**A Comparison of Movie Recommendation Engines**

By: Sophia George

Student number: **501004962**

Sourcecode: <https://github.com/sophiafgeorge/CKME-136-Capstone>

**Abstract**

The ever-increasing volume of information across the web has rendered the need for consumers to filter through options, to make informed decisions. This hold true in media, as the ease of selection can generate more views, purchases and revenue. The movie industry has seen a substantial increase of revenue due to the avid use of recommendation systems.

Recommendation systems are algorithms that make product or service recommendations to customers by narrowing down their search and presenting them with suggestions that they are likely to consume. One of the most popular recommendation systems is the collaborative system. The collaborative system can either use the behavior of the user to make recommendations by suggesting products that similar users like (user-based collaborative filtering) or by identifying items based on the user’s preferences (item-based collaborative filtering).

This Capstone project will focus on creating various recommendation systems to create profitable recommendations using the MovieLens 100K dataset, which contains 100,000 rows and 38 attributes (<https://grouplens.org/datasets/movielens/100k/>). Matrices would be created and incorporated with similarity measures and algorithms such as Pearson Correlation, Cosine Similarity, and KNN to ensure profitable movie recommendations to consumers.

# Literature Review

In the last 30 years recommendation systems, have been an avid topic in both academia, and media. Ease of selection can generate more views, purchases and revenue for media outlets. The movie industry has seen a substantial increase of revenue due to the avid use of recommendation systems. The unique ability of the system to generate loyalty and revenue, while bettering the customer experience has allowed it to become popular amongst media databases such as Netflix and MovieLens.**1**

Recommender systems are algorithms that make product or service recommendations to customers by narrowing down their search and presenting them with suggestions that they are likely to consume. There are three types of recommendations systems: content based, collaborative filtering and hybrid. Content based systems recommends consumers, items with similar content to the items in their purchase history. The most widely used recommendation systems is the collaborative system. The collaborative system can either use the behavior of the user to make recommendations by suggesting products that similar users like (user-based collaborative filtering) or by identifying items based on the user’s preferences (item-based collaborative filtering)**.2** Most Hybrid system use demographic data such as sex, and occupation in combination with collaborative methods to obtain a combination of benefits.

Collaborative filtering can be further categorized into two types: memory or neighborhood-based systems, and model-based systems. Memory based typically use similarity measures such as Pearson Correlation and Cosine to calculate similarity between consumers. **3** Collaborative Neighborhood systems are extremely prevalent in academia and are considered the benchmark of comparison for new discoveries. Model Based systems seek to determine what the user decision would be for products they have not consumed. Machine learning approaches such as Matrix Factorization, Principal Component Analysis, and Singular Value Decomposition are used frequently in this field.**4** This Capstone project will focus on creating various recommendation system using the MovieLens 100K dataset by creating matrices, incorporating them with similarity measures and algorithms such as Pearson Correlation, Cosine Similarity, and KNN, to ensure profitable movie recommendations to consumers.

In order to understand how this capstone fits into the diverse spectrum of collaborative recommendation systems in academia, it is imperative that data analysts understand how the different model variations have addressed the inherent deficiencies of a collaborative recommender engine.

The author, Bernal outlines how a singularity approach can enhance the traditional similarity measures (Cosine, KNN, and Pearson Correlation) by identifying key differences when comparing two user’s preferences. **5** He focused on exploring the relationships between rare similarities that are uncommon when compared to the user population as an alternative to considering similarities that are apparent for the majority and deemed them a powerful insight. This method was used on the MovieLens 1M, Netflix and FilmAffinity databases. The rating systems were transformed by categorizing by 4 to 5-star ratings as positive rank and anything below 3 stars as a negative one. This contextual information was used in addition to the numeric information of the ratings while running the K-nearest neighbors algorithm which generated an improvement in accuracy, precision, mean of squared errors, recall and percentage of predictions, when compared to neighborhood systems methods**6**.

Similarly, researchers Jesús Bobadilla, Fernando Ortega, and Antonio Hernando, sought to build on the Pearson Correlation similarity measures in collaborative item-based recommender systems as well by transforming the data into positive and negative ratings. This provided contextual data in addition to the numeric ratings of the MovieLens, FilmAffinity and Netflix datasets. The proportions of common votes and the total votes were analyzed and users with similar voting engagement were compared pairwise throughout the entire datasets.**7** The Jaccard method was used to calculate the proportion of items users had in common, the items they did not have, and compare them with the total number of item they voted for. The combination of Jaccard measure with another similarity calculation called Mean of Squared Differences was used outperform Pearson correlation on the ratings attribute alone on the three datasets. The proposed metric yielded better results in all fields (accuracy, precision, recall, perfect predictions) except for coverage**8**.

Combining implicit information with explicit data on users is another fascinating approach to revolutionize collaborative systems. The One-Class Collaborative Filtering article outlines the drawback of collaborative filtering by using Once Class collaborative filtering (OCCF) where only binary data of the users is analyzed through implicit feedback**.9** The team of researchers argue that this is a more realistic model than the traditional collaborative approaches that rely on explicit user feedback. Two methods were used to achieve this, combining the scores from all sources of user information independently to produce the recommendation, and incorporating rich implicit information into the Collaborative filtering model. The study found that rich user information performs better than traditional collaborative recommender models with weak neighbor relations. Evaluation metrics selected to showcase these finding were Mean Average Precision (MAP) and Mean Percentage Ranking.**10**

In addition to incorporating new measures in the foundational models, finding ways to address how recommender systems can evolve when new or unrated items are included into the model is a pressing concern. The article on prediction uncertainty in collaborative filtering by Mingyue Zhang, Xunhua Guo, and Guoqing Chen beautifully articulates the major drawbacks with traditional collaborative systems when dealing with sparsity and scalability. Sparsity becomes a problem when the recommender system is missing key information such as movie ratings and is unable to make accurate recommendations due to the limited data available**11**.

The article proposes the need to address the sparsity concern and improve the traditional neighborhood collaborative recommender system by analyzing the uncertainty associated with each user prediction.**12**Two method used to model the uncertainty, posterior rating distribution and conﬁdence level of prediction to create a new ranking method called Ranking with Prediction Uncertainty (RPU). The basic collaborative filtering ranking is used as a benchmark for evaluating RPU. This ranking method can be calculated by incorporating Bayes Theorem with the statistics of the training set to compute the posterior rating to inevitably calculate the maximum value of the item set in question.( 16)**13** Multiple collaborative filtering techniques such as K Nearest Neighbors and Singular Value Decomposition were used alongside the RPU to achieve greater accuracy in recommendations.

There is clearly an ever-increasing need in media for recommender systems to be scalable due to mass media databases like Netflix that need models to perform just as well on extremely large data. Authors, Yunhong Zhou, Dennis Wilkinson, Robert Schreiber and Rong Pan describe how they designed an algorithm for the Netflix Prize (popular collaborative filtering challenge) that addresses traditional collaborative filtering concerns such as sparsity and scalability. The weighted regularization of Alternating-Least-Squares (ALS-WR) algorithm was adopted and increased performance with increased features and iterations on a parallel Matlab, in a clustered Linux environment. This discovery underlines the possibility for recommendation systems to scale to extremely large datasets while still having great accuracy (Root Mean Square Error score of 0.8985)**14**.

