FINAL REPORT FOR ALDA PROJECT

GROUP P24

Brett Reier
Department of Computer Science
North Carolina State University
Raleigh, NC 27606
bpreier@ncsu.edu

Lohith Sowmiyan Panjalingapuram Senthilkumar
Department of Computer Science
North Carolina State University
Raleigh , NC 27606
lpanjal@ncsu.edu

Ishika Gandhi
Department of Computer Science
North Carolina State University
Raleigh , NC 27606
igandhi2@ncsu.edu

Maya Pavan
Department of Computer Science
North Carolina State University
Raleigh , NC 27606
mpavan2@ncsu.edu

Keywords Netflix Prize Data · Ensemble Method

1 Background

Machine Learning applications are increasingly becoming more and more widespread in daily environments. For instance, such applications are incorporated in "chat boxes, language translations apps and social media platforms" 1 . Today, we can see companies transitioning and relying more and more on machine learning applications. One example of a company deeply drenched in utilizing machine learning is Netflix. As a streaming service with a wide range of accessible media, Netflix understands the importance of the architecture of its recommendation system. The company understands that "as a monthly subscription service, member satisfaction is tightly coupled to a person's likelihood to retain with our service, which directly impacts our revenue" 2. Further they state that their main goal is "main task of our recommender system at Netflix is to help our members discover content that they will watch and enjoy to maximize their long-term satisfaction 2. In fact, Netflix has invested millions of dollars into its recommendation system because of its importance in user retention. Knowing that the recommendation system is very important in the ultimate revenue of Netflix we propose a project to develop a system. This project will employ to solve the problem of inaccuracies of recommending movies and shows to customers which leads to lower retention on the Netflix platform?

2 Introduction

In this project, we plan to predict the rating a user will give on a movie from the user's past ratings. The project is largely inspired by the Netflix Grand Prize Solution. The Netflix contest held in 2008 was a contest held by Netflix for data scientists to come up with the most effective collaborative filtering algorithm that can predict user ratings to movies. Netflix released a public dataset so that the public can use it for the algorithms, called the Netflix Prize Data. It contains the ratings for approximately 1700 movies by 480189 different users on a scale from 1 to 5, collected between 1998 and 2005. To determine the effectiveness of the collaborative filtering model, root mean squared error (RMSE) was used. As a baseline, using the movie average to predict ratings has an RMSE of 1.01. The Netflix solution at the time had an RMSE of 0.95, and the winning solution had an RMSE of 0.85. The winning solution used an ensemble approach for their algorithm, which combines multiple independent models to create a larger, more effective model, because different model architectures excel at different tasks.

The solution also in addition to ratings, used additional techniques to find movie similarity, such as comparing the users who are rating them, and comparing the title similarity. This ensemble solution combined many different kinds of models, including but not limited to k-NN, factorization, and RBMs. k-NN is k nearest neighbor, and it involves looking

at the most similar points in order to make a classification. Factorization is also called Singular Value Decomposition and is arguably the most effective single procedure for collaborative filtering. It involves using techniques such as Principal Component Analysis (PCA) and alternating least squares. RBMs are Restricted Boltzmann Machines and are based on neural networks. Some other models include Restricted Boltzmann Machines with Gaussian visible units, combinations of models, and making an enhanced training set by inputting qualifying predictions. These models can then all be combined to make one giant ensemble solution that was the most accurate way to predict the ratings for the Netflix contest.

For our project, we intend to create our own solution to the Netflix recommendation challenge. We will compare the effectiveness of our solution to the Netflix solution at the time and to the winning solution. We also plan to get additional insights from the data, such as which movies have the strongest associations, as well as clustering similar movies and users together into definable groups.

3 Method

The methodology of our proposed approach includes preprocessing and cleaning of the two datasets. This included removing repeating or null values. We did not think it was necessary to conduct pca for this project. This is because the netflix data that was used already had a limited amount of features. After preprocessing, the dataset was merged for ease of use. Our predictor value is the rating for the score. Gaining inspiration from Netflix's Grand Prize award submission paper, our group decided to conduct collaborative filtering utilizing the ensemble method. This method involves combining multiple algorithms to produce a more effective model due to the nature of different tasks excelling at different architectures with this particular type of data. The three algorithms that we pursued included Cosine Similarity, Restricted Boltzmann Machine and ANOVA techniques. Using these three algorithms we created five different classifiers for our ensemble. We had one classifier using a Restricted Boltzmann Machine, one classifier using ANOVA techniques, one classifier using each movie's average score as a predictor value, and two classifiers using cosine similarity. One of these cosine similarity classifiers used the mode of similar users to predict ratings and one used a weighted average of similar users.

