

# Movie Recommendation Systems

## Sp22 SDS 323: Statistical Learning and Inference Final Project Proposal

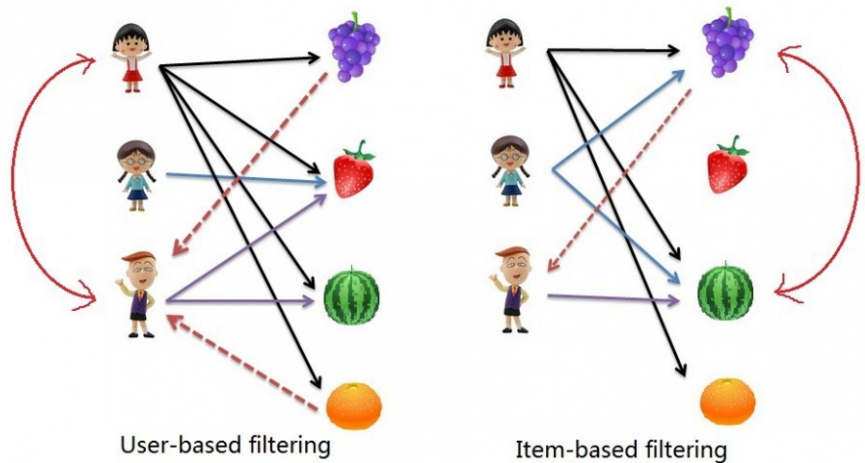
Hayley Zorkic, Ben Howell, Matthew Bradley, Ayanna Fisher

---

### Introduction

#### *Background*

Recommendation systems have become an integral component for modern web applications in order to personalize user experiences and increase engagement, with the natural end goal of earning more money. Platforms like Netflix, Hulu, Goodreads, Amazon, Spotify, etc. use recommendation systems in order to suggest content personalized to you based on your past and present behaviors in the context of these platforms. A recommendation is a machine learning system that provides *relevant* suggestions to customers. In the same way a friend may suggest that you read a book based on books you've read before or books that a majority of your friend group has enjoyed, we can suggest books, movies, and other content for users by using algorithmic approaches.



## *Approach*

For our recommendation system, we would like to suggest movies to a new user based on the movie preferences of similar users—this approach is known as “user-user collaborative-based filtering.” For example, following the image above, if user A is a 15 year old female student who likes grapes, strawberries, watermelon, and oranges, and user C is a 15 year old female student who likes strawberries and watermelon, a user-user collaborative-based filtering model may suggest user A would also like grapes and oranges. **We will employ linear regression, matrix-factorization, kNN, and the k-means algorithms in order to find users who enjoy similar movies as new users and then suggest 10 new movies for new users based on how the new users rate movies.** Due to the variations between different algorithms, the variables that can be used depend on the specific nature of the algorithm. Therefore, certain models will only be trained on model ratings as their input, while other algorithms will include additional demographic variables and information.

## *Data Collection*

Our data is from the “Movielens data sets” collected by a group at the University of Minnesota. The dataset consists of 100,000 ratings from 1 to 5, 943 users who have all reviewed at least 20 movies, user demographic data (age, gender, occupation etc...), and 1,682 movies and information about each movie (genre, year released etc...). Our analysis will be based on this demographic and movie data in order to predict which movies others will enjoy, through classification and clustering analyses.

---

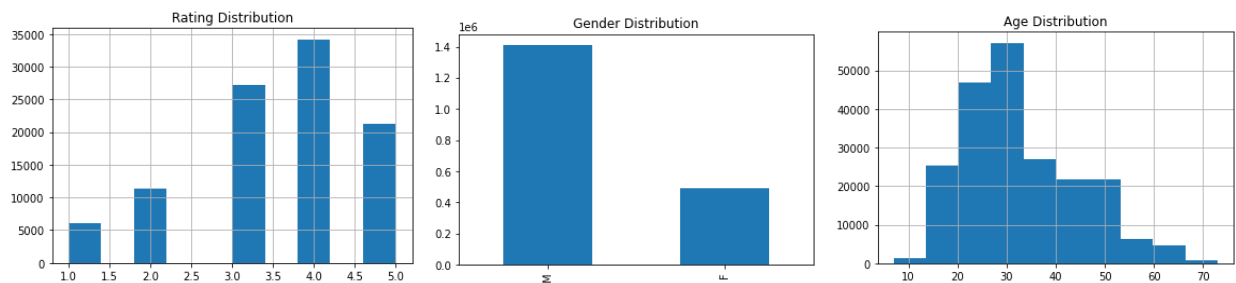
## Methods

### *Data Extraction*

The MovieLens data was downloaded as a multi-file relational database. We combined the u.data, u.user, u.genre, u.occupation, and u.item datasets in order to gather a final dataset which is pictured below:

