs of each model according to its specific needs. To elaborate, the data contained 100,000,000 rows in total. Analysis of this volume of data requires strong computing power that surpasses the scope of many of the average computers. Running these models is computationally expensive. The business needs to assess whether implementing the SVD model boasting only slightly more accuracy, yet is expensive to run, will yield a justifiable return on investment. Furthermore, the business needs to determine the satisfaction of the user with simpler, less expensive algorithms. For example, a company such as IMDb providing free streaming services may find that a less expensive model such as the weighted average model is sufficient for their needs. However, this simpler model may not be appropriate for a company such as Netflix with paying customers who may have higher expectations for their streaming experience. Further, more current data collection and analysis may aid the business in making the determination of which recommendation model to implement in business practice.

**References**

<https://www.kaggle.com/laowingkin/netflix-movie-recommendation/data?select=qualifying.txt>

Vidiyala, R., (2020). How to Build a Movie Recommendation System. *Towards Data Science*. https://towardsdata science.com/ how-to-build-a-movie-recommendation-system-67e321339109