Firstly, Cosine similarity can provide content-based recommendations based on item similarities. We can use cosine similarity to find what users are most similar to each other and predict ratings using those similar users scores. Cosine Similarity stands out in this context for several reasons. Effective comparison is made possible by its emphasis on direction rather than magnitude, particularly in high-dimensional environments such as the vast user-movie rating matrix. Moreover, cosine similarity disregards the scale or magnitude of ratings, emphasizing the alignment of user preferences, making it robust in scenarios where users might rate movies differently but still share similar tastes. We have implemented the similarity between users based on their rating patterns for movies using cosine similarity which helps identify users with similar tastes. One, we have defined a function which computes a weighted average of these ratings, where the weights are the similarities between the target user and similar users. Finally, we demonstrated by predicting a rating for a specific user and movie based on this collaborative filtering approach. Second try of using Cosine Similarity starts by zero-centering user rating vectors and employing NearestNeighbors with cosine distance to find similar users. For each user in the test set, it identifies similar neighbors from the training set based on their ratings for movies. It then predicts ratings by considering ratings given by these similar users to the specific movie.

Secondly, RBMs can capture complex interactions and user behavior patterns. The Restricted Boltzmann Machine is a neural network that consists of a layer of observed and a layer of hidden random variables, with a set of connections between them. The difference between an RBM and a regular neural network is that in an RBM, the hidden layer nodes are not connected to each other, and the visible layer nodes are not connected to each other. This makes it more simple and more efficient compared to a typical neural network. It is a generative framework that models a distribution over visible variables. SVD++ blended with Restricted Boltzmann Machines can achieve a ten percent increase in accuracy over Netflix's existing algorithm. The RBM initializes weights and biases and defines methods for sampling hidden and visible nodes, as well as training the model using contrastive divergence. The training loop runs for multiple epochs, iterates through user observations in batches, performs Gibbs sampling for each user to approximate hidden nodes, and calculates the training loss. Post-training, it then looks at the probability each hidden node is turned on, which makes it good for predicting binary data.

Lastly, ANOVA techniques can help select relevant item features for content-based filtering. ANOVA is a method in which is used to determine differences between research results against groups and determine if there is a relationship between them. The reasoning why ANOVA was used for our application was firstly, other papers cited the classifier in their own assemble method when using the Netflix Data. It utilizes K-Means clustering to group movie ratings based on user behavior. It conducts ANOVA tests on these clusters to find the cluster with the most distinct rating patterns. Then,

it assigns mean ratings from significant clusters to predict ratings for similar clusters in the test data. Overall, it's a clustering-based collaborative filtering approach for movie ratings prediction using K-Means and ANOVA.

These diverse models are complemented by our exploration of ensemble methods, which aim to further enhance our recommendation system's performance. By combining the various techniques, including Cosine Similarity, RBMs, and ANOVA, we aimed to create a robust and accurate collaborative filtering algorithm that excels in capturing latent factors, complex user behavior patterns, and providing relevant content-based recommendations. The incorporation of SVD++ with RBMs holds the potential to achieve a remarkable 10 percent increase in accuracy compared to Netflix's existing algorithm. Furthermore, ANOVA techniques are instrumental in selecting item features that are most relevant for content-based filtering, making our recommendation system more effective and user-oriented. This ensemble approach pushes the boundaries of recommendation systems, ultimately benefiting the community with a personalized and efficient user experience.

4 Experiment Setup

Datasets: To conduct this project we used the python language and wrote code in VS studio as well as Google Collab. We were able to create a folder in google drive which consisted of all of our data and python files.The dataset used for this project is the Netflix Price Data. This dataset was downloaded from UCI Repository and is the same dataset in which BellKor's paper was written. This data set includes the training dataset file, movie titles file and the probe dataset. The training set includes 1000 text files each representing a particular movie containing the features movie_id, customer_id, and ratings.We aggregated these files into a single dataset and included the movie_id as the new feature. The movie id's range from 1 to 1700- sequentially. There are 480189 users and the ratings are scaled from 1 to 5. The rows of the dataset are 5010199. The dates are in the format of year, month and day. The movies file consists of the movie_id, year of release and title. Movies that were used in the data set were released from the years between 1998 and 2005.