user_id	item_id	rating	timestamp	age	gender	occupation	zip_code	job	movie_id	...	Film_Noir	Horror	Musical	Mystery	Romance	Sci_Fi	Thriller	War	Western	gender_id
196	242	3	881250949	49	M	writer	55105	20	242	...	0	0	0	0	0	0	0	0	0	0
186	302	3	891717742	39	F	executive	00000	6	302	...	1	0	0	1	0	0	1	0	0	1
22	377	1	878887116	25	M	writer	40206	20	377	...	0	0	0	0	0	0	0	0	0	0
244	51	2	880606923	28	M	technician	80525	19	51	...	0	0	0	0	1	0	0	1	1	0
166	346	1	886397596	47	M	educator	55113	3	346	...	0	0	0	0	0	0	0	0	0	0

This is a dataframe with 99991 rows (unique reviews) 33 columns (variables). The columns can be separated into three important groups: 1. MovieTitles, 2. Demographic information, 3. Rating. Some distributions of important categorical variables are included below:

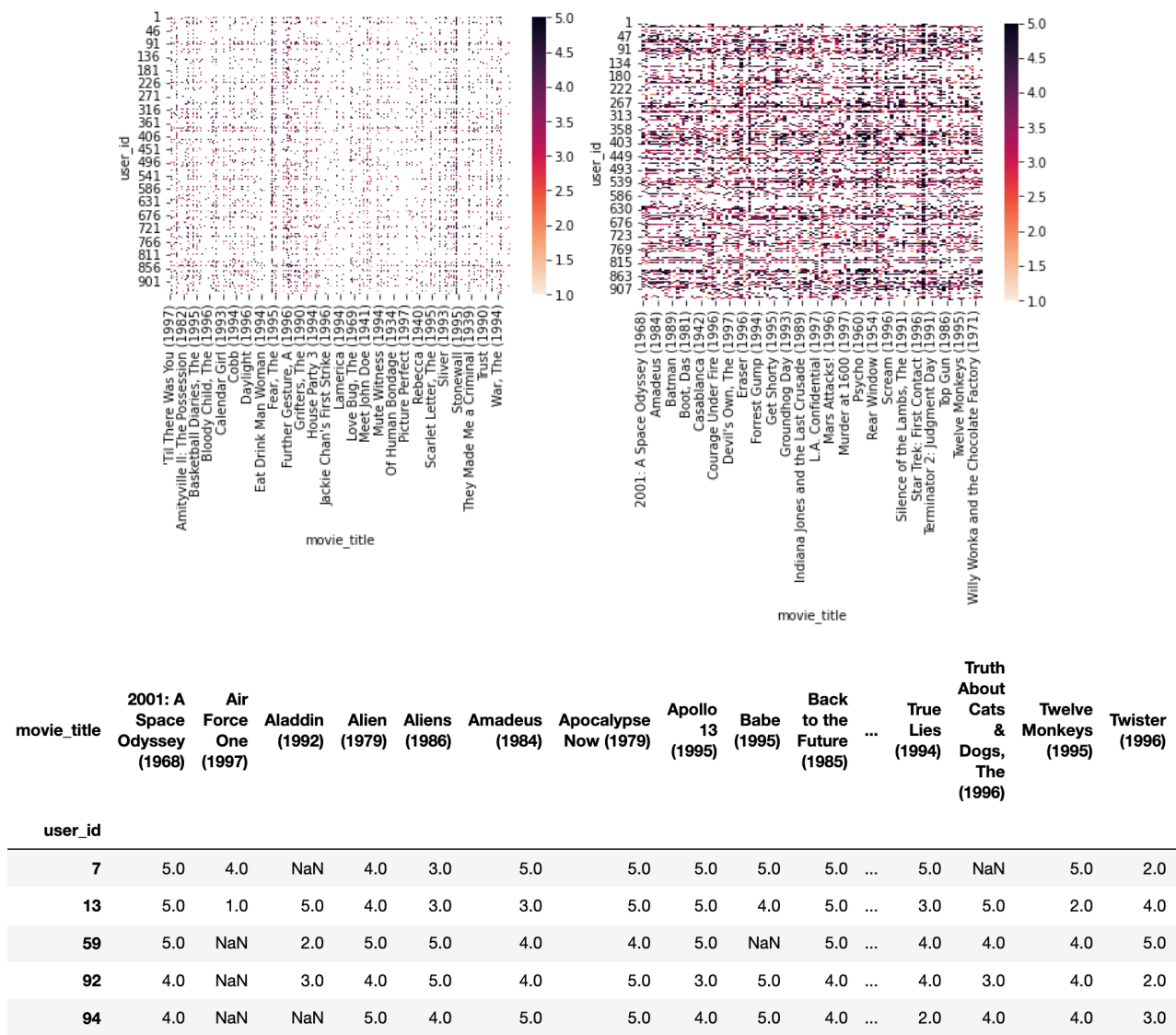


### *Data Transformation and Reduction*

We implemented four algorithms to generate 10 movie recommendations to users: linear regression, K-NN, K-means, and matrix-factorization. Because we cannot (easily) use categorical variables in K-means clustering or Matrix-factorization, we had to create two

different datasets for our analyses, one dataset with categorical variables, and a second without categorical variables.

