Netflix Prize: Factors of a User's Ratings

CSC 422 | Project Group 9 | Final Report

Link to GitHub Repository: https://github.ncsu.edu/dsbuchan/CSC422-G9-Project

Andrew Hoffman
North Carolina State University
Raleigh, NC
achoffm3@ncsu.edu

Daniel Buchanan
North Carolina State University
Raleigh, NC
dsbuchan@ncsu.edu

Zain Malik
North Carolina State University
Raleigh, NC
zmmalik@ncsu.edu

1. BACKGROUND & RELATED WORKS

With the advent of video streaming and subscription services, it has become imperative for companies to hone and refine their product in order to keep their customers engaged and happy with the content they provide, as customers dissatisfied with the service can easily terminate their subscription at any time. Even if a company boasts a large number of television shows and movies, it is overwhelming for a user to parse through all of it and decide what to spend their time watching.

In order to solve this predicament, recommendation systems have been developed to tailor a user's experience on the service and offer relevant items. Depending on the service in question, there are a multitude of parameters that could influence what gets recommended to the user; one of the main interactions between a user and what gets recommended to them are their reviews and ratings of a service's products [2]. For instance, if a user enjoys a movie, they will rate it highly (i.e. 5 stars) and, on the contrary, if a user hates a movie, they will rate it poorly (i.e. 1 star).

For this project, our group has selected the Netflix Prize dataset, containing over 100 million ratings by 480 thousand randomly-chosen, anonymous users over the course of 7 years, from October 1998 to December 2007 [5], [6]. This collection of ratings contains the user's ID number, the movie they rated, the actual rating (an integer between 1 and 5 stars), and the date that the user submitted their rating. In addition, we also have access to the catalog of 17,770 movies that these users rated, which has the English name of the movie and the release date of the movie. As noted within the README, the release date listed may refer to the theatrical or DVD release; the release date ranges from 1890 to 2005.

The Netflix Prize was a competition Netflix held in order to improve their recommendation algorithm. The above information is a part of the dataset that was provided to all teams by Netflix. However, our group has found supplementary material to aid in and enhance the experiment. [7] This dataset is akin to the file in the Netflix Prize file containing the list of movies, but includes the genre of the film according to what is listed in the IMDB database. However, not all movies within the Netflix Prize dataset were in the IMDB database, so only 12,279 of the total 17,770 movies have a list of genres associated with them. A movie can have more than one genre attached to it.

Due to the importance of recommendation systems and the profits involved, a lot of work and effort has been put into refining and perfecting it. One such approach is content-based recommendation, which focuses on a user's interests and compares that to the description of the item in order to decide if it should be recommended to said user [3]. Another approach is collaborative filtering: comparing one user to others with similar tastes and interests and recommending content from those users [4]. Of course, with both of these strategies, the crux comes from determining what a given user "likes". There are many factors in play here that all vary depending on what the subject is, such as views, ratings, viewing retention, and more.

Content-based recommendations and collaborative filtering are not mutually exclusive, so sometimes a hybrid approach is taken [1]. There are multiple instances of the hybrid method being used with varying amounts of weights between content-based and collaborative filtering.

2. METHOD

To predict the user rating for a movie, we will be using K-Nearest Neighbor (KNN) and K-Means Clustering (clustering). Both these techniques are similar in that they relate similar ratings to make a prediction. For the Netflix Prize dataset, this is exactly what we want because it is likely that ratings with similar attributes like time and genre will score similarly.

Both KNN and K-Means Clustering will have to at least beat the baseline method of the mean rating of the given movie. Before running any of these algorithms, we know that for the dataset, the mean rating is 3.6 and the median rating is 4.0. This would be the baseline prediction

made for a rating if the movie was not known. Given a movie, we can search the ratings for that movie and average it to make another prediction. The baseline for this experiment is to perform better than the average ratings given by each movie to predict user ratings. Additionally, we know that the distribution of user ratings is as follows.