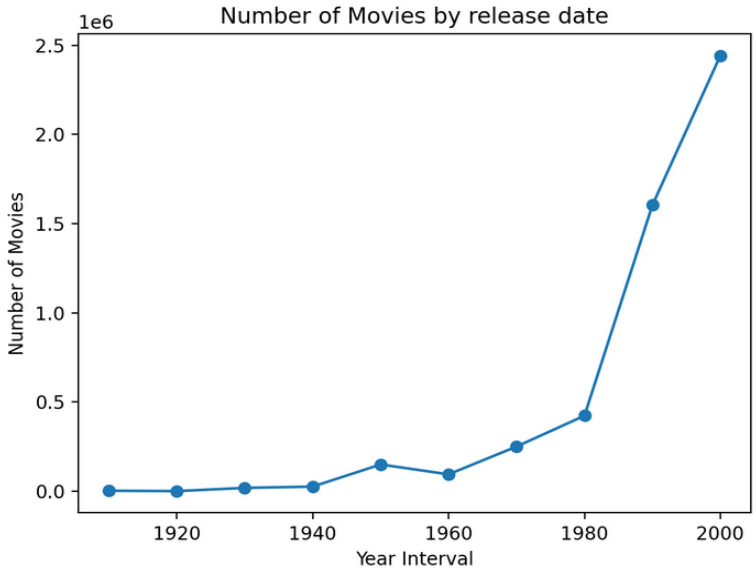

Figure 1: A graph that represents the release date of movies on the x-axis and the number of movies on the y-axis.

With respect to the formats of the files, the movie_titles, probe dataset, training_set and qualifying dataset were saved in .txt format. To turn the training set txt file to csv we created the column name "movie_id" and assigned the value to 1. We then created a pivot table with mean rating and the count rating for each respective movie id To preprocess the data, we had to filter out most of our data due to the inability of our machines to compute. Since most of the data is coming from the most popular movies and most active users, this will also help with preventing overfitting with movies that are very rarely seen by users. This can be shown in Figure 2 where it is shown that half of the total ratings are coming from only the 37 most popular movies.