For our numeric-only dataset, we had to remove the categorical variables and transform the data such that we ended up with a matrix where the indices were user-ids, the columns were movie titles, and the cells corresponded to the users respective rating of the movie. We visualized the data with a heat map and found that there were an overwhelming amount of NaN values in the data, so we filtered the data for users who reviewed more than 10 movies and movies that had at least 200 reviews. Below we included the before (left) and after (right) filtering heatmaps as well as the head of the final data matrix.

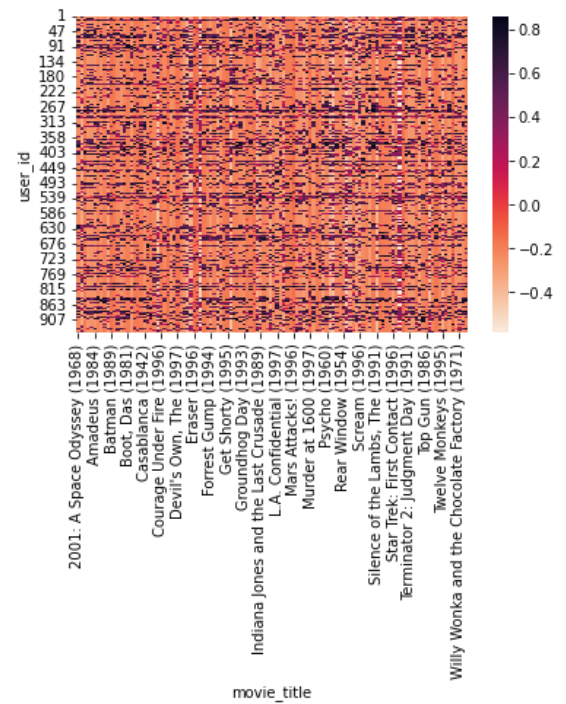


Not all users reviewed all movies, so there are still several NaN values in the data, thus we eliminated the NaNs by converting all of the NaNs to 0 and standardizing the ratings by applying the normalization equation to each row in the dataset which made the mean zero, more negative reviews negative and more positive reviews positive:

$$new\_row = \frac{row - row\ mean}{row\ max - row\ min}$$

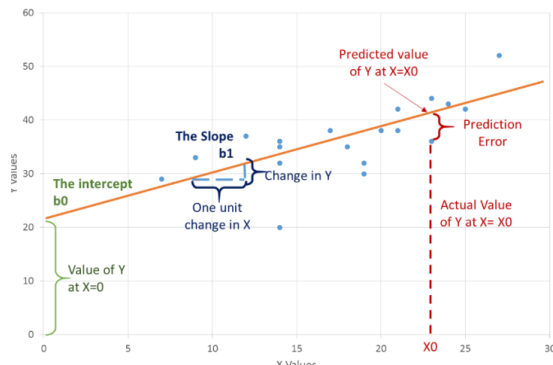
The final heat map included on the right shows the standardized reviews for users and movies.

For our dataset that included numeric and categorical variables, we also filtered the data to select users who reviewed more than 10 movies and movies that had at least 200 reviews.



## 1. What are the top 10 movie recommendations using Linear Regression ?

Figure: Example of Linear Regression



We chose to use Linear Regression as our initial analysis tool because it is the default regression tool, and incredibly easy and simple to implement in Python. Additionally, when we submitted our project proposal, we received feedback that we should include a basic model, such as a Linear Regression, as that would give us a starting point to

work off and improve upon throughout our project. Linear Regression evaluates the target variable on a continuous scale, then fits a formula and coefficients to predict the target variable.

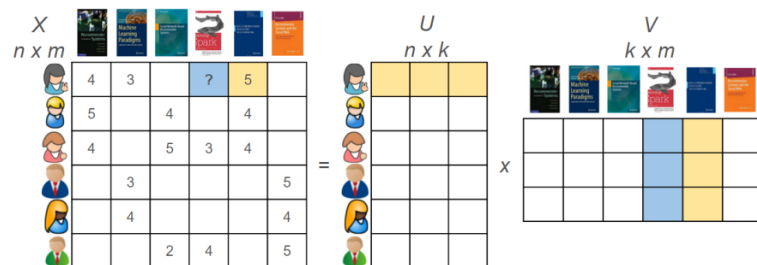
However, despite the ease with which one can implement a Linear Regression model in Python, it has notable flaws that make it a poor model to use for a final recommendation product. Recommendation models, or even rating models, work best when you treat the rating as a classification problem, in which the base linear regression is unable to differentiate between the ratings as classes.