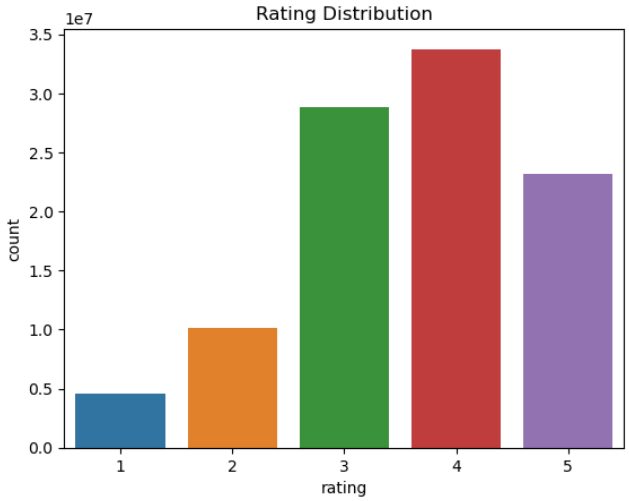

Figure 1: **The distribution of ratings amongst users**

For KNN, we will be using k-NN with Euclidean distance to determine the rating a given user will give a movie. This will take the average of the ratings of the k nearest neighbors to predict how the user is likely to rate the movie. With KNN, the next important aspect of the algorithm to determine is the number of neighbors to use to determine the rating. If we were to use one neighbor, then it would just be that neighbor's rating and if we used all ratings for the movie then it would just be the average rating of the movie. We will then use hyperparameter tuning to determine the best k for the KNN algorithm.

Another important aspect of the KNN algorithm is how we will determine the distance between ratings. Because some parameters are discrete (genre) while others are continuous (time) we will have to use hamming distance as the majority of our attributes are discrete.

The reason clustering will be used is to extrapolate grouping of users from the data. As we will already have an algorithm to determine the similarity between ratings from KNN, we can then use that to discover any groups of similarity that may exist in the data. This could lead to some interesting discoveries like movie trends that may help better classify or understand the data. To determine the number of clusters that we will use for K-Means clustering we will use the Elbow Method. For this method we will take the average within-cluster squared error for each number of clusters. We will then graph the result and determine the elbow point on the graph.

Compared to other techniques like Decision Trees and Deep Learning algorithms, KNN and clustering better fit the dataset as they look directly at similar ratings. The downside of Decision Trees for this dataset is that there are so many different movies, genres, and times that it would be difficult to find splits for these traits. The reason that deep learning was not selected, despite its ability to create an accurate model for the dataset, is that it is more costly to train and would not be able to be trained on most of the dataset due to its cost. In our model, we want to train on as much data as possible so artificial neural networks are not the best fit for what we want to achieve with our model.

The novel aspect of our method will come from the supplementary datasets that we researched or created ourselves. In addition to the Netflix Prize dataset, we will also be using another dataset. The additional dataset will contain the genres of movies as gleaned from their respective IMDB page, if applicable. There are 20 different genres in the dataset and a movie can have any number of genres.

The reason this data is relevant to better predict the Netflix prize dataset is for the following reasons: 1. Including genre will add additional information to the movies that the user is rating. Such information will allow us to find genres that the user likes and genres that the user dislikes if there are any. 2. Because the times of the ratings are given in the Netflix Prize dataset, having the time since the release may be a predictor of its relevance. 3. The overall addition of data will allow for more accurate clustering of data, and closer neighbors in the dataset

3. PLAN & EXPERIMENT

We have multiple datasets that will be combined and altered to generate the data set that will be used in the experiment.

The data set “combined_data_1-4.txt” contains data in the format CustomerId,Rating,Date per user rating. All user ratings are listed below and grouped by the MovieID. The CustomerId ranges from 1 to 2649429, with 480189 users. The MovieIDs range from 1 to 17770 sequentially. The ratings range from 1 to 5 in the form of an integer. The dates are of format YYYY-MM-DD. The “movie_titles.csv” contains data in the format MovieId, YearOfRelease,Title. The MovieIDs range from 1 to 17770, the YearOfRelease ranges from 1890 to 2005, and the Title is just the title of the movie. The other file provided in the Netflix Prize dataset is “qualifying.txt”, but is not useful to us as it does not contain the numerical rating. Instead, this data set only consists of the MovieID followed by a colon,

then a grouping of CustomerIDs and dates. The MovieIDs, CustomerIDs, and dates are in the same format as previously mentioned. The "probe.txt" file has MovieIDs followed by a grouping of CustomerIDs.