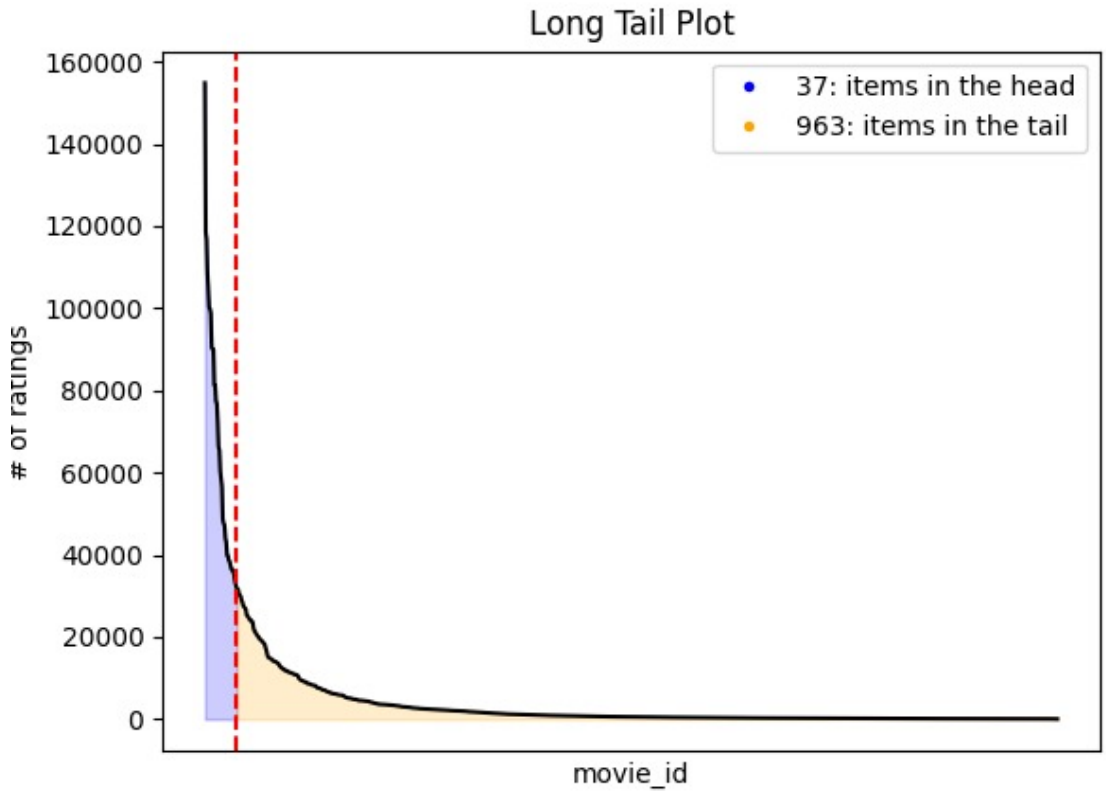

Figure 2: Visual representation of the most popular movies with movie_id placed on the x-axis and the number of ratings placed on the y-axis

Firstly, we grouped the data by movie_id and calculated the count of ratings for each movie. We also grouped the data by user_id and calculated the count of ratings for each user. We then sorted the movies and users based on their rating count in descending order from highest to lowest. This was done because as you can see in figure 2, the most popular movies made up most of the data which is why we filtered based on the most popular movies.Using these lists we then took only the users and movies with the highest counts for our filtered data set. We settled on using the top 8% of movies and top 8% of users by count. We found that this gave us the best balance between accuracy and computation speed. This reduced the amount of total entries from 5,010,199 down to 123,667. The number of movies was reduced from 1000 to 80 and the number of users was reduced from 404555 to 32364.

user_id	6	7	10	25	33	42	59	79	83	87	...	2649370	2649375	2649376	2649378	2649388	2649401	2649404	2649409	2649426	2649429
title																					
100 Days Before the Command	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11:14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44 Minutes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6ixtynin9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 3: Data after preprocessing

Hypothesis: Before conducting any results using our dataset and methods, we predict that our ensemble method will have a significantly larger RMSE compared to the winner of the Netflix Prize competition. Some of the questions that we considered were how much of an effect does one poor classifier have on the ensemble algorithm? Does a lower RMSE mean the classifier will help the ensemble model more than a classifier with a higher RMSE?

5 Experiment Design

We then selected the relevant features for all of our classifiers, which are user_id, movie_id and rating. Once these features were selected we split the data into a training and testing set with 80% in the training set and 20% in the test set.

After this we then ran each classifier on the training data and evaluated it against the testing data. The details for each individual classifier are as follows:

Cosine Similarity Weighted Average: In this approach, we first grouped by 'movie_id' and count the number of ratings for each movie. Then sorted the movies based on the number of ratings in descending order to get the most popular movies. Then generated a pivot table that summarizes the average rating given by each user for each movie along with the count of ratings given by each user for each movie. Then we tried to sample 20,000 users from a dataset of 75,000 users, preserving a proportional distribution of user ratings, by first calculating the count of ratings per user and then selecting users based on these counts but did not use that further due to insufficient information like missing genres, etc. Then converted to a Compressed Sparse Row matrix using SciPy's 'csr_matrix' for efficient handling of sparse data. After this, created a user similarity matrix, and calculated the weighted sum of ratings, and used the similarity scores between users and the ratings given by a specific user for all movies, returning the predicted rating. This process gives a final RMSE of 1.00.

```
[ ] user_similarity

    array([[1.        , 0.52466027, 0.57691127, ..., 0.32533737, 0.45688996,
            0.22031296],
           [0.52466027, 1.        , 0.40477173, ..., 0.30114491, 0.56819128,
            0.28841562],
           [0.57691127, 0.40477173, 1.        , ..., 0.20174251, 0.31878749,
            0.42077916],
           ...,
           [0.32533737, 0.30114491, 0.20174251, ..., 1.        , 0.29027236,
            0.        ],
           [0.45688996, 0.56819128, 0.31878749, ..., 0.29027236, 1.        ,
            0.15444591],
           [0.22031296, 0.28841562, 0.42077916, ..., 0.        , 0.15444591,
            1.        ]])
```

Figure 4: User Similarity Matrix

Cosine Similarity Mode: With the Cosine Similarity Mode classifier, we start with a matrix consisting of the users as rows and the movies as columns. We first turned all instances where a user has not watched a movie from a 0 into NaN. This is to make sure that the similarity between users is based on their ratings and not the fact that they both haven't seen a specific movie. We then normalized the data based on the mean of each user, since some users tend to rate higher or lower than others. After this, we then found the 50 closest users to each user using cosine similarity has the distance metric. 50 is the value we found that gave the best accuracy with not much deduction in runtime speed. Using these 50 users we then took all of their scores for each row of test data and found the most common score among them. These make up our final predictions and resulted in an RMSE value of 1.08.