We trained a linear regression model on all the observations in the dataset, giving us the ability to predict movie rating as a combination of demographic and movie-specific factors. To make this possible for use as a “recommendation”, we structured the model so that it takes the user’s demographic information, combines it with all the movie options, then returns the 10-highest predicted ratings for the movie list.

In general, the linear regression approach was not the best approach to use for either movie rating or for movie recommendations. With that acknowledgement that the linear regression structure is a clumsy fit for a recommendation model, we believe that our approach to the question makes sense. Such a model would not be fit for commercial deployment, but satisfies our needs as an exploratory model for our project.

## 2. What are the top 10 movie recommendations using a standard matrix factorization algorithm?

Figure: Example of Matrix-Factorization algorithm



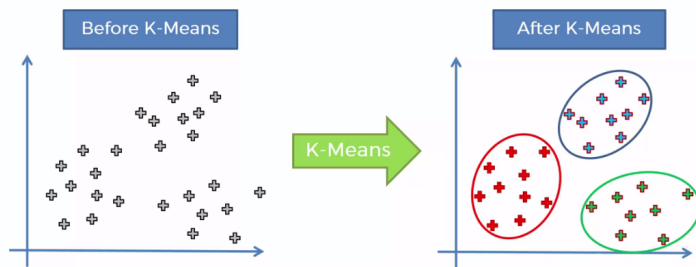
Collaborative filtering is a technique used predominantly for recommender systems like the one we are aiming to create and analyze for this project. Recommender systems filter information provided

by users to predict factors such as rating, or in our case, probability of preference for a given product, like a movie. This personalized information uses the interactions and data collected from other users to recommend products based on their agreeance/similarity in ratings. As described previously, we want to suggest movies to existing or new users based on ratings given by other, existing users provided by our dataset.

When attempting to execute the drafted code, we found that the item-to-item method would be the best way for analysis instead of user-to-user, since there are more users than movies and there's more available variability/dynamic when working with the ratings in this way. To quantify the similarity of the movies that would be considered to be recommended, the analysis was distance based, in which we were specifically using Pearson's Correlation to predict.

### 3. What are the 10 movie recommendations from K-Means Clustering ?

Figure: Example of K-means clustering



K-means clustering is an unsupervised algorithm that categorizes data into  $k$  clusters. In a 2D dataframe, we could plot points in a scatter plot and easily observe clusters of data, however when we

approach higher-dimension data such as those above 4, it becomes very difficult for humans to visualize and conceptualize these clusters. In these cases we entrust unsupervised clustering algorithms to find observed patterns in the data.

This algorithm is commonly used in fields such as customer segmentation, which is what inspired the use of this algorithm. In this project, we wanted to suggest movies to new users based on how they compare to existing users. We can employ k-means to develop user-profile clusters. When we have new user data, we classify them into an existing cluster, and predict movies they would like based on the top movies of other users in the same user-profile cluster.

Specifically, we did this by creating a data matrix of reviews with different movies as columns and different users as rows. Not all users have reviewed all movies, so we are left with several NaNs. We then fill in the NaNs by filling in the values with zero and normalizing each row (details in the pre-processing section above).

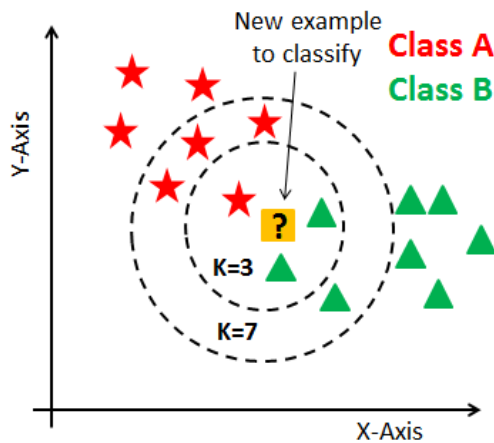
To select the number of clusters, we employ the elbow method by visually inspecting the sum of squared distance between each point and the centroid in a cluster and selecting the point that appears as the lines “elbow”. Next, we initialize the k-means model with the chosen  $k$  value, fit the model to the standardized review matrix data.

Finally, we take a user profile including a list of movie ratings, obtain their most likely cluster group, and then return the top 10 movie recommendations of that user profile cluster.