The research work of Xiaofeng Yuana, Lixin Hana, Subin Qian, Guoxia Xua, and Hong Yanc on SVD, depicts a collaborative model recommender systems’ approach to alleviating sparsity by increasing density in the model. ISVD is created by and improving the quality of recommendations by generating more useful ratings on the training data and incorporating it in the SVD model. **15**Imputed data is developed in the rating matrix after which neighbor selection has been carried out and the similarity is calculated in on both the user similarity matrix and the item one. Once it is created it is blended into the SVD Framework to create the final improved recommendation engine which displays a higher density in training data and outperforms accuracy in SVD methods such as RSVD, and Social Regulation.**16**

Alternatively, Urszula Kużelewsk’s research on a hybrid recommendation measures distinguish how clustering can be fused with recommender systems and reduce sparsity by associating different users with different clusters. This categorization of similar vector patterns reduces computation time by comparing the active users with the clusters instead of a full dataset(cluster based)**17** The methods used were modified K-means andCluster-onlywhich is used in recommender systems by subspace clustering. Similar users were used in offline recommendations to identify users with similar rating frequency. In the online portion the recommended items are created by not comparing to the full dataset but to their clustered segments of users. The clustering algorithms were compared with similarity measure such as Cosine, Pearson Correlation and Euclidean distance using Mahout Apache library and evaluated by RMSE between the predicted and actual preferences.**18** Although the approach yielded lower precision rates, it also allowed for quicker computation time than traditional memory based collaborative methods that identify like users by using K-Nearest Neighbor approach.

Deep learning’s layering ability is a complex and fascinating approach which has become a popular choice in media collaborative filtering competitions such as Netflix. A team of researches consisting of Feng Xue and JianDong Xu, Kai Liu, and Richang Hong from China, as well as Xiangnan He and Xiang Wang from Singapore collaborated to uncover more complicated aspects of consumer decision-making to understand which itemset in a user profile is more indicative of a decision. Non-linear and higher-ordered relationships take precedence by considering the interaction amongst all relatable item pairs, in order to go beyond the traditional similarity measures between two items. A modification of the neural network architecture called DeepICF **19**(deep variant of item based collaborative filtering page 33:4) was selected. Multi-hot encoding was used in the input layer of users’ interacted items, to use a set of vectors to reflect al the interactions of each user instead of just on the user id in traditional embedding layers. The embedding layer and the pairwise interaction lay beneath the Multiple layers (Deep Interaction Layers) to capture deeper facets of user decisions when recommending a prediction score The model was evaluated by looking for the higher HR and NDCG scores and compared it to approaches like Bayesian Personalized ranking and ItemKNN of which DeepICF+ outperformed the majority.**20**

# Dataset

This capstone project will utilize the MovieLens 100K dataset. The data was collected by the GroupLens Research Project at the University of Minnesota. This Capstone project will focus on creating various recommender engines such as, popular and collaborative recommendation systems using the MovieLens 100K dataset, which contains 100,000 rows and 38 attributes. The data for the users and items are numeric and are consecutively ordered from 1 and is tab separated. The data set is available using the following url: <https://grouplens.org/datasets/movielens/100k/>

It consists of:

* **100,000 ratings** (1-5) from 943 users on 1682 movies.
* Each user has rated **at least 20 movies.**
* Data is tab separated
* Data for the users and items are numeric and are consecutively ordered from 1
* Simple demographic info for the users (age, gender, occupation, zip)
* Genre information of movies (value of 1 denotes movie belongs to that genre and 0 otherwise)

|  |  |
| --- | --- |
| **Attribute Name** | **Attribute Description** |
| **user\_id** | User ID |
| **item\_id** | Item ID |
| **rating** | Rating from 1 to 5 stars |
| **timestamp** | Time of rating. Time stamp are in unix second since 1/1/1970 UTC |
| **movie\_id** | Movie ID |
| **movie\_title** | Title of the movie |
| **release\_date** | Release date |
| **video\_release\_date** | Date of Video Release |
| **imdb\_url** | IMDB URL link |
| **unknown** | Unknown genre type of movie |
| **action** | Genre type of movie |
| **adventure** | Genre type of movie |
| **animation** | Genre type of movie |
| **children** | Genre type of movie |
| **comedy** | Genre type of movie |
| **crime** | Genre type of movie |
| **documentary** | Genre type of movie |
| **drama** | Genre type of movie |
| **fantasy** | Genre type of movie |
| **film\_noir** | Genre type of movie |
| **horror** | Genre type of movie |
| **musical** | Genre type of movie |
| **history** | Genre type of movie |
| **romance** | Genre type of movie |
| **sci\_fi** | Genre type of movie |
| **thriller** | Genre type of movie |
| **war** | Genre type of movie |
| **western** | Genre type of movie |
| **age** | Age of user |
| **gender** | Gender of user |
| **occupation** | Occupation of User |
| **zipcode** | Zip code of user |

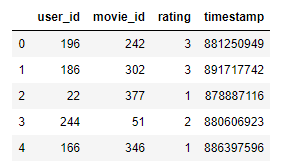
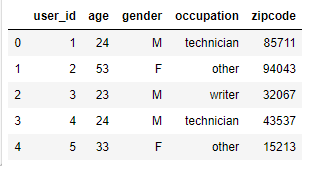
# Approach

**1) Data Preparation**

The data that was provided was split between three main files: u.data, u.item, and u.user. Each file needed to be loaded and merged into one single data frame in Python. **(Figures 1.0-1.2)** The dataset did not need much cleaning as it came with the genre attributes HotOneEncoded into numeric data types for ease of use when running statistical models. Categorical attributes such as gender, and occupation required alterations and were HotOneEncoded and transformed into numeric attributes for similarity measures and algorithms to run on the dataset. This process is outlined in the Exploratory Data Analysis Stage (**Figures 3.19 to 3.21)**

**2) Feature Selection**

Unnecessary columns such as timestamp, video\_release\_date (which contained all null values), imdb\_url, zip code and release\_date were dropped from the data frame. These attributes were dropped because they would not be included in the movie\_matrix (user\_id as the index, movie\_titles as the columns, and ratings as the values). The contents of the matrix were filtered by conditions such as age and occupation. The business problem was centered around providing similar movies to consumers that they would likely consume. The dataset was used to determine the similar movies to general user base as well as consumer who were part of a certain demographic of occupation and age. Therefore, timestamp, video\_release\_date (which consisted of all null values), imdb\_url, zipcode and release\_date were dropped from this dataset.



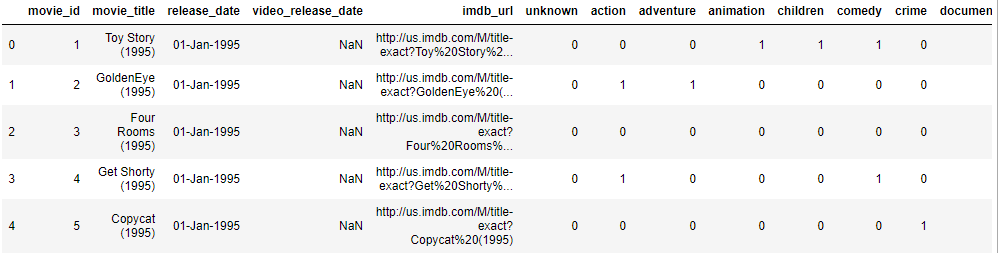
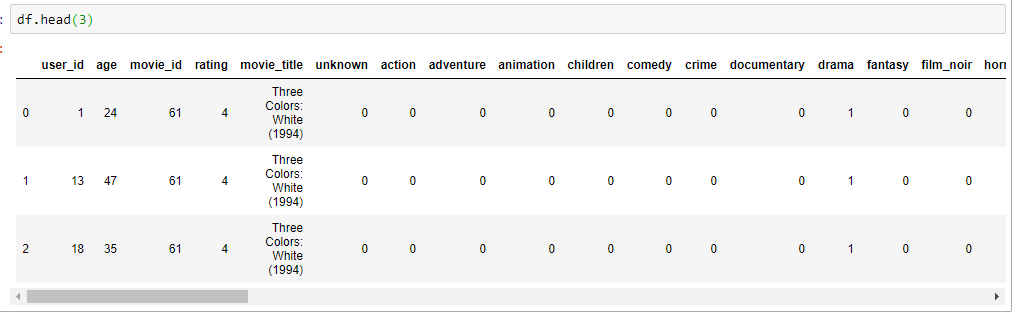
 Figure 1.0 Figure 1.1

Figure 1.2

**3) Exploratory Data Analysis**

To understand the data and its attributes, functions such as describe, shape, dtypes and groupby were implemented to retrieve statistical data from numeric attributes, understand the distribution of the of categorical variables, as well as the data types of the columns. **(Figure 3.0 to 3.16)** Discoveries such as, average rating being 3.52, average age of a consumer being 32.96, most users having the occupation of student, and homemakers making up the least demographic of users by occupation were documented.