Average Score: With the average score classifier, we first calculated the average score for each movie. We then found the percentile of the average score of each movie in comparison to all of the other movies. This is to make the ratings more spread out since each movie has a similar average score. We then took this percentile and mapped it to a prediction by using thresholds similar to the original data distribution. 5% of the original ratings are 1 so if a movie's rating percentile is 0.02, that is mapped to one. After getting these baseline predictions we then mapped these to the actual ratings by using linear regression. We got an RMSE of 1.23 after doing this.

RBM: With the RBM classifier, we first need to make the input data binary. We did this by considering all ratings that are a 1 or 2 and a 0 and ratings 3 or above as a 1. We then had to decide on a few parameters such as the number of hidden nodes and number of epochs. After running tests, we found that using 300 hidden nodes and 3 epochs was most efficient. We now run the 3 epochs, and in each epoch we use the RBM model on batches of users until we go through every user. For each epoch, we also update weights using backpropagation with the Adam optimizer algorithm. After

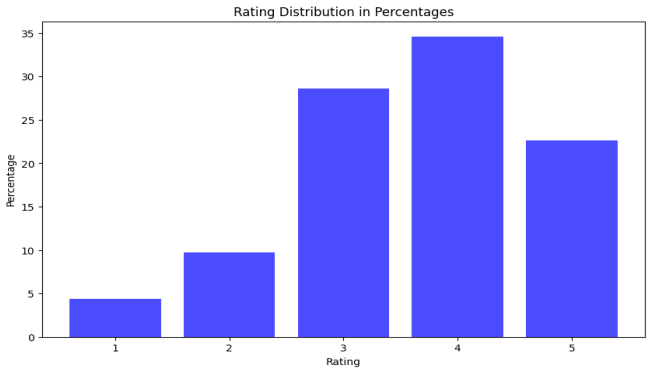

Figure 5: The Rating Distribution in Percentages of the dataset with the respective rating on the x-axis and the percentage of that rating in the dataset.

the RBM model was trained, we analyzed the probability that each visible node would be activated for each input value. We took these probabilities and transformed them into discrete rating predictions. We then created a linear regression mapping with these predictions and the testing data set in order to predict the final ratings. With this, we achieved a final RMSE of 1.03.

ANOVA: With the Anova classifier, using the sorted dataset (previously mentioned in the preprocessing section of this paper) we selected the relevant features for clustering. These features included user_id, movie_id and rating. After this step we then created a ratings matrix for both training and test sets. We then conducted K Means clustering on the training set with 100 assigned clusters. We originally tested for 5 clusters which we do not deem the more appropriate method. After the clusters were initialized, p values and t-statistics were calculated for each respective cluster and the most significant cluster was determined by the lowest p-value. After conducting the anova test we then mapped the values through linear regression and viewed the results in a histogram where we compared the actual versus predicted ratings.

Figure 6: Displays the hundred clusters that were created based on the rating the user gave to specific movies

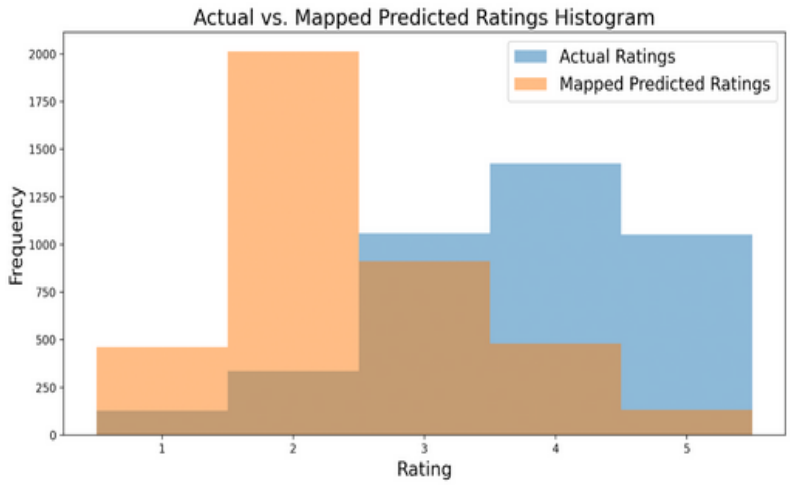

Figure 7: Displays a Histogram of the Actual vs Mapped Predicted Ratings after mapping the ANOVA rating values through linear regression