The "neflix_genres.csv" contains a movieid and genres. The movieid corresponds with the MovieIDs in the "movie_titles.csv" file, so it is an integer ranging from 1 to 17770. The genres is an attribute that labels the movie as a "Documentary", "Animation", and so on as a list separated by a "|" character.

From the "movie_titles.csv" and "combined_data_1-4.txt" we will also be adding the attribute of 'release' to the dataset used for our experiment. This attribute will be the release year of the movie that is being rated in a given row of ratings. We will also be improving the 'date' attribute of the rating by exchanging it for the difference between the date of the rating and the release date of the movie.

Another attribute we will consider is the difference in time between the release of a movie and the date of the review. There could be bias in the ratings depending on how many years are between the movie release and review. For instance, older (in respect to when the date of the review was) movies could be subject to more scrutiny than newer movies that could have hype or "shiny object syndrome". Alternatively, older movies could be looked upon more favorably due to nostalgia and time passing. To assert that this is true, we will compare our experiment with random ratings to assert that the information was useful.

We predict that the information from a movie's genre, release date, and years since the release of the movie will be enough information to make an accurate prediction of a given rating. This information should give an accurate prediction as it should consider movies with similar genres tended to be rated similarly. Also, the ratings of movies that are old or new, will likely follow a trend of deteriorating, increasing, or stagnant ratings as it ages which should be captured by analyzing similar ratings.

The first part of our experiment will be to combine the datasets into one table of data with the users. To do this, genres will need to be matched with the ID of the movie in the original dataset. All ratings with movie IDs that do not exist in the genre dataset should be dropped, which does remove some data but still maintains plenty of ratings to create our models. The time difference (taking the difference between the year of rating and release year) will also have its own column called "deltaYears".

Once the column for genre and deltaYears have been added, we will normalize the data to ensure that each attribute has equal influence on the models. Then, we will apply K Nearest Neighbor. For this, we will use hyperparameter tuning on the training dataset to determine the number of neighbors to consider. The values of k that we will be tuning to are 10, 100, 1000. Additionally, we will be recording the error for each k and comparing them based of the lowest testing error. The way that we will determine the distance between attributes since all but the date are continuous will be through hamming distance. For this hamming distance, we will be looking at similarities in dates, genre, and event for the rating.

For clustering, we will either use the K-Means. We will split data into training and testing first. Then, we will run K-Means with different values of k on the training data. We will determine our final number of clusters by graphing the mean squared error for each number of clusters. We will then analyze the graph to determine the elbow point, then using the number of clusters at that point for our final model from K-Means.

Lastly, we will need to compare our models from K-Means and KNN to that of the baseline. The baseline will be the average rating of the movies to predict the rating. If our models can outperform this simple prediction on the test dataset provided then we will consider our experiment a success. We will also be graphing some of the K-Means clusters to see if we can discover any trends or commonalities that may exist in the dataset.

4. RESULTS

4.1 Results

We presented a novel approach to the Netflix Prize challenge through the use of new datasets. The one drawback of including these datasets is that they did not contain all the movies that were used in the original Netflix prize dataset. For this reason, of the 17,770 movies in the original dataset, we could only use 12,279 of them as the rest did not have an associated genre in the genre dataset. Below is the format of our dataset:

	movie	user	rating	date	release	deltaYears	genreDrama	\
0	1	1488844	3	2005-09-06	2004.0	1.0	1	
1	1	822109	5	2005-05-13	2004.0	1.0	1	
2	1	885013	4	2005-10-19	2004.0	1.0	1	
3	1	30878	4	2005-12-26	2004.0	1.0	1	
4	1	823519	3	2004-05-03	2004.0	0.0	1	

	genreCrime	genreMystery	genreSciFi	...	genreFilmNoir	genreComedy
0	1	1	0	...	0	0
1	1	1	0	...	0	0
2	1	1	0	...	0	0
3	1	1	0	...	0	0
4	1	1	0	...	0	0

	genreSport	genreDocumentary	genreFamily	genreFantasy	genreHistory
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

	genreHorror	genreMusical	genreWar
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0