In addition, various plots such as boxplots, and correlation heatmaps were used to see the distribution of the attributes and their relationship with other attributes such as ratings. Most ratings are between 3 and 4. Thefigures **3.11 and 3.12** showcased that most of the movies were not rated and were given a rating of zero. This finding demonstrated that the data did not contain much explicit rating information, which is a primary determining factor in many item-based recommender systems. Also, most movies had between 100-300 rating reviews, which indicated where a threshold for rating reviews could be drawn if considered in the recommender model.

**Figure 3.3** ofthe boxplot of occupation attribute revealed the outliers by occupation who tended to rate movies lower and higher than the average rating of 3.5. Healthcare workers and executives tended to give lower rating scores than average while artists, lawyers and users with no occupation were more generous in their ratings.

Grouping the ratings column in a data frame with average rating for each movie and having the number ratings added for each movie column next to the average rating column depicted that some movies have hardly any reviews compared to movies with over 100 reviews.**( Figure 3.15)**

During the exploratory data analysis phase the attributes occupation and gender were HoteOneEncoded and transformed into numeric data types, so that they could be included in correlation calculations**. (Figure 3.19 to 3.21)**

The most revealing finding was gained from a joint plot of ratings vs number of ratings. It showcased that the more ratings a movie gets the higher the rating it receives. Therefore, more popular movies received higher ratings**(Figure 3.16)**

Correlation heatmaps, plots and tables revealed some insightful information as well. When seeing if any variables in the data frame were highly correlated with one another there was only a strong negative correlation between gender\_M and gender\_F attributes which makes perfect sense given that the inverse of a male is a female. It is common practice to eliminate one of the two highly correlated variables, if one entering data in a model. However, the features remained in the dataset as they were not the focus of the study were not included in the recommender engine. **(Figure 3.25)**

The Dataset was then converted into a matrix as a subsequent step to building a recommender engine. The columns in the matrix were the movie\_titles and the rows were the user\_ids. Each column represented all the ratings of a movie by all the users. The ratings appeared as NAN where a user didn't rate a certain movie. This matrix was used to compute the correlation between the ratings of a single movie and the rest of the movies in the matrix. Pandas pivot table function was used to create the movie\_matrix. **(Figure 3.17)**

The pandas sort\_values function was utilized, and ascending was set to false in order to sort the movies from the most rated. The Head() function was used to view the top 10 most rated movies. Star Wars place at the top of the most rated movie list cemented its place as the movie of choice in this study. **(Figure 3.26)**

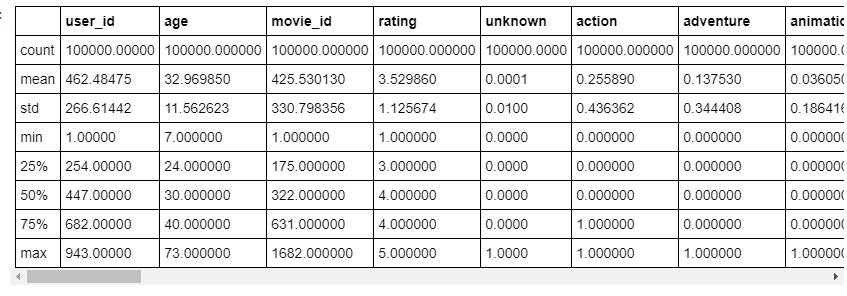


Figure 3.0

Describe function revealed the average rating is 3.52 and the average age of a consumer was 32.96as, the quadrants of the boxplots standard deviation, as well as the min and maximum values of the attributes.



Figure 3.1 Used shape function to determine how many rows and columns were in the data frame after merging the files into one data frame. There were 100K rows and 26 attributes initially

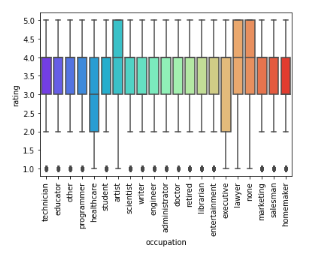




Figure 3.3

The box plot of the occupation column vs the rating score showcases that the average rating of 3.5 was the average rating score given to movies for the majority of the 21 occupation categories. Outliers consisted of occupations that went against the norm. Healthcare workers and executives tended to give lower rating scores than average while artists, lawyers and users with no occupation were more generous in their ratings.

Figure 3.4

Grouped the categorical occupation column by size to determine the distribution of user occupation, and found that most of the users are students and, the least occupation was homemaker.

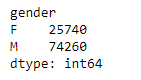


Figure 3.5

About 75% of user are male and about 25% are female

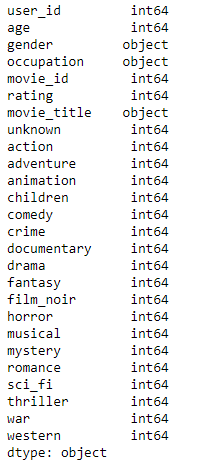


Figure 3.6 Figure 3.7

Checked which attributes are numeric with dtypes function. Checked to see if there were any nulls in data frame. There were no nulls in data frame:

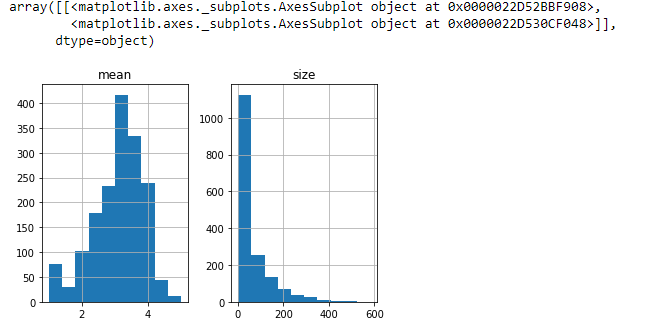


Figure 3.11 Figure 3.12

Made various plots to see the distribution of ratings. Most ratings are between 3 and 4. The second figure shows us that most of the movies were not rated and were given a zero. Also, that most movies have between 100-300 rating reviews.

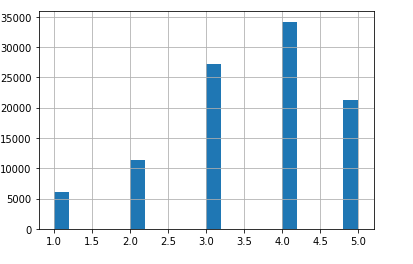


Figure 3.13

Most ratings are between 3 and 4.

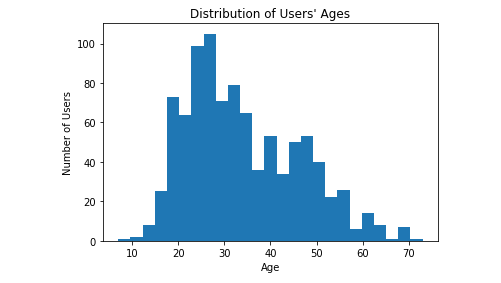


Figure 3.14

Most users are between 20 and 40.

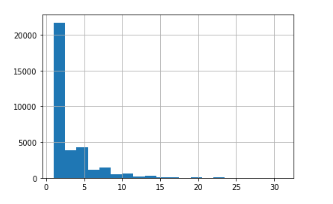


Figure 3.15

Most movies have 0 ratings by age, but there is an increase after 2.5 ratings.