After running each classifier, we then took the results from each one to make a final classification with our ensemble. To do this, we took the average predicted rating from each classifier on each row of test data, and created a linear regression model using that average rating and the actual rating. We then used this to predict the final ratings for the test data.

6 Results

Our final ensemble model had an RMSE of 0.99. This is comparable to the RMSE of the Netflix algorithm at the time of the competition in 2008 which had an RMSE of 0.95. Within our ensemble, the highest performing classifiers were our Cosine Similarity weighted average classifier with an RMSE of 1.01, and our Restricted Boltzmann Machine classifier and ANOVA classifier with RMSEs of 1.03. Our lowest performing classifiers were our Cosine Similarity mode classifier with an RMSE of 1.08 and our average score classifier with an RMSE of 1.23.

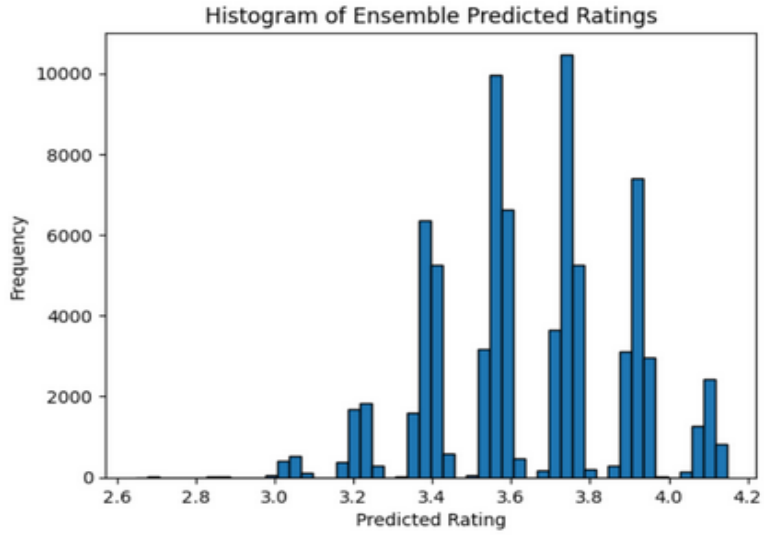

Figure 8: The histogram plotting the frequency of predicted ratings of the Ensemble method.

In terms of our hypotheses, our initial hypothesis was correct in that our RMSE value was higher than Netflix's. We also found that a higher RMSE does not necessarily mean the model is the least important to the ensemble. For instance in the "insert article", "stated that it was not optimal to minimize the RMSE of the individual predictors, further only the RMSE of the ensemble counts. Taking the average score classifier out of the ensemble hurts the ensemble more than taking out the ANOVA classifier which had a much lower RMSE. We also found that one bad classifier does not negatively affect the ensemble much, because its off prediction is made up for by the other predictions.

7 Conclusion

In our project, we have successfully completed parts of the preprocessing phase, including data cleaning and merging of the Netflix Prize Data. We've also implemented the k-NN (k nearest neighbor) algorithm as one of the four models inspired by Netflix's approach to collaborative filtering. This approach combines multiple algorithms to create a more effective model.

The main objective of our project is to forecast user ratings for movies based on those users' previous ratings and recommending relevant movies. We have successfully achieved an RMSE of 0.99. The Netflix Grand Prize Solution, which sought to enhance collaborative filtering algorithms for forecasting user ratings, served as the model for this project. Cosine Similarity, Restricted Boltzmann Machine (RBM), and Anova techniques are the four models that we investigated. RBMs model complex user behavior patterns, and ANOVA aids in choosing pertinent item features for content-based filtering. Whereas, Cosine Similarity offers recommendations based on content. Mixture of these resulted in an ensemble approach which chose the best model for the desired use case.