#### 4. What are the top 10 movie recommendations using a kNN classifier ?

Figure: Example of K-NN



For the k Nearest Neighbors Classification approach we started by subsetting the data to only include movies with over 200 reviews and reviewers with at least 10 reviews. The resulting dataset contained about 32,000 observations. This is especially useful for the kNN approach because our dataset, before subsetting, has nearly 100,000 observations, and kNN is computationally expensive and slow compared to other approaches. Especially because we need to test

different values of k in order to minimize error, subsetting data will be very useful. kNN is appropriate for this data because there are some non-normal variables and not all variables have a linear relationship with the response variable. Because kNN is nonparametric, it makes no assumptions about linearity so our results may be more accurate than if we used a parametric model.

Once we had our data subsetting, we split it into train and test datasets with a training dataset making up 66.7% of the data and a test set making up 33.3% of the data. We then scaled the training data because kNN, since it is a distance based algorithm, will be more accurate if the independent variables are on the same scale, otherwise certain variables may dominate the model. We fit 20 kNN models, with k's ranging from 1 to 20, and found the k value where error stopped decreasing as drastically. We then fit a kNN classifier model with this number of neighbors, and analyzed the confusion matrix to understand how well our fit was. We also looked at recall ( $\frac{\text{True positives}}{\text{True positives} + \text{false positives}}$ ) and precision ( $\frac{\text{True positives}}{\text{True positives} + \text{false negatives}}$ ). Lastly, we recorded the top movies recommended by the kNN model and what their predicted ratings were.

---

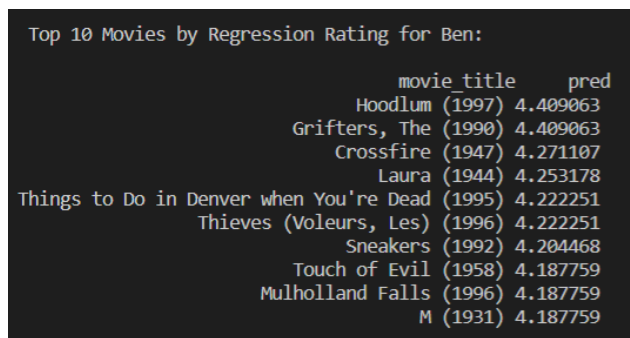
## Data Analysis

### *1. What are the top 10 movie recommendations using Linear Regression ?*

While the linear regression model did not score in a manner that would lend it to production or commercial use, we believe our approach returned a decent baseline model. As we did with our other models, we subsetting the data so that our dataset consisted of movies with over 200 ratings and reviewers who had reviewed more than 20 films. We trained the linear regression on 67% of the remaining observations in the dataset, and used the remaining 33% to evaluate the model. Originally, we did *not* apply the stricter filtering approach to the model, but when we applied the stringent filtering, we saw different movies recommended, which was a good sign that it had an impact.

Our base linear regression model was trained on a combination of demographic variables, as well as variables regarding the movie, which consisted of one-hot encoding of the movie genre. The demographic variables included age (how old was the reviewer), gender\_id (a binary 0/1 variable where 1 represents female), and job.

*Figure: Top-10 Movies by Linear Regression Predicted Rating for Simulated User (Ben)*



```
Top 10 Movies by Regression Rating for Ben:
      movie title      pred
Hoodlum (1997) 4.409063
Grifters, The (1990) 4.409063
Crossfire (1947) 4.271107
Laura (1944) 4.253178
Things to Do in Denver when You're Dead (1995) 4.222251
Thieves (Voleurs, Les) (1996) 4.222251
Sneakers (1992) 4.204468
Touch of Evil (1958) 4.187759
Mulholland Falls (1996) 4.187759
M (1931) 4.187759
```

Once the model was trained, it made the most sense to apply it by taking a user's demographic information as an input, then using that in the new test dataset for each movie title.

Once those predictions were made with the new user demographic data, it's a simple matter to sort the values by predicted rating and return the top-10 list.

As an example of this approach, we simulated the user input data by using one of our authors, Ben Howell, as the test subject, meaning that the above list is for a 21-year-old male student. The highest predicted rating belongs to Hoodlum, which is closely followed by Grifters.

## 2. What are the top 10 movie recommendations using a standard matrix factorization algorithm?

Before further wrangling our data, we created a mockup of a correlation matrix for 8 of the major variables within our dataset. Since we were using a standard matrix factorization algorithm was being used to test which variables coincide the most within the chosen dataset, we simply had to create a correlation matrix to observe which quantitative variables had the value closest to 1, without being 1 or -1 exactly as that would be the variable compared to itself. The variables found to be the most (dis)similar in this initial test were, age & job with a correlation of -0.36, rating & movie\_id with -0.18, and age & rating with -0.047.