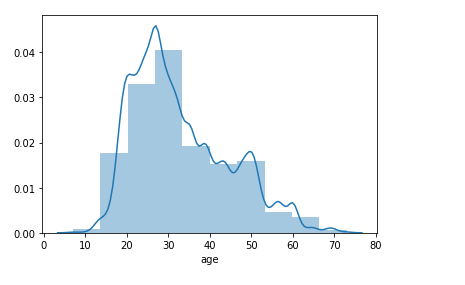


Figure 3.13

Age has an approximate normal distribution

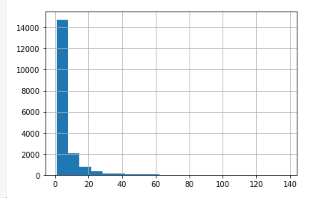


Figure 3.14

Most movies have 0 ratings by occupation, but there is an increase after 10 ratings

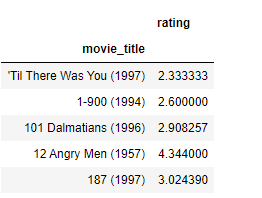
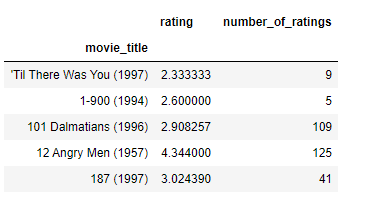


Figure 3.15

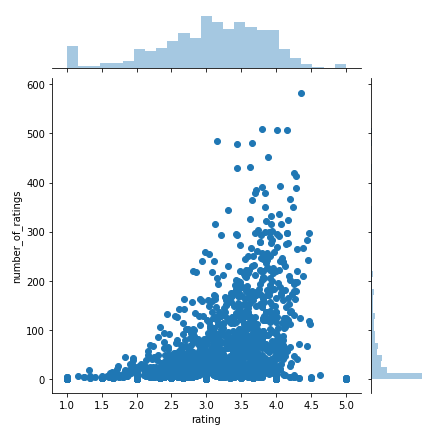
Grouped ratings column in a data frame with average rating for each movie. The number ratings column was added for each movie column next to the average rating column. Can clearly see that some movies have hardly any reviews compared to movies with over 100 reviews.

Figure 3.16

The joint plot of ratings vs number ratings showcases that the more ratings a movie gets the higher the rating it receives. Therefore, more popular movies received higher ratings.

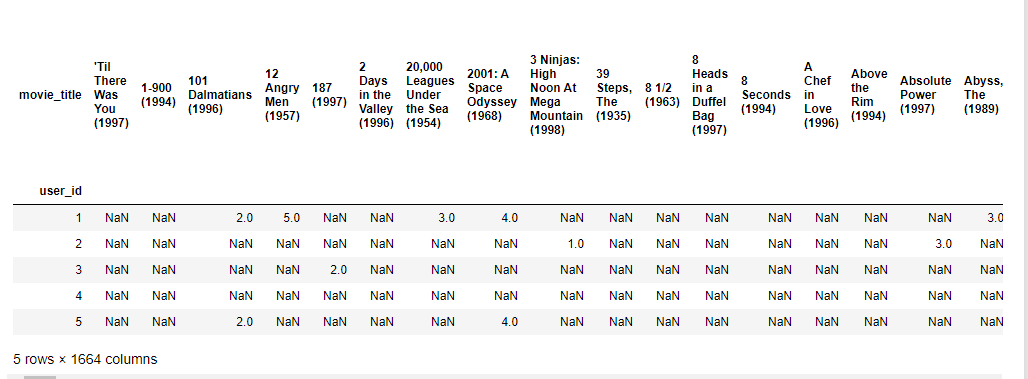


Figure 3.17

Dataset was converted into a matrix with the movie\_titles as the columns, the user\_id as the index and the ratings as the values. The columns in the matrix were the movie\_titles and the rows were the user\_ids. Each column represented all the ratings of a movie by all the users. The ratings appeared as NAN where a user didn't rate a certain movie. This matrix was used to compute the correlation between the ratings of a single movie and the rest of the movies in the matrix. Pandas pivot table function was used to create the movie\_matrix.

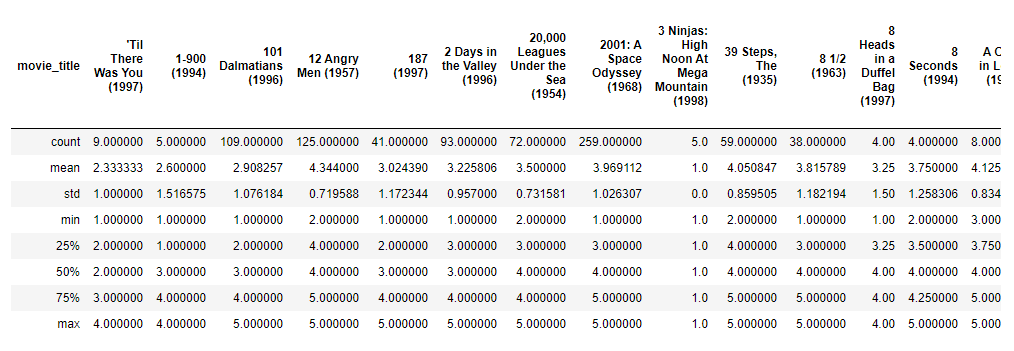
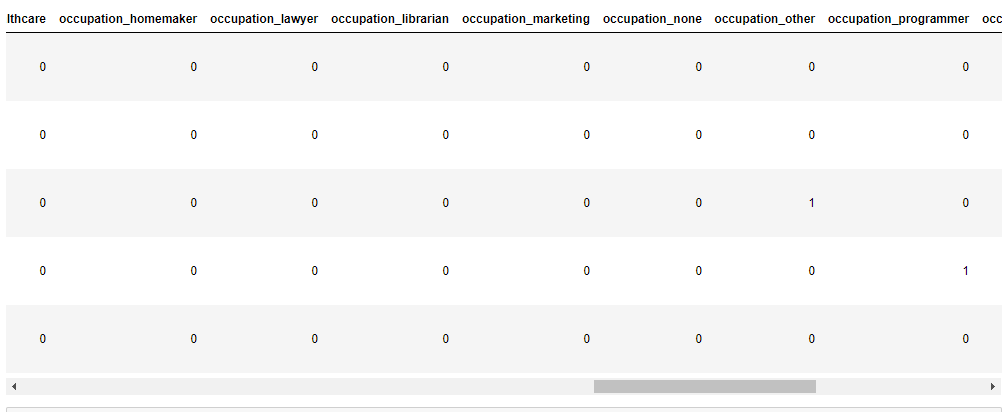


Figure 3.18

Movie matrix with user\_id as the index, ratings as the values, and movie\_titles as the attributes. He describe function was used to view statistical data such as the average rating for each movie, and the standard deviation.



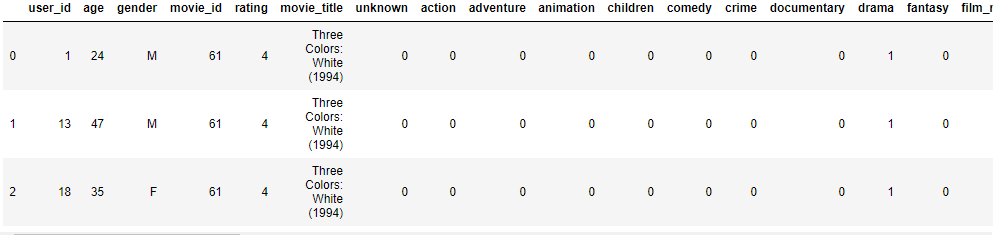


Figure 3.19

Changed the categorical/object attribute Occupation into dummy variables that are HotOneEncoded to obtain numeric attributes that can be used in various similarity models. The original categorical occupation column was then dropped from our data frame.