Figure 2: **Format of the dataset used for the experiment.**

As seen, we have added a few different attributes to better predict the rating of a movie. From a data set with movie release dates, we added the release year of the movie as an attribute. We also added the difference between the release year of the movie and the date that the movie was rated in the attribute deltaYears. This attribute serves to determine the change in rating due to the aging of a movie. The reason we are including deltaYears along with the date as they contain similar information, is that we think knowing the age of a movie will be more indicative of a user's rating than purely the date when the rating was given. Hence in our analysis, we will replace the 'date' column with the 'deltaYears' and 'release' date.

The other attributes that we added are the genres for the movie. We made columns for each genre and used a 0 to indicate that the genre does not apply to the movie and a 1 to indicate that the genre does apply to the movie. These attributes allow us to gain more information about the rating, and allow similarities to be drawn between different movies.

Using KNN, we were able to predict a rating just given the attributes of the genre of the movie, what the movie is, and the time the rating occurred after its release. We optimized the parameter, k, to determine how many neighbors would give the best score on the test dataset. We would then choose the k that had the best accuracy score on the test data set.

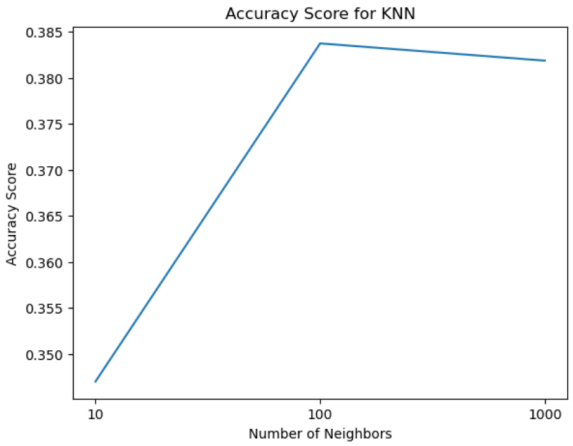

Figure 3: **Comparing Accuracy Score for different Ks used in KNN**

As seen in the figure, there was a major difference in accuracy between 10 neighbors and 100 neighbors. Between 100 and 1000 neighbors, however, there was not a significant difference. Notably, there is a decrease in the accuracy score between 100 and 1000 neighbors indicating that 100 neighbors is the better parameter. The best parameter, because it has the best accuracy score, is 100 neighbors to predict a rating. This represents a marginal improvement in accuracy over a random guess of ratings, which would have had an accuracy of around 20% as guessing a rating randomly from 1 to 5 has a 1 out of 5 chance of getting it correct. So it does represent some improvement, but still only has an accuracy of around 38%. To further analyze the model we calculated the root mean squared error (RMSE) for each of the models. The information that RMSE gives us that the accuracy score does not is how close the prediction was to the actual value. Shown below is the RMSE calculated for each model.

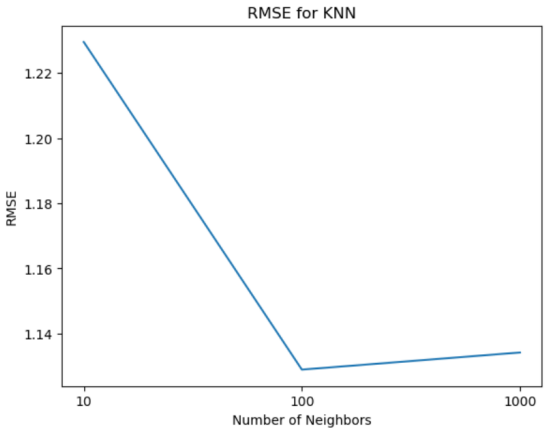

Figure 4: **Comparing Root Mean Squared Error for different Ks used in**

In the above figure, it also shows a significant improvement between using 10 neighbors and using 100 neighbors, but not much of a difference between 100 neighbors and 1000 neighbors. What RMSE shows that is not demonstrated by the accuracy score is how close the model was to predicting the correct value. This is accomplished by RMSE by taking the difference between the predicted value and the actual value, taking the mean of these differences, then taking the square root of that difference. What this tells us, similar to the accuracy score, is that the model using 100 neighbors is the best model as it predicted closest to the actual values.