Figure: Correlation matrix

	user_id	item_id	rating	timestamp	age	job	movie_id	gender_id
user_id	1.000000	0.007786	-0.009345	0.017033	-0.075986	0.104361	0.007786	0.028086
item_id	0.007786	1.000000	-0.183815	0.044195	-0.001901	0.000972	1.000000	0.032098
rating	-0.009345	-0.183815	1.000000	-0.018312	0.047199	-0.027173	-0.183815	-0.000223
timestamp	0.017033	0.044195	-0.018312	1.000000	0.153953	0.027172	0.044195	0.035340
age	-0.075986	-0.001901	0.047199	0.153953	1.000000	-0.369569	-0.001901	-0.033479
job	0.104361	0.000972	-0.027173	0.027172	-0.369569	1.000000	0.000972	-0.015315
movie_id	0.007786	1.000000	-0.183815	0.044195	-0.001901	0.000972	1.000000	0.032098
gender_id	0.028086	0.032098	-0.000223	0.035340	-0.033479	-0.015315	0.032098	1.000000

With these results being the strongest of the variables used in the test, it indicated that, all in all, the correlation within the dataset was weak. To further the analysis of the data, we reduced the dataset to only consider 200 movies, which reduced the shape from 100,000 entries to 943, making it more manageable to analyze. With that, we ran Pearson's Correlation tests to observe which movies a user that has watched a specific movie would both most likely and least likely watch depending on the rating they gave the original movie.

As a specific example, we took the movie, *2001: A Space Odyssey*, and recommended 10 movies, printing out the correlation to demonstrate how likely the user would at least watch the recommended films. We also did the opposite for which films the user who watched *2001: A Space Odyssey* should avoid watching.

Figure: Pearson's Correlation Recommendation

```
-----
List of 10 recommended movies to a user who has liked '2001: A Space Odyssey'
movie_title
2001: A Space Odyssey (1968)      1.000000
Clockwork Orange, A (1971)        0.388071
People vs. Larry Flynt, The (1996) 0.327292
Apocalypse Now (1979)             0.312847
Godfather, The (1972)             0.305717
Fargo (1996)                      0.299882
Terminator, The (1984)            0.260066
Brazil (1985)                    0.254415
Graduate, The (1967)             0.252987
Alien (1979)                     0.248089
Name: 2001: A Space Odyssey (1968), dtype: float64

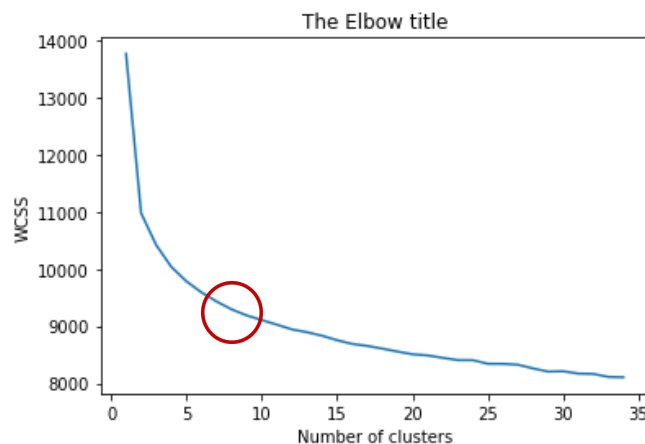
-----
List of 10 movies to NOT recommend a user who liked '2001: A Space Odyssey'
movie_title
Air Force One (1997)              -0.282994
Murder at 1600 (1997)            -0.222437
In & Out (1997)                  -0.195379
Saint, The (1997)                -0.182773
Beauty and the Beast (1991)      -0.171573
Conspiracy Theory (1997)         -0.164735
Full Monty, The (1997)           -0.149286
Evita (1996)                    -0.142785
Eraser (1996)                   -0.126674
Phenomenon (1996)               -0.122596
Name: 2001: A Space Odyssey (1968), dtype: float64
-----
```

This recommendation system can be seen in the figure above, revealing that the movie with the most probability to be recommended to a user that has watched *2001: A Space Odyssey* would be *Clockwork Orange, A* (1981), while *Air Force One* (1997) would be the least likely to be recommended to the same user.

### 3. What are the 10 movie recommendations from K-Means Clustering ?

After filtering our matrix such that we only have users that reviewed at least 10 movies and only included movies with at least 200 reviews, we had a dataframe with the dimensions of (854, 118): 854 users, 118 movies. We set the NaN values to zero and then normalized the data in the data frame to get our final matrix—we will use this matrix to generate clusters.