In the future, there are many other ideas we can explore to improve our research. One of these is to incorporate the date field into our models, since none of them currently are using this feature of the data set. This would look at the relationship between ratings and dates over time, and potentially be able to make our models more accurate. We can also implement the popular collaborative filtering algorithm SVD, or Singular Value Decomposition, into our ensemble. Further, perhaps if we had a faster computational power we could keep a higher percentage of data and explore whether we would be able to achieve the same results. This would use matrix operations and possibly incorporate more accuracy into our ensemble.

8 Appendix

Maya:
Data Cleaning: Organized the data for Clustering
Methods Development: Clustering and ANOVA
Exploration: Display of Histogram for ANOVA, Line graphs and Clustering Graph
Results Analysis: RMSE for ANOVA
Conclusions Drawing: Drew conclusions for ANOVA
Presentation Preparation: Completed and edited appropriate slides
Final Report Creation: Added figures and completed ANOVA, Clustering and Background, Experiment Set Up sections.

Brett:
Data Cleaning: Wrote the scripts to preprocess our initial dataset that we took from Kaggle.
Methods Development: Proposed the idea to make an ensemble classifier. Did the RBM classifier, average score classifier, cosine similarity mode classifier. Created the final ensemble classifier.
Exploration: Visualization of the dataset ratings. Visualization of all the predicted ratings from each classifier.
Results Analysis: Analyzed each of our classifiers and used this to optimize each one in the ensemble to make the ensemble make more accurate predictions.
Conclusions Drawing: RBM, Cosine Similarity Mode, Average Score, and Ensemble conclusions.
Presentation Preparation:. Made the slides outline and detailed what sections we should have. Completed each one with basic information. Added images and visualizations for each classifier.
Final Report Creation: Wrote the results section and experimental design sections of my three classifiers. Helped modify the experiment setup and conclusion sections to account for the changes we made since the midterm report. Converted our google document into LaTeX. Added figures for any necessary sections.

Ishika:
Data Cleaning: Handled missing ratings through zero imputation for top movies taken into consideration.
Methods Development: Prediction of ratings based on Cosine Similarity Weighted Average and KNN- based recommendations
Exploration: Created pivot tables and explored ways to filter data to get the best results like considering popular movies, movies with highest number of ratings or performing stratified sampling on the data.
Results Analysis: RMSE for Cosine Similarity Weighted Average
Conclusions Drawing: Drew Conclusions for Weighted Average Rating Cosine Similarity
Presentation Preparation: Overall formatting and edited relevant slides to my implementation.
Final Report Creation: Contributed in the Experiment Design, Methods and Conclusion

Lohith:
Data Cleaning: Downloaded the dataset from UCI and created a preprocessing pipeline to create the final dataset.
Methods Development: KNN-based movie recommendations and Cosine Similarity Weighted Average
Exploration: Visualization of the dataset using long-tail plot and pivot tables.
Results Analysis: RMSE for Cosine Similarity weighted average classifier
Conclusions Drawing: Weighted average Rating Cosine Similarity and KNN-Based recommendations conclusion.
Presentation Preparation: Edited the relevant slides to my implementation
Final Report Creation: Contributed with the Experiment Setup Part

Github Repository: https://github.ncsu.edu/bpreier/engr-ALDA-Fall2023-P24

References

Koren Y. Volinsky C Bell, R.M. The bellkor 2008 solution to the netflix prize. Journal of Important Research, 2008.

Andrey Feuerverger. Yu He. Shashi Khatri. Statistical significance of the netflix challenge. Statist. Sci. 27(2): 202-231, 2012.

Michael To scher, Andreas Jahrer. The bigchaos solution to the netflix grand prize. Journal of Important Research, 2009.

Avcontentteam. (2023, September 14). How machine learning is used on social media platforms in 2023?. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2023/04/machine-learning-for-social-media/

Kenton, W. (n.d.). Analysis of variance (ANOVA) explanation, formula, and applications. Investopedia. https://www.investopedia.com/terms/a/anova.asp: :text=Analysis

NETFLIX prize and SVD - ups. (n.d.). http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf

Norouzi, M. (2007). CONVOLUTIONAL RESTRICTED BOLTZMANN MACHINES FOR FEATURE LEARNING (thesis).

Steck, H., Baltrunas, L., Elahi, E., Liang, D., Raimond, Y., Basilico, J. (2021). Deep Learning for Recommender Systems: A Netflix case study. AI Magazine, 42(3), 7–18. https://doi.org/10.1609/aimag.v42i3.18140