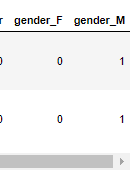




Figure 3.20

Did the same for the gender attribute. Leaving us with 47 attributes.

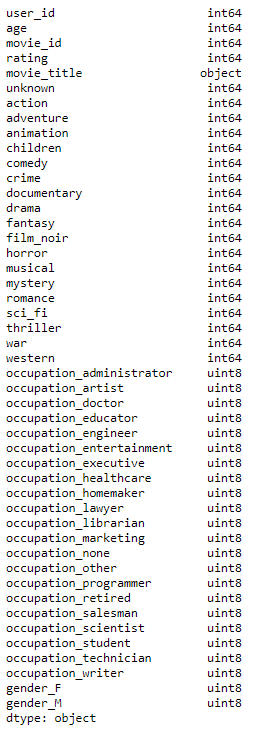


Figure 3.21

Attribute types were viewed with the dtype function to verify if occupation attributes have been converted into numeric columns that can be entered into statistical models.

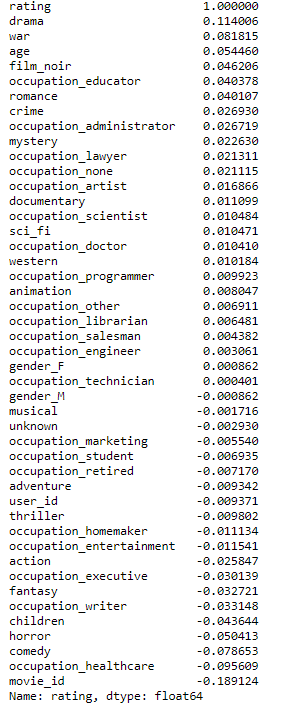
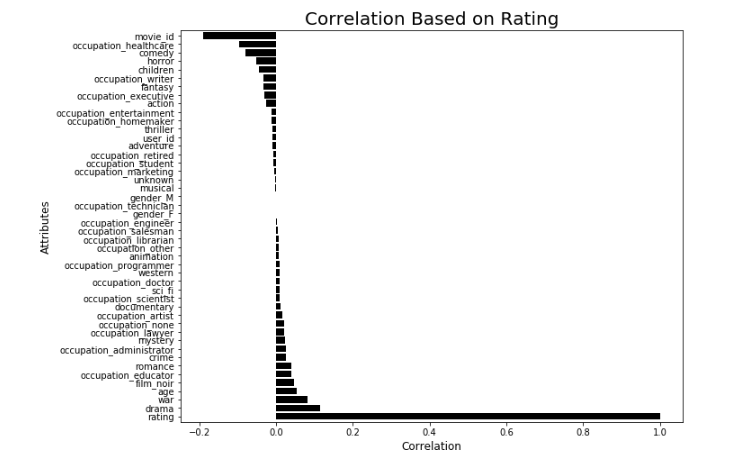


Figure 3.23

Looked at the most correlated attributes with the rating column. The top 5 correlated attributes with the rating column, other than ratings itself were: drama, war, age, film\_noir, and occupation\_educator. The presence of age and an occupation in the top 5, inspired to see if adding these attributes to the movie\_ matrix could better the outcome of the recommender system.

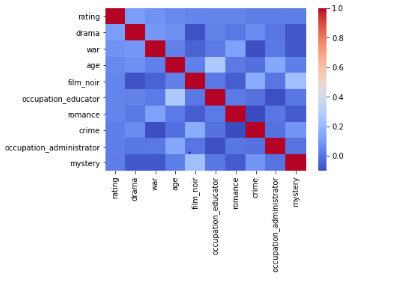


Figure 3.22

Figures 3.23 and 3.24

Produced a head map with the seaborn library to display the top 10 correlated attributes with rating to see if there was a significant correlation between them. Figure 3.23 displays the correlated vales to ratings in descending order. There is not a strong negative or positive correlation between these attributes

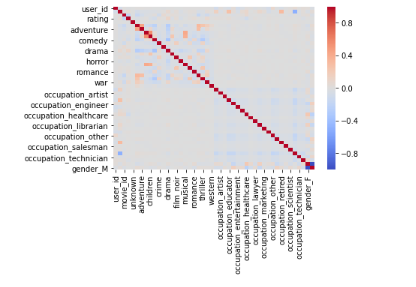


Figure 3.25

There is only a strong negative correlation between gender\_M and gender\_F attributes which makes perfect sense given that the inverse of a male is a female. It is common practice to eliminate one of the two highly correlated variables, if you are entering your data in a model. However, the features remained as they are not the focus of the study would not be included in the recommender engine.

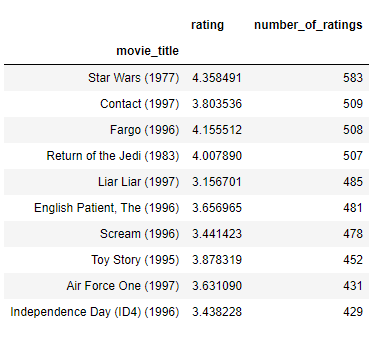


Figure 3.26

The most rated movies were displayed and one of them was chosen to work with in this simple recommender system. The pandas sort\_values function was utilized, and ascending was set to false in order to sort the movies from the most rated. Head() function was used to view the top 10.

**4) Build Recommender Engines**

A series of matrices, entailing insightful information such as individual user individual preferences and average ratings for each movie will be incorporated with a various similarity measures and algorithms to build movie recommender engines.

1. Most Popular Movie Recommender System
2. Collaborative Item-based Movie Recommender with Pearson Correlation
3. Collaborative Item-based Movie Recommender with Pearson Correlation Age under 25
4. Collaborative Item-based Movie Recommender with Pearson Correlation Age over 30
5. Collaborative Item-based Movie Recommender with Pearson Correlation Occupation Administrator
6. Collaborative Item-based Movie Recommender with Pearson Correlation Occupation Student
7. Collaborative Item-based based Movie Recommender with KNN Baseline, Pearson Baseline
8. Collaborative Item-based based Movie Recommender with KNN Baseline, Cosine
9. Collaborative Item-based based Movie Recommender with KNN Basic, Cosine
10. Collaborative Item-based based Movie Recommender with KNN Basic, MSD
11. Collaborative Item-based based Movie Recommender with KNN Baseline, Pearson

**Model 1:**

**Popularity Based Movie Recommender Engine:**

As per the histogram in **(Figure 3.12)** in the exploratory data analysis stage, it was apparent that the data set contained many movies that were not rated, which was safely be assumed that not many users watched these movies. One of the main concerns in recommender systems that needs to be addressed is the Cold Start Problem. The Cold Start Problem occurs when there is a new user or a new item in the recommendation system, and there isn’t enough implicit rating information available to generate a similarity between two movies. To ameliorate this problem popular recommendation systems are used to generate recommendations for new consumer, by generating the most bought products, highest rated lists, or for this dataset, a highest and most rated movie recommender engine.