Figure: Elbow plot for K-means



In order to select the optimal number of clusters, we created an “Elbow Plot” which plotted the within clusters sum of squares (WCSS) value against the number of clusters. Upon visual inspection, the elbow appears to be between 5-10, so we chose 7 as our k value.

Figure: Top-10 movies chosen by K-means ratings.

#### Top 10 Movies by K-Means Ratings:

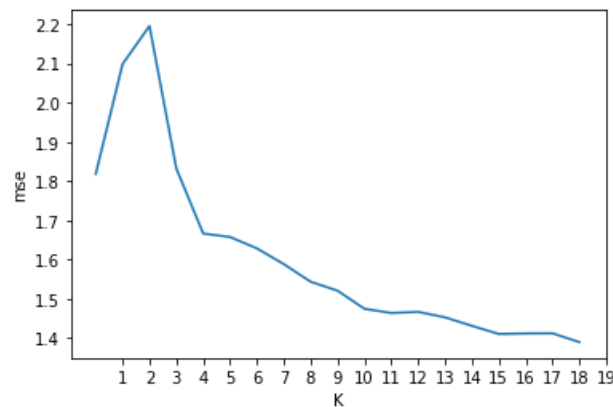
Babe (1995)	4.600000
GoodFellas (1990)	4.500000
Casablanca (1942)	4.444444
2001: A Space Odyssey (1968)	4.400000
Boat, Das (1981)	4.380952
Fugitive, The (1993)	4.375000
Graduate, The (1967)	4.363636
Wizard of Oz, The (1939)	4.333333
Field of Dreams (1989)	4.333333
Schindler's List (1993)	4.294118

Then, we fit a K-means classifier using the standardized data matrix with  $k=7$ . The resulting groups had the following number of users in them {4: 168, 6: 129, 2: 127, 0: 125, 3: 107, 1: 104, 5: 94}. We appended the cluster labeled to the non-standardized numeric data matrix and replace the NaN values with ‘’. In order to generate a “test” recommendation, we isolated a user\_id from the numeric data matrix taking note of their cluster. In this case, we took user 2 from group 6. We identify the movies the user has already reviewed, remove those from potential recommendations. and then take the average rating of all of the users in the same cluster (cluster 6) for each movie and output the movies with the average top 10 highest ratings within the cluster. These are the top 10 movie recommendations from the K-Means algorithm.

#### 4. What are the top 10 movie recommendations using a kNN classifier ?

The first step for kNN analysis, once the data was cleaned, was to choose the numbers of neighbors,  $k$ . For this, we fit 20 kNN classifiers to our training data, and evaluated the mean squared error of our predictions of the test data set to the actual test values. We found that error steadily decreased until around 15 then begin to level off:

Figure: kNN error values based on number of neighbors.



Because of this, we chose 15 as our  $k$  value for the analysis. We then fit our kNN classifier with 15 neighbors to the data and made predictions of the test set.

In this case, like the linear regression model, the test and training sets included different user's demographic information as an input, then used that in the new test dataset for each movie title. Once those predictions were made with the new user demographic data, it was easy to sort the values by predicted rating and return the top-10 list.

Overall, our predictor tends to think people will like movies more than they actually do, with the majority of its predictions being 3 or 4 stars. We can see in the heat map of the confusion matrix that even for ratings that were actually 1 or 2 stars, the classifier thinks ratings will be higher.



Figure: kNN Confusion matrix heat map:

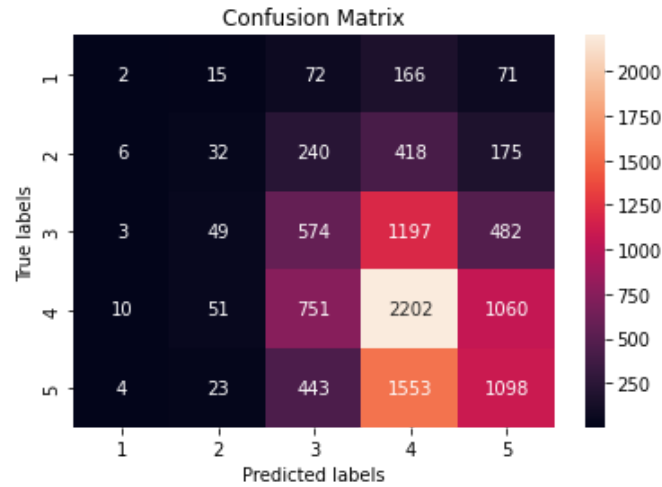


Table: kNN precision, recall, and f score:

	precision	recall	f1-score	support
1	0.08	0.01	0.01	326
2	0.19	0.04	0.06	871
3	0.28	0.25	0.26	2305
4	0.40	0.54	0.46	4074
5	0.38	0.35	0.37	3121
accuracy			0.37	10697
macro avg	0.26	0.24	0.23	10697
weighted avg	0.34	0.37	0.34	10697

Overall, the classifier struggled to accurately predict ratings when looking at precision and recall. Part of this may be the fact that in order to have a true positive, the classifier must correctly predict the numerical rating, so predicting a 3 or a 5 when a rating is actually 4 is seen as equally wrong as predicting a 1 when a rating is actually 5. Our overall precision (using weighted average) was 0.34, and overall recall was 0.36.