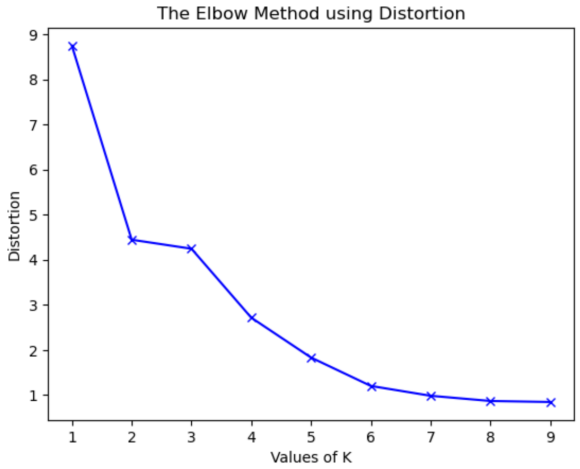

Figure 5: **Graph showing the elbow method implementation to find number of clusters to use**

The above figure has the mean squared error for the different number of clusters we used to analyze this dataset. We chose 6 because it is the point where a significantly steep slope meets a far more gradual slope. In the figure below is a few of the clusters we found in the dataset.

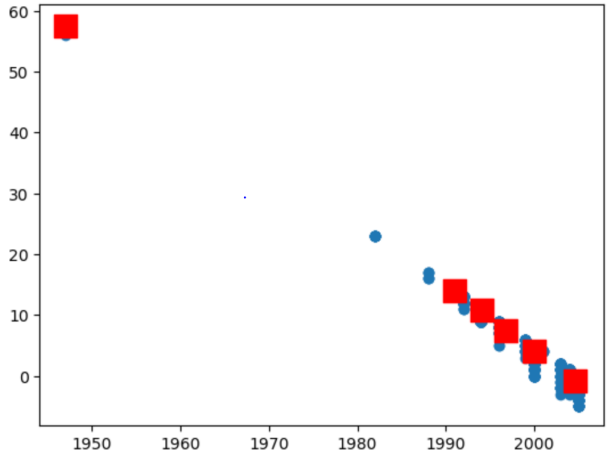

Figure 6: **Graph showing a group of clusters within a certain range**

The above figure shows clusters of users that have similar attributes. There were 6 groups of users that identified similarly for the most part with the 6 clusters. Clustering is based on grouping the data according to similarities.

4.2 Discussion

Despite KNN being able to give us decent results, the algorithm ran exceptionally slow on our data. The reason for this is that our data size was exceptionally large, making KNN not an ideal model to predict ratings at scale. One reason for this exceptional run time is that we are looking at each rating to find which ratings are the most similar to each other. A possible solution that could make the runtime of KNN faster with the downside of discarding much of the meta-information about the movie by representing a user as a row rather than their rating for each movie as a column.

Also significant is that we had to run a random sample to be able to run KNN, which meant that we did not use all the data available to us. This is also one of the reasons that the model created is not ideal, as the dataset is so large that it is not realistic to get the most accurate prediction without the cost of significantly extended runtime.

The algorithm for K-means ran quickly on the sample training data that we extracted and gave us very useful information. The information provided by k-means is useful because individuals that fall into demographics that like one of the movies in a specific cluster may also like other movies that fall into the same cluster. We could potentially use the factor of past movie rating a user gave for a movie that falls into one of the specific clusters and use that to give a similar prediction for the other new movie that is in the same cluster. A limitation is potentially the fact

that we did not use the entire data set and only used bits of the data.