The main concept behind this recommender engine was that popular movies receive more reviews and generally have high rating scores, making it likely that it would be well received by a new consumer. This was substantiated in the EDA analysis with a joint plot of ratings vs number of ratings. Please see **Figure 4.0** below:

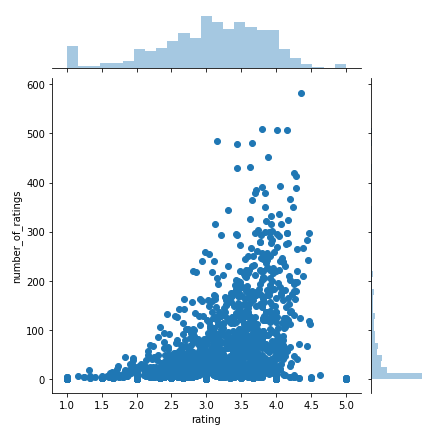


Figure 4.0

The popularity-based movie recommender was generated by grouping the data frame of the dataset by movie\_title and average rating (**Figure 4.1).** A number\_of\_ratings column was then appended to the data frame. (**Figure 4.2)** The data frame sorted further by movies that had the most rating review and were given a score of 4 or more. **(Figure 4.3)**

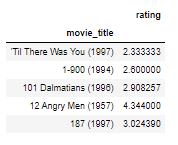


Figure 4.1

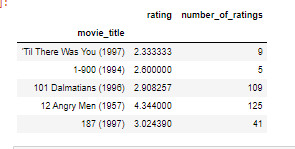


Figure 4.2

**Model 1:**

**Popularity Based Movie Recommender Engine:**

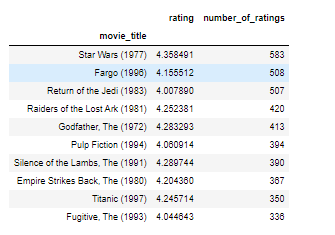


Figure 4.3

**Model 2:**

**Collaborative Item-based Movie Recommender Systems with Pearson Correlation**

Item based recommender engines are by far the most popular type of recommender system academia and generally outperform engines that make recommendation to consumer based on consumer similarity. This is usually due to the simple fact that it is easier to categorize an item and find similarities than it is to categorize a consumer, who could have a plethora of tastes and tendencies.

This item-based model with Pearson Correlation was achieved by assuming that a user watched film Star Wars (1977). The model looked for movies that were similar to Star Wars and recommended similar movies to the respective user. The recommendations were retrieved by computing the Pearson Correlation between the Star Wars movie’s ratings and the ratings of the rest of the movies in the movie\_ matrix. Pearson correlation was selected as the similarity measure due to its well documented performance in academia.

The first step was to create a data frame with the ratings of Star Wars by extracting the Star Wars ratings from the movie\_matrix.**(Figure 4.4)**The corrwith function was then used to compute the similarity between the Star Wars object and the movie matrix data frame. The function generates the pairwise correlation of rows or columns of two data frame objects. **(Figure 4.5)**

Most of the movies in the dataset were not rated, causing the movie\_matrix as well as the similar\_to\_star\_wars correlation results to have null values. The correlation results were then transformed into a data frame for aesthetic purposes and the null values were dropped. **(Figure 4.7)**

In the EDA phase of this study, it was revealed that there is a sharp decline in the number of ratings from 100. The threshold of 100 was set for the number of ratings a movie had to have to be included in the recommendations. This was an important step as without it, it would have been possible for highly rated movies with very few rating reviews to be recommended to the consumer. The Star Wars data frame was then joined with the number\_of\_ratings column in the ratings data frame and was sorted further by most correlated values to generate the movie recommendations for users who were looking for movies similar to Star Wars.**(Figure 4.9)**

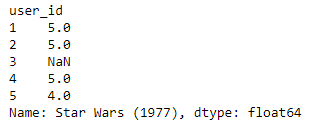


Figure 4.4

Star Wars data frame with the user\_id and the rating given for the Star Wars movie

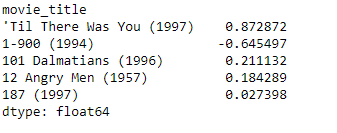


Figure 4.5

The output of the correlation of the Star Wars movie’s ratings and the ratings of the rest of the movies in the movie\_matrix. Sorted the correlated output in descending order to get the most correlated movies

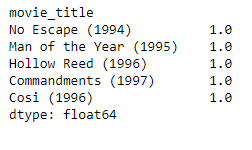


Figure 4.6

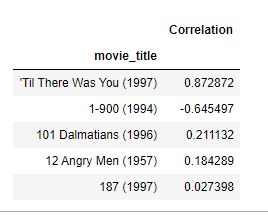


Figure 4.7

The matrix had many missing values since not all the movies were rated by all the users. Therefore, the null values were dropped, and the correlation results transformed into data frame,

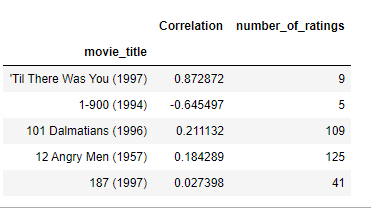


Figure 4.8

The Star Wars data frame was then joined with the number\_of\_ratings column in the ratings data frame

The movies that are most like Star Wars (1977) were obtained by creating a condition that limits the output to movies that have at least 100 reviews. This was done because most movies have at least 100-300 ratings reviews. Is imperative that movies that a few people rated highly are not included in the recommendation. The recommendations were then sorted by the correlation column and the first 10 were displayed.

It is not a surprise that the most correlated movie was Star Wars itself, other similar movies included Empire Strikes Back, Return of the Jedi and Raiders of the Lost Ark. All of which make perfect sense, as they are quite like Star Wars. What is surprising is that Empire Strikes Back only has a positive correlation of 0.75. One could assume that it would have been higher value, as it is the sequel to Star Wars.

This ignited the thought that this recommender system could be improved. In addition to a better similarity predicter, there is a cold start problem that needs to be addressed. Most of the movies have received a zero rating, which likely means that most movies were not watched by the users, have no rating reviews and would not be recommended. The dataset provided us with information such as age and occupation that can be included in the matrix to possibly create better performance and mitigate the cold start problem.

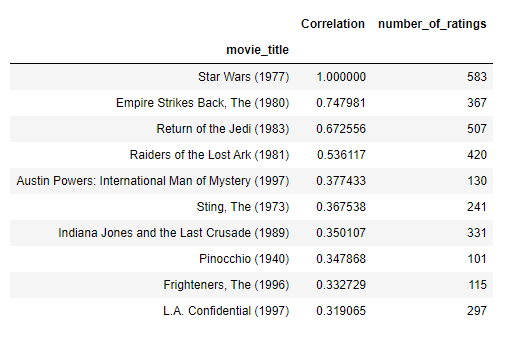


Figure 4.9

**Model 3:**

**Collaborative Item-based Movie Recommender Systems with Pearson Correlation Age: under 25**

The presence of the age attribute in the top 5 highly correlated values with ratings ignited the curiosity of determining whether enhancing the Collaborative Item-based Movie Recommender with Pearson Correlation with age could better the system’s recommendations.

This notion was actualized by following the same steps in the previous model but filtering the contents of the movie\_matrix with users restricted by the age condition under 25 years old.

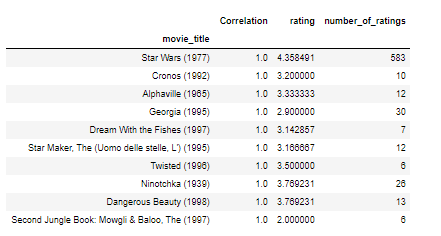
The threshold, as depicted in the EDA section was set to 2.5 ratings reviews given the increase in reviews after 2.4 review in the histogram **(Figure 3.15)**.

Figure 4.10

The output of the recommender restricted by the condition under 25 years of age. Only 272 rows were included in the movie\_matrix, making the data quite sparse. Making the output somewhat inaccurate

**Model 4:**

**Collaborative Item-based Movie Recommender Systems with Pearson Correlation Age: over 30**

Similar steps were taken as the previous model, but the contents were filtered by a condition over 30 years of age The threshold, as depicted in the EDA section was set to 2.5 ratings reviews given the increase in reviews after 10 reviews in the histogram.