Figure: Top 10 movies, and their predicted ratings, based on kNN

movie_title	
GoodFellas (1990)	4.465753
Godfather: Part II, The (1974)	4.424242
Empire Strikes Back, The (1980)	4.418033
Apocalypse Now (1979)	4.389610
Pulp Fiction (1994)	4.379562
Graduate, The (1967)	4.376623
Return of the Jedi (1983)	4.366279
Usual Suspects, The (1995)	4.341176
Seven (Se7en) (1995)	4.328947
Star Wars (1977)	4.324324

---

## Conclusions and Future Work

Upon reviewing our process and the four approaches that we took, linear regression, standard matrix factorization algorithm, K-Means clustering, and kNN approach, we felt that the K-Means clustering and standard matrix factorization approaches made the most sense and returned the most interpretable results.

The k-NN algorithm makes a lot of sense on the surface through identifying similar movies as it allows us to include demographic information into our predictions. However, through our model diagnostics, particularly our Confusion Matrix, we noticed that the model struggled mightily with properly identifying ratings below 3, which is not ideal because reviewers rarely rate movies low unless it was a very bad experience.

The standard matrix factorization algorithm was the most basic recommendation system that we explored and was quite solid. We appreciated that this approach easily returned both the most similar movies as well as a score that measured that similarity and relationship between the two. It made for a strong starting point that we were able to build on and compare to quite easily.

We felt that the K-Means clustering approach did a strong job of sort of combining the things that we liked about both the kNN and standard matrix factorization algorithms. The K-Means clustering is obviously similar to the kNN approach where they get their value from identifying similar values and what other observations are around them. Additionally, because we took the approach of identifying user-profiles from clusters, we significantly reduce the computational complexity and time of the analysis as it is not performing calculations on thousands of users and movies. It is important to note that we did not come up with a way to “fit” a brand new user to a pre-established cluster in order to provide them cluster-inspired recommendations, but rather, we used a pre-existing user in a known cluster to do so.

For future works, we **strongly** recommend developing a way to assign a new user to an existing cluster perhaps using another algorithm. This would make the K-Means implementation more generalizable and potentially able to be implemented in some kind of live production.

Another point that we identified as an avenue for interesting future work would be to scale the reviews by each reviewer so that the tendencies of an individual reviewer are accounted for within our algorithms that include categorical data as well. We did do this for our , if you have two reviewers, A & B, but A just naturally enjoys movies and consistently even rates “bad” movies higher, that will end up

having an effect on the model and the predicted ratings that we can generate for users. Scaling reviews to some sort of normal-esque distribution across users could make a lot of sense and even help the model perform better.

---

## **Group Member Contributions**

### **Hayley Zorkic**

- Idea generation: proposed project idea, wrote introduction/background section
- Data cleaning: creating numeric-only matrix, filtering numeric-only matrix data
- K-means clustering execution
- Final proposal writing: expanding key points, formatting

### **Ben Howell**

- Preliminary Analysis for Project Proposal
- Linear Regression execution
- Data Cleaning: merging data sources
- Manage version control
- Final proposal writing: Linear Regression sections, Conclusion, editing

### **Matthew Bradley**

- Data exploration: created plots/figures to explore distributions of variables and correlations between variables
- Data Cleaning: filtering non-numeric matrix data

- K-NN Algorithm execution and analysis
- Final proposal writing: creating outline, filling in key points, wrote kNN sections

### **Ayanna Fisher**

- Matrix factorization execution
- Data Cleaning: reduce dimensionality of data, transposing
- Final proposal writing: Collaborative Filtering based recommendation system (Matrix Factorization, Pearson's Correlation)

---

## Sources

### **Overview of Collaborative Filtering:**

<https://developers.google.com/machine-learning/recommendation/collaborative/basics>

### **For K-Means Implementation:**

[https://programming.rhysshea.com/K-means\\_movie\\_ratings/](https://programming.rhysshea.com/K-means_movie_ratings/)

### **For Matrix Factorization Implementation:**

<https://www.youtube.com/watch?v=3ecNC-So0r4>

---

## **Source Code**

[https://github.com/benhowell71/SDS-group-repo/tree/main/final\\_project](https://github.com/benhowell71/SDS-group-repo/tree/main/final_project)