Although our intentions and scope was never for practical implementation into a proper recommender system, one improvement for run time would be feature selection. Especially since each of the 20 genres had to have its own column for our implementation to work with our models, the dimensionality is sky high. In a similar vein, we could have attempted to isolate different attributes and determine which had the greatest effect on accuracy. Unfortunately, we ran out of time before we were able to give this the appropriate amount of attention to bring it into fruition.

5. CONCLUSIONS

One of the main lessons we learned from this project is working with such a large dataset. In all of our experience, we have never worked with data on the scale of hundred million entries, so the near instant or just a few seconds of runtime to complete and generate a model that we were more accustomed to with sklearn's library of toy datasets or other datasets used in homework assignments was a massive change to adapt to. When it came to data preparation, we had initially thought to generate the dataframe from reading in the original files every time, but merely preparing the data took too long for it to be sustainable for us to continue and be able to debug and do experiments.

While the sheer magnitude of the data was a factor we had to accept, we figured out that exporting the prepared data into a csv file would retain our progress and allow us to use the read_csv function to recreate the dataframe. This was a lot quicker than having to edit a new dataframe to our whims from the original datasets every time the kernel reset and we needed to run the models again.

Another consideration we came across was the appropriateness of a model for the data within a dataset we have available. For instance, the movieID and userID are nominal attributes that had questionable utility for a model like KNN. A nominal attribute like userID is useful in the context for predicting ratings for a user as it would give weight to that user's other ratings. However, there's no immediate connection or correlation between, say, userID 100 and userID 101, which is something that would not be accounted for in our model.

In a similar vein, were we to redo this project, it could have been more beneficial to split up via users with their ratings as opposed to split up via the ratings. As of now, the dataset consists of individual ratings of a particular movie made by a particular user. For our purposes of curating a user's experience by predicting how they would rate a movie, it perhaps would be better to alter the data formatting to have it based on userID and have a list of their ratings for movies. While the implementation of KNN and K-Means would have to be more complex than our current implementation, it would likely result in a more user-focused approach.

Recommendation services and predictive models are as imperative as ever in this day and age of information and data. Users can easily be overwhelmed by a mass of products, and they may be turned away or unsatisfied if those coming their way are not to their liking. There will always be work towards improving and optimizing ways to recommend products and services and predict ratings in order to keep consumer satisfaction and companies' profitability high.

REFERENCES

[1] Ido Guy et al., "Social Media Recommendation Based on People and Tags", SIGIR 2010 Proceedings - 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. N.p., 2010. DOI: https://doi.org/10.1145/1835449.1835484

[2] Satya Prakash Sahu, Anand Nautiyal and Mahendra Prasad, "Machine Learning Algorithms for Recommender System - a Comparative Analysis", International Journal of Computer Applications Technology and Research, vol. 6.2, 2017. https://www.ijcat.com/archives/volume6/issue2/ijcatr06021005.pdf

[3] Pazzani, M.J., & Billsus D. 2007. Content-based recommendation systems. The Adaptive Web, 325-341. DOI: https://doi.org/10.1007/978-3-540-72079-9_10

[4] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. 1992. Using Collaborative Filtering to Weave an Information Tapestry. Commun. ACM 35, 12 (Dec. 1992), 61-70. DOI: https://doi.org/10.1145/138859.138867

[5] James Bennett and Stan Lanning. 2007. *The Netflix Prize*. https://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings/The-Netflix-Prize-Bennett.pdf

[6] https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data?select=README

[7] https://github.com/tommasocarraro/netflix-prize-with-genres/blob/master/netflix_genres.csv

6. MEETING SCHEDULE

Meeting Time	Duration	Attendance
4/11 8pm	1 hour	Andrew, Daniel, Zain
4/18 8pm	1 hour	Andrew, Daniel, Zain
4/23 3pm	1 hour	Andrew, Daniel, Zain
4/24 8pm	2 hour	Andrew, Daniel, Zain

Netflix Prize: Factors of a User’s Rating