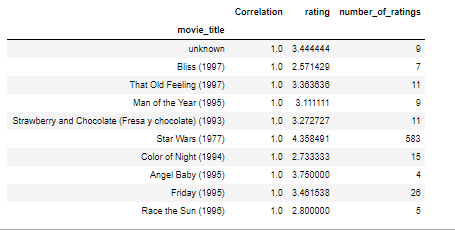




Figure 4.11

The output of the recommender restricted by the condition over 20 years of age. Only 534 rows were included in the movie\_matrix, making the data bigger than the last model but still quite sparse. Making the output somewhat inaccurate as well.

**Model 5:**

**Collaborative Item-based Movie Recommender Systems with Pearson Correlation Occupation: Administrator**

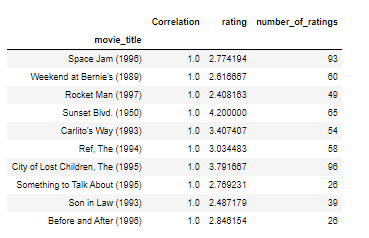




Figure 4.12

The output of the recommender restricted by the condition occupation as an administrator. Only 79 rows were included in the movie\_matrix, making the data extremely sparse. Making the output somewhat inaccurate as well.

**Model 6:**

**Collaborative Item-based Movie Recommender Systems with Pearson Correlation Occupation: Student**

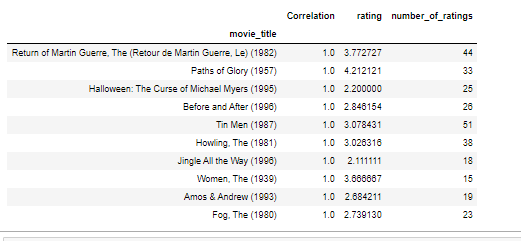




Figure 4.14

The output of the recommender restricted by the condition occupation as a student. Only 196 rows were included in the movie\_matrix, making the data extremely sparse. Ecen though student are the largest demographic by occupation, the output is still somewhat inaccurate.

**Model: 7**

**Collaborative Item-based Movie Recommender System with KNN Baseline and Pearson Baseline Similarity**

The previous models used either a groupby function or Pearson similarity to locate similar items. K Nearest Neighbors algorithm is another commonly used method to generate an item-based recommender system. In supervised datasets it is common practice to train an algorithm on 80 percent of the data to make the algorithm learn the data and make prediction and compare the prediction performance with the actual results on (untrained 20% of the data) the test set. This methodology ignited the notion of building a recommender engine with the KNN classification algorithm that incorporated similarity measures and verified its performance on a new data through training and testing.

The KNN engines were created with the surprise library package in python. This package was specifically built to easily build recommender engines with various algorithms and parameters. The MovieLens 100K dataset and the training and testing function (split 80% training, 20% testing) are built in, and can be called easily.

A defined similarity function was created with the parameters of similarity functions and algorithm type. The selected model was then fitted to the training set and the predictions were made and compared with the actual result of the test set. Root Mean Squared Error was then called to evaluate the predicted and actual values. Another function was defined to map the movie\_id to the movie name and vice versa. This function was implemented in a for loop that ran through the entire dataset. A show similar movies function was then created to match the movie id to the movie input (Star Wars) and to generate the 10 nearest neighbors.

**Model: 7**

**Collaborative Item-based Movie Recommender System with KNN Baseline and Pearson Baseline Similarity**

**RMSE: 0.6404**

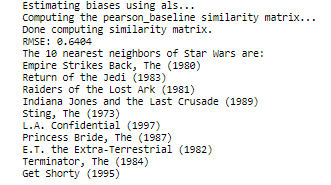


Figure 4.17

**Model: 8**

**Collaborative Item-based Movie Recommender System with KNN Baseline and Cosine Similarity**

**RMSE: 0.5945**

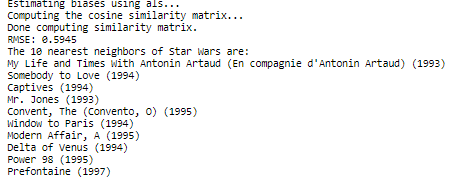


Figure 4.18

**Model: 9**

**Collaborative Item-based Movie Recommender System with KNN Basic and Cosine Similarity**

**RMSE: 0.4893**

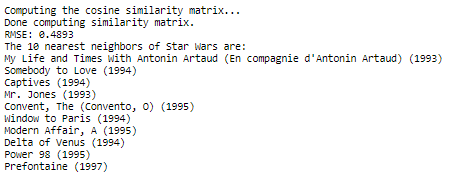


Figure 4.19

**Model: 10**

**Collaborative Item-based Movie Recommender System with KNN Basic and Mean Squared Differences Similarity**

**RMSE: 0.5083**

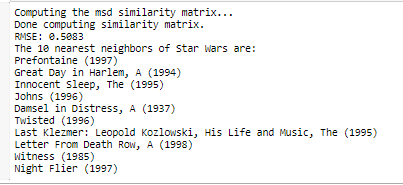


Figure 4.20

**Model: 11**

**Collaborative Item-based Movie Recommender System with KNN Baseline and Pearson Similarity**

**RMSE: 0.6244**

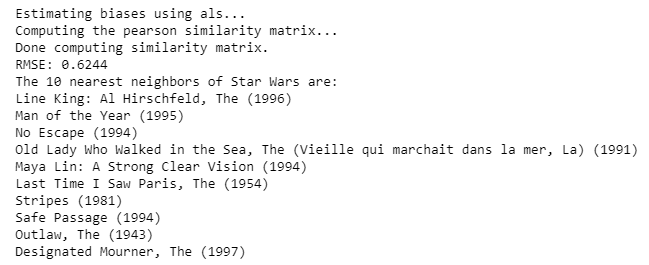


Figure 4.21

**5} Evaluation and Conclusion**

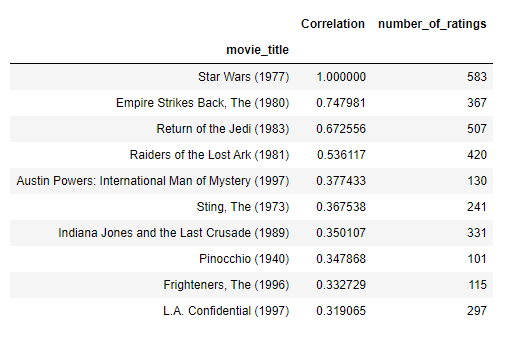
The Recommender systems will be evaluated based on their ability to recommend movies that can generate a profit for a movie distribution company such as Netflix. The positive aspects as well as the drawbacks of the systems will be considered.

The popular movies recommender would generate money for businesses where most of the consumers base do not rate the movies or leave personal information on file. This simple engine ameliorates the issue of the Cold Start problem by recommending movies that have a minimum of 100 reviews and a minimum rating of 4 out of 5. In the EDA phase a linear relationship between movies that had a high number of ratings and a high rating score was well documented. This finding showcased that there is a high likelihood that the average consumer would like some of the recommendations provided by this model.

One of the major drawbacks of the popular movie recommender model is that recommendations are not personalized to a specific user or item. Also, the movie recommendations outputs well known film, which increases the likelihood that the user has already watched most of them, and that the recommendations would not inspire them to watch or purchase more content.

The Collaborative models with Pearson Correlation provided personalized results and recommended 10 movies that are similar to a movie (Star Wars) the consumer prefers. This personalization would restrict a user to options that match their needs, making them more likely to consume. The convenience and relevancy of the models where conditions on age and occupation were implemented on the movie matrix, generated correlation values of 1.0 for all of the recommendations. The correlation of 1.0 indicates an almost identical similarity to the movie in question, while a -1.0 depicts an inverse relationship. When the movie\_matrix became constricted to only people of a certain age group or people of a certain occupation, the rating data on movies becomes very scarce. This scarcity lead to poor recommendations as the first recommendations other than the Star Wars action movie itself, for the two models were Bliss, a romantic film and Cronos, a horror film. These movies would not be considered an almost identical match to Star Wars. Therefore, the Collaborative Item-based Movie Recommender Systems with Pearson Correlation alone out performed the other two models in its family .The use of the full movie\_matrix content, generated relevant results(second recommendation was the sequel to Star Wars) and more feasible correlation numbers.

The main disadvantage of this family of item-based recommendation systems, is that if an item doesn’t have any implicit rating data, then no recommendations can be made for it. Given that most of the movies in the dataset have a rating of 0 and are unwatched, it is highly probably that movies that are not very popular would not receive recommendations or be included in the engine’s output. Please see **Figures 5.0 to 5.2.**



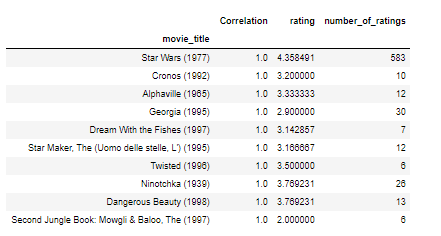
Figure 5.0 Recommendations for users who watched Star Wars

Figure 5.1

Recommendations for users under 25 who watched Star Wars

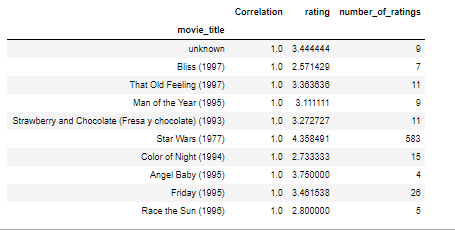


Figure 5.2

Recommendations for user over 30 who watched Star Wars

In all the KNN models in this study, the Root Mean Squared Error can be generated due to the KNN classifier algorithms ability to utilize the training and testing sets in the surprise library. The training and testing ability allow for the predicted k-nearest neighbors vs the actual results to be compared. **Root Mean Square Error** (RMSE) is the standard **deviation** of the residuals (prediction **errors**). The lower the RMSE the better the model performance.

The best performing KNN model that generated the lowest Root Mean Square Error was KNN Basic with Cosine similarity, with an RMSE of 0.4893. This is a very low RMSE, which gives credence to the similarity power of the model and its ability to yield favorable result for consumers. Favorable results make it much easier for users to filter through their options and consume movie content, which equals to quick and easy transactions or views for the business.

|  |  |  |
| --- | --- | --- |
| **KNN Model Type** | **Parameters** | **RMSE** |
| KNN Baseline | Pearson Baseline, Item-based | 0.6404 |
| KNN Baseline | Pearson, Item-based | 0.6244 |
| KNN Baseline | Cosine, Item-based | 0.5945 |
| KNN Basic | Cosine, Item-based | 0.4893 |
| KNN Basic | Mean Squared Differences, Item-based | 0.5083 |

Figure 5.3

KNN Comparison Table

Similar to the Collaborative item-based models with Pearson similarity, the negative aspects of the model was that popular movies had a tendency to be recommended, and un-popular movies ran the risk of not having enough information to generate recommendations or be included in the recommendation list.

Determining the best model overall is quite a difficult task as each model can satisfy different needs for various movie distribution businesses. Given that the purpose of an item based recommendation system is to generate similar results for an inputted item in an accurate way, the KNN Basic model with Cosine similarity would most likely fit the needs of an average movie distributer due to it having a low RMSE and a proven ability to perform well on new data information. Making it the preferred model out of the eleven recommender engines in this study.

**Conclusion**

In conclusion, various recommender models were compared and evaluated on how profitable the recommendations could be for a movie distribution business. Lack of implicit rating data from the users had a detrimental effect on the recommendation output of all the item-based models and presented some items as similar when they were just the most similar of the limited data provided. Although each model had the ability to remedy various business concerns, the recommender systems that could personalize a user’s recommendation results and retain high performance on new data outperformed the other models. KNN Basic with Cosine Similarity reigned as the model with the performance to securely generate revenue for a movie distribution business in this study.

**References**

1. Najafabadi, M. K., & Mahrin, M. N. R. (2016). A systematic literature review on the state of research and practice of collaborative filtering technique and implicit feedback. *Artificial intelligence review*, *45*(2), 168.
2. Xue, F., He, X., Wang, X., Xu, J., Liu, K., & Hong, R. (2019). Deep item-based collaborative filtering for top-n recommendation. *ACM Transactions on Information Systems (TOIS)*, *37*(3), 1-6.
3. Zhang, M., Guo, X., & Chen, G. (2016). Prediction uncertainty in collaborative filtering: Enhancing personalized online product ranking. *Decision Support Systems*, *83*, 12.
4. Yuan, X., Han, L., Qian, S., Xu, G., & Yan, H. (2019). Singular value decomposition based recommendation using imputed data. *Knowledge-Based Systems*, *163*, 485-494.
5. Bobadilla, J., Ortega, F., & Hernando, A. (2012). A collaborative filtering similarity measure based on singularities. *Information Processing & Management*, *48*(2), 204-211.
6. Bobadilla, J., Ortega, F., & Hernando, A. (2012). A collaborative filtering similarity measure based on singularities. *Information Processing & Management*, *48*(2), 213-217.
7. Bobadilla, J., Serradilla, F., & Bernal, J. (2010). A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, *23*(6), 520-523.
8. Bobadilla, J., Serradilla, F., & Bernal, J. (2010). A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, *23*(6), 525-528.
9. Li, Y., Hu, J., Zhai, C., & Chen, Y. (2010, October). Improving one-class collaborative filtering by incorporating rich user information. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 959-963).
10. Li, Y., Hu, J., Zhai, C., & Chen, Y. (2010, October). Improving one-class collaborative filtering by incorporating rich user information. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 965-968).
11. Zhang, M., Guo, X., & Chen, G. (2016). Prediction uncertainty in collaborative filtering: Enhancing personalized online product ranking. *Decision Support Systems*, *83*, 10-13.
12. Zhang, M., Guo, X., & Chen, G. (2016). Prediction uncertainty in collaborative filtering: Enhancing personalized online product ranking. *Decision Support Systems*, *83*, 115-18.
13. Zhang, M., Guo, X., & Chen, G. (2016). Prediction uncertainty in collaborative filtering: Enhancing personalized online product ranking. *Decision Support Systems*, *83*, 20-21.
14. Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008, June). Large-scale parallel collaborative filtering for the netflix prize. In *International conference on algorithmic applications in management* (pp. 340-348). Springer, Berlin, Heidelberg.
15. Yuan, X., Han, L., Qian, S., Xu, G., & Yan, H. (2019). Singular value decomposition based recommendation using imputed data. *Knowledge-Based Systems*, *163*, 485-490.
16. Yuan, X., Han, L., Qian, S., Xu, G., & Yan, H. (2019). Singular value decomposition based recommendation using imputed data. *Knowledge-Based Systems*, *163*, 492-494.
17. Kuzelewska, U. (2014). Clustering algorithms in hybrid recommender system on movielens data. *Studies in logic, grammar and rhetoric*, *37*(1), 125-128.
18. Kuzelewska, U. (2014). Clustering algorithms in hybrid recommender system on movielens data. *Studies in logic, grammar and rhetoric*, *37*(1), 136-139.
19. Xue, F., He, X., Wang, X., Xu, J., Liu, K., & Hong, R. (2019). Deep item-based collaborative filtering for top-n recommendation. *ACM Transactions on Information Systems (TOIS)*, *37*(3), 1-8.
20. Xue, F., He, X., Wang, X., Xu, J., Liu, K., & Hong, R. (2019). Deep item-based collaborative filtering for top-n recommendation. *ACM Transactions on Information Systems (TOIS)*, *37*(3), 20-25.